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

Интеллектуальные Системы.

Теория и приложения

ТОМ **24** ВЫПУСК 2 * **2020**

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.

Спонсором издания является:

ООО «Два Облака»
Разработка корпоративных информационных систем
http://www.dvaoblaka.ru

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

Применение методов искусственного интеллекта для планирования персональной стратегии обучения

Бекташев Р.А.[1]

Аннотация

В статье поставлена задача планирования персональной стратегии обучения и приведены результаты применения различных подходов к ее решению на примере курса обучения программированию.

Ключевые слова: компьютерная обучающая система, предсказание элементов последовательности, нейронная сеть, рекуррентная нейронная сеть, компактное дерево предсказаний

1. Введение

Анализ данных в образовании (Educational Data Mining, EDM) становится эффективным инструментом в руках исследователей, который позволяет совершенствовать процесс обучения как со стороны студента, так и со стороны преподавателя. Решая широкий спектр задач, современные алгоритмы анализа данных позволят сгладить возможные различия в подготовке студентов, а также повысить ее средний уровень, что, несомненно, полезно для университетов. Новая университетская образовательная модель [1] ставит задачу построения такой информационно-образовательной среды, которая сможет служить платформой для применения предложенных методов анализа данных, способствуя более эффективному использованию ресурсов университета, студента и преподавателя. Подход с использованием вычислительных технологий успешно

[1] *Бекташев Руслан Анварович* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: ruslanbektashev@gmail.com.

Bektashev Ruslan Anvarovich — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

применяется во многих бизнес-моделях (рекомендательных системах, голосовых помощниках и т. п.), поэтому, даже учитывая специфичность образовательной модели, и здесь можно рассчитывать на аналогичные результаты.

В университетах программы курсов обычно разделены на темы, излагаемые во время лекций. Каждая тема закрепляется решением набора задач на семинарах. Для проверки решений этих задач преподаватели могут использовать автоматическую систему в которую студенты будут отправлять свои решения. Система может вести учет различных данных в процессе обучения (например, время решения задачи, количество неудачных попыток, качество решения и т. д.). Собранные данные можно использовать для выявления применяемых шаблонов, типичных ошибок и соответствующей корректировки плана обучения. Однако, не вся собранная информация сразу готова к обработке — ввиду того, что решением задачи является текст на некотором языке, оценка качества решения представляет собой отдельную непростую задачу. Такие свойства решения, как: уникальность, краткость, соответствие ограничениям и др. обычно выявляются преподавателем вручную, но этот процесс может быть частично автоматизирован. Для этого необходимо формальное определение понятия *оценка свойства решения*, после чего станет возможным построение соответствующих алгоритмов.

2. Постановка задачи

2.1. Основные понятия

Множество всех задач некоторого раздела обозначим P.

Определение 1. *Конечная последовательность слов в некотором конечном алфавите символов является решением задачи.*

Множество всех решений некоторой задачи $p \in P$ обозначим A_p.

Определение 2. *Множество* $S = \{s \mid 0 \leqslant s \leqslant 100,\ s \in \mathbb{R}\}$ *будем называть шкалой оценивания.*

Введем полный порядок на множестве решений относительно некоторого свойства f:

Определение 3. $\forall\, a, b \in A_p$ *верно одно из утверждений:*

- $a \leqslant_f b$, *если* b *в большей степени обладает свойством* f, *чем* a

- $a \geqslant_f b$, *иначе*

Определение 4. *Монотонное отображение* $F\colon A_p \to S$ *является оценкой свойства решения.*

Выбранная шкала оценивания может обеспечить инъективность оценки свойства решения, хотя это, вообще говоря, необязательно. Для некоторых свойств решения имеет смысл ввести упрощенную шкалу оценивания. К примеру, свойство решения быть верным (т. е. решать поставленную задачу) может принимать два значения: «да» (1) или «нет» (0).

Определение 5. *Пусть* $M \subset \mathbb{Z}_+$. *Монотонное отображение* $I\colon S \to M$ *назовем интерпретацией оценки свойства решения.*

Для вычисления значения F на практике необходимо построить соответствующий алгоритм, который принимает на вход решение, а на выходе выдает оценку свойства решения. Однако, определенная ранее шкала оценивания непрерывна, поэтому следует дискретизировать образ оценки свойства решения:

Определение 6. *Отображение* $\bar{F} = F \circ I$ *назовем дискретной оценкой свойства решения.*

2.2. Построение дискретного пространства признаков

Любое отображение F кодирует некоторое свойство f решения задачи $p \in P$. Большинство таких свойств бессмысленны или вырождены. Однако, некоторыми из этих свойств руководствуются преподаватели при оценивании решения $a \in A_p$ вручную. К примеру: решение проходит 7 из 10 тестов, подготовленных преподавателем для данной задачи; имеется 80-процентная схожесть с решениями данной задачи у другого студента.

Определение 7. *Множество* $\mathbb{F} = \{f_1, f_2, \ldots, f_n\}$ *свойств решения порождает пространство признаков*

$$T_a = S_1 \times S_2 \times \cdots \times S_n \simeq S^n \subset \mathbb{R}^n$$

решения a *задачи* p. *Компонента* t_i *точки* $t = (t_1, t_2, \ldots, t_n)$ *характеризует степень обладания свойством* f_i *решения* a.

Пространство признаков можно дискретизировать, если вместо F использовать $\bar{F}$ с интерпретацией $\bar{S} = \{0, 1, \ldots, 100\}$.

Определение 8. *Пространство признаков решения a, порожденное свойствами $\bar{\mathbb{F}}$, назовем дискретным пространством признаков и обозначим $\bar{T}_a$.*

Для построения необходимого нам дискретного пространства признаков следует определить множества осмысленных свойств решения каждой задачи из некоторого раздела, которые можно эффективно оценить.

Определение 9. *Объединение свойств решения всех задач из раздела порождает универсальное пространство признаков для раздела, обозначим его T_P .*

Основную сложность здесь представляет автоматическая оценка некоторого свойства.

Задача 1. *Построить алгоритм $V_{\bar{F}}$ оценивающий заданное свойство f со сложностью не более $O(l)$, где l - количество символов в решении.*

При вычислении оценки некоторых свойств решения, описываемых регулярными выражениями, можно построить конечный автомат, распознающий слова допускаемые этими выражениями. Примером такого свойства может служить «наличие или отсутствие некоторого множества слов». Для свойства «компилируется» алгоритмом является процесс компиляции [9], который не может быть реализован классическим конечным автоматом. Существуют также свойства, задача оценки которых алгоритмически неразрешима, например - «корректно решает поставленную задачу» [10]. Для решений, имеющих вид текстов программ на некотором языке программирования, подобные свойства рассматриваются в направлении верификации программ [11]. В нашем случае формальное доказательство корректности программы не обязательно, достаточно лишь доказательства корректности на некотором нетривиальном наборе входных данных - ограниченного тестирования черного ящика.

2.3. Планирование персональной стратегии обучения

Элемент $t \in \bar{T}_a$, отражающий совокупные свойства решения, обозначает степень освоения навыков, тренируемых или тестируемых задачами P. Последовательность

$$T_k = (t_0, t_1, \ldots, t_k),\ t_i \in \bar{T}_P,\ i = \overline{0,k}$$

отражает процесс обучения в виде результатов решения k последовательных задач из раздела.

Степень освоения студентом навыков или тем в данный момент времени можно также представить в виде точки в конечномерном дискретном пространстве $\bar{S}_1 \times \bar{S}_2 \times \cdots \times \bar{S}_m \simeq \bar{S}^m$, где каждой оси соответствует один навык или тема. Последовательность

$$L_q = (l_0, l_1, \ldots, l_q),\ l_i \in \bar{S}^m,\ i = \overline{0, q}$$

такая, что

$$l_i \preccurlyeq l_{i+1},\ i = \overline{0, q-1}$$

является кривой обучения [4] и описывает процесс обучения студента во времени с 1-го до q-го тактов измерения. Для данного типа кривых разработан алгоритм поиска ближайшего соседа [4], который можно использовать для планирования персональной стратегии обучения. Однако, условие монотонности в представлении данных имеет существенный недостаток — в таком случае не учитывается склонность студента к забыванию части информации при длительном обучении без закрепления.

Алгоритм поиска ближайшего соседа заключается в разбиении пространства $\bar{S}^m$ на такие сектора, что при попадании объекта-запроса в один из секторов, поиск сводится к перебору объектов внутри данного сектора. Фактически задается такое отображение точек

$$h\colon \bar{S}^m \to \{1, \ldots, k\},$$

что в соответствии с характеристиками точек суммарная характеристика сектора одинакова для всех секторов. Каждой точке s пространства $\bar{S}^m$ ставится в соответствие тройка (v, r, k), где: $v = \|s\|_\Sigma$, $r = \|s - c\|_\Sigma$, $k = \arg\min\{\|s - (v, \ldots, 0)\|_\Sigma, \ldots, \|s - (0, \ldots, v)\|_\Sigma\}$.

Номер сектора, которому принадлежит точка $s \in \bar{S}^m$ по заданной тройке вычисляется по формуле

$$h(s) := m(100m + 1) \cdot l(v, r) + m \cdot g(r) + k,$$

где

$$l(v, r) = \left\lfloor \frac{v}{g(r) + 1} \right\rfloor,\ g(r) = \begin{cases} r, & r \leqslant 1 \\ r - 1, & r > 1 \end{cases}$$

В случае, если сектор с номером $h(s)$ не содержит объектов, находятся соседние сектора, в которых поиск продолжается. Соседние сектора определяются как объединение трех множеств:

$$\{h(s_1) \mid s_1 \to (l(v, r), g(r), i),\ 0 \leqslant i \leqslant m,\ i \neq k\},$$

$$\{h(s_2) \mid s_2 \to \begin{cases}(l(v,r)-1, g(r), k), & l(v,r) > 0\\ (l(v,r)+1, g(r), k), & (l(v,r)+1)(g(r)+1) \leqslant 100m\end{cases}\},$$

$$\{h(s_3) \mid s_3 \to \begin{cases}(l(v,r), g(r)-1, k), & g(r) > 0\\ (l(v,r), g(r)+1, k), & r \leqslant 100m\end{cases}\}.$$

Доказательство корректности алгоритма и оценка его сложности приведены в [4].

Заметим, что переход от последовательности типа T к последовательности типа L, без условия монотонности элементов последней, прост. В самом деле, достаточно покомпонентно перебирать элементы T и заполнять компоненты элементов L, оставляя наибольшую из компонент предыдущего элемента L и текущего элемента T.

Сформулируем задачу планирования стратегии обучения используя кривую обучения. Каждый студент находится в некотором начальном состоянии l_0, которому соответствует кривая $L_0 = (l_0)$. Цель обучения — через q тактов измерения оказаться в состоянии $l_{max} = (100, \ldots, 100)$, то есть освоить все навыки и темы некоторого раздела. Кривая, соответствующая успешному обучению:

$$L_q = \underbrace{(l_0, \ldots, l_{max})}_{q+1}$$

Планирование стратегии обучения заключается в предсказании d будущих состояний студента для некоторой начальной кривой L_i, чтобы назначать ему задачи, тренирующие соответствующие навыки, и закреплять освоенные темы.

Зафиксируем целевую функцию для оптимизации - по результатам решения планируемой задачи студент должен наиболее далеко продвинуться к цели l_{max}. Введем для задачи следующие параметры:

- c - сложность
- r - степень освоения тем, необходимая для решения
- a - максимальная степень освоения тем, возникающая в результате решения

Каждой задаче $p \in P$ поставим в соответствие тройку (c, r, a)

$$e\colon P \to \mathbb{N} \times \bar{S}^m \times \bar{S}^m = P^e.$$

Определим отображение:

$$\varphi\colon \bar{S}^m \times P^e \to \bar{S}^m$$

$$\varphi(l, e(p)) = \frac{1}{c} \cdot (r - l) + \|(r - l)\| \cdot a$$

Заметим, что максимум выбранной функции существует при любом нормировании образа.

Задача 2. *Построить такое продолжение L_{i+d} заданной последовательности L_i, что:*

$$\varphi(l_i, e(p)) \to \max_{p \in P},$$

где образ отображения нормирован некоторой метрикой.

Выбор метрики определяет способ выбора оптимальной задачи. В случае, например, манхэттенской метрики $\| \ \|_1$, оптимальной задачей является та, которая закрепляет много различных еще не освоенных тем.

Стоит отметить, что предположение о том, что студент решит назначенную задачу - вероятностное, а значит последующие предположения должны, вообще говоря, опираться на условную вероятность предшествующих событий.

3. Применение методов искусственного интеллекта для планирования персональной стратегии обучения

3.1. Построение дискретного пространства признаков

В рамках данного курса разработано пространство признаков с общими и тематическими навыками.

Общие навыки:

Свойство	*Интерпретация*
Компилируется	$\{0, 1\}$
Протестировано	$\{0, \ldots, 100\}$
Уникально	$\{0, \ldots, 100\}$
Сдано вовремя	$\{0, \ldots, 100\}$
Соответствует ограничениям	$\{0, 1\}$
Оформлено в едином стиле	$\{0, 1\}$

Оценка признаков уникальности, соответствия ограничениям и оформления в едином стиле представляют собой отдельные нетривиальные задачи.

Тематические навыки:

Свойство	*Интерпретация*
Битовые операции	$\{0, 1\}$
Сортировки	$\{0, 1\}$
Алгебра логики	$\{0, 1\}$
k-значная логика	$\{0, 1\}$
Структуры и классы	$\{0, 1\}$

Оценка тематических свойств в некоторых случаях также представляет собой непростую задачу. Для некоторых свойств были созданы модули автоматического их обнаружения в решениях.

3.2. Планирование персональной стратегии обучения

В задачах продолжения последовательностей хорошо себя зарекомендовали рекуррентные нейронные сети, а именно: LSTM (Long-Short Term Memory) [5], CW-RNN (Clockwork Recurrent Neural Network) [6], GRU (Gated Recurrent Unit) [7] и др. Данные сети были использованы в экспериментах с искусственными данными процессов обучения 4 групп студентов по 50 человек на курсе «Практикум на ЭВМ» Ташкентского филиала МГУ им. М. В. Ломоносова.

Каждому студенту были назначены 3 случайных задачи разной сложности, остальные задачи назначались на основании принадлежности студента к одному из выделенных классов:

- отличники - решают задачи любой сложности
- хорошисты - в сложных задачах иногда испытывают трудности
- троечники - затрудняются решить задачи любой сложности в некоторых разделах
- двоечники - с трудом решают самые простые задачи

В процессе имитации обучения некоторые студенты переходили из одного класса в другой в результате ускоренного роста генерируемых результатов начиная с некоторого момента. Для такого класса студентов

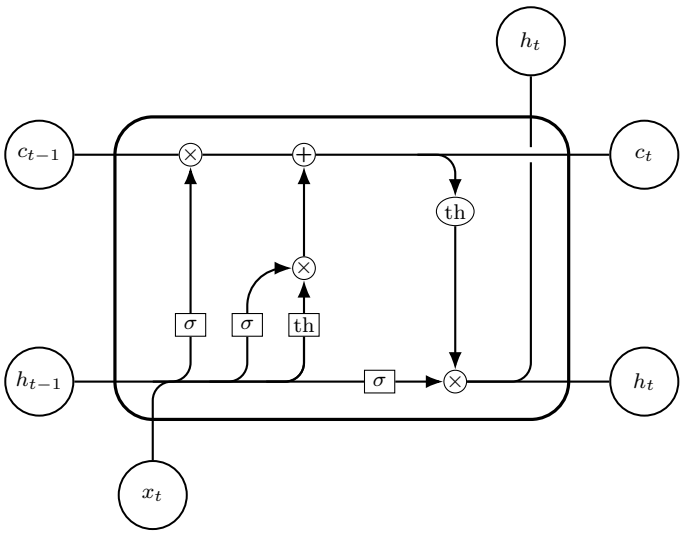


Рис. 1. Ячейка LSTM

необходимо учитывать *давние* результаты и не повышать сложность назначаемых задач слишком быстро, давая им возможность закрепить результат.

Используя построенную в итоге историю обучения применим нейронные сети для предсказания успеха в решении назначенных задач начиная с 4-ой. Опустим вероятностную природу процесса планирования и будем предполагать, что каждое новое предположение производится на основании уже произошедших событий. Для каждого метода измерим точность предсказания, как отношение верных предсказаний к общему их количеству.

Функционально ячейка LSTM задается следующими уравнениями:

$$h_t = o_t \cdot \mathrm{th}(\bar{c}_t)$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot \bar{c}_t$$

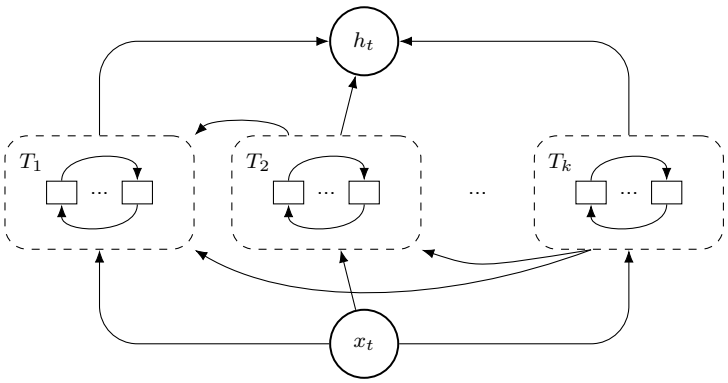


Рис. 2. Архитектура CW-RNN: нейроны скрытого слоя разделены на k групп. Нейроны i-ой группы подключены друг к другу и имеют период T_i. Группа i подключена к группе j, если $T_i < T_j$.

где

$$\begin{aligned}
\bar{c}_t &= \mathrm{th}(W_c \cdot [h_{t-1}, x_t] + b_c) \\
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)
\end{aligned}$$

Уравнения ячейки GRU принимают следующий вид:

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \hat{h}_t$$

где

$$\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t]) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t]) \\
\hat{h}_t &= \mathrm{th}(W \cdot [r_t \cdot h_{t-1}, x_t])
\end{aligned}$$

Результат применения нейронных сетей представлен в таблице 1.

Для конечномерного дискретного пространства признаков можно применить алгоритм предсказания следующего элемента в последовательности, основанный на модели *компактное дерево предсказаний*

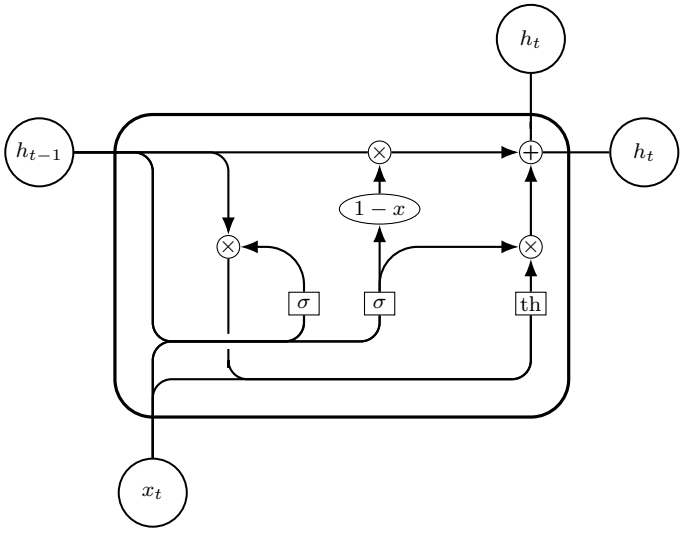


Рис. 3. Ячейка GRU

Нейронная сеть	*Искусственные данные*	*Реальные данные*
LSTM	84.12	56.87
CW-RNN	85.77	54.90
GRU	89.43	58.01

Таблица 1. Точность предсказания, %

(Compact Prediction Tree, CPT, [8]). CPT состоит из трех структур данных: *дерева предсказаний* (PT), *обратного индекса* (II), *массива указателей* (LT).

PT. Пустое дерево предсказаний состоит из одного узла - корня. В процессе обучения модели дерево заполняется по следующему алгоритму: для очередной последовательности проверить наличие ее первого элемента среди потомков корня; если таковой потомок имеется, то перейти к поиску следующего элемента последовательности в потомке; иначе, вставить оставшиеся элементы последовательности ветвью, начиная с последнего узла, для которого нашлись совпадения или корня, если совпадений не было вообще.

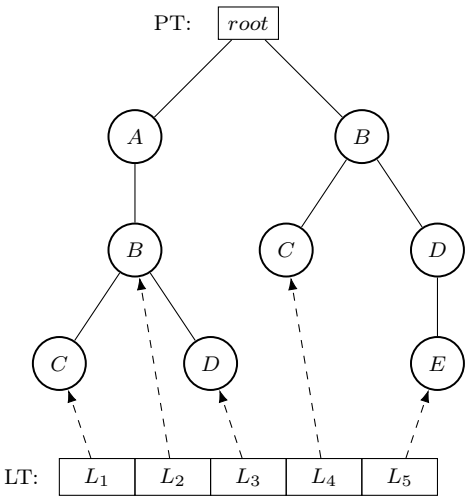
II. Обратный индекс является двумерной двоичной матрицей, строки которой соответствуют уникальным элементам последовательностей, содержащихся в PT. Если элемент x содержится в последовательности y, то в соответствующей ячейке матрицы $\text{II}_{x,y}$ находится 1, иначе - 0.

LT. Массив указателей содержит указатели на последние элементы последовательностей в PT доступные по номерам последовательностей.

Алгоритм: Заполнение CPT

дано: CPT, кривая обучения L
текущий узел $\leftarrow$ корень PT
цикл $l \in L$
  если множество потомков текущего узла $\neq \varnothing$ и содержит l
    текущий узел $\leftarrow$ потомок, равный l
  иначе
    добавить l в множество потомков текущего узла
    текущий узел $\leftarrow$ новый потомок, равный l
  конец если
конец цикла
обновить II и LT

Для заданной последовательности $X = (x_1, x_2, \ldots, x_n)$ процесс предсказания следующего элемента заключается в следующем: для некоторого m, $1 < m < n$, выберем все последовательности, содержащие последние m элементов $\{x_{n-m+1}, x_{n-m+2}, \ldots, x_n\} \subset X$ в любом порядке и в любой позиции. Такие схожие с X последовательности, обозначим их $\mathbb{Y}$, находятся битовым перемножением всех строк II, соответствующих выбранным элементам. Для каждой последовательности $Y \in \mathbb{Y}$ выделим подпоследовательность, начинающуюся с элемента, следующего за



II:

	L_1	L_2	L_3	L_4	L_5
A	1	1	1	0	0
B	1	1	1	1	1
C	1	0	0	1	0
D	0	0	1	0	1
E	0	0	0	0	1

Рис. 4. Пример структуры CPT

Модель	*Искусственные данные*	*Реальные данные*
CPT	94.36	85.55

Таблица 2. Точность предсказания, %

последним элементом схожим с X. Каждому элементу y таких подпоследовательностей поставим в соответствие два числа:

- s - число вхождений элемента y в последовательностях $\mathbb{Y}$
- $c = \frac{s}{r}$, где r - число последовательностей в PT, содержащих y (т.е. сумма строки матрицы II, соответствующей элементу y)

Элементы, соответствующие минимальному s - кандидаты. Искомым элементом является кандидат с наименьшим c.

Результат применения CPT с использованием приведенного выше пространства признаков представлен в таблице 2.

4. Заключение

Описанные выше методы анализа данных применяются в системе поддержки преподавания курса «Практикум на ЭВМ» в Ташкентском филиале МГУ им. М. В. Ломоносова. Курс ставит перед студентами цель

получить навыки разработки на языке C/C++ самодостаточных программных продуктов, решающих поставленные преподавателем задачи. Посредством лекций, семинаров и практических занятий материал курса должен быть постепенно усвоен студентами. Этот курс обширный и покрывает 4 года обучения:

	1 семестр	2 семестр
1 год	Простые алгоритмы	Дискретная математика
2 год	Операционные системы	Основы ООП
3 год	Дискретная оптимизация	Численные методы, часть 1
4 год	Численные методы, часть 2	-

Каждый раздел содержит набор практических задач разной сложности и тематики, а практическим занятиям предшествуют соответствующие лекции или курсы. Согласно плану обучения, студент должен решить некоторое случайное подмножество задач из раздела для перехода к следующему. В противном случае преподаватель может назначить студенту дополнительные задания, обычно заключающиеся в решении нового подмножества задач. В построении этого подмножества преподаватели опираются на рекомендации представленных в статье методов.

Из проведенных экспериментов следует, что применение CPT для решения задачи планирования персональной стратегии обучения выдает неплохой результат, когда подход с использованием нейронных сетей указывает на необходимость уточнения пространства признаков или корректировки архитектуры применяемых нейронных сетей. В этом направлении будут проводиться дальнейшие исследования.

Список литературы

[1] Алисейчик П. А., Вашик К., Ж. Кнап, Кудрявцев В. Б., Шеховцов С. Г., Строгалов А. С., "Моделирование процесса обучения", *Интеллектуальные системы*, **10** (2006), 85.

[2] Алисейчик П. А., Вашик К., Кудрявцев В. Б., Строгалов А. С., "Компьютерные обучающие системы", *Интеллектуальные системы*, **8** (2004), 20.

[3] Алисейчик П. А., Кудрявцев В. Б., Строгалов А. С., "Автоматная модель обучения", *Материалы VIII Международного семинара «Дискретная математика и и ее приложения», изд мехмат ф-т МГУ*, 2004, 7.

[4] Аширматов Б. Д., "Многомерный поиск в обучающих системах", *Интеллектуальные системы*, **15** (2011), 14.

[5] Sepp Hochreiter, Jürgen Schmidhuber, "Long Short-Term Memory", *Neural Computation*, 1997, №9(8), 1735-1780.

[6] Jan Koutník, Klaus Greff, Faustino Gomez, Jürgen Schmidhuber, “A Clockwork RNN”, 2014, arXiv: https://arxiv.org/abs/1402.3511.

[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, 2014, arXiv: https://arxiv.org/abs/1406.1078.

[8] Ted Gueniche, Philippe Fournier-Viger, Vincent S. Tseng., “Compact Prediction Tree: A Lossless Model for Accurate Sequence Prediction”, Advanced Data Mining and Applications: 9th International Conference, 2013, № II.

[9] Ахо А., Ульман Дж., *Теория синтаксического анализа, перевода и компиляции*, **1**, Мир, 1978.

[10] Котов В. Е., Сабельфельд В. К., *Теория схем программ*, Наука, 1991.

[11] Камкин А. С., *Введение в формальные методы верификации программ*, Москва, 2018.

Using AI approach to plan personal learning strategy

Bektashev R.A.

Abstract

In this article the problem of planning personal learning strategy is set and results of applying various approaches to its solution are presented.

Keywords: computer education system, sequence elements prediction, neural network, recurrent neural network, compact prediction tree

Технология дистилляции знаний для обучения нейронных сетей на примере задачи бинарной классификации

Бирюкова В.А.[1]

С использованием технологии обучения нейронных сетей, дистилляции знаний, были получены модели, решающие задачу бинарной классификации с производительностью, превышающей производительность сети-учителя примерно в 5 раз при несущественном падении качества. Сверточная нейронная сеть ResNet-18 была обучена двумя способами по данной технологии (с помощью предобученной сети ResNet-50) и классическим методом. Введено понятие степени неуверенности модели на множестве объектов как величины отклонения предсказаний нейронной сети от принимаемых за ответ значений. Были также проведены эксперименты по рекурсивному применению технологии дистилляции знаний.

Ключевые слова: дистилляция знаний, бинарная классификация, остаточная нейронная сеть, сверточная нейронная сеть, степень неуверенности модели на множестве объектов, рекурсивное обучение нейронных сетей.

1. Введение

В настоящее время благодаря появлению технологии сверточных нейронных сетей и высокопроизводительных средств бурно развивается направление построения искусственных систем для распознавания визуальных образов [1],[2],[3],[4]. При этом трудоемким является процесс подготовки обучающей базы, как правило, требующей большого количества вручную размеченных изображений, что сильно увеличивает стоимость

[1] *Бирюкова Вероника Андреевна* — студент каф. высшей математики института кибернетики РТУ МИРЭА, e-mail: biryukovaveronika@mail.ru.

Biryukova Veronika Andreevna — student, MIREA - Russian Technological University, Institute of Cybernetics, Chair of Higher Mathematics.

разработок. Также возникает необходимость размещения на портативных устройствах небольших нейронных сетей, способных выдавать достаточно точные результаты [5]. Поэтому для решения ряда задач может быть использована технология обучения относительно простых сетей на базе изображений без ручной разметки, но с использованием предварительно обученной более мощной модели. Для этого разработана методика обучения нейронных сетей – дистилляция знаний (Knowledge Distillation) [6],[7]: пусть имеется мощная модель (или ансамбль моделей), которая обучена улавливать сложные признаки (features) и выдавать достаточно точные предсказания. Такую нейронную сеть называют сетью-учителем. С её помощью обучается более простая модель (сеть-студент), которую можно обучить так, что она будет давать результаты, сопоставимые с результатами сети-учителя.

Целью данной работы является исследование применения технологии Knowledge Distillation в рамках задачи бинарной классификации. В этой работе сверточная нейронная сеть ResNet-18 («остаточная» нейронная сеть [8]) была обучена по трем сценариям. Сеть, обученную по первому сценарию, то есть она обучалась на базе данных с ручной разметкой, в дальнейшем будем называть классическим студентом. Второй и третий сценарии предполагают обучение по меткам, которые являются предсказаниями сети, полученной из предобученной сети ResNet-50 [8], причем в разных сценариях эти метки использовались в различных формах представления. Далее эти модели будем называть soft-студент и hard-студент. ResNet-50 была взята из нейросетевой библиотеки Keras. Полученные результаты были изучены, а также были проведены эксперименты с рекурсивным применением исследуемой технологии. Эксперименты проводились на базе изображений кошек и собак Kaggle «Dogs vs. Cats» [9].

2. Основные понятия

Машинное обучение (англ. machine learning, ML) — класс методов искусственного интеллекта, целью которых является создание математических моделей, способных выполнять конкретные задачи без использования явных инструкций, а основываясь на выявленных в процессе обучения эмпирических закономерностях в данных, на которых она обучалась.

Искусственная нейронная сеть (нейронная сеть) или нейросетевая модель – это математическая модель, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. У модели данного типа есть тренируемые параметры, которые меняются во время процедуры обучения, и гиперпараметры, параметры модели, требующие ручной настройки и задаваемые до процесса обучения. Тренируемые параметры, далее именуемые параметрами, – это веса и смещения (weights and biases) [10]. Гиперпараметрами являются, например, архитектура сети (строение модели), функция потерь, оптимизатор. Общие понятия, связанные с нейронными сетями, можно посмотреть в работе [10]. В этом исследовании рассматриваются сверточные нейронные сети специального вида, называемые «остаточными» нейронными сетями, ResNet [8]. Эти сети используют дополнительные соединения между слоями для нивелирования эффекта «затухающего градиента» [11]. С помощью нейронных сетей можно решать различные задачи машинного обучения, например, задачу классификации.

Постановка задачи классификации. Дана обучающая база – множество объектов (изображений, аудиофайлов и т.д.). Каждому элементу этого множества соответствует некоторый класс (возможны варианты задачи, где один объект сопоставляется нескольким классам). Требуется, чтобы модель после процесса обучения на этой базе могла правильно предсказывать принадлежность объектов, которые не использовались во время обучения, к одному из заданных классов.

Если работа ведется с изображениями, то эта задача будет относиться к области компьютерного зрения «Computer Vision» [12]. Компьютерное зрение – это теория и соответствующая реализация интеллектуальных систем [13], способных извлекать полезную информацию из изображений.

Одним из способов обучения моделей и, в частности нейронных сетей, является метод «обучения с учителем» [10]. Для обучения модели используются множество объектов и множество меток. Обучающим множеством называется совокупность пар «объект, метка», между составляющими которых существует некоторая зависимость. На основе этих данных требуется восстановить эту зависимость, то есть построить модель, способную для любого возможного корректного входного изображения выдать достаточно точный ответ (предсказание). Важно, чтобы обучаемая модель была способна к обобщению, то есть к адекватному отклику на данные, выходящие за пределы обучающего множества. Если же мо-

дель научилась достаточно точно отвечать на объекты множества, на котором она обучалась, а на новые изображения – нет, то такой эффект называется «переобучением» модели [10].

Для обучения модели обучающее множество делится на тренировочное, проверочное и тестовое множества. В ходе обучения нейронной сети объекты из тренировочного множества поступают на вход модели, а на выходе получаются соответствующие значения, которые сравниваются с метками, выступающими в качестве «правильных» ответов, посредством чего вычисляется величина ошибки нейронной сети с помощью заранее выбранной функции потерь (Loss function). После этого осуществляется процесс обратного распространения ошибки [10], благодаря чему получают значения для корректировки параметров сети. Как именно корректируются параметры, зависит от важного механизма – оптимизатора. После корректировки параметров процедура обучения повторяется, пока не выполнится заданный критерий остановки.

Как правило, обучение проходит по «эпохам». Эпохой называется часть обучающего процесса, во время которого используется всё тренировочное множество. По итогам эпохи обычно оценивают качество работы сети на проверочном множестве [14]. После этого начинается новая эпоха, и тренировочное множество изображений снова обрабатывается моделью. Как правило, в течение одной эпохи тренировочные объекты подаются не по одному и не все сразу, а порциями. То есть на вход сети поступает какая-то выборка (mini-batch) из тренировочного множества, она целиком проходит через сеть, и только после этого начинается процедура корректировки параметров. Размер выборки определяется заранее, таким образом это гиперпараметр сети.

Обычно для улучшения результатов обучения, перед тем как поступить на вход модели, изображения проходят процедуру предобработки, например: значения пикселей нормируются, или картинки приводятся к соответствующему разрешению. Эти преобразования применяются к каждому изображению. Также часто в процессе обучения модели используется аугментация данных – искусственное увеличение обучающего множества за счет преобразования имеющихся изображений, при котором полученное изображение продолжает принадлежать классу исходного объекта [15]. В этом случае при каждом вызове изображение меняется случайным образом в рамках заданных преобразований. Аугментация данных позволяет минимизировать возможность переобучения и способствует обучению нейронной сети обобщать образы.

За результат обучения принимается нейронная сеть с теми значениями параметров, при которых модель показала наилучшее качество работы на проверочном множестве.

После процесса обучения оценивается качество работы сети на тестовом множестве, то есть на тех изображениях, которые модель не обрабатывала в процессе обучения. Тем самым определяется итоговая оценка результативности модели.

Для задачи бинарной классификации в данной работе используется бинарная кросс-энтропия (Binary Cross-Entropy) [16]. Эта функция выводится с помощью метода максимального правдоподобия. Формула данного выражения для одного объекта:

$$CE(p, y) = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)), \tag{1}$$

где y и p – метка и предсказание сети на этом объекте соответственно.

Для выборки из n объектов:

$$Loss(\bar{p}, \bar{y}) = \frac{1}{n} \sum_{i=1}^{n} CE(p_i, y_i), \tag{2}$$

где $\bar{p}$ – вектор предсказаний нейронной сети на n объектах с компонентами p_i (предсказание модели на i-ом изображении); $\bar{y}$ – вектор меток этих n объектов с компонентами y_i (метка i-ого изображения).

В данном исследовании в качестве оптимизатора применяется алгоритм Adam (adaptive moment estimation) [17], сочетающий в себе следующие две идеи. Во-первых, идею из численных методов о накоплении градиента, «если некоторое время двигаться в определённом направлении, то следует туда двигаться некоторое время и в будущем». Во-вторых, идею более слабого обновления параметров как реакции на типичные признаки для какого-нибудь из заданных классов, выявленные на объекте из обрабатываемой выборки.

Параметры сети корректируются на каждой t-ой выборке, номер этой выборки меняется в следующих пределах:

$$t \in \{1, \ldots, P \cdot S\}, \tag{3}$$

где P – количество эпох, S – количество выборок, обрабатываемых сетью за одну эпоху.

Этот оптимизатор использует понятия экспоненциально взвешенного скользящего среднего для градиента функции потерь на t-ой выборке m_t:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \ , \tag{4}$$

где g_t – градиент функции потерь, β_1 – гиперпараметр сети, обычно полагается $\beta_1 = 0.9$;

и оценки средней нецентрированной дисперсии на t-ой выборке υ_t:

$$\upsilon_t = \beta_2 \upsilon_{t-1} + (1 - \beta_2) g_t^2 \ , \tag{5}$$

где g_t – градиент функции потерь, β_2 – гиперпараметр сети, обычно полагается $\beta_2 = 0.999$.

Как правило, напрямую оценку дисперсии и скользящее среднее не используют. Их искусственно увеличивают на первых выборках. Общие формулы для рассматриваемых величин следующие:

$$\widehat{m_t} = \begin{cases} \frac{m_t}{1-\beta_1^t}, & \text{если } 0 < t < k_1 \\ m_t \quad , & \text{если } t \geq k_1 \end{cases} \ , \tag{6}$$

$$\widehat{\upsilon_t} = \begin{cases} \frac{\upsilon_t}{1-\beta_2^t}, & \text{если } 0 < t < k_2 \\ \upsilon_t \quad , & \text{если } t \geq k_2 \end{cases} \ , \tag{7}$$

где k_1 и k_2 часто полагаются 10 и 1000 соответственно.

Скорость обучения – это параметр обучения нейронных сетей, позволяющий управлять величиной коррекции весов на каждой выборке.

В алгоритме Adam этот параметр меняется по следующей формуле:

$$\eta_t = \eta_0 \cdot \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \ , \tag{8}$$

где η_0 – начальное значение скорости обучения.

В библиотеке Keras реализован механизм дополнительного изменения скорости обучения в этом оптимизаторе с помощью задаваемого до обучения параметра d:

$$\eta_t = \eta_0 \cdot \frac{1}{1 + d \cdot (t-1)} \cdot \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \ . \tag{9}$$

Формула обновления параметра сети на t-ой выборке θ_t:

$$\theta_t = \theta_{t-1} - \frac{\eta_t}{\sqrt{\widehat{\upsilon_t}} + \epsilon} \, \widehat{m_t} \ , \tag{10}$$

где θ_{t-1} – значение этого параметра после обновления на $(t-1)$-ой выборке, ϵ – сглаживающий параметр (гиперпараметр сети), который в данной работе полагается равным $\epsilon = 10^{-7}$. Этот гиперпараметр необходим, чтобы избежать деления на 0.

Для оценки качества работы нейронной сети используется оценка точности (accuracy), которую максимизируют в процессе обучения. Для задачи бинарной классификации на одном объекте эта оценка вычисляется по следующей формуле:

$$acc(p, y) = \begin{cases} 1, \text{ если } (p > th \text{ и } y > th) \text{ или } (p \leq th \text{ и } y \leq th) \\ 0, \text{ если } (p > th \text{ и } y \leq th) \text{ или } (p \leq th \text{ и } y > th) \end{cases} \ , \tag{11}$$

где th – пороговое значение, которое в данной работе равно 0.5; метка объекта y и предсказание модели на этом объекте p являются действительными числами из отрезка $[0, 1]$.

Для выборки из n объектов (оценивается следующим образом):

$$acc(\bar{p}, \bar{y}) = \frac{1}{n} \sum_{i=1}^{n} acc(p_i, y_i) \ , \tag{12}$$

где $\bar{p}$ – вектор предсказаний нейронной сети на n объектах, с компонентами p_i (предсказание модели на i-ом изображении); $\bar{y}$ – вектор меток этих n объектов, с компонентами y_i (метка i-ого изображения) .

3. Обучение нейронных сетей

База данных, использованная для задачи бинарной классификации, представляет собой 25 000 изображений кошек и собак (12 500 для каждого класса) с сайта Kaggle [9]. Во всех экспериментах полагается, что, если сеть выдает значение строго больше 0.5, то она классифицирует этот объект как «изображение с собакой». В противном случае, если выходное значение модели меньше или равно 0.5, то нейронная сеть предсказывает принадлежность данного изображения к классу «изображение с кошкой». Всё множество изображений было поделено на тренировочное (70%), проверочное (15%) и тестовое (15%) множества. Таким образом, количество тренировочных объектов составило 17500 изображений. Проверочных – 3750. Тестовых – 3750. При работе со всеми моделями изображения загружались в формате «RGB» в виде четырехмерного массива (размер выборки × высота × ширина × количество цветовых каналов). Поскольку использовался формат «RGB», то количество цветовых каналов равнялось трем. В каждой ячейке этого массива указывалось значение одного из цветовых каналов соответствующего пикселя изображения, равное целому числу из диапазона $[0, 255]$. Для каждой модели использовалась одинаковая предобработка данных:

1) Исходное изображение преобразовывалось к разрешению 224×224 пикселей с помощью интерполяционного метода ближайшего соседа.

2) Производилась нормировка значений массива, то есть каждый его элемент был поделен на 255, после всех преобразований предобработки и аугментации.

При обучении hard- и soft-студентов в качестве меток i-ого изображения y_i выступили значения, сформированные по значению предсказания сети-учителя на этом объекте p_i. Для hard-студента:

$$y_i = \begin{cases} 1, \text{ если } p_i > 0.5 \\ 0, \text{ если } p_i \leq 0.5 \end{cases}, \tag{13}$$

а для soft-студента:

$$y_i = p_i \,. \tag{14}$$

Все модели оценивались на тестовом множестве с ручной разметкой с помощью двух генераторов изображений: один генератор подавал на вход нейронной сети картинки, которые прошли только процедуру предобработки, а другой генератор для оценки качества работы нейронной сети изменял картинки, как и генераторы, использовавшиеся при обучении этой модели, то есть с применением преобразований для аугментации данных. Для сетей, которые обучались не на базе с ручной разметкой, проводилась также оценка на тестовом множестве с разметкой сети-учителя – как с аугментацией, так и без неё.

Генераторы, использовавшие помимо предобработки преобразования для аугментации данных, меняли исходные изображения случайным образом, поэтому оценивание проводилось несколько раз. В таблицах с результатами представлены наилучшие значения для функции потерь согласно формуле (2) и для оценки качества работы сети в соответствии с формулой (12).

3.1. Обучение сети-учителя

В роли сети-учителя выступала нейронная сеть ResNet-50, преобразованная в соответствии с поставленной задачей: использовалась только часть сети, отвечающая за выделение признаков на изображении. К ней был добавлен слой Dropout [10] с долей «забывания» равной 0.5 и полносвязный слой [10] с одним нейроном. К выходу этого слоя применили сигмоидальную функцию активации [10]. Схема сети-учителя изображена на рисунке 1:

В качестве оптимизатора сети использовался метод Adam со скоростью обучения 10^{-6} и значением параметра d из формулы (9) равным 0.01. Параметры β_1 и β_2 из формул (4)–(9) равнялись соответственно 0.9 и 0.999. Функция потерь – бинарная кросс-энтропия. Для оценки качества работы нейронной сети вычислялась точность модели в соответствии с формулой (12). При обучении этой нейронной сети для аугментации данных использовалась симметрия относительно вертикальной оси, проходящей через центр изображения. Обучение модели проходило в течение 25 эпох с выборкой размера 32. Динамика значений функции потерь и точности представлена на рисунке 2:

Результаты обучения сети-учителя:

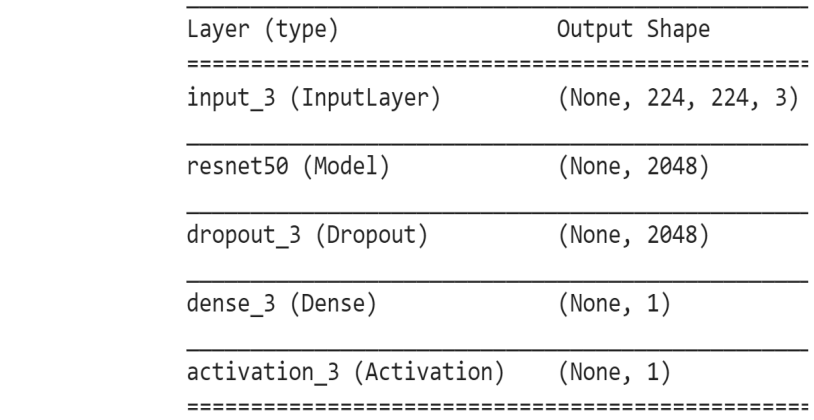


Рис. 1. Схема сети-учителя, полученной из ResNet-50

Множество \ Функция		Функция потерь	Точность
Тренировочное множество (лучшее значение в процессе обучения)		0.00079	0.99983
Проверочное множество (лучшее значение в процессе обучения)		0.09638	0.97526
Тестовое множество с ручной разметкой	Без аугментации	0.10340	0.97333
	С аугментацией	0.10310	0.97547

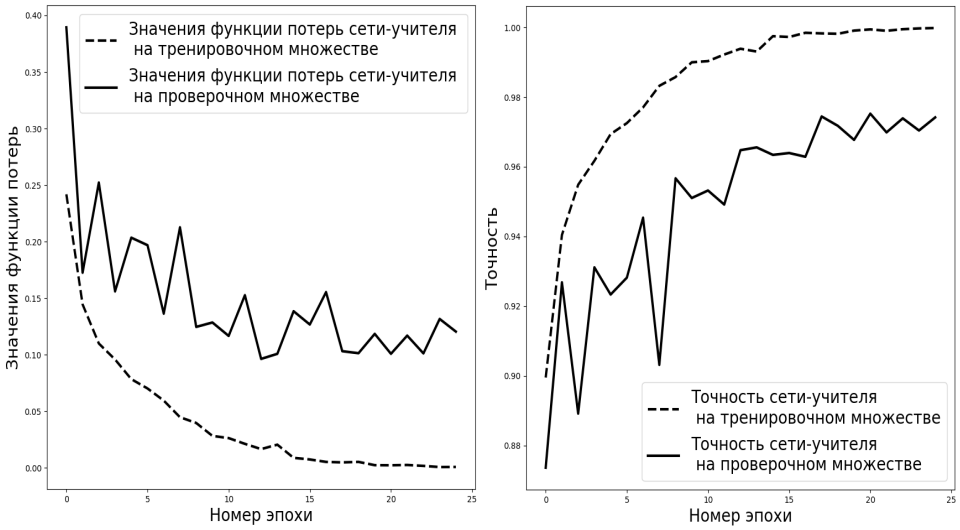


Рис. 2. Динамика значений функции потерь и точности сети-учителя

3.2. Обучение студентов

Модель ResNet-18 была реализована в библиотеке Keras с архитектурой из работы [8]. После слоя Average Pooling [10] его выход (массив 1x1x512) был преобразован к вектору размерности 512. К нему был присоединен полносвязный слой с одним нейроном. К выходу этого слоя применялась сигмоидальная функция активации.

Данная модель была обучена трижды: классическим способом, и два раза при обучении метки формировались по предсказаниям сети-учителя. В первом случае использовалась формула (13), а во втором – формула (14). Во всех трех процедурах использовались одинаковые гиперпараметры и аугментация данных. Модели обучались в течение 100 эпох с выборкой размера 32. В роли оптимизатора выступал алгоритм Adam со скоростью обучения 0.0001 и нулевым значением параметра d из формулы (9). Параметры β_1 и β_2 из формул (4)–(9) полагались соответственно 0.9 и 0.999. Функция потерь – бинарная кросс-энтропия. Для оценки качества работы нейронной сети вычислялась точность модели в соответствии с формулой (12).

Применялась аугментация данных 2 видов: симметрия относительно вертикальной оси, проходящей через центр изображения, и сдвиги в

горизонтальном и вертикальном направлениях. Для сдвигов устанавливалась максимальная доля от общей ширины или высоты изображения, на которую могла сдвигаться картинка в соответствующем направлении. В процессе обучения величины сдвигов для каждого входного изображения выбирались случайно. В экспериментах использовалось значение 0.4. Появившиеся при сдвигах «пустоты» заполнялись с помощью интерполяционного метода ближайшего соседа.

3.2.1. Классический студент

При обучении классического студента в качестве меток использовались значения 0 или 1, полученные при ручной разметке базы данных. Процесс изменения значений функции потерь и точности во время обучения модели можно увидеть на рисунке 3:

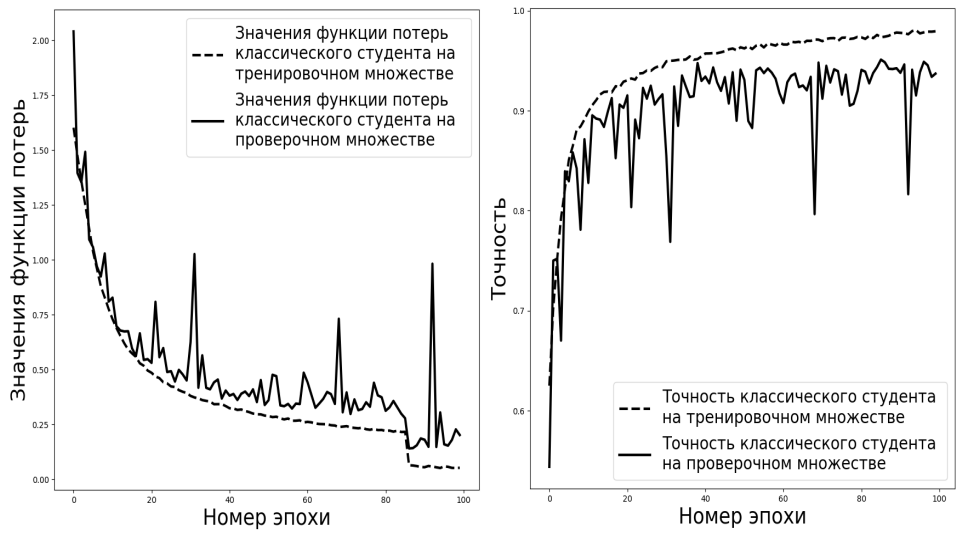


Рис. 3. Динамика значений функции потерь и точности классического студента

Результаты обучения классического студента:

Множество \ Функция		Функция потерь	Точность
Тренировочное множество (лучшее значение в процессе обучения)		0.05183	0.98048
Проверочное множество (лучшее значение в процессе обучения)		0.14111	0.95105
Тестовое множество с ручной разметкой	Без аугментации	0.14394	0.96160
	С аугментацией	0.30252	0.94987

3.2.2. Hard-студент

При обучении данным способом в качестве меток использовались предсказания сети-учителя, распределенные на отрезке [0,1], преобразованные в соответствии с формулой (13). Статистика обучения hard-студента представлена на рисунке 4:

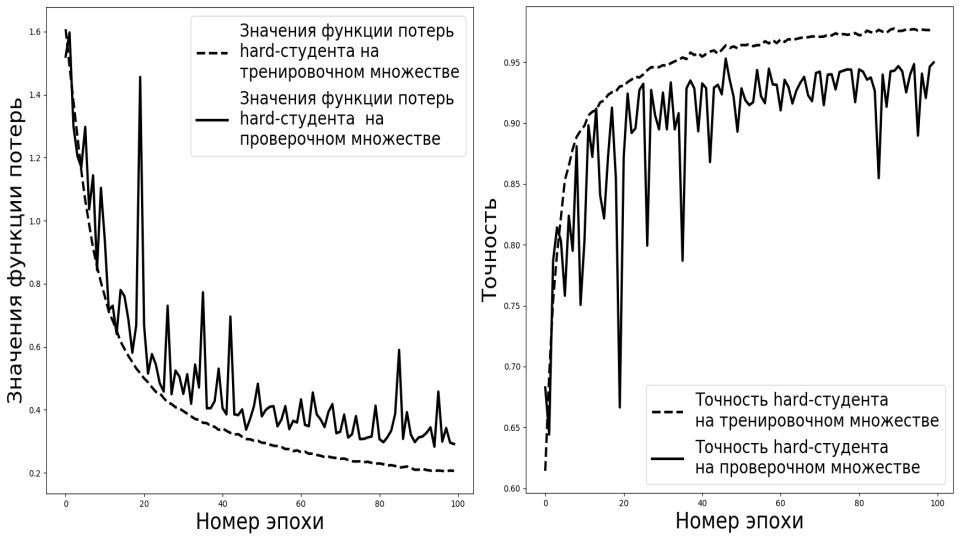


Рис. 4. Динамика значений функции потерь и точности hard-студента

Результаты обучения hard-студента:

Множество \ Функция		Функция потерь	Точность
Тренировочное множество (лучшее значение в процессе обучения)		0.20442	0.97762
Проверочное множество (лучшее значение в процессе обучения)		0.28316	0.95293
Тестовое множество с ручной разметкой	Без аугментации	0.13251	0.95440
	С аугментацией	0.36285	0.94560
Тестовое множество с предсказаниями сети-учителя в качестве меток	Без аугментации	0.12143	0.95493
	С аугментацией	0.35090	0.94667

3.2.3. Soft-студент

В процессе обучения этого студента в роли меток выступали непосредственно предсказания сети-учителя. Таким образом, при обучении модель аппроксимировала не значения $\{0, 1\}$, а действительные числа из отрезка [0,1]. Динамика обучения этой модели показана на рисунке 5:

Результаты обучения soft-студента:

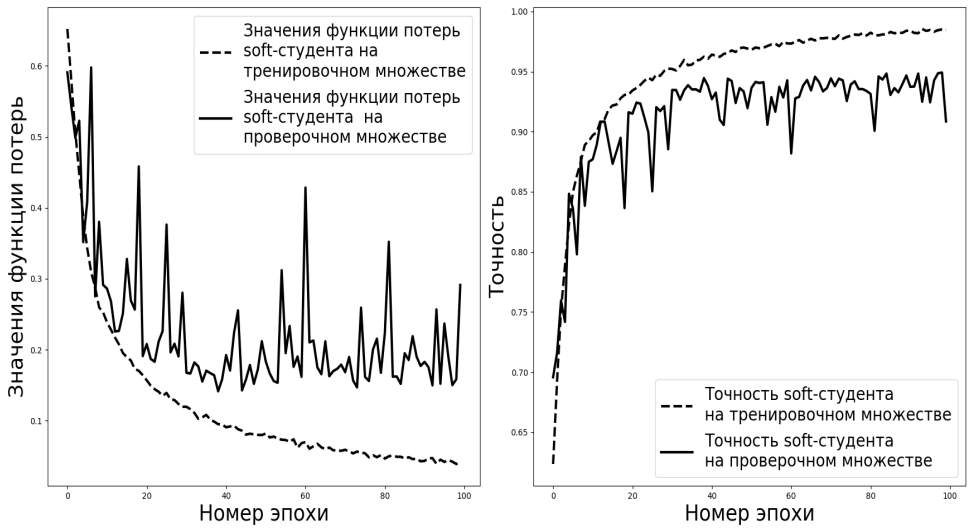


Рис. 5. Динамика значений функции потерь и точности soft-студента

Множество \ Функция		Функция потерь	Точность
Тренировочное множество (лучшее значение в процессе обучения)		0.03890	0.98546
Проверочное множество (лучшее значение в процессе обучения)		0.14147	0.94917
Тестовое множество с ручной разметкой	Без аугментации	0.12629	0.96320
	С аугментацией	0.14906	0.95040
Тестовое множество с предсказаниями сети-учителя в качестве меток	Без аугментации	0.13455	0.96053
	С аугментацией	0.15217	0.95120

4. Исследование обученных нейронных сетей

4.1. Производительность

Среднее время работы нейронных сетей оценивалось на тестовом множестве, состоящем из 3750 изображений. Средний размер изображения составил 0.152 мегапикселя.

Оценка производительности проводилась на устройстве без графического ускорителя с процессором Intel Core i7-7500U, тактовой частотой 2.7 ГГц, 2 ядрами и 16 ГБ оперативной памяти. На этом устройстве были получены следующие результаты:

- Среднее время обработки одного изображения нейронной сетью ResNet-18 – 0.07168 секунды.
- Среднее время обработки одного изображения нейронной сетью ResNet-50 – 0.35452 секунды.
- Общее время обработки тестового множества изображений нейронной сетью ResNet-18 – 268.80433 секунд.
- Общее время обработки тестового множества изображений нейронной сетью ResNet-50 – 1329.45384 секунд.

Таким образом, сеть ResNet-18 в среднем выполняет обработку одного изображения на 0.28284 секунды быстрее модели ResNet-50. А на всём тестовом множестве преимущество составляет 1060.64951 секунд. Получается, что сеть ResNet-18 в среднем работает практически в 5 раз быстрее.

4.2. Статистика распределений предсказаний

Рассмотрим статистику предсказаний всех четырех обученных нейронных сетей на тренировочном, проверочном и тестовом множествах. Сначала построим распределение относительных частот предсказаний моделей, значения которых принадлежат отрезку [0,1] с шагом $h = 0.05$.

1. Тестовое множество. В нем 3750 изображений, по 1875 изображений кошек и собак. Значения относительных частот предсказаний каждой сети приведены в следующих таблицах.

Учитель:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.48027	0.00427	0.0048	0.00293	0.0008
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00053	0.00187	0.00133	0.00133	0.0008
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00107	0.00053	0.0008	0.00107	0.00107
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00187	0.00213	0.0032	0.00347	0.48587

Классический студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.4512	0.00987	0.00853	0.00347	0.00267
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00347	0.00267	0.004	0.00187	0.00267
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00293	0.00187	0.00293	0.0032	0.00133
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00213	0.00347	0.0032	0.00773	0.4808

Hard-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.44587	0.01467	0.0072	0.00507	0.00667
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00373	0.00533	0.00533	0.00427	0.00373
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00293	0.00453	0.0024	0.00533	0.004
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00373	0.00347	0.01013	0.01307	0.44853

Soft-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.476	0.00827	0.00293	0.00347	0.00267
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00267	0.00293	0.00293	0.0032	0.0008
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00267	0.00187	0.0024	0.0008	0.0024
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.0048	0.00373	0.0048	0.00933	0. 46133

Графическое представление этой статистики можно видеть на рисунке 6:

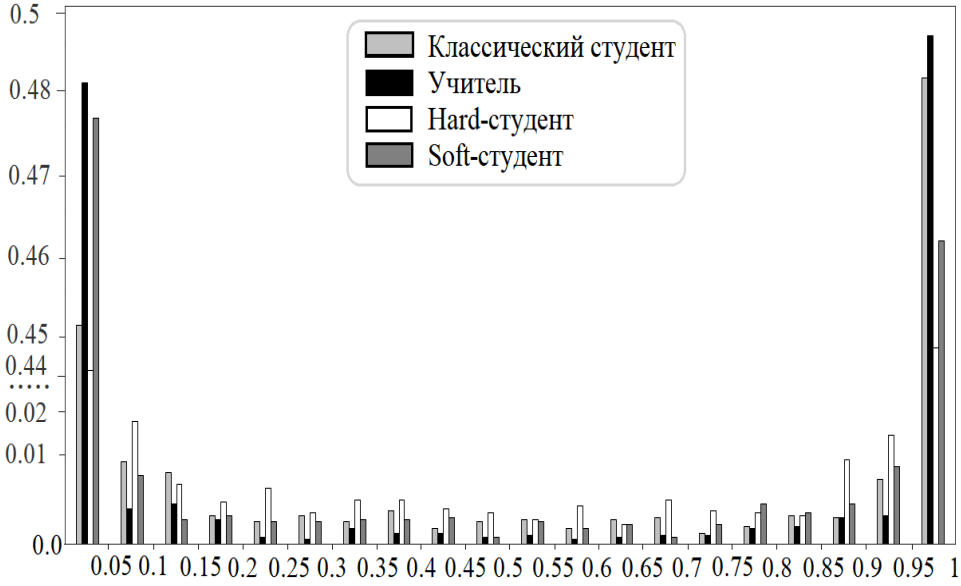


Рис. 6. Распределение относительных частот предсказаний обученных нейронных сетей на тестовом множестве

Заметно, что у всех рассматриваемых сетей в предсказаниях явно преобладают значения очень близкие к 0 или 1, а на остальных проме-

жутках значения во много раз меньше и не превышают 0.015. Можно сказать, что все сети с большой степенью уверенности, то есть очень близко (в данном случае с погрешностью 0.05) к 0 или 1, предсказывают «кошку» или «собаку». Самое уверенное поведение демонстрирует, что вполне ожидаемо, сеть-учитель. Между студентами нельзя выявить наиболее неуверенную сеть – на внутренних полуинтервалах наибольшие значения имеют разные студенты.

2. Проверочное множество. В нем также по 1875 изображений каждого класса. Распределения относительных частот предсказаний для всех моделей изложены в таблицах ниже.

Учитель:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.4808	0.00347	0.00373	0.00107	0.00133
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00107	0.00187	0.00053	0.0008	0.0024
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00133	0.00133	0.0008	0.00133	0.00293
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00267	0.0032	0.00293	0.00373	0.48267

Классический студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.45387	0.0104	0.00693	0.00427	0.00507
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00213	0.0016	0.00373	0.00133	0.00187
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00213	0.00267	0.0016	0.00213	0.00133
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00267	0.004	0.0048	0.00773	0.47973

Hard-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.44907	0.01493	0.00933	0.00427	0.00693
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.0048	0.00373	0.004	0.00373	0.0032
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.0024	0.00373	0.00427	0.0048	0.00267
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00693	0.00587	0.0072	0.01787	0.44027

Soft-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.4752	0.00747	0.00373	0.00453	0.00267
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.0024	0.00187	0.00293	0.00187	0.00213
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00187	0.0024	0.00293	0.00213	0.0016
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00533	0.00453	0.00693	0.012	0.45547

Общая картина осталась примерно такой же: как и на тестовом множестве все сети с очень большой степенью уверенности предсказывают или «кошку», или «собаку». А на внутренних промежутках относительные частоты предсказаний не превышают значения 0.015. Студенты стали немного увереннее – большее количество предсказаний стало попадать в промежутки более близкие к краям отрезка $[0, 1]$.

Результаты для проверочного множества изображены на рисунке 7:

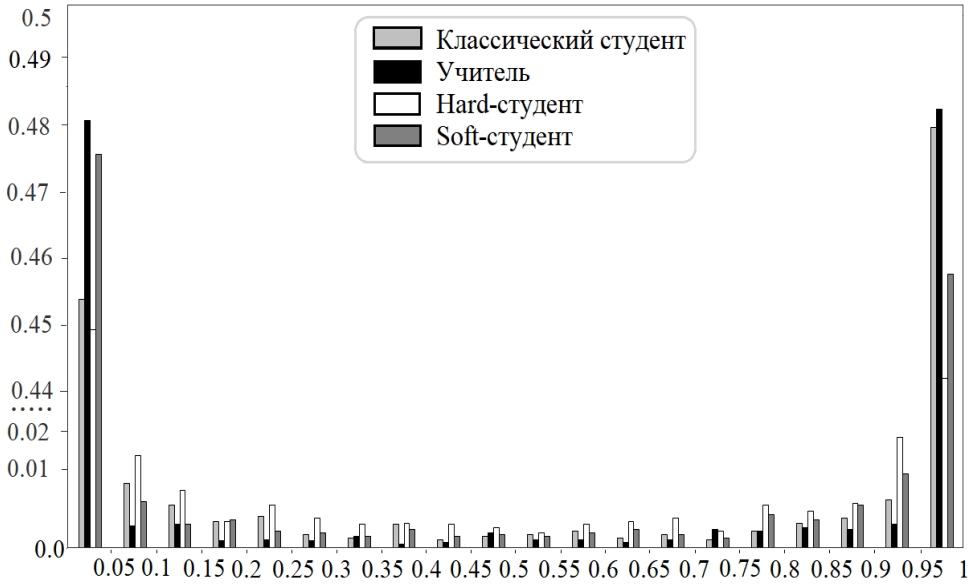


Рис. 7. Распределение относительных частот предсказаний обученных нейронных сетей на проверочном множестве

3. Тренировочное множество. В нем 17500 изображений, по 8750 изображений «кошек» и «собак». Относительные частоты предсказаний обученных моделей представлены в таблицах далее.

Учитель:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.49966	0.00011	0.00006	0.00006	0.0
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.0	0.00006	0.0	0.0	0.0
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.0	0.00006	0.00011	0.00006	0.0
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00011	0.00011	0.00023	0.00046	0.49891

Классический студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.47222	0.00869	0.00451	0.00337	0.00217
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00149	0.0016	0.00114	0.00131	0.00137
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00137	0.00126	0.00097	0.00189	0.002
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.002	0.002	0.00291	0.00571	0.48200

Hard-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.46382	0.01274	0.00594	0.00451	0.00406
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.00371	0.00371	0.00229	0.00268	0.00257
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00229	0.00251	0.00263	0.00291	0.00337
промежуток	[0.75, 0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00331	0.00451	0.00691	0.01297	0.45257

Soft-студент:

промежуток	[0, 0.05)	[0.05,0.1)	[0.1,0.15)	[0.15,0.2)	[0.2,0.25)
частота	0.49189	0.00406	0.00149	0.00114	0.0008
промежуток	[0.25,0.3)	[0.3, 0.35)	[0.35,0.4)	[0.4,0.45)	[0.45, 0.5)
частота	0.0008	0.00103	0.00086	0.00051	0.0004
промежуток	[0.5,0.55)	[0.55,0.6)	[0.6,0.65)	[0.65,0.7)	[0.7,0.75)
частота	0.00109	0.00126	0.0008	0.0008	0.00131
промежуток	[0.75,0.8)	[0.8,0.85)	[0.85,0.9)	[0.9,0.95)	[0.95,1]
частота	0.00171	0.00251	0.0036	0.00589	0.47806

Результаты для тренировочного множества изображены на рисунке 8:

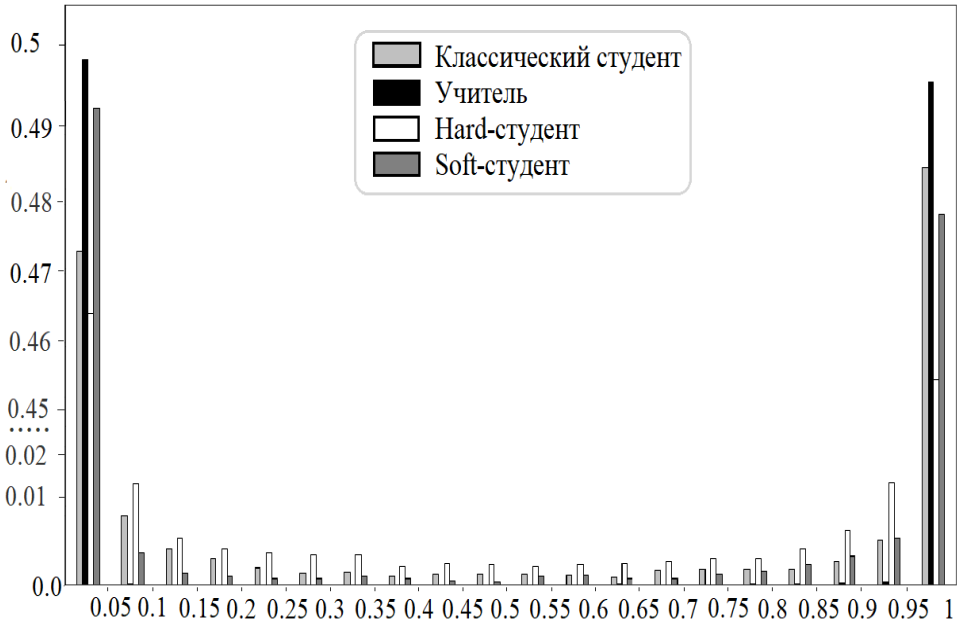


Рис. 8. Распределение относительных частот предсказаний обученных нейронных сетей на тренировочном множестве

Поскольку сети обучались на этом множестве, то и предсказания получаются наиболее уверенными. У каждой сети больше 91% ответов оказываются на промежутках [0,0.05) и [0.95, 1]. У сетей-студентов также в крайние промежутки попадает предсказаний больше, чем на тестовом или проверочном множествах. Видно расслоение графиков – наиболее уверенным является учитель, наименее уверенно ведет себя hard-студент. Промежуточное положение занимают классический студент и soft-студент. Точнее, при предсказаниях менее 0.5 soft-студент везде имеет меньшие значения, чем классический студент, но чем ближе к единице, тем менее уверенным он становится. Возможно, это связано с тем, что soft-студент обучался по ответам сети-учителя. Было выявлено, что сеть-учитель ошибается больше на изображениях с собаками. Ввиду этого, можно сказать, что эта модель более не уверена в их отношении, что естественно отражается в предсказаниях, выступивших в качестве меток при обучении soft-студента. Тем самым сеть-учитель может передавать свою неуверенность студенту.

4.3. Степень неуверенности на множестве

Для задачи бинарной классификации введем понятие «степень неуверенности модели на множестве», вычисляемая по следующей формуле:

$$unc = \frac{1}{n}\sum_{i=1}^{n}(p_i - y_i)^2 \ , \tag{15}$$

где n – мощность множества, на котором вычисляется степень неуверенности, y_i – значение, принимаемое за ответ модели на этом объекте, p_i – предсказание модели на i-ом изображении этого множества. В рассматриваемой задаче за ответ модели на i-ом объекте принимается предсказание сети, преобразованное в соответствии с формулой (13), то есть $y_i \in \{0, 1\} \, \forall \, i = \overline{1, n}$.

В следующей таблице приведены степень неуверенности и точность для каждой из рассматриваемых нейронных сетей на тренировочном, проверочном и тестовом множествах (с ручной разметкой). В каждой клетке таблицы выше диагонали указана точность модели на соответствующем столбцу множестве. А ниже диагонали – степень неуверенности, посчитанная по формуле (15).

Множество / Сеть	Тренировочное множество	Проверочное множество	Тестовое множество
Учитель	0.99983 / 0.00006	0.97526 / 0.00252	0.97333 / 0.00197
Классический студент	0.98048 / 0.00262	0.95105 / 0.00402	0.96160 / 0.00467
Hard-студент	0.97762 / 0.005	0.95293 / 0.00674	0.95440 / 0.00713
Soft-студент	0.98546 / 0.00163	0.94917 / 0.00421	0.96320 / 0.00403

4.4. Рекурсивное применение технологии Knowledge Distillation

Был дважды проведен эксперимент по рекурсивному обучению сетей ResNet-18 по технологии Knowledge Distillation. В каждом испытании

на новом шаге в качестве меток выступали предсказания сети ResNet-18, полученной на предыдущем шаге. Эти предсказания предварительно преобразовывались по формулам (13) и (14), на первом шаге использовались предсказания hard-студента и soft-студента в первом и втором экспериментах соответственно. При обучении всех моделей для чистоты эксперимента использовались те же гиперпараметры обучения, как и при обучении сетей-студентов. Таким образом, был использован оптимизатор Adam со скоростью обучения 0.0001 и нулевым значением параметра d из формулы (9). Параметры β_1 и β_2 из формул (4)–(9) равнялись 0.9 и 0.999 соответственно. Для оценки качества работы моделей вычислялась точность сети в соответствии с формулой (12). Функция потерь – бинарная кросс-энтропия. Каждая модель обучалась 100 эпох с выборкой размера 32. При рекурсивном обучении применялась аугментация данных, как и при обучении сетей-студентов.

В следующих таблицах для каждого эксперимента приведены точности и значения степени неуверенности на всех рассматриваемых множествах для сетей, полученных в ходе этих испытаний. В каждой клетке таблицы выше диагонали указана точность, а ниже – степень неуверенности модели на соответствующем множестве.

Результаты рекурсивного обучения с помощью hard-студента:

Сеть \ Множество	Тренировочное множество	Проверочное множество	Тестовое множество
Hard-студент	0.97762 0.005	0.95293 0.00674	0.95440 0.00713
ResNet-18 №1	0.98225 0.00391	0.94621 0.0069	0.95760 0.006
ResNet-18 №2	0.97463 0.00379	0.94567 0.00569	0.94880 0.00515
ResNet-18 №3	0.98025 0.00292	0.94809 0.0062	0.94773 0.00668
ResNet-18 №4	0.98117 0.0021	0.94567 0.00556	0.94293 0.00575
ResNet-18 №5	0.96336 0.00616	0.94621 0.00866	0.93600 0.00895
ResNet-18 №6	0.97281 0.00352	0.94594 0.00572	0.93520 0.00561

Результаты рекурсивного обучения с помощью soft-студента:

Множество / Сеть	Тренировочное множество	Проверочное множество	Тестовое множество
Soft-студент	0.98546 0.00163	0.94917 0.00421	0.96320 0.00403
ResNet-18 №1	0.98397 0.00268	0.95535 0.00529	0.96320 0.00454
ResNet-18 №2	0.98363 0.00636	0.95697 0.00934	0.95733 0.00935
ResNet-18 №3	0.97933 0.00937	0.95966 0.01236	0.95093 0.01272
ResNet-18 №4	0.97670 0.01249	0.95858 0.01547	0.95307 0.01576
ResNet-18 №5	0.97384 0.01433	0.95320 0.01795	0.94507 0.01856
ResNet-18 №6	0.97258 0.01940	0.95643 0.02302	0.94587 0.02268

В обоих экспериментах наблюдается относительное снижение точности в процессе рекурсивного обучения, но и в том, и в другом случаях небольшое – менее 3%. Однако в отношении степени неуверенности в этих экспериментах получены различные результаты. В первом случае значение степени неуверенности на каждом множестве практически не меняется на всех шагах – отличие от hard-студента не более 0.003. Во втором эксперименте, наоборот, наблюдается явный рост значений этой величины на каждом множестве. При этом точность на тестовом множестве у сети, обученной в ходе эксперимента с soft-студентом, всегда больше, чем у модели, полученной на аналогичном шаге в опыте с hard-студентом.

Кроме того, для рекурсивно обученных сетей были изучены распределения относительных частот предсказаний на тестовом множестве. Для первого эксперимента, поскольку значения степени неуверенности не увеличивались и оставались очень малыми, то и распределения предсказаний остались примерно такими же, как и для hard-студента. Напротив, в ходе эксперимента с soft-студентом с каждым новым шагом всё большее

количество предсказаний попадало во внутренние полуинтервалы за счет уменьшения значений в крайних промежутках. В большей степени эта закономерность проявилась на отрезке $[0.5,\ 1]$, то есть на промежутках, попадание в которые интерпретируется как принадлежность объекта к классу «изображение с собакой». Эту динамику можно увидеть на рисунке 9:

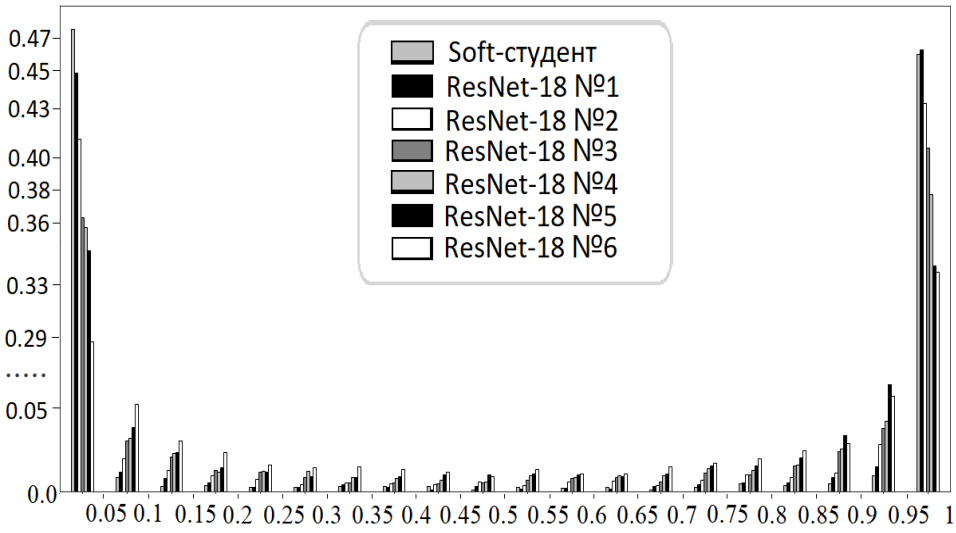


Рис. 9. Распределение относительных частот предсказаний на тестовом множестве рекурсивно обученных с помощью soft-студента нейронных сетей

Таким образом, помимо общего увеличения значения степени неуверенности у рекурсивно обученных с помощью soft-студента сетей, наблюдается наследование характера распределения этой неуверенности. Тем самым, получает подтверждение предположение о возможности наследования неуверенности обучаемой сетью от своей сети-учителя.

5. Выводы

В ходе данной работы подтверждена возможность обучения по технологии Knowledge Distillation несложных моделей ResNet-18, способных выдавать результаты, сопоставимые с аналогичными измерениями для сети-учителя, полученной из нейронной сети ResNet-50, для задачи бинарной классификации. При этом использование сети-студента позво-

ляет увеличить производительность вычислений по сравнению с сетью-учителем практически в 5 раз.

Все три сети-студента дают примерно одинаковые результаты, причем, по сравнению с классическим студентом, soft-студент показал лучший результат на тестовом множестве с ручной разметкой, как с применением аугментации данных, так и без неё. Hard-студент, в свою очередь, продемонстрировал результаты ненамного хуже классического студента, но соответствующие значения параметров были получены практически в два раза быстрее: после 86 эпох – у классического и после 47 эпох – у hard-студента.

Были изучены распределения относительных частот предсказаний на различных множествах для каждой обученной нейронной сети. Замечена закономерность между характером ошибок сети-учителя и soft-студента, проявившаяся в переносе неуверенности сети-учителя в предсказании принадлежности изображения к классу «изображение с собакой» на аналогичное поведение, выявленное у soft-студента. У hard-студента это влияние не столь заметно, поскольку детали подобного рода нивелируются в форме представления предсказаний сети-учителя, выступавших в качестве меток в процессе обучения этой модели, в отличие от soft-студента, который в процессе обучения аппроксимировал непосредственно предсказания сети-учителя. Такая же тенденция выявлена при рекурсивном обучении моделей по данной технологии.

В процессе изучения распределений относительных частот предсказаний было введено понятие степени неуверенности модели на множестве как величины отклонения предсказаний нейронной сети от значений, принимаемых за ответ модели на рассматриваемом множестве. Эту величину можно рассматривать как интуитивно понятную меру способности модели к обобщению. На всех обученных сетях значение степени неуверенности очень небольшое, что может означать, что способность к обобщению у моделей, обученных по технологии Knowledge Distillation, сохраняется.

Также рекурсивно были обучены модели ResNet-18 по технологии Knowledge Distillation, с обеими формами представления предсказаний сети-учителя. Подобная процедура в обоих случаях показала вполне стабильные результаты: обученные в ходе каждого эксперимента модели делают достаточно точные предсказания на всех множествах, выступая наравне с сетями-студентами. Более того были получены нейронные сети, которые не уступают на тестовом множестве моделям, обученным с помощью более сложной нейронной сети.

Дальнейшие исследования будут направлены на изучение возможности применения данной технологии для других задач.

Автор выражает благодарность доценту кафедры МаТИС механико-математического факультета МГУ, к.ф.-м.н. Часовских Анатолию Александровичу и доценту кафедры высшей математики института Кибернетики РТУ МИРЭА, к.т.н. Парфенову Денису Васильевичу за постановку задачи и помощь в исследовании.

Список литературы

[1] Earnest Paul Ijjina, Dhananjai Chand, Savyasachi Gupta, Goutham K, *Computer Vision-based Accident Detection in Traffic Surveillance*, arXiv: arxiv.org/abs/1911.10037.

[2] Omid Bazgir, Ruibo Zhang, Saugato Rahman Dhruba, Raziur Rahman, Souparno Ghosh, Ranadip Pal, *REFINED (Representation of Features as Images with Neighborhood Dependencies): A novel feature representation for Convolutional Neural Networks*, arXiv: arxiv.org/abs/1912.05687.

[3] Sara Sabour, Nicholas Frosst, Geoffrey E Hinton, *Dynamic Routing Between Capsules*, arXiv: arxiv.org/abs/1710.09829.

[4] Jo Schlemper, Chen Qin, Jinming Duan, Ronald M. Summers, Kerstin Hammernik, $\sum$*-net: Ensembled Iterative Deep Neural Networks for Accelerated Parallel MR Image Reconstruction*, arXiv: arxiv.org/abs/1912.05480.

[5] Ramit Pahwa, Manoj Ghuhan Arivazhagan, Ankur Garg, Siddarth Krishnamoorthy, Rohit Saxena, Sunav Choudhary, *Data-Driven Compression of Convolutional Neural Networks*, arXiv: arxiv.org/abs/1911.12740.

[6] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, *Distilling the Knowledge in a Neural Network*, arXiv: arxiv.org/abs/1503.02531.

[7] Gaurav Menghani, Sujith Ravi, *Learning from a Teacher using Unlabeled Data*, arXiv: arxiv.org/abs/1911.05275.

[8] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun, *Deep Residual Learning for Image Recognition*, arXiv: arxiv.org/abs/1512.03385.

[9] Система организации конкурсов по исследованию данных «Kaggle», *Соревнование «Dogs vs. Cats», www.kaggle.com/c/dogs-vs-cats/overview*.

[10] Саймон Хайкин, *Нейронные сети: полный курс*, 2-е издание, Издательский дом «Вильямс», Москва, 2006, 1104 с.

[11] George Philipp, Dawn Song, Jaime G. Carbonell, *The exploding gradient problem demystified – definition, prevalence, impact, origin, tradeoffs, and solutions*, arXiv: arxiv.org/abs/1712.05577.

[12] Ян Эрик Солем, *Программирование компьютерного зрения на языке Python*, ДМК Пресс, Москва, 2016, 312 с.

[13] В.Б.Кудрявцев, Э.Э. Гасанов, А.С. Подколзин, *Основы теории интеллектуальных систем*, МАКС Пресс, Москва, 2016, 612 с.

[14] Bao-Gang Hu, Ran He, XiaoTong Yuan, *Information-Theoretic Measures for Objective Evaluation of Classifications*, arXiv: arxiv.org/abs/1107.1837.

[15] Mitchell A. Gordon, Kevin Duh, *Explaining Sequence-Level Knowledge Distillation as Data-Augmentation for Neural Machine Translation*, arXiv: arxiv.org/abs/1912.03334.

[16] Tyler Sypherd, Mario Diaz, Lalitha Sankar, Peter Kairouz, *A Tunable Loss Function for Binary Classification*, arXiv: arxiv.org/abs/1902.04639.

[17] Diederik P. Kingma, Jimmy Ba, *Adam: A Method for Stochastic Optimization*, arXiv: arxiv.org/abs/1412.6980.

The technology of knowledge distillation for training neural networks on example of binary classification

Biryukova V.A.

Using the technology of training neural networks, the knowledge distillation, models were obtained that solve the binary classification problem with the productivity that is about five times higher than the performance of the teacher network with an insignificant drop in quality. The convolutional neural network ResNet-18 was trained in two ways by this technology (using the pre-trained network ResNet-50) and by the classical method. The concept of the degree of uncertainty of the model on objects' set is introduced as the quantity of the deviation of the neural network predictions from the values accepted for the answer. The experiments on the recursive application of the knowledge distillation technology were also conducted.

Keywords: knowledge distillation, binary classification, residual neural network, convolutional neural network, degree of uncertainty of the model on objects' set, recursive training of neural networks.

Часть 2.
Специальные вопросы теории интеллектуальных систем

Необходимые и достаточные условия существования изображения с заданным кодом

Алексеев Д.В.[1]

Вводится кодирующая функция, инвариантная относительно аффинных отображений и исследуются ее свойства. Найдены необходимые и достаточные условия того, что данный набор чисел является кодом невырожденного изображения.

Ключевые слова: код изображения, кодирование изображений, аффинная эквивалентность.

Введение

В задачах распознавания часто возникает необходимость представления точечного изображения в виде какого-то кода. Одним из наиболее часто используемых является координатная запись, когда кодом изображения являются координаты его точек. Несмотря на определенные достоинства, эта кодировка не является инвариантной относительно геометрических преобразований, таких, как сдвиг, поворот, растяжение. Кроме того, она подразумевает привязку к некоторой фиксированной системе координат. В то же время представляется осмысленным рассматривать изображения, полученные в результате таких преобразований эквивалентными.

В работах В.Н. Козлова [4]–[5] предложена кодирующая функция, инвариантная относительно аффинных преобразований. Было показано, что совпадение кодов двух плоских изображений является необходимым

[1] *Алексеев Дмитрий Владимирович* — кандидат физ.-мат. наук, с.н.с. лаборатории Проблем теоретической кибернетики мех.-мат. ф-та МГУ, e-mail: dvalex@rambler.ru

Alekseev Dmitriy Vladimirovich — Candidate of Physical and Matematical Sciences, senior staff scientist, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Problems of Theorecical Cybernetics Lab.

и достаточным условием их аффинной эквивалентности. В данной работе вводится модифицированная кодирующая функция и исследуются ее свойства. В частности, найдены необходимые и достаточные условия того, что некоторый набор чисел является кодом двумерного изображения.

Ранее в работе [2] были получены необходимые и достаточные условия того, что существует трехмерное изображение, обладающее данными плоскими проекциями.

В работе [5] была предложена кодирующая функция ρ, инвариантная к аффинным преобразованиям плоскости: $\rho_{ijk,lmp} = \frac{S(\triangle a_i a_j a_k)}{S(\triangle a_l a_m a_p)}$, где S означает площадь треугольника. Таким образом, набору из n точек ставится в соответствии код, состоящий из $(C_n^3)^2$ чисел. Очевидно, этот код является избыточным. Данная работа исследует вопрос, какова степень его избыточности. Для модифицированной кодирующей функции приводятся явные условия того, что множество чисел является кодом изображения. Для оригинальной функции кодирования также приводятся соответствующие условия (в неявном виде).

В данной работе рассматривается модифицированная кодирующая функция $r_{ijk,lmp} = \frac{S'(\triangle a_i a_j a_k)}{S'(\triangle a_l a_m a_p)}$, где S' — ориентированная площадь, т.е. площадь со знаком, зависящим от направления обхода вершин треугольника.

Для модифицированной кодирующей функции приводятся явные условия того, что множество чисел является кодом изображения. Для оригинальной функции кодирования также приводятся соответствующие условия (в неявном виде).

Дальнейшая структура статьи устроена следующим образом: В разделе 1 вводятся основные понятия и обозначения. В разделе 2 изучаются свойства матрицы кода изображения. В разделе 3 приводится основной результат — необходимые и достаточные условия того, что набор чисел является кодом невырожденного изображения. В разделе 4 формулируются выводы и планы на будущее.

1. Основные обозначения

Введем понятие ориентированной площади треугольника $S'(\triangle abc) = S(\triangle abc)$ в случае положительного направления обхода и $S'(\triangle abc) = -S(\triangle abc)$ в случае отрицательного. Положительным считается такое на-

правлением обхода, при котором треугольник расположен слева, т.е. направление против часовой стрелки.

Пусть на плоскости расположены точки $a_1, ..., a_n$. Будем называть множество $A = \{a_1, ..., a_n\}$ изображением. Изображение называется *вырожденным* в случае если все точки лежат на одной прямой и *невырожденным* в противном случае. Зафиксируем некоторую (евклидову) систему координат, будем обозначать координаты точки a_i как $X(a_i)$ и $Y(a_i)$. В дальнейшем для удобства будем обозначать отдельные индексы строчными латинскими буквами, а мультииндексы, т.е. вектора, состоящие из трех индексов, будем обозначать строчными греческими буквами $\alpha, \beta, \gamma, ...$ Компоненты мультииндекса будем обозначать $\alpha = [\alpha(1), \alpha(2), \alpha(3)]$, а треугольник[1] с соответствующими номерами вершин будем обозначать $\triangle_\alpha = \triangle a_{\alpha(1)} a_{\alpha(2)} a_{\alpha(3)}$. Введем отношение эквивалентности на мультииндексах — не будем различать мультииндексы, если один из них можно получить циклической перестановкой другого. Более формально введем отношение $\alpha \simeq \alpha'$ тогда и только тогда, когда перестановка $\begin{pmatrix} \alpha(1) & \alpha(2) & \alpha(3) \\ \alpha'(1) & \alpha'(2) & \alpha'(3) \end{pmatrix} \in S_3$ — четная. Будем назвать мультииндексом сопряженным к α и обозначать $\bar{\alpha}$ такой мультииндекс, что перестановка $\begin{pmatrix} \alpha(1) & \alpha(2) & \alpha(3) \\ \bar{\alpha}(1) & \bar{\alpha}(2) & \bar{\alpha}(3) \end{pmatrix} \in S_3$ — нечетная. Содержательный смысл этого такой — треугольники, соответствующие эквивалентным индексам имеют одинаковую ориентированную площадь, а треугольники, соответствующие сопряженным мультииндексам имеют площади, отличающиеся знаком. В дальнейшем под словом "мультииндекс"будет подразумеваться соответствующий класс эквивалентности.

Всего существует C_n^3 различных треугольников с вершинами в точках $a_1, ..., a_n$, если не учитывать ориентацию и $N = 2 \cdot C_n^3$ — с учетом ориентации.

Занумеруем все мультииндексы (т.е. соответствующие треугольники): $\alpha_1, ..., \alpha_N$, пусть $\mathcal{A} = \{\alpha_1, ..., \alpha_N\}$ — множество всех мультииндексов, обозначим $E : \alpha_i \mapsto i$ соответствующую функцию нумерации.

Рассмотрим множество отношений: $r_{ijk,lmp} = \frac{S'(\triangle a_i a_j a_k)}{S'(\triangle a_l a_m a_p)}$. Если треугольник $\triangle a_l a_m a_p$ вырожден, т.е. $S'(\triangle a_l a_m a_p) = 0$, то используем формальный символ $r_{ijk,lmp} = \infty$. Будем называть это множество кодом изображения $\{a_1, ..., a_n\}$. Подобная кодирующая функция была предложена в [1].

[1] Это обозначение будем использовать также и в случае, когда треугольник вырожден, т.е. все точки лежат на одной прямой.

Определение 1. Рассмотрим матрицу $R = (r_{ij})$ размера $N \times N$ в которой на пересечении $i-$строки и $j-$го столбца расположен элемент $r_{ij} = r_{\alpha_i,\alpha_j} = \frac{S'(\triangle_\alpha)}{S'(\triangle_\beta)}$. Таким образом, элементы кода изображения расположены в виде квадратной таблицы, в которой строки и столбцы занумерованы мультииндексами (треугольниками). Будем называть R *матрицей кода изображения.*

Замечание 1. В дальнейшем будет использоваться также обозначение $r_{\alpha\beta} = R_{E(\alpha)E(\beta)}$, т.е. строки и столбцы матрицы кода могут быть занумерованы мультииндексами.

Пример 1. Рассмотрим трапецию $a_1a_2a_3a_4$, основания которой a_1a_2 и a_4a_3 относятся как $1:2$ (см. рис. 1). Занумеруем мультииндексы как

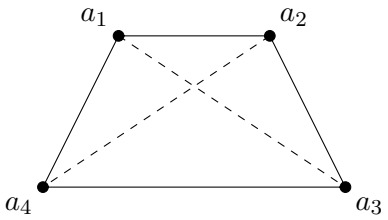


Рис. 1. К примеру 1.

в таблице 1. Заметим, что последние 4 мультииндекса являются сопряженными к первым четырем, так что достаточно построить только часть матрицы кода изображения, соответствующую первым 4 строкам и столбцам. Она будет иметь вид $R_4 = \begin{pmatrix} 1 & 1 & 1/2 & 1/2 \\ 1 & 1 & 1/2 & 1/2 \\ 2 & 2 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{pmatrix}$. А полная матрица кода будет иметь вид $R = \begin{pmatrix} R_4 & -R_4 \\ -R_4 & R_4 \end{pmatrix}$.

2. Свойства матрицы кода изображения

1) $r_{\alpha\alpha} = 1$ или ∞, при любых $\alpha \in \mathcal{A}$ (**рефлексивность**).

2) При любых $\alpha, \beta \in \mathcal{A}$, таких, что $r_{\alpha\beta} \notin \{0, \infty\}$ выполняется $r_{\beta\alpha} = r_{\alpha\beta}^{(-1)}$ (**антисимметричность**[2]).

i	α_i
1	$1,2,3$
2	$1,2,4$
3	$1,3,4$
4	$2,3,4$
5	$1,3,2$
6	$1,4,2$
7	$1,4,3$
8	$2,4,3$

Таблица 1. Таблица мультииндексов

3) При любых $\alpha, \beta, \gamma \in \mathcal{A}$, таких, что $r_{\alpha\beta}, r_{\beta\gamma} \notin \{0, \infty\}$, выполняется $r_{\alpha\gamma} = r_{\alpha\beta} \cdot r_{\beta\gamma}$ (**транзитивность**).

4) Пусть заданы перестановки $\pi, \sigma \in S_3$ и мультииндексы $\alpha, \beta \in \mathcal{A}$. Пусть мультииндексы $\alpha' = \pi(\alpha)$ и $\beta' = \sigma(\beta)$ получены применением перестановок π и σ к мультииндексам α и β, соответственно, т.е. $\alpha' = [\alpha(\pi(1)), \alpha(\pi(2)), \alpha(\pi(3))]$ и $\beta' = [\beta(\sigma(1)), \beta(\sigma(2)), \beta(\sigma(3))]$. Тогда либо $r_{\alpha'\beta'} = (-1)^\pi \cdot (-1)^\sigma \cdot r_{\alpha\beta}$, либо $r_{\alpha\beta} = \infty = r_{\alpha'\beta'}$ (**согласованность с перестановками индексов**).

5) Пусть $i_1, i_2, i_3, i_4 \in \{1, \ldots, N\}$, $\alpha_1 = [i_2, i_3, i_4]$, $\alpha_2 = [i_3, i_4, i_1]$, $\alpha_3 = [i_4, i_1, i_2]$ и $\alpha_4 = [i_1, i_2, i_3]$. Тогда для произвольного $\beta \in \mathcal{A}$ выполнено равенство[3] $r_{\alpha_1\beta} + r_{\alpha_3\beta} = r_{\alpha_2\beta} + r_{\alpha_4\beta}$ (**аддитивность**).

Свойства 1–3 очевидны. Свойство 4 следует из изменения ориентированной площади при перестановках вершин. Свойство 5 следует из подсчета площади четырехугольника $a_{i_1}a_{i_2}a_{i_3}a_{i_4}$ (см. рис. 2) двумя различными способами. Действительно,

$$S'(a_{i_1}a_{i_2}a_{i_3}a_{i_4}) = S'(\triangle a_{i_2}a_{i_3}a_{i_4}) + S'(\triangle a_{i_4}a_{i_1}a_{i_2}) = \\ = S'(\triangle a_{i_3}a_{i_4}a_{i_1}) + S'(\triangle a_{i_1}a_{i_2}a_{i_3}).$$

Разделив обе части на $S'(\triangle_\beta)$, получаем требуемое равенство.

Возникает вопрос: являются ли эти условия достаточными для того, чтобы матрица была матрицей кода некоторого изображения? Ниже будет приведен контрпример, показывающий, что это не так.

[2]Если $r_{\alpha\beta} = 0$, то $r_{\beta\alpha} = \infty$. Обратное, вообще говоря, неверно.

[3]В случае нулевого знаменателя равенство рассматриваем как формальное $\infty + \infty = \infty + \infty$.

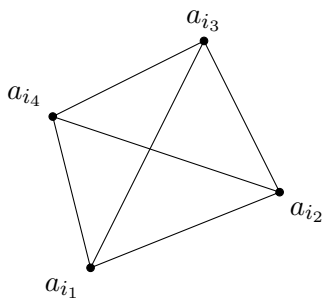


Рис. 2. Аддитивность площадей

Сначала докажем вспомогательную лемму:

Лемма 1. *Пусть треугольник $\triangle_\beta$ — невырожденный, обозначим $\rho_\alpha = r_{\alpha\beta}$, $\alpha \in \mathcal{A}$. Пусть на плоскости задана евклидова система координат. Тогда существует аффинное преобразование F, переводящее точки $a_i, i = 1, ..., n$ в $c_i = F(a_i)$, такое, что координаты $X(c_i) = \rho_{i,\beta(3),\beta(1)}$, $Y(c_i) = \rho_{i,\beta(1),\beta(2)}$.*

Доказательство. Не ограничивая общности рассуждений можно считать, что $\beta = [1, 2, 3]$. Построим аффинное преобразование A, переводящее $a_1 \mapsto c_1(0, 0)$, $a_2 \mapsto c_2(1, 0)$ и $a_3 \mapsto c_3(0, 1)$ (такое преобразование существует и единственно). Обозначим (x_i, y_i) — координаты $c_i = A(a_i)$. Тогда $S'(\triangle c_1c_2c_3) = \frac{1}{2}$, $S'(\triangle c_3c_1c_i) = \frac{1}{2}x_i$ и $S'(\triangle c_1c_2c_i) = \frac{1}{2}y_i$. Следовательно, $\rho_{3,1,i} = \frac{S'(\triangle c_3c_1c_i)}{S'(\triangle c_1c_2c_3)} = x_i$ и $\rho_{3,1,i} = \frac{S'(\triangle c_3c_1c_i)}{S'(\triangle c_1c_2c_3)} = y_i$, что и требовалось доказать.

Следствие 1. *Если треугольник $\triangle_\beta$ — невырожденный и $\rho_{i,\beta(1),\beta(2)} = \rho_{j,\beta(1),\beta(2)} = \rho_{k,\beta(1),\beta(2)}$, то точки a_i, a_j, a_k лежат на одной прямой.*

Доказательство. По лемме 1, существует аффинное преобразование, переводящее a_i, a_j, a_k в точки c_i, c_j, c_k с одинаковыми ординатами $Y(c_i) = Y(c_j) = Y(c_k)$, а, следовательно, лежащие на одной прямой.

Следствие 2. *Для того, чтобы два невырожденных изображения A и B были аффинно–эквивалентны, необходимо и достаточно, чтобы матрицы их кодов совпадали (при некоторой нумерации).*

Замечание 2. Это следствие является аналогом теоремы 1 из [4] (для другой кодирующей функции).

Доказательство. Необходимость вытекает из сохранения отношений ориентированных площадей при аффинных преобразования. Докажем достаточность. В невырожденном изображении всегда найдется невырожденный треугольник $\triangle_\beta(A) = \triangle a_i a_j a_k$. Из совпадения кодов следует, что соответствующий треугольник $\triangle_\beta(B) = \triangle b_i b_j b_k$ — тоже невырожденный. Тогда по лемме 1 можно построить аффинные преобразования $F_1 : A \to C$ и $F_2 : B \to C$, где точки изображения C имеют координаты $X(c_i) = \rho_{i,\beta(3),\beta(1)}$, $Y(c_i) = \rho_{i,\beta(1),\beta(2)}$). Следовательно, A и B аффинно–эквивалентны, что и требовалось доказать.

Покажем на примере, что свойства 1–5 не являются достаточными для существования изображения, обладающего данной матрицей кода.

Пример 2. Рассмотрим правильный пятиугольник с вершинами a_1,..., a_5. Обозначим точки пересечения диагоналей b_1,..., b_5 (см. рис. 3). Расположим точки m_1, m_2, m_3 внутри треугольников $\triangle a_1 a_2 b_4$, $\triangle a_4 a_5 b_2$ и $\triangle a_3 b_1 b_5$, соответственно. Поместим в эти точки единичные массы. Для треугольника $\triangle a_i a_j a_k$ рассмотрим суммарную массу точек, расположенных внутри него. Будем называть эту массу, взятую со знаком $+/-$ в зависимости от направления обхода, псевдо-площадью треугольника $\triangle a_i a_j a_k$ и обозначать $S^*(\triangle a_i a_j a_k)$. Очевидно, для псевдо-площади выполняется закон аддитивности. Рассмотрим матрицу $R = (r_{\alpha\beta})$, $r_{\alpha\beta} = S^*(\triangle_\alpha)/S^*(\triangle_\beta)$. свойство 4 вытекает из определения псевдо-площади, а свойство 5 — из ее аддитивности.

Предположим, что матрица R является матрицей кода некоторого изображения $a'_1, ..., a'_5$. Заметим, что

$$r_{123,123} = r_{124,123} = r_{125,123} = 1,$$

тогда, по следствию 1, точки a'_3, a'_4 и a'_5 расположены на одной прямой. Поэтому площадь треугольника $\triangle a'_3 a'_4 a'_5$ должна быть равна нулю, следовательно, $r_{345,123} = 0$. Но $r_{345,123} = 1 \neq 0$. Противоречие.

Замечание 3. Можно более строго определить понятие псевдо-площади. Для этого надо разместить рис. 3 на комплексной плоскости и интерпретировать точки как элементы $\mathbb{C}$. Рассмотрим мероморфную функцию $f(z) = \frac{1}{z-m_1} + \frac{1}{z-m_2} + \frac{1}{z-m_3}$, тогда в качестве определения псевдо-площади треугольника можно взять интеграл по его контуру:

$$S^*(\triangle a_i a_j a_k) = \frac{1}{2\pi i} \oint\limits_{\triangle a_i a_j a_k} f(z)\mathrm{dz}.$$

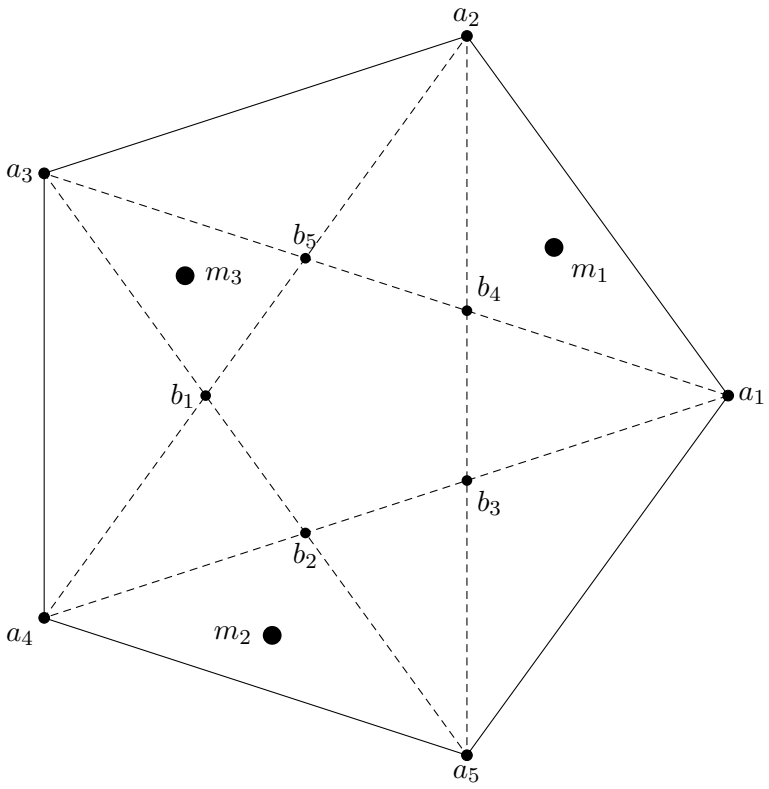


Рис. 3. К примеру 2.

3. Основные результаты

Итак, условий 1–5 недостаточно для существования изображения с указанным кодом. Теорема, приведенная ниже, отвечает на вопрос — какие дополнительные условия надо наложить для существования такого изображения.

Теорема 1. *Пусть матрица R удовлетворяет условиям 1–5. Пусть существуют $\alpha, \beta \in \mathcal{A}$, для которых $r_{\alpha,\beta} \neq \infty$ Тогда для того, чтобы R была матрицей кода некоторого невырожденного изображения, необходимо и достаточно, чтобы при любых $i, j = 1, ..., n$ выполнялось равенство*

$$\rho_{\beta(1),i,j} = \rho_{i,\beta(3),\beta(1)} \cdot \rho_{j,\beta(1),\beta(2)} - \rho_{j,\beta(3),\beta(1)} \cdot \rho_{i,\beta(1),\beta(2)}, \qquad (1)$$

где $\rho_\alpha = r_{\alpha,\beta}$.

Доказательство.

Необходимость. Не ограничивая общности рассуждений будем считать, что $\beta = [1,2,3]$. Рассмотрим отображение из доказательства леммы 1. Точки a_s, $s = 1,2,3$ перейдут в точки с координатами $c_1(0,0)$, $c_2(1,0)$ и $c_3(0,1)$, соответственно, $S'(\triangle c_1c_2c_3) = \frac{1}{2}$. Точки a_i, a_j перейдут в c_i и c_j, с координатами $X(c_i) = \rho_{i,3,1}$, $Y(c_i) = \rho_{i,1,2}$, $X(c_j) = \rho_{j,3,1}$, $Y(c_j) = \rho_{j,1,2}$. Тогда площадь треугольника $\triangle c_1c_ic_j$ можно найти по известной формуле из аналитической геометрии:

$$S(\triangle c_1c_ic_j) = \frac{1}{2} det \begin{pmatrix} X(c_i) & Y(c_i) \\ X(c_j) & Y(c_j) \end{pmatrix} = \frac{1}{2}(\rho_{i,3,1} \cdot \rho_{j,1,2} - \rho_{j,3,1} \cdot \rho_{i,1,2}).$$

Поделив обе части на $S'(\triangle c_1c_2c_3) = \frac{1}{2}$, получаем равенство(1).

Достаточность. Пусть равенство (1) выполнено. Построим множество точек $\{a_i : i = 1, ..., N\}$ с координатами $X(a_i) = \rho_{i,3,1}$, $Y(a_i) = \rho_{i,1,2}$. Построим матрицу кода этого изображения $R^* = (r^*_{\alpha\beta})$ и покажем, что она совпадает с данной матрицей R. Будем обозначать $\rho^*_\alpha = r^*_{\alpha\beta}$. Пусть $\alpha = [i,j,k]$, рассмотрим пересечение $P = \{i,j,k\} \cap \{1,2,3\}$. Содержательно P — это множество общих индексов у α и β. Возможны случаи

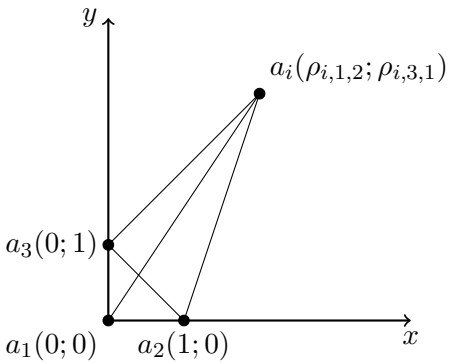


Рис. 4. Случай $|P| = 2$.

- Случай $|P| = 3$. Это означает, что $\alpha = [j,i,k]$ — некоторая перестановка индексов $1,2,3$, т.е. либо $\alpha = \beta$, либо $\alpha = \bar{\beta}$. Тогда $S'(\triangle_\beta) = \frac{1}{2}$ и $S'(\triangle_{\bar{\beta}}) = -\frac{1}{2}$. Следовательно, $\rho^*_\beta = 1 = \rho_\beta$ и $\rho^*_{\bar{\beta}} = 1 = \rho_{\bar{\beta}}$.

- Случай $|P| = 2$ и $1 \in P$ (см. рис. 4). Содержательно это означает, что треугольники $\triangle_\alpha$ и $\triangle_b$ имеют две общие вершины, одна из которых расположена в начале координат. Один из индексов не принадлежит множеству $\{1, 2, 3\}$, не ограничивая общности будем считать, что это i. Если оставшиеся индексы — 1 и 2, то[4] $\alpha = \gamma$ или $\alpha = \bar{\gamma}$, где $\gamma = [1, 2, i]$. При $\alpha = \gamma$ имеем

  $$S'(\triangle_\gamma) = S'(\triangle a_i a_1 a_2) = \frac{1}{2} Y(a_i) = \frac{1}{2} \rho_{i,1,2}.$$

  Разделив на $S'(\triangle_\beta) = 1/2$, получим $\rho^*_\gamma = \rho_\gamma$. Если же $\alpha = \bar{\gamma}$, то $\rho^*_\alpha = \rho^*_{\bar{\gamma}} = -\rho^*_\gamma = -\rho_\gamma = \rho_{\bar{\gamma}}$.

  Случай, когда $P = \{1, 3\}$ доказывается аналогично.

- Случай $P = \{2, 3\}$ (см. рис. 4). Содержательно это означает, что треугольники $\triangle_\alpha$ и $\triangle_b$ имеют две общие вершины — a_1 и a_2. Один из индексов не принадлежит множеству $\{2, 3\}$, не ограничивая общности будем считать, что это i. Тогда $\alpha = \delta$ или $\alpha = \bar{\delta}$, где $\delta = [i, 3, 2]$. Если $\alpha = \delta$, то по свойству 5 (аддитивности)

  $$\rho^*_\alpha = \rho^*_{i,3,2} = \rho^*_{i,3,1} + \rho^*_{i,1,2} - \rho^*_{1,2,3} = \rho_{i,3,1} + \rho_{i,1,2} - \rho_{1,2,3} = \rho_{i,3,2} = \rho_\alpha.$$

  Третье равенство здесь следует из доказанных ранее равенств $\rho^*_{i,1,2} = \rho_{i,1,2}$ и $\rho^*_{i,3,1} = \rho_{i,3,1}$. Если $\alpha = \bar{\delta}$, то $\rho^*_\alpha = \rho^*_{\bar{\delta}} = -\rho^*_\delta = -\rho_\delta = \rho_{\bar{\delta}}$.

- Случай $P = \{1\}$ (см. рис. 5). Содержательно это означает, что треугольники $\triangle_\alpha$ и $\triangle_\beta$ имеют единственную общую вершину, расположенную в начале координат. Не ограничивая общности рассуждений, что оставшиеся две точки это a_i и a_j. Рассмотрим $\triangle a_1 a_i a_j$, его ориентированная площадь равна

  $$S'(\triangle a_1 a_i a_j) = \frac{1}{2} \cdot \det \begin{pmatrix} X(a_i) & Y(a_i) \\ X(a_j) & Y(a_j) \end{pmatrix} = \frac{1}{2} \cdot (\rho_{i,3,1} \cdot \rho_{j,1,2} - \rho_{j,3,1} \cdot \rho_{i,1,2}).$$

  Разделив на $S'(\triangle a_1 a_2 a_2) = 1/2$, получим $\rho^*_{1,i,j} = \rho_{i,3,1} \cdot \rho_{j,1,2} - \rho_{j,3,1} \cdot \rho_{i,1,2} = \rho_{1,i,j}$, где последнее равенство вытекает из (1).

- Общий случай — произвольные i, j, k. Воспользуемся тем, что как ρ так и ρ^* обладают свойством аддитивности. Тогда $\rho^*_{i,j,k} = \rho^*_{1,i,j} +$

[4] Напомним, что равенство мультииндексов понимается с точностью до циклической перестановки их компонент.

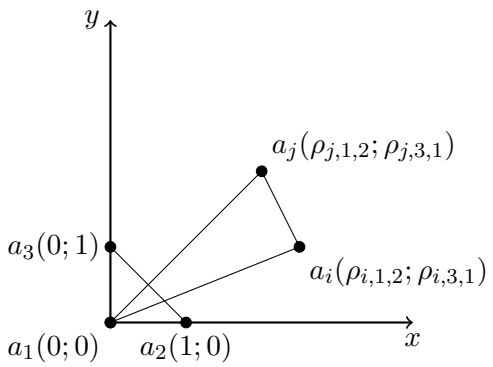


Рис. 5. Случай $P = \{1\}$.

$\rho^*_{1,j,k} - \rho^*_{1,i,k}$, что, как показано в предыдущем случае равно $\rho_{1,i,j} + \rho_{1,j,k} - \rho_{1,i,k} = \rho_{i,j,k}$, что и требовалось доказать.

Возвращаясь к кодам, не учитывающим ориентацию ([4])

Определение 2. Набор чисел $s_{\alpha,\beta} \in \{\pm 1\}$, $\alpha, \beta \in \mathcal{A}$ будем называть *расстановкой знаков*. Расстановку знаков будем назвать согласованной, если для любых $\alpha, \beta, \gamma \in \mathcal{A}$ выполнены равенства

1) $s_{\alpha\beta} \cdot s_{\beta\gamma} = s_{\alpha\gamma}$;

2) $s_{\pi(\alpha)\sigma(\beta)} = (-1)^\pi \cdot (-1)^\sigma \cdot s_{\alpha\beta}$, $\pi, \sigma \in S_3$.

Замечание 4. Для согласованной расстановки знаков, очевидно выполняются равенства $s_{\alpha\alpha} = 1$ и $s_{\alpha\beta} = s_{\beta\alpha}$ при любых $\alpha, \beta \in \mathcal{A}$.

Следствие 3. *Набор чисел $r^*_{\alpha\beta}$ является кодом невырожденного изображения тогда и только тогда, когда существует согласованная расстановка знаков $s_{\alpha,\beta}$, такая, что для $r_{\alpha\beta} = s_{\alpha\beta} \cdot r^*_{\alpha\beta}$ выполнены условия 1–5 и (1).*

4. Заключение

Полученные результаты полностью описывают множество возможных кодов невырожденных изображений. В дальнейшем планируется построение аналогичных условий для других кодирующих функций, например, сохраняющих проективную эквивалентность.

Список литературы

[1] Агниашвили П.Г., "Однозначность восстановления изображения по его коду в n-мерном случае", *Интеллектуальные системы*, **15**:1–4 (2011), 293–332.

[2] Алексеев Д.В., "К вопросу о восстановлении трехмерного тела по его плоским проекциям", *Интеллектуальные системы. Теория и приложения*, **21**:4 (2017), 66–85.

[3] Козлов В.Н., "Доказательность и эвристика при распознавании визуальных образов", *Интеллектуальные системы*, **14**:1–4 (2010), 35–52.

[4] Козлов В.Н., *Элементы математической теории зрительного восприятия*, Изд–во ЦПИ при мех.–мат. ф–те МГУ, Москва, 2001, 128 с.

[5] Козлов В.Н., "О кодировании дискретных фигур", *Дискретная математика*, **8**:6 (1996), 57–61.

[6] Kozlov V.N., "Image Coding and Recognition and Some Problems of Stereovision", *Pattern Recognition and Image Analysis*, **7**:4 (1997), 448–466.

Necessary and sufficient conditions for the existence of an image with a given code

Alekseev D.V.

The article introduces an image encoding function which is invariant with respect to affine transform. The properties of the encoding funciton are investigated. Necessary and sufficient conditions are found for a given set of numbers to be a code of nonsingulari image.

Keywords: image code, image encoding, affine equivalence.

О самоорганизующейся системе светофоров, обеспечивающих бесперебойное движение транспорта

Муравьев А.К.[1]

В работе рассматривается кооперативная игра о доставке транспорта на прямоугольной решетке. Вводятся понятия затора и столкновения автомобилей на ней, а так же автоматов как агентов в светофорах решетки, регулирующих движение.

Показано существование структуры автоматов с локальной областью видимости, приводящей к самоорганизации системы светофоров, обеспечивающих сколь угодно длительную корректную доставку автомобилей без заторов и столкновений для любого количества выходов решетки.

Ключевые слова: кооперативная игра, доставка, организация движения транспорта, агент, однородная структура, автомат.

1. Введение

Рассматривается следующая кооперативная игра. Дана прямоугольная сетка с m столбцами и n строками (Рис. 1). Назовем ее *решеткой* или *полем*.

Ребра решетки — *дороги*, по которым будут двигаться автомобили. Каждое ребро вмещает лишь один автомобиль.

На левой границе поля — $n+1$ *вход*, куда будут заезжать автомобили. Присвоим им номера от 0 до n с порядком нумерации снизу вверх.

Игра происходит дискретно. В каждый такт времени случайно выбирается u входов без повторений (из равномерного распределения на всех

[1] *Муравьев Артем Константинович* — эксперт Департамента Анализа Данных и Моделирования, "Банк ВТБ"(ПАО), e-mail: muravev.a.k@yandex.ru

Muravev Artem Konstantinovich — expert, Department of Data Analysis and Modelling, VTB Bank

входах), и в них заезжает по 1 автомобилю. За один такт времени автомобиль перемещается с ребра, на котором находится, на какое-то другое, имеющее с первым общую вершину.

Перед началом игры выбирается $u < (n+1)$ узлов на правой границе поля, и объявляются *выходами*. Автомобиль, приезжающий в выходной узел, исчезает, и считается доставленным.

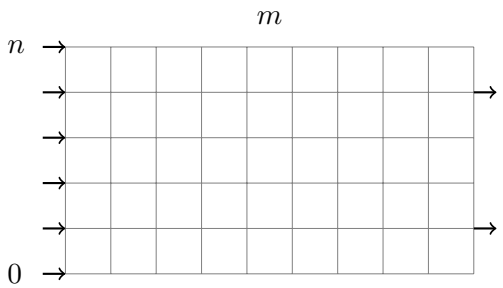


Рис. 1

В каждом узле поля установим *светофор*, регулирующий движение автомобилей через этот узел. По каждой из горизонтальных дорог движение организовано исключительно от входов к выходам (слева направо), а по вертикальным — либо вверх, либо вниз. Соответственно, светофор в каждый такт времени выбирает одно из входящих в него направлений (сверху, снизу или слева), переключает свой сигнал в какое-то выходное направление (вверх, вниз или вправо), и пропускает автомобиль туда.

При этом допустим, что светофорам на i-й снизу горизонтальной полосе на переключение необходимо $\frac{i}{n+2}$ времени. Такое допущение позволит разрешать конфликты, связанные с порядком переключения светофоров, и тем самым получить лавинообразную смену логики пропуска, в которой нижние светофоры будут «ведущими», а те, что сверху от них — «ведомыми» . При этом каждый светофор все же будет переключаться за один такт игры. Подобный подход использовался в [1, с. 152] для разрешения спорных ситуаций при параллельном обходе информационного графа несколькими вычислителями в модели параллельных алгоритмов поиска.

Если на выбранном выходном направлении на близжайшем к светофору ребре уже стоит автомобиль, и его не пропускает следующий светофор, то и текущий светофор не может пропустить автомобиль в выбранном направлении, и автомобиль стоит на нем в ожидании пропуска.

Столкновением назовем ситуацию, когда два светофора, находящиеся на одном и том же ребре решетки, выпустили автомобили на это ребро с противоположных сторон (Рис. 2).

Задержкой автомобиля на светофоре назовем ситуацию, когда на светофор с нескольких сторон одновременно приехали автомобили, и он их направляет в одну и ту же сторону, в следствие чего один из автомобилей проезжает на текущем такте времени, заставляя другой в течение этого такта стоять на месте и ждать (Рис. 3). При этом будем считать, что задержек не возникает, если автомобили одновременно приезжают на светофор, и светофор их направляет по разным выходным направлениям (Рис. 4, 5). Аналогом этому в реальной жизни будет организация движения на нерегулируемых перекрестках, где при проезде одновременно двух автомобилей из разных въездов в разные выезды они одновременно выезжают на перекресток, одна из них пропускает другую, и обе завершают маневр практически одновременно.

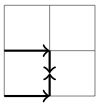


Рис. 2

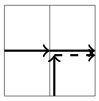


Рис. 3

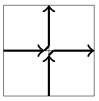


Рис. 4

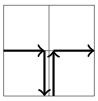


Рис. 5

Затором назовем ситуацию, когда входной светофор, на который на данном такте заезжает автомобиль, не может пропустить его в выбранное им направлении из-за задержки автомобиля на нем.

Цель игры — организовать доставку автомобилей на выходы таким образом, чтобы при любой длительности игры не создавалось заторов и столкновений. При этом игроками считаем светофоры, регулирующие проезд автомобилей через соответствующие вершины, а m — длину поля — считаем параметром игры, который выбираем изначально в зависимости от n, u.

Автор выражает благодарность профессору Э. Э. Гасанову за постановку задачи и помощь в работе.

2. Постановка задачи

Будем считать, что в светофорах находятся *автоматы*, имеющие конечное число состояний, каждое из которых определяет какое-либо поведение по пропуску автомобилей через него. Каждый автомат принимает на вход ситуацию по въезжающим на него и выезжающим с него авто-

мобилям, а так же состояния и аналогичные показатели со светофоров в некоторой его окрестности.

Определим *типы светофоров* на поле:

(1-4) Не угловые светофоры, находящиеся на соответствующей границе поля: верхней, нижней, левой или правой;

(5-8) Угловые светофоры (по одному типу на каждый угол);

(9) Внутренние светофоры - не попавшие в предыдущие типы.

И потребуем, чтобы система автоматов в светофорах была *однородной* внутри каждого из типов.

Требуется определить структуру автоматов (а именно - структуру автомата для каждого из типов) таким образом, чтобы в процессе игры описанная система светофоров самоорганизовалась на обеспечение доставки автомобилей со случайных входов на определенные выходы, не известные до начала игры, без заторов и столкновений.

3. Описание решения

3.1. Игра с одним выходом

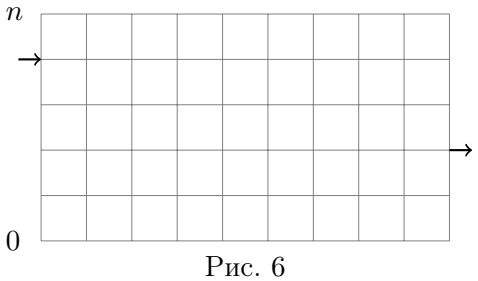


Рис. 6

Рассмотрим вариант игры, когда $u = 1$ (Рис. 6). То есть, на каждом такте времени в какой-то один случайный вход (выбранный из равномерного распределения на всех входах) заезжает автомобиль, а задача агентов на светофорах - доставить их все на выход, случайно выбранный до начала игры, и не известный изначально, не создавая при этом заторов и не сталкивая автомобили друг с другом.

Определение 1. Статичная стратегия игры - *это стратегия (структура автоматов в светофорах), при которой в процессе обучения каждый из светофоров переходит в одно единственное финальное состояние, в котором и остается в дальнейшем.*

Определение 2. Динамичная стратегия игры - *это стратегия, не являющаяся статичной.*

Теорема 1. *Для $m \geq 1$ и любых n существует статичная стратегия игры всех игроков, приводящая к сколь угодно длительной доставке автомобилей со случайных входов на определенный выход сети без заторов и столкновений.*

Доказательство. Рассмотрим конфигурацию светофоров, изображенную на Рис. 7. Потребуем на каждом из светофоров на правой границе поля установить приоритет при проезде для автомобилей, подъезжающих по горизонтальным дорогам, нежели тем, что уже едут по правой границе. Полученную конфигурацию будем обозначать как $K_{n,m}$.

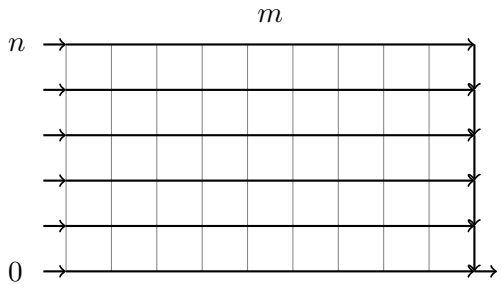


Рис. 7: $K_{n,m}$

Очевидно, что такая конфигурация обеспечивает доставку на выход в углу решетки без столкновений. Покажем, что она также обеспечивает отсутствие заторов. Для этого последовательно докажем несколько предложений.

Предложение 1. *Конфигурация $K_{1,1}$ обеспечивает доставку автомобилей в правый нижний угол на поле 1×1 без заторов.*

Доказательство. Для удобства введем обозначения светофоров (S_i) и дорог (a_i^j) на поле, как показано на Рис. 8.

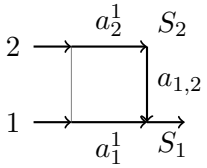


Рис. 8

Учитывая, что на S_1 приоритет при проезде имеют автомобили, подъезжающие со стороны дороги a_1^1, то на этом светофоре задержка может возникнуть только со стороны дороги $a_{1,2}$. Тогда на входе 1 тут заторов быть не может, поэтому для затора при выбранной конфигурации необходима задержка на светофоре S_2 со стороны a_2^1.

Очевидно, что из-за вышеупомянутого приоритета на светофоре S_1 автомобиль, заехавший на такте t на вход 1 (на ребро a_1^1), обязательно покинет это ребро на такте $t+1$.

Теперь рассмотрим случай, когда на такте t автомобиль заехал во вход 2 (на ребро a_2^1). Получается, что в момент t в первый вход автомобиль не заезжал, и на ребре a_1^1 на такте t пусто, потому как даже если туда и заезжал автомобиль ранее, он уже проехал в силу его приоритета на светофоре S_1. Из-за этого на такте $t+1$ автомобиль, возможно бывший на ребре $a_{1,2}$, проехал на выход, и в любом случае ребро $a_{1,2}$ освобождается, и туда проезжает автомобиль с ребра a_2^1.

Тем самым, получаем, что автомобиль, заехавший на такте t на вход 2 (на ребро a_2^1), точно покинет это ребро на такте $t+1$.

Итого получаем, что для каждого из светофоров $S_i, i = 1, 2$, справедливо утверждение, что на S_i не возникнет задержек со стороны a_i^1. То есть, не возникнет заторов на входах 1, 2.

□

Предложение 2. *Конфигурация $K_{n,1}$ обеспечивает доставку автомобилей в правый нижний угол на поле $n \times 1$ без заторов.*

Доказательство. Введем обозначения, аналогичные предыдущим (Рис. 9).

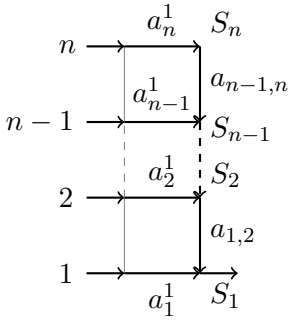


Рис. 9

Будем доказывать предположение индукцией по n.

- *База:* $n = 1$ — очевидно; $n = 2$ — по Предл. 1.

- *Индуктивный переход:* Пусть утверждение выполнено для $n - 1$. Докажем его для n.

  Добавление дополнительного входа не влияет на ситуацию на ребрах $a_i^1, i = 1 \dots n-1$, так как у автомобилей на этих дорогах на светофорах $S_i, i = 1 \dots n-1$, приоритет. При этом для $n-1$ по предположению индукции на S_i не может быть задержек со стороны дорог a_i^1. Поэтому и для n на светофорах $S_i, i = 1 \dots n-1$, со стороны дорог a_i^1 задержек быть не может, соответственно, не может быть заторов на входах $1 \dots n-1$.

  Покажем, что на S_n также не может быть задержки со стороны дороги a_n^1.

  Пусть на такте t автомобиль заехал во вход n (на ребро a_n^1). Тогда что в момент t ни в какой из входов $1 \dots n-1$ автомобили не заезжали, и на ребрах $a_i^1, i = 1 \dots n-1$, на такте t пусто, потому как даже если туда и заезжали автомобили ранее, они уже проехали в силу отсутствия задержек на светофорах $S_i, i = 1 \dots n-1$, со стороны дорог $a_i^1, i = 1 \dots n-1$. Из-за этого на такте $t+1$ цепочка автомобилей, возможно бывших на ребрах $a_{i,i+1}, i = 1 \dots n-1$, проезжает вперед к выходу, и в любом случае ребро $a_{n-1,n}$ освобождается, и туда проезжает автомобиль с ребра a_n^1.

  Тем самым, получаем, что автомобиль, заехавший на такте t на вход n (на ребро a_n^1), точно покинет это ребро на такте $t+1$. То есть, задержек на светофоре S_n не возникнет, соответственно, не возникнет и заторов на входе n.

□

Предложение 3. *Конфигурация $K_{n,m}$ обеспечивает доставку автомобилей в правый нижний угол на поле $n \times m$ без заторов.*

Доказательство. Снова вводим соответствующие обозначения (Рис. 10).

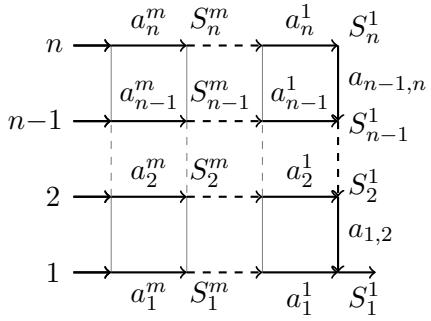


Рис. 10

Докажем предложение индукцией по m.

- *База:* $m = 1$ — по Предл. 2.
- *Индуктивный переход:* Пусть выполнено для $m - 1$. Докажем для m.

  Для $m - 1$ по предположению индукции не возникнет заторов на светофорах $S_i^m, i = 1 \dots n$, поэтому для m на тех же самых светофорах не возникнет задержек со стороны дорог $a_i^m, i = 1 \dots n$. Поэтому заторов на входах $1 \dots n$ так же не возникнет.

□

Таким образом, получили, что конфигурация $K_{n,m}$ обеспечивает требуемую доставку на поле $n \times m$. Однако, лишь в правый нижний угол.

Пусть теперь выход на l-й горизонтальной дороге, $l > 0$. В таком случае рассмотрим модернизацию конфигурации $K_{n,m}$, изображенную на Рис. 11:

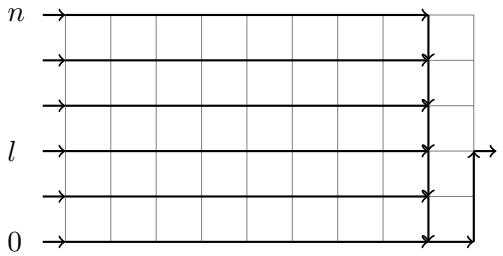


Рис. 11

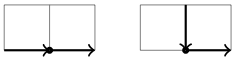
В таком случае так же выполнены все необходимые условия.
Заметим наконец, что здесь $m \geq 1$. □

Покажем теперь, что можно организовать обучение агентов (светофоров) показанной стратегии в процессе игры с заранее неизвестными параметрами.

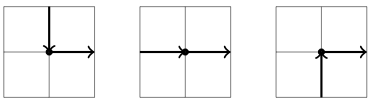
Теорема 2. *Существует алгоритм обучения светофоров (то есть конфигурация автоматов) с локальным полем видимости радиуса 1, приводящий к организации требуемой доставки.*

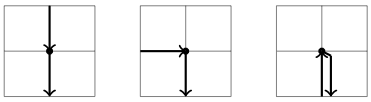
Доказательство. Определим состояния автоматов для разных типов светофоров:

1. Светофоры на горизонтальной границе поля (тип HE):

   (1) 

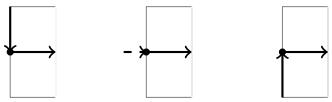
2. Внутренние светофоры (тип $CORE$):

   (1) 

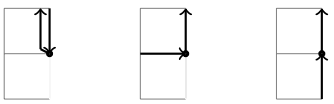
   (2) 

   Здесь так же определим, что на каждом из светофоров приоритет при проезде имеет автомобиль, приближающийся слева.

3. Светофоры на вертикальной границе со входами (тип $VEIN$):

   (1) 

4. Светофоры на вертикальной границе с выходами (тип $VEOUT$):

   (1) 

   Здесь так же определим, что приоритет при проезде будет иметь автомобиль, приближающийся слева.
   Светофоры на углах снизу и сверху будут повторять второй патерн поведения (1), и его же в зеркальном (относительно горизонтали) виде соответственно.

Опишем теперь структуру автоматов в светофорах.

Для светофоров типов HE, $VEIN$, $VEOUT$ состояние всего одно, изначально его и фиксируем.

Опишем автомат для типа $CORE$ (Рис. 12). Петли на состояниях будем опускать, подразумевая, что все входы, не описанные в триггерах, переводят состояние в себя же.

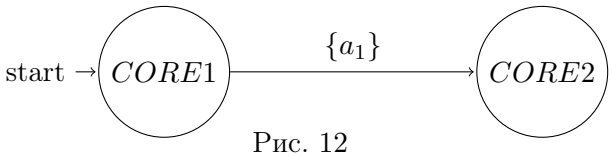
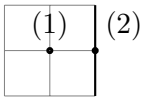


Рис. 12

Опишем указанный на схеме триггер перехода a_1.



$$(1) - CORE1, (2) - VEOUT \Rightarrow$$

$$a_1 : (1) : CORE1 \to CORE2$$

□

3.2. Игра с двумя выходами

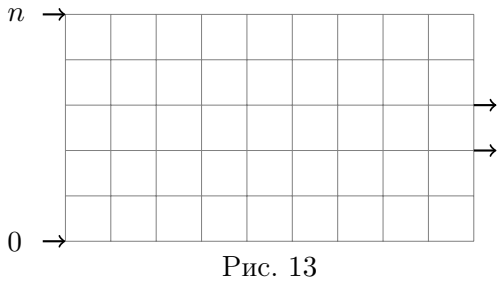


Рис. 13

Теперь рассмотрим вариант игры, когда $u = 2$ (Рис. 13). То есть, на каждом такте времени в два случайных входа (выбранных без повторений из равномерного распределения на всех входах) заезжает по автомобилю, а задача агентов на светофорах - доставить их все в два выхода, случайно выбранные до начала игры, и не известные изначально, не создавая при этом заторов и не сталкивая автомобили друг с другом.

Теорема 3. *Для $m \geq n+1$ и любых n существует динамичная стратегия игры всех игроков, приводящая к сколь угодно длительной доставке автомобилей с двух случайных входов на два определенных выхода сети без заторов и столкновений.*

Доказательство. Опишем конфигурацию, гарантирующую сколь угодно длительный поток доставки автомобилей из двух случайных (на каждом ходу) входов до двух случайных (выбранных один раз до начала игры) выходов без заторов и столкновений (Рис. 14).

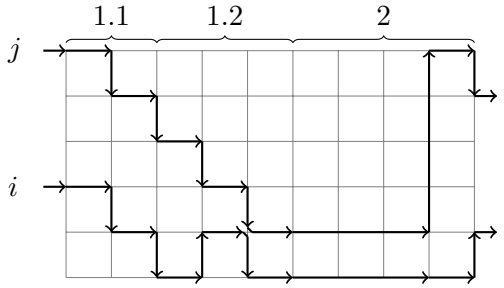


Рис. 14

Назовем вертикальные полосы поля слоями.

Доставку разделим на два этапа:

1. доставка по слоям с 1 по n;
2. доставка с n-го слоя на выходы.

Разделим этап 1 дополнительно на 2 подэтапа:

Этап 1.1: Организуем передвижение автомобилей сквозь слои 1...n параллельно, чтобы автомобили, заехавшие в одно время, на каждом из тактов времени с 1 по n находились на одном слое, и в одном положении (на вертикальной дорожке, или на горизонтальной). При этом двигаем их «лесенкой», смещая к нижней границе поля.

Очевидно, что нижний автомобиль достигнет нижней границы раньше. Как только это происходит - начинается

Этап 1.2: нижний автомобиль начинает «ожидать» верхнего у нижней границы, сохраняя при этом скорость движения сквозь слои (и очередность проезда дорог: то горизонтальная, то вертикальная). Двигается он в это время по следующей траектории (Рис. 15):

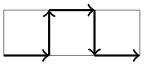


Рис. 15

Так происходит до тех пор, пока верхний автомобиль, двигаясь в это время по все той же «лесенке» , не приблизится к нижнему на расстояние в 2, либо 1 ячейку (если $(j-i)$-четно, то они сблизятся на расстояние в 2 ячейки, а дальше сближать нельзя, так как они столкнутся на светофоре между ними; если $(j-i)$-нечетно (как на Рис. 13), то они сблизятся на расстояние в 1 ячейку). Как только это произошло - верхний автомобиль начинает двигаться параллельно нижнему (повторяя его траекторию), пока они вместе не достигнут n-го слоя.

На n-й слой автомобили выезжают параллельно друг другу по горизонтальным дорогам, расположенным около нижней границы поля, так что:

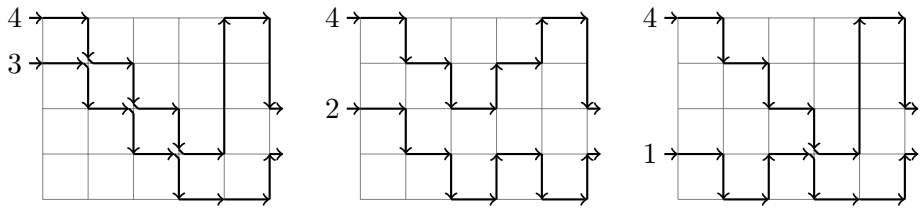
- Нижний автомобиль выезжает либо по нижней граничной дороге, либо по соседней к ней;

- верхний автомобиль выезжает по дороге, параллельной той, с которой выезжает нижний, и отстоящей от нее вверх на 1 либо 2 ячейки.

Этап 2: Перемещаем автомобили по горизонтальным дорогам прямо до предпоследнего вертикального ряда светофоров, на нем разводим их в противоположные вертикальные направления, доводя до противоположных границ поля, и доставляя через углы решетки по дальней границе до нужных выходов.

Получаем, что при таком раскладе всегда автомобиль, заехавший в верхний вход, едет на верхний выход, а тот, что заехал в нижний вход - отправляется в нижний выход. Тем самым, потом автомобилей на верхние входы равен потоку автомобилей на верхний выход, и аналогично с нижними входами и выходом.

Приведем примеры описанной выше стратегии для $n = 4, m = 5$, выходов с индексами 1 и 2, и всевозможными парами входов (Рис. 16):



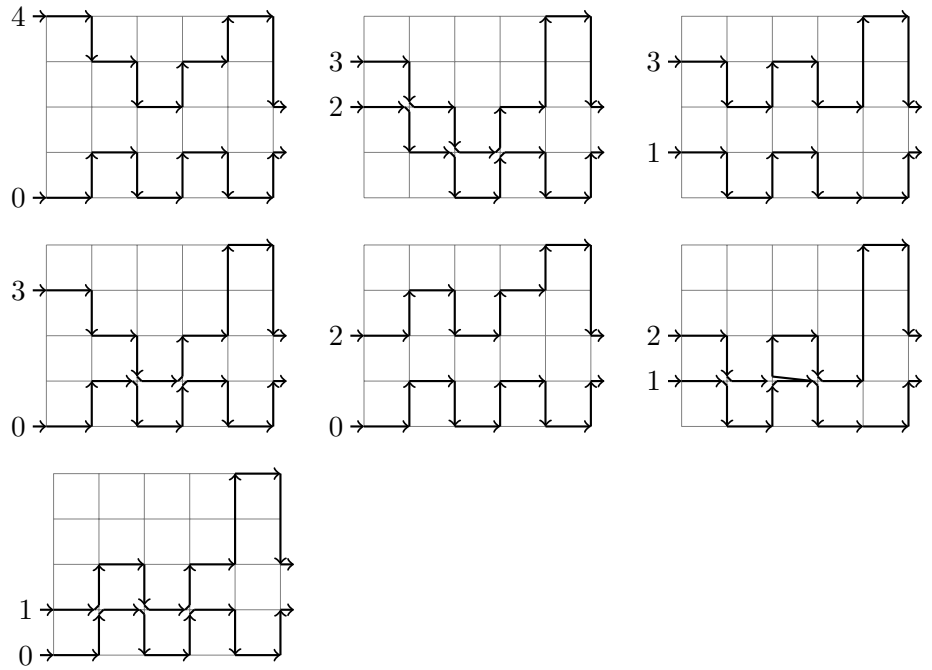


Рис. 16

Покажем теперь, что данная стратегия верна.

Очевидно, что автомобили с разных входов при такой стратегии не встретятся друг с другом. Поэтому достаточно доказать, что не будет образовываться заторов и столкновений из-за автомобилей, заехавших в разные такты времени.

На Этапе 1 - автомобили движутся с постоянной скоростью по слоям, поэтому:

1. автомобили, въехавшие друг за другом с разницей в 1 такт, будут ехать по разным направлениям (одни - вертикально, другие - горизонтально), поэтому заторов и столкновений не возникнет;

2. автомобили, въехавшие с разницей в $\geq$ 2 такта, будут ехать на разных слоях, поэтому встретиться они не могут, следовательно, заторов и столкновений так же не возникнет.

На Этапе 2 - доставка к каждому из выходов превращается в аналог доставки с одним выходом, описанной выше. Там для нее и показана корректность.

Осталось показать ограничения на длину поля m. Для первого этапа, очевидно, необходима длина $m_1 = n$ (так как в худшем случае, когда верхний автомобиль едет с последнего входа, для сдвига его на нижнюю около-граничную дорогу необходимо ровно n слоев). Для второго этапа - справедливо ограничение, показанное выше для игры с одним выходом, то есть $m_2 \geq 1$. В итоге получаем

$$m = m_1 + m_2 \geq n + 1.$$

□

Покажем теперь, что можно организовать обучение агентов (светофоров) показанной стратегии в процессе игры с заранее неизвестными параметрами.

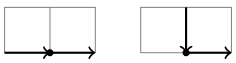
Теорема 4. *Существует алгоритм обучения светофоров (то есть конфигурация автоматов) с локальным полем видимости радиуса* 2*, приводящий к организации требуемой доставки.*

Доказательство. Определим состояния автоматов для разных типов светофоров:

1. Светофоры на горизонтальной границе поля (тип HE):

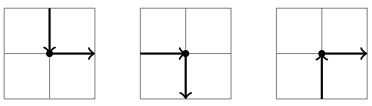
   (1) 

   Светофоры границы на 1 этапе.

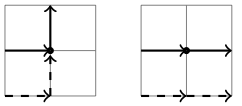
   (2) 

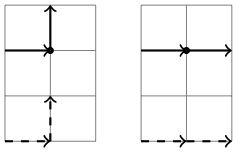
   Светофоры границы на этапе 2.

2. Внутренние светофоры (тип $CORE$):

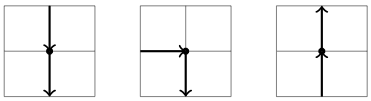
   (1) 

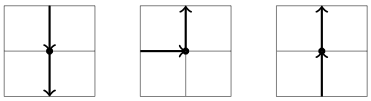
   Начальное состояние внутреннего светофора с логикой «сближения».

(2)  , иначе (1)

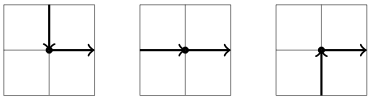
(3)  , иначе (1)

Здесь (2) / (3) обозначают поведение светофора, который повторяет за светофором ниже него на 1 / 2 ячейки, если на этот светофор автомобиль приехал одновременно и параллельно тому, что приехал на текущий.

(4) 

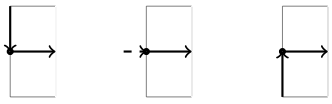
(5) 

(4) / (5) - это состояния светофоров на предпоследней вертикальной границе.

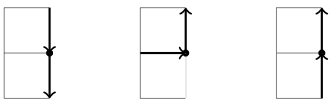
(6) 

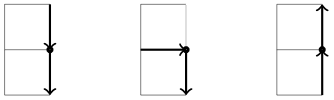
(6) - состояние внутреннего светофора на этапе 2, где автомобили едут параллельно друг другу.

3. Светофоры на вертикальной границе со входами (тип $VEIN$):

(1) 

4. Светофоры на вертикальной границе с выходами (тип $VEOUT$):

(1) 

(2) 

Оба состояния эквивалентны с точки зрения нашей стратегии доставки.

Светофоры на углах снизу и сверху будут повторять второй паттерн поведения (1) или (2) соответственно.

Опишем теперь структуру автоматов в светофорах.

Для светофоров типа $VEIN$ состояние всего одно, посему тут нечего обучать, изначально фиксируем его.

Для светофоров типа $VEOUT$ расставим изначально случайно либо (1), либо (2), кроме угловых, про которые описано выше.

Опишем автоматы для типов HE (Рис. 17) и $CORE$ (Рис. 18). Петли на состояниях будем опускать, подразумевая, что все входы, не описанные в триггерах, переводят состояние в себя же.

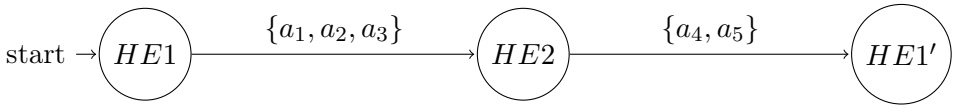


Рис. 17

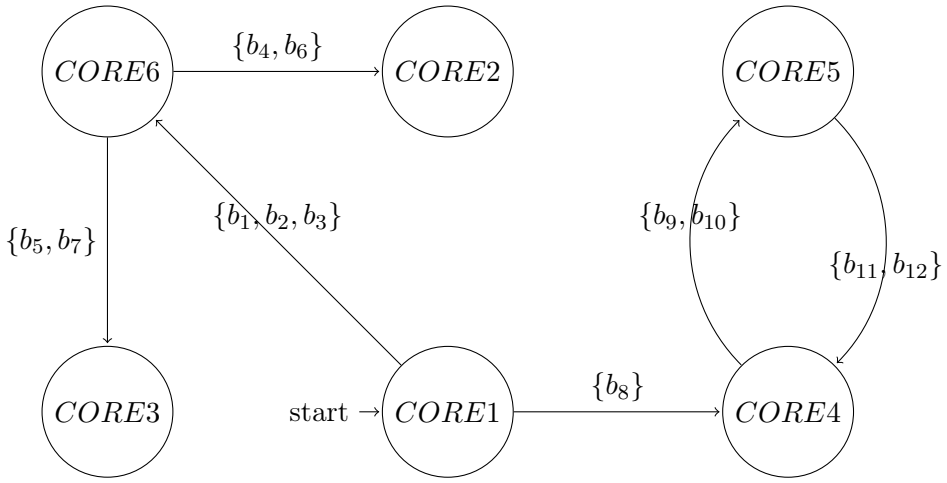
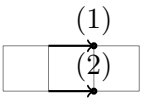


Рис. 18

$HE1'$ здесь - состояние автомата, в котором поведение агента совпадает с поведением $HE1$, но состояние отличается от изначального.

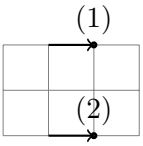
Опишем указанные на схемах триггеры переходов. Толстые линии будут обозначать участки дорог, на которых в рассматриваемый момент времени стоят автомобили.

1. $\{a_1\}$



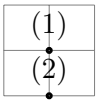
$(1) - CORE1, (2) - HE1 \Rightarrow$
$a_1 : (2) : HE1 \to HE2$

2. $\{a_2\}$



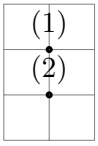
$(1) - CORE1, (2) - HE1 \Rightarrow$
$a_2 : (2) : HE1 \to HE2$

3. $\{b_1\}$



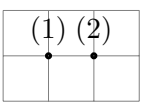
$(1) - CORE1, (2) - HE2 \Rightarrow$
$b_1 : (1) : CORE1 \to CORE6$

4. $\{b_2\}$



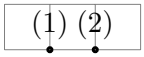
$(1) - CORE1, (2) - CORE6 \Rightarrow$
$b_2 : (1) : CORE1 \to CORE6$

5. $\{b_3\}$



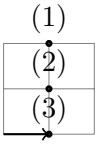
$(1) - CORE6, (2) - CORE1 \Rightarrow$
$b_3 : (2) : CORE1 \to CORE6$

6. $\{a_3\}$



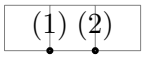
$(1) - HE2, (2) - HE1 \Rightarrow$
$a_3 : (2) : HE1 \to HE2$

7. $\{a_4, b_4, b_5\}$



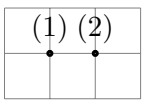
$(1) - CORE6, (2) - CORE6, (3) - HE2 \Rightarrow$
$b_4 : (1) : CORE6 \to CORE3$
$b_5 : (2) : CORE6 \to CORE2$
$a_4 : (3) : HE2 \to HE1'$
Здесь рассматривается случай, когда на нижний светофор автомобиль приехал, а ни на один из верхних в радиусе 2 не приехало ничего.

8. $\{a_5\}$



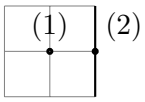
$(1) - HE2, (2) - HE1' \Rightarrow$
$a_5 : (1) : HE2 \to HE1'$

9. $\{b_6, b_7\}$



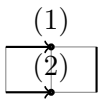
$(1) - CORE6, (2) - CORE2 \Rightarrow$
$b_6 : (1) : CORE6 \to CORE2;$
$(1) - CORE6, (2) - CORE3 \Rightarrow$
$b_7 : (1) : CORE6 \to CORE3$

10. $\{b_8\}$



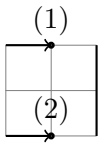
$(1) - CORE1, (2) - VEOUT \Rightarrow$
$b_8 : (1) : CORE1 \to CORE4$

11. $\{b_9, b_{11}\}$



$(1) - CORE4$, на (2) так же заезжает автомобиль $\Rightarrow$
$b_9 : (1) : CORE4 \to CORE5$;
$(2) - CORE5$, на (1) так же заезжает автомобиль $\Rightarrow$
$b_{11} : (2) : CORE5 \to CORE4$

12. $\{b_{10}, b_{12}\}$



$(1) - CORE4$, на (2) так же заезжает автомобиль $\Rightarrow$
$b_{10} : (1) : CORE4 \to CORE5$
$(2) - CORE5$, на (1) так же заезжает автомобиль $\Rightarrow$
$b_{12} : (2) : CORE5 \to CORE4$

В итоге, после первого проезда пары, где один из автомобилей идет из самого дальнего (левого верхнего) входа, получим, что у все светофоры типа HE на нижней границе до n-го проезда находятся в состояниях $HE1/HE1'$, а все после - в состоянии $HE2$. Внутренние светофоры до n-го слоя, удаленные от нижней границы на 1, будут все в состоянии $CORE2$, а те, что удалены на 2 ячейки - в состоянии $CORE3$. Остальные внутренние светофоры останутся в состоянии $CORE1$. В силу определения правил проезда для вышеупомянутых состояний, получим обученный первый этап.

Все внутренние светофоры после n-го проезда и до предпоследнего будут в состоянии $CORE6$, а те, что на предпоследнем слое, будут адаптироваться под пары автомобилей, приезжающие с первого этапа. Тем самым, и в силу определения правил проезда для состояний, получим обученный второй этап. $\square$

3.3. Игра с произвольным числом выходов

Наконец, рассмотрим вариант игры, когда $u > 2$.

Теорема 5. *Для $m \geq n - u + 1$ и любых n и u существует динамичная стратегия игры всех игроков, приводящая к сколь угодно длительной доставке автомобилей с и случайных входов на и определенных выходов сети без заторов и столкновений.*

Доказательство. Обобщим стратегию, разработанную для двух входов, на общий случай. Примеры: Рис. 19 ($n = 6, m = 10, u = 3$), Рис. 20 ($n = 8, m = 11, u = 4$).

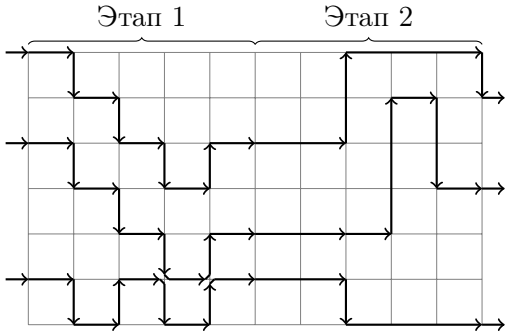


Рис. 19

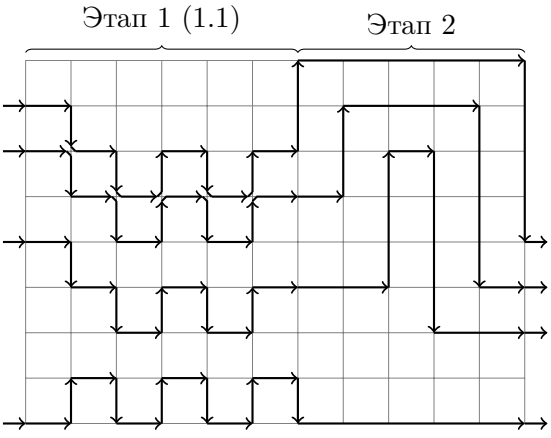


Рис. 20

На первом этапе будем вести автомобиль с нижнего входа так же, как и в прошлый раз, прижав к нижней границе, но не до n-го слоя, а до $n - u + 2$-го.

Второй снизу ведем так же, как вели верхний в случае двух входов, - «прижимаем» к нижнему на 2, либо 1 ячейку, и ведем параллельно.

И так далее, каждый следующий по порядку с низу автомобиль ведем, «прижимая» к предыдущему на 2, либо 1 ячейку.

Для корректного сближения в таком случае нам понадобится $(n - u + 2)$ слоя. Данную границу устанавливаем как необходимое количество слоев для «прижатия» всех автомобилей друг к другу снизу поля (в ситуации, когда самый дальний автомобиль заезжает в n-й вход, а расстояния между каждой парой соседних автомобилей после «прижатия» равно 1, дальний автомобиль нужно будет доставить на $(u - 1)$-ю горизонтальную дорогу, для чего нужно $(n - u + 1)$ вертикальный сдвиг, и 1 слой оставляем на выезд на горизонтальную дорогу в конце).

Имеем $m_1 \geq n - u + 2$.

Далее, на первой части Этапа 2, как и в прошлый раз, отправляем автомобили прямо в сторону выходов, параллельно друг другу, до $(m - 2u + 3)$ слоя (см. ниже).

Пусть номера выходов в порядке возрастания: $j_1 ... j_u$.

На Этапе 2:

1. Отправляем первый снизу автомобиль в нижний выход строго по нижней границ;

2. Все остальные въехавшие автомобили переводим на вертикальные параллельные дороги с направлением вверх, сохраняя «очередность», то есть $(u - i)$-й снизу автомобиль поедет по i-й вертикальной дороге Этапа 2 вверх;

3. Затем разворачиваем дороги через горизонтальные проезды вниз, обеспечивая при этом сближение траекторий каждой пары соседних автомобилей до расстояния 1 между ними;

4. Затем ведем $u - i$-й автомобиль до i-го сверху входа, и по горизонтальным проездам выводим на выход. То есть, с ближайшей к концу поля вертикальной дороги автомобиль направляем на последний выход, со второй с конца - на предпоследний, и т.д.

Для такой схемы на втором этапе необходимое количество слоев:

$$m_2 \geq 2(u - 1) - 1 = 2u - 3.$$

Учитывая, что на Этап 1 по-прежнему необходимо количество слоев

$$m_1 \geq n - u + 2,$$

получаем итоговую оценку на длину поля, необходимую для реализации описанной схемы:

$$m = m_1 + m_2 \geq n - u + 2 + 2u - 3 = n + u - 1.$$

При такой стратегии так же получаем выполнение всех необходимых условий и разделение потоков с разных входов на разные выходы. □

Покажем теперь возможность обучения в данном случае.

Теорема 6. *Для $m >= n + u - 1$ существует алгоритм обучения светофоров (то есть конфигурация автоматов) с локальным полем видимости радиуса 2, приводящий к организации требуемой доставки.*

Доказательство. Использовать подход, в точности аналогичный случаю с 2 выходами в контексте обучения тут не получится, так как он основан на переходах $\{a_1, a_2\}$, где обнаруживается, с какого момента можно пускать автомобили напрямую через слои (то есть где заканчивается «сближение» автомобилей у нижней границы). Это обнаружение происходит при параллельном «достаточно близком» проезде двух автомобилей по нижней границе и рядом с ней, что в случае двух входов позволяет определить порог слоев «сближения». В случае же $u > 2$ входов такой порог должен быть установлен в момент, когда на одном слое проезжают параллельно u автомобилей, причем каждая пара соседних находится «достаточно близко» друг к другу. В общем случае такую ситуацию не получается отловить при радиусе видимости светофора $< n$.

Однако, если исключить из стратегии часть Этапа 2, где автомобили едут параллельно друг другу, то можно не отлавливать слой, с которого начинается отправка автомобилей по прямой после «сближения« (то есть конец Этапа 1), а вместо этого зафиксировать $2u - 3$ слоя для этапа 2, а все, что до них - отдать под поведение «сближения» (то есть по всем слоям 1 этапа автомобили будут двигаться «змейкой», как двигались при «сближении», просто после сближения их траетории будут параллельны траетории движения нижнего автомобиля, описанной выше в алгоритме доставки для $u = 2$). Пример: Рис. 21 ($n = 6, m = 10, u = 3$).

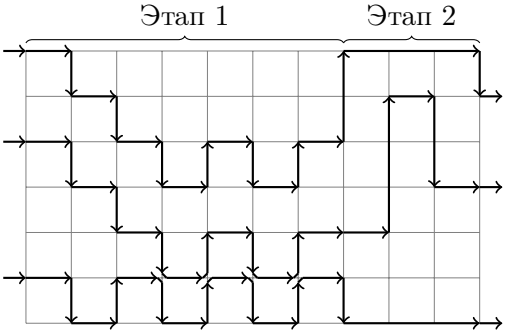


Рис. 21

Недостаток данной стратегии в том, что средняя скорость движения по слоям меньше, чем в предыдущей стратегии, в силу отсутствия того, что слои, по которым раньше мы пропускали автомобили прямо, теперь подчиняются логике «сближения», поэтому при использовании нынешней стратегии автомобили будут доставляться медленнее.

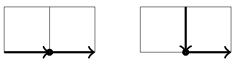
Однако, жертвуя скоростью доставки, мы получаем стратегию, для которой возможно обучение. Как раз его и осталось описать.

Начнем с описания состояний и структуры автоматов для разных типов светофоров:

1. Светофоры на горизонтальной границе поля (тип HE):

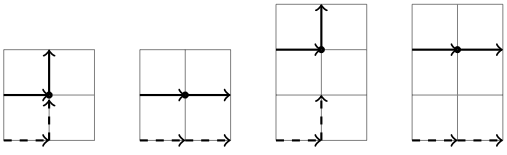
(1) 

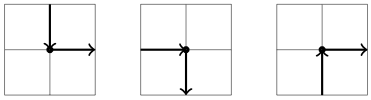
Это состояние для HE-светофоров на Этапе 1.

(m_i),
$i \in \{1, ...2u-3\}$ 

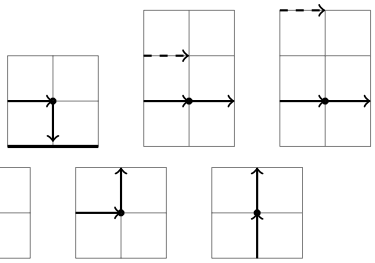
Это группа состояний светофоров второго этапа (послойно).

2. Внутренние светофоры (тип $CORE$):

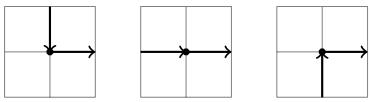
(1) 



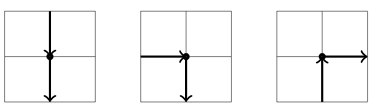
Здесь имеется в виду, что если на светофор приезжает автомобиль, и на соседний к нему снизу светотофор, либо на соседний через 1 ячейку, в то же время параллельно приезжает автомобиль, и его направляет либо вправо, либо вниз, то и текущий светофор поступает так же, направляя вправо или вниз соответственно. Во всех остальных случаях его срабатывают правила, описанные на нижних рисунках.
Это состояние $CORE$-светофоров на Этапе 1.

(m_i),
$i \in \{1, ...2u-3\}$ 

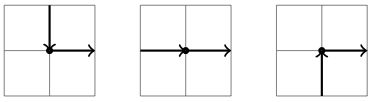
Это группа состояний светофоров второго этапа (послойно).
Первое правило означает, что если светофор находится рядом с нижней границе, то автомобили слева он направляет на границу.
Если на светофор приехал автомобиль, и параллельно ему приехал автомобиль на соседний сверху светофор, либо отстоящий на 2 ячейки наверх, то текущий светофор направляет машинку прямо.
Во всех остальны случаях он пользуется правилами, изображенными на нижних рисунках.

(f_i),
$i \in \{1, ...u-1\}$ 

Имеем группу состояний, описывающих одно и то же поведение, и использующихся для разной длины горизонтальных проездов к выходам после разворота на вертикали.

(r_i),
$i \in \{1, ...u-2\}$ 

Данная группа состояний, описывающих одно и то же поведения, нужна для организации поворотов параллельных траекторий движения автомобилей с горизонтального на вертикальное направление вниз, то есть перед проездом к выходам.

(l_i),
$i \in \{1, ...u-2\}$ 

Данная группа состояний, описывающих одно и то же поведения, нужна для организации поворотов параллельных траекторий движения автомобилей с вертикального направления на горизонтальное, влево, во время разворота с попарным «сближением» на расстояние 1.

3. Угловые светофоры при границе с выходами (тип ANG):

(1) 

Начальное состояние для нижнего угла.

(2) 

Начальное состояние для верхнего угла.

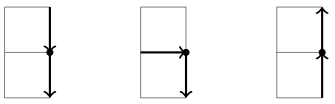
(f_1) 

Состояние, когда нижний угол является выходом.

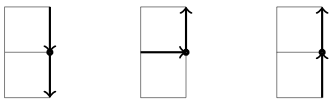
(f_u) 

Состояние, когда верхний угол является выходом.

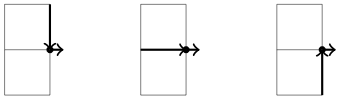
4. Светофоры на вертикальной границе с выходами (тип $VOUT$):

(1) 

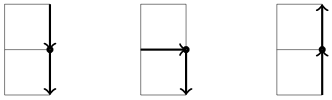
Изначальное состояние рассматриваемых светофоров.

(2) 

Состояния светофоров под нижним входом.

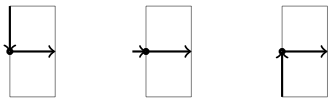
(f_i), $i \in \{0, ...u\}$ 

Группа состояний для обозначения различных выходов.

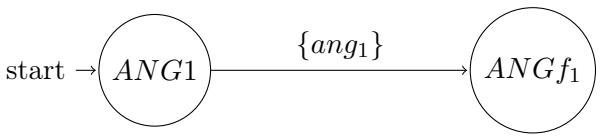
(f'_i), $i \in \{1, ...u\}$ 

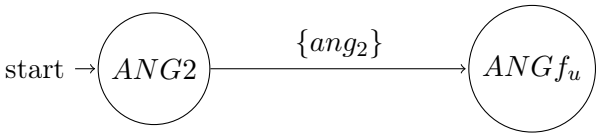
Группа состояний с поведением, аналогичным первому, обозначающие невыходные светофоры, с индексом, соответствующим ближайшему снизу выходному светофору.

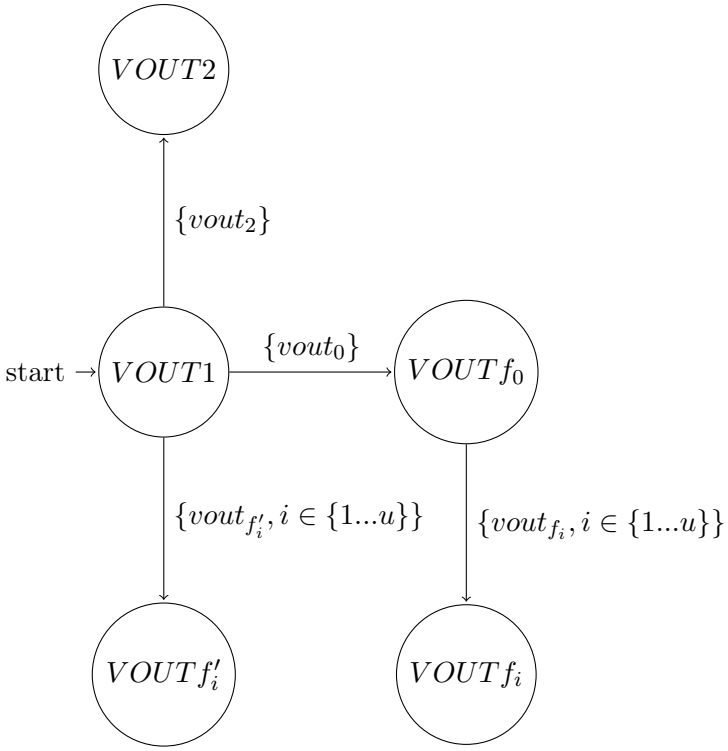
5. Светофоры на вертикальной границе со входами (тип $VEIN$):

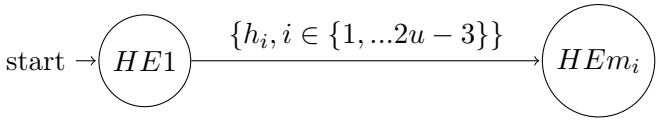
(1) 

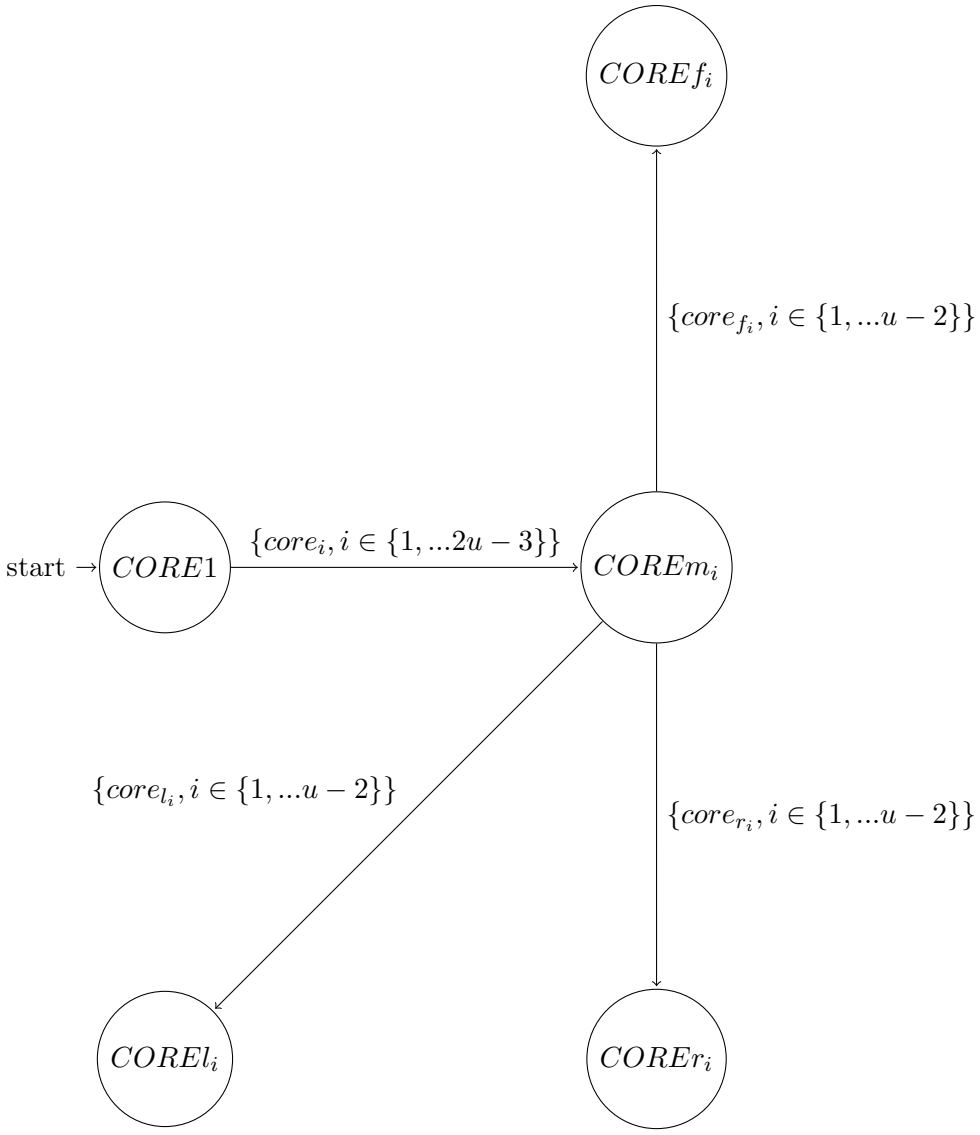
Структура автоматом (для всех типов кроме $VEIN$, у которых всего одно состояние):




start
$ANG2$
$\{ang_2\}$
$ANGf_u$


$VOUT2$
$\{vout_2\}$
start
$VOUT1$
$\{vout_0\}$
$VOUTf_0$
$\{vout_{f'_i}, i \in \{1...u\}\}$
$\{vout_{f_i}, i \in \{1...u\}\}$
$VOUTf'_i$
$VOUTf_i$

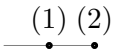

start
$HE1$
$\{h_i, i \in \{1, ...2u-3\}\}$
HEm_i



На рисунках группы состояний объединяем в одно, оставляя лишь общее обозначение (напр. $VOUTf_i$), однако подразумеваем, что к каждому из состояний группы идет по ребру.

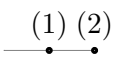
Приступим к описанию переходов:

1. $\{he_1\}$



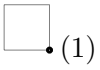
$(1) - HE1, (2) - ANG \Rightarrow$
$he_1 : (1) : HE1 \rightarrow HEm_1$

2. $\{he_i\}$



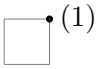
$(1) - HE1, (2) - HEm_i, i \in \{1, ...2u - 2\} \Rightarrow$
$he_{i+1} : (1) : HE1 \rightarrow HEm_{i+1}$

3. $\{ang_1\}$



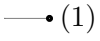
$(1) - ANG1, (1)$ - объявлен выходом $\Rightarrow$
$ang_1 : (1) : ANG1 \rightarrow ANGf_1$

4. $\{ang_2\}$



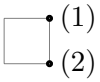
$(1) - ANG2, (1)$ - объявлен выходом $\Rightarrow$
$ang_2 : (1) : ANG2 \rightarrow ANGf_2$

5. $\{vout_0\}$



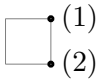
$(1) - VOUT1, (1)$ - объявлен выходом $\Rightarrow$
$vout_0 : (1) : VOUT1 \rightarrow VOUTf_0$

6. $\{vout_2\}$



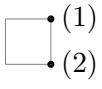
$(1) - VOUT1, (2) - ANG1$ или $VOUT2 \Rightarrow$
$vout_2 : (1) : VOUT1 \rightarrow VOUT2$

6. $\{vout_{f_1}\}$



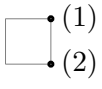
$(1) - VOUTf_0, (2) - VOUT2$ или $ANG1 \Rightarrow$
$vout_{f_1} : (1) : VOUTf_0 \to VOUTf_1$

7. $\{vout_{f_i'}\}$



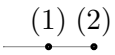
$(1) - VOUT1, (2) - VOUTf_i$ или $VOUTf_i', i \in \{1, ...u\} \Rightarrow$
$vout_{f_i'} : (1) : VOUT1 \to VOUTf_i'$

8. $\{vout_{f_i}\}$



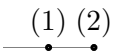
$(1) - VOUTf_0, (2) - VOUTf_i$ или $VOUTf_i', i \in \{1, ...u-1\} \Rightarrow$
$vout_{f_{i+1}} : (1) : VOUTf_0 \to VOUTf_{i+1}$

9. $\{core_i\}$



$(1) - CORE1, (2) - VOUT$ или $ANG \Rightarrow$
$core_1 : (1) : CORE1 \to COREm_1$
$(1) - CORE1, (2) - COREm_i, i \in \{1, ...2u-4\} \Rightarrow$
$core_{i+1} : (1) : CORE1 \to COREm_{i+1}$

10. $\{core_{f_i}\}$

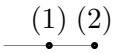


$(1) - COREm_1, (2) - VOUTf_i, i \in \{2, ...u-1\} \Rightarrow$
$core_{f_{u-i}} : (1) : COREm_1 \to COREf_{u-i}$
$(1) - COREm_k, (2) - COREf_i, i \in \{2, ...u-1\} \Rightarrow$
$core_{f_{i-1}} : (1) : COREm_k \to COREf_{i-1}$

11. $\{core_{r_i}\}$

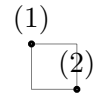
$(1) - COREm_1, (2) - ANG \Rightarrow$
$core_{r_1} : (1) : COREm_1 \to CORErr_1$
$(1) - COREm_k, (2) - CORErr_i, i \in \{1, ...u-3\} \Rightarrow$
$core_{r_{i+1}} : (1) : COREm_k \to CORErr_{i+1}$

12. $\{core_{l_{u-2}}\}$



$(1) - COREm_k, (2) - CORErr_{u-2} \Rightarrow$
$core_{l_{u-2}} : (1) : COREm_k \to COREl_{u-2}$

13. $\{core_{l_i}\}$



$(1) - COREm_k, (2) - COREl_i, i \in \{2, ...u-2\} \Rightarrow$
$core_{l_{i-1}} : (1) : COREm_k \to COREl_{i-1}$

В итоге через $max\{2u-2, n\}$ шагов игры второй слой будет обучен, а первый - обучен изначально.

Теорема доказана. □

Список литературы

[1] Э. Э. Гасанов, В. Б. Кудрявцев, *Теория хранения и поиска информации*, ФИЗМАТЛИТ, М., 2002.

On the self-organizing system of the traffic lights that ensure uninterrupted traffic

Muravev A.K.

The present paper considers the cooperative game about the transport delivery on the rectangular grid. The concepts of congestion and cars collision on it, as well as the automata as the agents in

the traffic lights of the grid regulating movement of transport, are introduced.

It's proved that there exists a structure of the automata with local area of visibility leading to the self-organization of the system of traffic lights ensuring an arbitrarily long, correct transport delivery without any congestions and cars collisions for any number of grid's outputs.

Keywords: cooperative game, delivery, traffic organization, agent, homogeneous structure, automaton.

Минимизация числа состояний нечёткого автомата с помощью интервальных формальных понятий

Панкратьева В.В.[1]

В настоящей статье рассматривается связь между задачей минимизации числа состояний нечёткого автомата и проблемой поиска интервальных формальных понятий с максимальным объёмом.

Метод кластеризации, основанный на поиске интервальных формальных понятий, позволяет объединить состояния нечёткого автомата в подмножества со сходными строками достоверностей перехода в другие состояния. При этом мера близости строк определяется заранее заданным параметром σ. В работе показано, что для определённого вида нечётких матриц перехода, начиная с некоторого момента поведение исходного нечёткого автомата, как и поведение минимизированного автомата, стабилизируется. Кроме того, доказано, что при минимизации автомата достоверность слова, распознаваемого нечётким автоматом, не уменьшается. Этот факт позволяет сравнить нечёткий язык, распознаваемый исходным автоматом, и язык, распознаваемый минимизированным.

Ключевые слова: нечёткие автоматы, интервальные формальные понятия, нечёткие языки.

1. Введение

Начиная со второй половины прошлого столетия активно исследуются различные математические модели, в которых некоторые параметры определены неоднозначно или не определены вовсе. Прекрасным инструментом для работы с такими моделями является созданная в 1965 году

[1] *Панкратьева Вера Викторовна* — аспирант каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: vera.pankratieva@gmail.com.

Pankratieva Vera Viktorovna — graduate student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

теория нечётких множеств [1]. Нечёткие методы применяются в различных прикладных разделах математики, в том числе в теории графов, теории формальных понятий, а также в теории автоматов и в других областях.

Конструкция нечётких автоматов впервые появилась в 1969 году в работе [2] и является обобщением понятия детерминированного автомата, в котором функция переходов в другое состояние не является однозначно определённой, а представляет собой некоторую оценку (степень нечёткости) из частично-упорядоченного множества. Чаще всего в качестве этого множества используется отрезок [0,1], в этом случае функция перехода представляет собой нечёткую матрицу, элементами которой являются числа из отрезка. Удобнее всего использовать не весь числовой отрезок, а значения из него с некоторым заданным постоянным шагом, например, шаг длиной 0.2 порождает шесть числовых значений: 0, 0.2, 0.4, 0.6, 0.8, 1. Для различных символов входа автомата такие нечёткие матрицы, вообще говоря, различны. В качестве операций умножения и сложения нечётких матриц используются стандартная операция min-max умножения и операция взятия максимума соответственно. Если входной алфавит представляет собой один символ, то его повторение отвечает возведению в степень нечёткой матрицы переходов.

Зачастую нечёткие матрицы имеют очень большие размеры, что вызывает неудобство при работе с ними, а также при интерпретации результатов работы модели. Одним из решений этой проблемы является минимизация числа состояний нечёткого автомата. Эта проблема может быть решена за счёт объединения в одно состояние нескольких «похожих» состояний нечёткого автомата. В нечёткой матрице переходов размера $n \times n$ состоянию с номером i соответствует строка степеней нечёткости, с которыми i-е состояние переходит во все остальные состояния автомата. Таким образом, если два или более состояний автомата имеют одинаковые или схожие строки степеней нечёткости в матрице, то такие состояния можно объединить в одно состояние, так как они ведут себя при функционировании автомата похожим образом. Данная задача полностью соответствует задаче нахождения интервальных формальных понятий.

Анализ формальных понятий — математическая теория, основы которой были заложены Б. Гантером и Р. Вилле в фундаментальной работе [3]. Анализ формальных понятий успешно применяется в различных системах кластеризации. При этом часто возникает необходимость работы с большим массивом данных, который представляет собой числовой контекст (таблицу). Существует несколько обобщений анализа фор-

мальных понятий именно для числовых и нечётких контекстов. Одно из обобщений — узорные структуры, введённые С. Кузнецовым и Б. Гантером [4], частным случаем которых являются формальные понятия с операцией интервального пересечения (слияния). Интервальные формальные понятия удобно использовать при работе с нечёткими контекстами для решения задач кластеризации, то есть разбиения множества строк на группы, определяемые похожими распределениями числовых данных.

Проблема поиска интервального формального понятия, содержащего наибольшее количество строк, NP-трудна [5], однако на практике для его нахождения может успешно применяться приближённый алгоритм [6].

Матрица переходов нечёткого автомата может интерпретироваться как нечёткий контекст, при этом задача уменьшения числа состояний автомата соответствует задаче кластеризации. В результате кластеризации получается новый нечёткий автомат, состояния которого соответствуют найденным кластерам.

Множество всех интервальных формальных понятий (найденных кластеров) образует полную решётку, что согласуется с результатами работы [7].

Одной из практически важных задач в теории нечётких автоматов является распознавание языков. Нечёткий автомат можно рассматривать как устройство, распознающее язык над входным алфавитом, при этом каждому слову ставится в соответствие степень его принадлежности языку.

Часто работа с данными требует применения таких моделей, которые бы позволяли выделить подмножество слов в языке, обладающих высокой (не ниже заранее заданного числа) степенью достоверности. В то же время степень достоверности распознаваемого автоматом слова может измениться при минимизации числа состояний исходного автомата. Поэтому естественным образом возникает задача о сравнении языков, распознаваемых до и после минимизации.

2. Основные понятия и определения

С детальным введением в теорию решёток и теорию порядков можно ознакомиться в работе [8], а основные определения, относящиеся к теории формальных понятий, содержатся в работе [3].

Определение 1. Бинарное отношение R на множестве M называется *отношением частичного порядка* (или просто *частичным порядком*), если оно удовлетворяет следующим условиям для всех элементов:

1) рефлексивность: xRx;
2) антисимметричность: если xRy и $x \neq y$, то не выполняется yRx;
3) транзитивность: xRy и $yRz \Rightarrow xRz$.

Для R мы часто используем символ $\leq$ (для R^{-1} символ $\geq$) и пишем $x < y$, если $x \leq y$ и $x \neq y$. Как обычно, мы читаем $x \leq y$ как «x меньше или равно y». *Частично упорядоченное множество* — это пара $(M, \leq)$, где M — множество, а $\leq$ — отношение частичного порядка на M.

Определение 2. *Верхняя (нижняя) полурешётка* — это частично упорядоченное множество $(M, \leq)$ такое, что для любых элементов $x, y \in M$ существует единственная точная верхняя (нижняя) грань.

Определение 3. *Полурешёточной операцией* на множестве M называется такая операция $\sqcap\colon M \times M \to M$, что для некоторого $e \in M$ и любых $x, y, z \in M$ выполнены следующие свойства:

- $x \sqcap x = x$ (идемпотентность);
- $x \sqcap y = y \sqcap x$ (коммутативность);
- $(x \sqcap y) \sqcap z = x \sqcap (y \sqcap z)$ (ассоциативность);
- $e \sqcap x = e$.

Определение 4. *Решёткой* называется упорядоченное множество $(L, \leq)$, которое является одновременно верхней и нижней полурешёткой.

Определение 5. Пусть $(P, \leq_P)$ и $(Q, \leq_Q)$ — частично упорядоченные множества. *Соответствием Галуа* между этими множествами называется пара отображений $\varphi\colon P \to Q$ и $\psi\colon Q \to P$ (каждое из которых является *оператором Галуа*) таких, что для любых $p_1, p_2 \in P$ и $q_1, q_2 \in Q$ выполнены следующие свойства:

- $p_1 \leq_P p_2 \;\Rightarrow\; \varphi(p_1) \geq_Q \varphi(p_2)$ (антиизотонность);
- $q_1 \leq_Q q_2 \;\Rightarrow\; \psi(q_1) \geq_P \psi(q_2)$ (антиизотонность);
- $p \leq_P \psi(\varphi(p))$ и $q \leq_Q \varphi(\psi(q))$ (изотонность).

Определение 6. Двойное применение оператора Галуа, а именно $\psi(\varphi(\cdot))$ и $\varphi(\psi(\cdot))$ является *оператором замыкания*. *Оператор замыкания* $\overline{(\cdot)}$ на M — отображение, относящее *замыкание* $\overline{X} \subseteq M$ каждому подмножеству $X \subseteq M$ и удовлетворяющее следующим условиям:

- $X \leq Y \;\Rightarrow\; \overline{X} \leq \overline{Y}$ (монотонность);
- $X \leq \overline{X}$ (экстенсивность);
- $\overline{\overline{X}} = \overline{X}$ (идемпотентность).

Сформулируем определение узорной структуры так, как оно дано в работе [4].

Определение 7. *Узорная структура* — это тройка $(G, (D, \sqcap), \delta)$, где G — множество объектов, $(D, \sqcap)$ — полная полурешётка всевозможных описаний, а $\delta\colon G \to D$ — функция, которая сопоставляет каждому объекту из множества G его описание из D.

Соответствие Галуа между подмножествами множества объектов и описаниями для узорной структуры $(G, (D, \sqcap), \delta)$ определяется следующим образом:

$$A^{\square} := \prod_{g \in A} \delta(g), \qquad \text{где} \;\; A \subseteq G,$$
$$d^{\square} := \{g \in G \mid d \sqsubseteq \delta(g)\}, \qquad \text{где} \;\; d \in D.$$

Определение 8. *Узорное понятие узорной структуры* $(G, (D, \sqcap), \delta)$ — это пара (A, d), в которой $A \subseteq G$ — подмножество множества объектов, $d \in D$ — одно из описаний из полурешётки, такая, что $A^{\square} = d$ и $d^{\square} = A$; A называется *узорным объёмом* понятия, а d — его *узорным содержанием*.

Частным случаем узорного понятия является интервальное формальное понятие. В качестве D берутся строки числового контекста и рассматриваются как кортежи интервалов нулевой длины. Интервальное формальное понятие — это пара (A, d), где A — подмножество множества объектов, а d — кортеж интервалов, границы которых определяются наименьшим и наибольшим значениями соответствующей компоненты в описаниях всех объектов из A.

Интервальные формальные понятия удобно применять при работе с числовыми контекстами, когда необходимо разделять все данные на кластеры, содержащие объекты с похожими «распределениями» числовых данных в строке.

Если ввести параметр σ для ширины интервального понятия (разницы между наименьшим и наибольшим значениями каждой компоненты), то кластером можно будет назвать интервальное понятие с наибольшим объёмом, такое, что ширина по каждой компоненте не превосходит σ.

Далее приведём определения, относящиеся к теории нечётких автоматов, которые можно найти в работах [1], [2] и [9].

Определение 9. Пусть S — произвольное непустое множество элементов, $|S| = k$. *Нечётким подмножеством множества* S называется совокупность $\mu(S) = (\mu(s_1), \mu(s_2), \ldots, \mu(s_k))$, где $\mu(s_i) \in [0, 1]$, $1 \leq i \leq k$.

То есть для каждого элемента множества S указывается степень нечёткости или достоверности, с которой этот элемент принадлежит множеству S. На практике в качестве степеней нечёткости удобно использовать не произвольные точки отрезка [0, 1], а точки, полученные при делении отрезка на равные части, например, при делении на 10 *количество* α *степеней нечёткости* будет 11: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, а при делении на 5 значение α равно 6. Очевидно, что множество нечётких подмножеств множества S образует полную решетку.

Определение 10. *Нечёткая матрица* — это числовая матрица, элементы которой принадлежат отрезку [0,1].

Определение 11. *Нечётким языком* L *в алфавите* A называется произвольное нечёткое подмножество множества всех слов A^* в этом алфавите

$$L\colon A^* \to [0, 1].$$

Определение 12. *Нечёткий конечный инициальный автомат* — это четвёрка $F = (A, Q, \varphi, P_I)$, где $A = \{a_1, a_2, \ldots, a_m\}$ — конечный входной алфавит, $Q = \{q_1, q_2, \ldots, q_k\}$ — конечное множество состояний, $\varphi : Q \times A \times Q \to [0, 1]$ — функция переходов, P_I — вектор-строка начального состояния (нечёткое подмножество Q).

Запись $\varphi : Q \times A \times Q \to [0, 1]$ интерпретируется следующим образом: нечёткий автомат, находящийся в состоянии q_1, при действии входного символа a переходит в состояние q_2 с некоторой степенью нечёткости из отрезка [0, 1]. Эту степень нечёткости удобно обозначать как величину $\mu_a(q_1, q_2)$. Функция φ порождает для каждого входного символа a нечёткую матрицу $T(a)$ размера $k \times k$, $T(a) = [\mu_a(q_i, q_j)]$, где $1 \leq i \leq k$, $1 \leq j \leq k$, $a \in A$.

Определение 13. *Автономный автомат* — это автомат, входной алфавит которого состоит из одного символа, то есть $A = \{a\}$.

Для определения нечёткого языка, распознаваемого автоматом F, необходимо указать, каким образом вычисляется степень достоверности распознаваемого слова.

Пусть w — слово из A^* длины n. Слово w можно записать как последовательность входных символов $\omega_1\omega_2 \ldots \omega_n$ из алфавита A. При подаче на вход одного из символов ω_i нечёткий автомат переходит из состояния q_j в состояние q_{j+1} с некоторой степенью достоверности $\mu_{\omega_i}(q_j, q_{j+1})$ из отрезка [0,1]. Тогда j-я «траектория» множества «траекторий» Ω автомата F при прочтении слова w выглядит следующим образом:

$$q_{j_0} \longrightarrow q_{j_1} \longrightarrow \ldots \longrightarrow q_{j_{n-1}} \longrightarrow q_{j_n},$$

где q_{j_0} — начальное состояние, а q_{j_n} — состояние автомата, в которое он попадает после прочтения слова w при прохождении по j-й «траектории». При этом последовательность изменений степеней достоверности перехода в следующее состояние выглядит как следующая цепочка:

$$\mu(q_{j_0}) \to \mu_{\omega_1}(q_{j_0}, q_{j_1}) \to \mu_{\omega_2}(q_{j_1}, q_{j_2}) \to \ldots \to \mu_{\omega_{n-1}}(q_{j_{n-2}}, q_{j_{n-1}}) \to \mu_{\omega_n}(q_{j_{n-1}}, q_{j_n}),$$

где $\mu(q_{j_0})$ — достоверность начального состояния j-й «траектории». Для j-й «траектории» можно ввести обозначение $M^j(w) = \min\limits_{0 \le l \le n-1} \mu_{\omega_l}(q_{j_l}, q_{j_{l+1}})$, определяющее минимальное значение достоверности состояний нечёткого автомата при прочтении слова w.

Определение 14. *Степень достоверности распознаваемого автоматом F с помощью множества P_F слова w* определяется по формуле:

$$L(F)(w) = \max_j(\min(\mu(q_{j_0}), \mu(q_{j_n}), M^j(w))),$$

где $|w| = n$, q_{j_0} — начальное состояние для j-й «траектории», $\mu(q_{j_0})$ — достоверность начального состояния, q_{j_n} — состояние, в котором окажется нечёткий автомат при прохождении по j-й «траектории» после прочтения слова w, $\mu(q_{j_n})$ — соответствующий состоянию q_{j_n} элемент нечёткого подмножества $P_F = \mu(Q)$ множества состояний Q. Заметим, что начальное состояние P_I представляет собой нечёткую вектор-строку, а множество P_F можно интерпретировать как нечёткий вектор-столбец.

Другими словами, среди всех цепочек изменений состояний и достоверностей находится такая «траектория», что минимальная степень нечёткости (достоверности) перехода из одного состояния в другое максимальна среди всех возможных «траекторий». Очевидно, что $L(F)(w)$ принимает единственнное значение, при этом «траекторий» из множества Ω, реализующих $L(F)(w)$, может быть много.

Таким образом, нечёткий автомат F с помощью множества финальных состояний P_F определяет степень достоверности каждого слова из A^*.

Определение 15. Множество распознаваемых автоматом F с помощью множества P_F слов с соответствующими им степенями достоверности называется *нечётким языком, распознаваемым автоматом F с помощью множества P_F*, и обозначается $L(F)$.

Иногда требуется найти подмножество слов из $L(F)$, у которых степень принадлежности не меньше какого-то наперёд заданного числа β. Такое подмножество нечёткого языка обозначается $L^{\beta}(F)$.

3. Минимизация числа состояний нечёткого автомата

Рассмотрим автономный нечёткий инициальный автомат $F=(A, Q, \varphi, P_I)$, где $A = \{a\}$, $Q = \{q_1, q_2, \ldots, q_k\}$ — конечное множество состояний, $\varphi : Q \times A \times Q \to [0, 1]$ — функция переходов, которая определена нечёткой матрицей перехода $T(A)$ размера $k \times k$, P_I — начальное распределение состояний (под словом "распределение" мы подразумеваем кортеж достоверностей элементов нечёткого подмножества, при этом, в отличие от теоретико-вероятностного значения этого термина, сумма достоверностей элементов нечёткого подмножества не обязательно равна 1).

На пересечении i-й строки и j-го столбца нечёткой матрицы перехода указана степень достоверности, с которой состояние q_i переходит в состояние q_j. То есть элементы i-й строки представляют собой степени достоверности, с которыми состояние q_i переходит во все состояния $\{q_1, q_2, \ldots, q_k\}$ при поступлении на вход символа a. Если в нечёткой матрице есть строки с таким же или в достаточной мере похожим распределением степеней достоверности, то можно все такие строки объединить в один кластер, то есть в одно новое состояние q'_i для нового нечёткого автомата $F' = (A, Q', \varphi', P'_I)$. Мера похожести двух строк m и n определяется выбранным параметром σ: строки похожи, если для любого столбца j, $1 \le j \le k$, выполняется неравенство $|(\mu_a(q_m, q_j)) - (\mu_a(q_n, q_j))| \le \sigma$.

Матрицу нечёткого перехода можно интерпретировать как нечёткий контекст, состояния которого $\{q_1, q_2, \ldots, q_k\}$ соответствуют множеству объектов G узорной структуры [4], а строки распределений, соответствующие состояниям — описаниями D для узорной структуры. Описание состояния q_i представляет собой кортеж длины k интервалов нулевой длины: $\{(\mu_a(q_i, q_1)), (\mu_a(q_i, q_2)), \ldots, (\mu_a(q_i, q_i)), \ldots, (\mu_a(q_i, q_k))\}$. Это соответствие определяет функцию $\delta : G \to D$, которая сопоставляет каждому объекту из множества G его описание из D.

Описанием для подмножества объектов J является кортеж длины k интервалов вида $\{\min\limits_{i \in J} \mu_a(q_i, q_l), \max\limits_{i \in J} \mu_a(q_i, q_l)\}$, где i — номера объектов (состояний), входящих в подмножество J, $1 \leq l \leq k$. Соответствие Галуа между подмножествами множества объектов (состояний) и множеством описаний (кортежей интервалов) для узорной структуры $(G, (D, \sqcap), \delta)$ определяется следующим образом:

$$J^{\square} := \prod_{g \in J} \delta(g), \qquad \text{где } J \subseteq G,$$

$$d^{\square} := \{g \in G \mid d \sqsubseteq \delta(g)\}, \quad \text{где } d \in D.$$

Узорным понятием узорной структуры (в нашем частном случае интервальным формальным понятием) называется пара (J, d) такая, что $J^{\square} = d$ и $d^{\square} = J$. Эти формулы можно интерпретировать в следующем смысле: подмножеству множества объектов J (оно называется узорным объёмом) соответствует описание d (узорное содержание), причем подмножество J состоит из максимального числа объектов, разделяющих это описание, и это описание нельзя уточнить.

Таким образом, поиск похожих состояний нечёткого автомата сводится к поиску интервальных формальных понятий с максимальным узорным объёмом и с таким узорным содержанием, что длина интервала в каждой компоненте содержания не превышает значения σ. В качестве первого кластера выбирается интервальное формальное понятие, для которого выполнены эти условия; после выбора объекты удаляются из нечёткого контекста и процедура поиска интервального формального понятия с максимальным узорным объёмом и узорным содержанием ширины не более σ повторяется для меньшего нечёткого контекста.

После всех возможных итераций этой процедуры исходный нечёткий контекст будет представлять собой объединение непересекающихся кластеров, являющихся интервальными формальными понятиями. На языке нечётких автоматов это означает, что множество состояний Q будет поделено на непересекающиеся подмножества состояний нечёткого

автомата:

$$Q = \bigcup_i Q_i, \quad Q_i \cap Q_j = \varnothing, \quad i, j \in \{1, \ldots, k\}, i \neq j.$$

В каждом таком подмножестве Q_i содержатся состояния с похожими распределениями степеней достоверности и, таким образом, каждое подмножество Q_i станет состоянием q'_i для нового нечёткого автомата $F' = (A, Q', \varphi', P'_I)$. Значения новой нечёткой матрицы перехода $T'(A)$ определяются так, что на пересечении строки i и столбца j находится максимальная степень достоверности:

$$\mu_a(q'_i, q'_j) = \max_{q_m \in Q_i, q_n \in Q_j} \mu_a(q_m, q_n),$$

а значения нового вектора начального состояния P'_I определяются соотношением $\mu(q'_i) = \max\limits_{q_m \in Q_i} \mu(q_m)$.

В классической теории анализа формальных понятий существует множество алгоритмов для поиска замкнутых объектов, а также для построения решёток формальных понятий. Самые распространённые алгоритмы и их особенности описаны в статье [11]. Для поиска замкнутых объектов в многозначных и нечётких контекстах, а также в узорных структурах, существует множество модификаций этих известных алгоритмов.

Однако вычислительная сложность алгоритмов для поиска формальных понятий может быть экспоненциальной от числа объектов, это показано в работе [3]; кроме того, результатом работы [5] является факт, что задача поиска интервального формального понятия, содержащего максимальное количество объектов, NP-трудна. Поэтому на практике для поиска кластеров нечёткого контекста удобно использовать жадный алгоритм, приближенно решающий данную задачу за линейное по количеству объектов контекста и полиномиальное по размерности время. Приближенный алгоритм и оценки вычислительной сложности подробно описаны в работе [6]. Корректность работы алгоритма кластеризации была протестирована на данных [12], и результат работы приближенного алгоритма совпал с реальными данными.

В случае нечёткого автомата с произвольным конечным входным алфавитом $A = \{a_1, a_2, \ldots, a_{s-1}, a_s\}$ каждому входному символу a_i соответствует своя нечёткая матрица перехода $T(a_i)$. Для того, чтобы два состояния q_m и q_n нечёткого автомата попали в один кластер, необходимо, чтобы разность достоверностей в m-й и n-й строках не превышала значения σ: $|\mu_a(q_m, q_j) - \mu_a(q_n, q_j)| \leq \sigma$, $1 \leq j \leq k$, и это должно быть выполнено для каждой нечёткой матрицы $T(a_i)$. Как и в случае автономного автомата, нечёткие матрицы переходов можно рассматривать как

нечёткие контексты. Если из всех матриц составить единую прямоугольную матрицу размера $k \times (k * s)$, (k строк и $k * s$ столбцов), то строки длиной $k * s$ можно рассматривать как описания узорной структуры, в которой состояния $\{q_1, q_2, \ldots, q_k\}$ соответствуют множеству объектов. В этой узорной структуре можно найти интервальные формальные понятия с максимальным узорным объёмом и шириной не более σ. Узорный объём найденных интервальных формальных понятий будет определять кластеры, то есть группы состояний с похожими распределениями достоверности в строках длиной $k * s$, и, следовательно, с похожими распределениями в каждой нечёткой матрице переходов $T(a_i)$. Все состояния найденного кластера объединяются в одно состояние, что позволяет минимизировать нечёткий автомат.

4. Поведение нечёткого автомата при минимизации числа состояний

При уменьшении числа состояний нечёткого автомата важно, чтобы поведение минимизированного нечёткого автомата соответствовало поведению исходного. Рассмотрим случай, в котором у нечёткой матрицы переходов автономного автомата главная диагональ состоит только из единиц. Очевидно, что если такая матрица допускает возможность кластеризации, то у матрицы с уменьшенным числом состояний главная диагональ также будет содержать максимальные значения.

Запись $X \geq Y$ означает, что $x_{ij} \geq y_{ij}$ для всех пар индексов (i, j). Напомним, что под операцией умножения нечётких матриц понимается min-max умножение, а именно $X \cdot Y = Z$, $z_{ij} = \max\limits_k \min\{x_{ik}, y_{kj}\}$.

Лемма 1. *Если у матрицы X на главной диагонали расположены единицы, то $X \cdot Y \geq Y$ и $Y \cdot X \geq Y$ для любой нечёткой матрицы Y.*

Доказательство. Пусть $Z = X \cdot Y$. По определению, $z_{ij} = \max\limits_k \min\{x_{ik}, y_{kj}\}$. При $k = i$ выполняется равенство $x_{ii}y_{ij} = \min\{x_{ii}, y_{ij}\} = \min\{1, y_{ij}\} = y_{ij}$, откуда $z_{ij} = \max\limits_k \min\{x_{ik}, y_{kj}\} \geq \min\{x_{ii}, y_{ij}\} = y_{ij}$.

Неравенство $Y \cdot X \geq Y$ доказывается аналогично. □

Лемма 2. *Для любого $s \geq 1$ имеют место соотношения*

$$X^{s+1} \geq X^s, \qquad X^s \geq X, \tag{1}$$

где X — нечёткая матрица с единичной главной диагональю. Поскольку элементы матриц X^s принадлежат конечному набору значений, соотношения (1) гарантируют, что, начиная с некоторого $n \in \mathbb{N}$ степени матрицы X стабилизируются: $X^n = X^{n+1} = \dots$.

Из этого утверждения с учётом того факта, что после кластеризации матрицы с единичной главной диагональю матрица с уменьшенным числом состояний также имеет единичную главную диагональ, вытекает важная теорема:

Теорема 1. Начиная с некоторого номера $m \in \mathbb{N}$ степени матрицы C с уменьшенным числом состояний также стабилизируются, то есть выполняется соотношение $C^m = C^{m+1}$.

Это означает, что поведение автономного нечёткого автомата с уменьшенным числом состояний совпадает с поведением исходного автомата.

Далее сформулируем теорему, относящуюся к произвольному нечёткому автомату:

Теорема 2. Если для любого символа $a \in A$ матрица нечёткого перехода $T(a)$ имеет единичную диагональ, то для любой бесконечной последовательности $P = \rho_1\rho_2 \dots \rho_m\rho_{m+1} \dots$ символов из алфавита A найдётся число $m \in \mathbb{N}$, начиная с которого поведение нечёткого автомата стабилизируется, т. е. $R_{m+1} = R_m$, где $R_{i+1} = R_i \cdot T(\rho_{i+1})$, $i \geq 1$, $R_1 = T(\rho_1)$.

Доказательство. Так как для любого символа $\rho_i \in A$ нечёткая матрица перехода $T(\rho_i)$ имеет единичную диагональ и $R_{i+1} = R_i \cdot T(\rho_{i+1})$, то в силу леммы 1 верно $R_{i+1} \geq R_i$. Таким образом, последовательность R_i увеличивается и в силу конечности набора значений нечёткой матрицы найдется номер $m \in \mathbb{N}$, такой, что начиная с m выполняется равенство $R_{m+1} = R_m$. □

5. Распознавание нечёткого языка при минимизации числа состояний нечёткого автомата

Нечёткий автомат $F = (A, Q, \varphi, P_I)$ с помощью множества P_F распознаёт нечёткий язык $L(F)$ и для каждого распознаваемого слова $w \in L(F)$ определена достоверность $L(F)(w) = \max\limits_j(\min(\mu(q_{j_0}), \mu(q_{j_n}), M^j(w)))$. Достоверность $L(F')(w)$ слова w в новом автомате $F' = (A, Q', \varphi', P'_I)$, полученном при минимизации числа состояний автомата F с помощью интервальных формальных понятий, связана с $L(F)(w)$ следующим образом:

Теорема 3. *Достоверность $L(F)(w)$ слова $w \in L(F)$ не уменьшается при минимизации числа состояний нечёткого автомата F.*

Доказательство. Рассмотрим произвольное непустое слово $w \in L(F)$, $w = \omega_1\omega_2 \ldots \omega_n$ и достоверность слова w равна $L(F)(w)$.

При прочтении автоматом слова $w = \omega_1\omega_2 \ldots \omega_n$ образуется j-я траектория, которая описывается цепочкой изменений достоверностей и цепочкой изменений состояний:

$$q_{j_0} \longrightarrow q_{j_1} \longrightarrow \ldots \longrightarrow q_{j_{n-1}} \longrightarrow q_{j_n},$$

где q_{j_0} — начальное состояние для траектории j, а q_{j_n} — состояние автомата, в которое он попадает после прочтения слова w, проходя по траектории $j \in \Omega$.

Выберем два произвольных соседних состояния q_{j_i} и $q_{j_{i+1}}$ на этой траектории. То, что состояния являются соседними, означает, что при прочтении символа ω_{i+1} автомат F переходит из состояния q_{j_i} в состояние $q_{j_{i+1}}$ с достоверностью $\mu_{\omega_{i+1}}(q_{j_i}, q_{j_{i+1}})$.

При кластеризации множества состояний автомата $F = (A, Q, \varphi, P_I)$ мы получим новый автомат $F' = (A, Q', \varphi', P'_I)$. Множество состояний Q является объединением непересекающихся подмножеств: $Q = \bigcup\limits_i Q_i$. Каждое из подмножеств Q_i определяет состояние q'_i в новом нечётком автомате F'.

Как указано в разделе 3, вектор начального состояния P'_I в новом нечётком автомате выбирается так, чтобы достоверность $\mu(q'_{j_0})$ была максимальной среди всех достоверностей состояний, попавших в один кластер; таким же образом выбираются значения нового принимающего множества P'_F (нечёткого вектор-столбца).

Для соседних на j-й траектории состояний q_{j_i} и $q_{j_{i+1}}$ существуют два варианта: или они попали после кластеризации в один кластер (новое состояние q'_λ автомата F'), или в разные кластеры (разные состояния q'_ν и q'_ζ). В первом случае, как указано в разделе 3, в качестве диагонального элемента новой нечёткой матрицы переходов $T'(\omega_{i+1})$ выбирается значение

$$\mu_{\omega_{i+1}}(q'_\lambda, q'_\lambda) = \max_{q_m, q_n \in Q_\lambda} \mu_{\omega_{i+1}}(q_m, q_n).$$

то есть максимальная среди всех достоверностей перехода из состояния q_m в q_n исходной нечёткой матрицы переходов $T(\omega_{i+1})$, где q_m и q_n попали в один кластер $Q_\lambda = q'_\lambda$. Очевидно, что для q_{j_i} и $q_{j_{i+1}}$ верно неравенство

$$\mu_{\omega_{i+1}}(q_{j_i}, q_{j_{i+1}}) \leq \max_{q_m, q_n \in Q_\lambda} \mu_{\omega_{i+1}}(q_m, q_n).$$

Аналогично, если состояния q_{j_i} и $q_{j_{i+1}}$ попали в разные кластеры q'_ν и q'_ζ соответственно, достоверность перехода из состояния q_{j_i} в состояние $q_{j_{i+1}}$ при прочтении символа ω_{i+1} равна

$$\mu_{\omega_{i+1}}(q_{j_i}, q_{j_{i+1}}) \leq \max_{q_m \in Q_\nu, q_n \in Q_\zeta} \mu_{\omega_{i+1}}(q_m, q_n),$$

где $Q_\nu = q'_\nu$ и $Q_\zeta = q'_\zeta$.

Согласно определению, достоверность слова w, определяемая нечётким автоматом F' с помощью принимающего множества P'_F, вычисляется по формуле: $L(F')(w) = \max_j(\min(\mu(q'_{j_0}), \mu(q'_{j_n}), M'^j(w)))$, где

$$M'^j(w) = \min_{0 \leq l \leq n-1} \mu_{\omega_l}(q'_{j_l}, q'_{j_{l+1}}).$$

Очевидно, что $M'^j(w) \geq M^j(w)$. Отсюда, с учётом неравенств $\mu(q'_{j_0}) \geq \mu(q_{j_0})$ и $\mu(q'_{j_n}) \geq \mu(q_{j_n})$, для j-й траектории имеем:

$$\min(\mu(q'_{j_0}), \mu(q'_{j_n}), M'^j(w)) \geq \min(\mu(q_{j_0}), \mu(q_{j_n}), M^j(w)).$$

Тогда, используя определение достоверности, получаем искомое неравенство:

$$L(F')(w) \geq L(F)(w). \qquad \square$$

Из теоремы вытекает утверждение о размере подъязыка $L^\beta(F)$:

Утверждение. Пусть $L^\beta(F)$ — язык, распознаваемый автоматом F с помощью множества P_F и содержащий слова с достоверностью не менее β, $L^\beta(F')$ — язык, распознаваемый минимизированным автоматом F' с помощью множества P'_F и содержащий слова с достоверностью не менее β. Тогда для любого β верно $L^\beta(F) \subseteq L^\beta(F')$.

6. Заключение

В настоящей работе рассматривается связь между задачей минимизации количества состояний нечёткого автомата и проблемой поиска интервальных формальных понятий с максимальным объёмом. Для исходного автомата $F = (A, Q, \varphi, P_I)$ для каждого входного символа из алфавита A определена матрица нечёткого перехода $T(A)$. Для каждого состояния из Q строка в нечёткой матрице показывает распределение достоверностей, с которыми автомат переходит во все состояния $\{q_1, q_2, \ldots, q_k\}$.

Если объединить состояния, имеющие похожие строки распределений, то можно получить новый автомат F' с числом состояний, меньшим, чем число состояний исходного автомата. Матрицу нечёткого перехода можно рассматривать как нечёткий контекст и применять методы теории формальных понятий. Контекст представляет собой узорную структуру, в которой объектам соответствуют состояния, а сложному описанию объектов — строки распределения достоверностей в нечёткой матрице. Нахождение строк контекста с похожими описаниями является задачей поиска интервальных формальных понятий с максимальным объёмом и содержанием, ширина которого по каждой компоненте ограничена значением σ. Существуют различные алгоритмы поиска интервальных понятий, а для больших размерностей нечёткого контекста можно использовать приближённый алгоритм.

Хорошим критерием того, что получаемый в результате кластеризации автомат наследует свойства исходного автомата, является схожее поведение их матриц нечёткого перехода при возведении в последовательные степени. В статье показано, что для случая нечётких матриц с единицами на главной диагонали как исходная, так и кластеризованная матрица автономного автомата обладают схожим свойством: последовательность их степеней является монотонно неубывающей и стабилизируется на некотором конечном шаге. Если входной алфавит содержит более одного символа и для каждого из них матрица нечёткого перехода обладает таким же свойством на диагонали, то результат последовательного умножения различных матриц перехода стабилизируется на некотором конечном шаге; это также верно для матриц с уменьшенным числом состояний.

Проблема уменьшения числа состояний нечёткого автомата порождает следующую интересную задачу: каким образом изменяется распознаваемый нечётким автоматом нечёткий язык? В работе доказано, что достоверность любого слова исходного нечёткого языка не уменьшается при кластеризации числа состояний нечёткого автомата. Этот факт позволяет заметить, что нечёткий язык, у которого каждое слово обладает достоверностью не меньше заданного параметра β исходного нечёткого автомата, является подмножеством языка с достоверностью слов не меньше β кластеризованного нечёткого автомата.

Дальнейшие исследования будут направлены на анализ свойств нечётких матриц, которые позволяют оценить, насколько и каким образом изменяется нечёткий язык при минимизации числа состояний нечёткого автомата с помощью интервальных формальных понятий.

Список литературы

[1] L. Zadeh, "Fuzzy Sets", *Information and control*, **8** (1965), 338–353.

[2] W. G. Wee, K. S. Fu, "A Formulation of Fuzzy Automata and its Application as a Model of Learning Systems", *IEEE Transactions on Systems Science and Cybernetics*, **5**:3 (July 1969), 215–223.

[3] B. Ganter, R. Wille, *Formal Concept Analysis*, Springer-Verlag, Berlin, 1999.

[4] B. Ganter, S. Kuznetsov, *Pattern Structures and Their Projections*, Preprint MATH-AL-14-2000, Technische Universit at Dresden, Herausgeber, Der Rektor, November, 2000.

[5] A. V. Galatenko, S. A. Nersisyan, D. N. Zhuk, "NP-Hardness of the Problem of Optimal Box Positioning", *Mathematics*, 2019, №7, 711.

[6] S. A. Nersisyan, V. V. Pankratieva, V. M. Staroverov, V. E. Podolskii, "A Greedy Clustering Algorithm Based on Interval Pattern Concepts and the Problem of Optimal Box Positioning", *Journal of Applied Mathematics*, 2017, 4323590.

[7] А. А. Максимов, "Исследование сложных информационных систем с использованием универсально-алгебраических конструкций нечетких автоматов", *Вестник Саратовского государственного социально-экономического университета*, 2006, №14(3), 126–128.

[8] G. Birkhoff, *Lattice Theory*, American Mathematical Society colloquium publications, **25**, Amer. Math. Soc., 1940.

[9] R. Bělohlávek, "Determinism and fuzzy automata", *Inf. Sci.*, **143**:1–4 (2002), 205–209.

[10] M. Kaytoue, S. Duplessis, S. O. Kuznetsov, A. Napoli, "Two FCA-Based methods for mining gene expression data", *Proc. 7th International Conference on Formal Concept Analysis (ICFCA'2009), LNAI 5548*, eds. S. Ferre, S. Rudoplh, Springer-Verlag, Berlin, Heidelberg, 2009, 251–266.

[11] S. O. Kuznetsov, S. A. Obiedkov, "Comparing performance of algorithms for generating concept lattices", *J. Expt. Theor. Artif. Intell.*, **14**:2–3 (2002), 189–216.

[12] J. Barretina, G. Caponigro, N. Stransky et al., "The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity", *Nature*, **483** (2012), 603–607.

Minimization of the Number of States of a Fuzzy Automaton Using Interval Pattern Concepts

Pankratieva V.V.

Relationship between the problem of minimization of the number of states of fuzzy automata and the problem of mining of interval pattern concepts with maximum extent is considered.

The clustering method based on interval pattern concept mining combines states of a fuzzy automaton into subsets with similar patterns of confidence of transition into other states, where patterns

are considered close to each other if the distance between them is bounded by a predetermined parameter σ. It is shown that for a certain type of fuzzy transition matrices the behavior of the original fuzzy automaton, as well as the behavior of the minimized one, eventually stabilizes. Moreover, it is proved that the membership value of each word recognized by the fuzzy automaton does not decrease after minimization. This fact allows comparing the fuzzy language recognized by the original automaton and the language recognized by the minimized automaton.

Keywords: fuzzy automata, interval pattern concepts, fuzzy languages.

Часть 3.
Математические модели

Клеточные автоматы с локаторами

Гасанов Э.Э.[1]

В работе вводится новый математический объект — клеточный автомат с локаторами. Он получается добавлением к клеточному автомату новой возможности - посылать сигналы в “эфир” и получать из “эфира” суммарный сигнал всех элементарных автоматов. Приведены примеры задач, решение которых существенно упрощается, если для их решения использовать клеточные автоматы с локаторами вместо традиционных клеточных автоматов.

Ключевые слова: клеточные автоматы, однородные структуры, задача о стрелках, конструирование движущихся изображений, построение кратчайшего пути.

1. Введение

Клеточные автоматы (другие названия: самовоспроизводящиеся автоматы и однородные структуры) 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, и в описанном ниже виде использовалось А. Берксом [4], Э. Муром [5], В. Б. Кудрявцевым, А. С. Подколзиным, А. А. Болотовым [6] и другими исследователями.

Клеточный автомат — это математический объект с дискретными пространством и временем. Каждое положение в пространстве представлено отдельной клеткой, а каждый момент времени — дискретным временным шагом или поколением. Состояние каждой пространственной клетки определяется очень простыми правилами взаимодействия. Эти

[1] *Гасанов Эльяр Эльдарович* — профессор каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: el_gasanov@gmail.com.

Gasanov Elyar Eldarovich — professor, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, Chair of Mathematical Theory of Intellectual Systems.

правила предписывают изменения состояния каждой клетки в следующем такте времени в ответ на текущее состояние соседних клеток. При этом для разных клеток правила изменения состояний могут быть разными.

Если в качестве преобразователя информации, стоящего в клетке пространства, выбрать конечный автомат, причем один тот же для всех клеток, то мы приходим к понятию однородной структуры. В таком случае содержательно клеточный автомат представляет собой бесконечную автоматную схему, построенную следующим образом. Рассмотрим k-мерное евклидово пространство. Разобьем его на гиперкубы с единичным ребром, ребра которых параллельны осям координат. В каждый гиперкуб поместим один и тот же конечный автомат V с m входами и одним выходом. Разветвим выход автомата и соединим с входами его соседей одинаковым образом для всех гиперкубов в пространстве. Получим бесконечную однородным образом устроенную автоматную схему, которая и называется клеточным автоматом. Последовательность состояний отдельных автоматов V, содержащую состояния всех автоматов схемы, будет образовывать состояние клеточного автомата. Последовательность состояний клеточного автомата, возникающая при синхронной работе всех составляющих его конечных автоматов, называется функционированием клеточного автомата.

Клеточные автоматы представляют собой дискретную математическую модель широкого класса реальных систем вместе с протекающими в них процессами, таких как физические среды, в которых реализуются тепловые и волновые явления, химические растворы с реакциями в них, биологические ткани, в которых происходит обмен веществ, технические схемы управления, производящие переработку механических и электрических сигналов, вычислительные схемы и т.п.

Если задать начальные состояния автоматов, то в схеме начнется изменение состояний автоматов, определяемое законами функционирования автоматов и связями между ними. Явление глобального изменения этих состояний и является главным объектом изучения в теории клеточных автоматов.

В данной работе вводится обобщение клеточного автомата, которое предлагается назвать клеточным автоматом с локаторами.

Одним из серьезных ограничений клеточных автоматов является ограниченность шаблона соседства, т.е. каждый автомат может видеть некоторое число своих соседей и тем самым сигналы в клеточных автоматах распространяются относительно медленно. В данной работе предла-

гается добавить клеточным автоматам возможность передавать некоторые сигналы всем элементарным автоматам одновременно, что позволит преодолеть свойство локальности.

Здесь можно вспомнить модель несжимаемой жидкости, в которой тоже сигналы моментально распространяются по всему объему. Похожая картина наблюдается в квантовой механике и в квантовых клеточных автоматах [7], когда изменение состояния одного автомата вызывает изменение состояния всех "запутанных" с ним автоматов. В работе [8] вводится понятие нелокальных клеточных автоматов. В этой работе нелокальность состоит в том, что для каждого элементарного автомата множество его соседей выбирается случайным образом, и таким образом соседними могут оказаться далеко отстоящие друг о друга элементарные автоматы.

В обычной жизни человек, когда хочет передать информацию не только видимым соседям, он может воспользоваться такими приемами, как подача световых сигналов с помощью сигнальных ракетниц. Еще более распространенным способом является использование радио- и телеэфира.

В данной работе тоже вводится понятие эфира. Считается, что каждый элементарный автомат может послать в эфир некоторый сигнал из конечного алфавита. Элементы алфавита образуют конечную аддитивную коммутативную полугруппу, а сам эфир представляет собой потенциально бесконечный сумматор сигналов элементарных автоматов, где в качестве суммы выступает определяющая операция данной полугруппы. На следующий такт каждый элементарный автомат получает из эфира суммарный сигнал и, учитывая его, изменяет свое состояние. В природе таким сумматором является эфир, который суммирует все радиосигналы естественным образом, и фактически каждый из приемников получает на вход один и тот же сигнал, и уже потом выделяет из общего сигнала нужную ему составляющую.

На этом принципе можно реализовать новый тип интегральных схем, используя в качестве сумматора некоторую подложку, на которую все элементарные автоматы будут сбрасывать некоторые коммутирующие или экстренные сигналы.

Введение эфира и возможности посылать в эфир сигналы позволяет мгновенно передавать сигналы на любые расстояния, и тем самым позволяет одному элементарному автомату управлять поведением сколь угодно далеко удаленного от него другого элементарного автомата. Рассматриваются клеточные автоматы с локаторами, которые могут полу-

чать сигналы из эфира с определенных направлений. Иными словами, у каждого элементарного автомата имеется несколько локаторов, направленных в разные стороны, и он может с помощью этих локаторов получать сигналы с этих направлений.

В работе вводится формальная модель клеточных автоматов с локаторами. Приводится решение некоторых классических и новых задач с помощью стандартных клеточных автоматов, а затем показывается, что эти же задачи с помощью клеточных автоматов с локаторами решаются значительно легче.

2. Понятие клеточного автомата с локаторами

Введем понятие клеточного автомата с локаторами, опираясь на определение клеточного автомата, взятое из [9].

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. В частности, в данной работе мы будем рассматривать два вырожденных случая: полный телесный угол, совпадающий с $\mathbb{R}^k$ без вершины угла, который будем обозначать через Ω, и телесные углы, равные одному лучу, такие телесные углы будем обозначать через вектора, являющиеся направляющими лучей.

Клеточным автоматом с локаторами называется восьмерка

$$\sigma = (\mathbb{Z}^k, E_n, V, E_q, +, L, \varphi, \psi),$$

где $\mathbb{Z}^k$ — множество k-мерных векторов с целыми координатами, $E_n = \{0, 1, \ldots, n-1\}$, $V = (\alpha_1, \ldots, \alpha_{h-1})$ — упорядоченный набор попарно различных ненулевых векторов из $\mathbb{Z}^k$, $E_q = \{0, 1, \ldots, q-1\}$, $+$ — коммутативная полугрупповая операция, заданная на E_q, $L = (\nu_1, \ldots, \nu_m)$ — упорядоченный набор попарно различных телесных углов в $\mathbb{R}^k$ с вершиной в начале координат, φ — функция, зависящая от переменных $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$, $\varphi : E_n^h \times E_q^m \to E_n$, $\varphi(0, \ldots, 0) = 0$, ψ — функция, зависящая от переменных $x_0, x_1, \ldots, x_{h-1}, z_1, \ldots, z_m$, $\psi : E_n^h \times E_q^m \to E_q$. Элементы множества $\mathbb{Z}^k$ называются *ячейками* клеточного автомата σ; элементы множества E_n называются *состояниями ячейки* клеточного автомата σ; набор V называется *шаблоном соседства* клеточного автомата σ; элементы множества E_q называются *сигналами вещания*; набор L

называется *шаблон 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 E_n, переменные $z_1, \ldots, z_m$ принимают значения из E_q. Состояние 0 интерпретируется как *состояние покоя*, а условие $\varphi(0, \ldots, 0) = 0$ — как условие сохранения состояния покоя.

Здесь нам нужно было вводить упорядочение шаблона соседства V и шаблон 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 функция вещания ψ зависят от переменных $x_0, x_{\alpha_1}, \ldots, x_{\alpha_{h-1}}, z_{\nu_1}, \ldots, z_{\nu_m}$, здесь индекс первой переменной есть нулевой вектор $0 = (0, \ldots, 0) \in \mathbb{Z}^k$. 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 просто как множество, а не упорядоченный набор.

В дальнейшем мы так и будем поступать: воспринимать шаблон соседства как множество векторов, а шаблон локаторов как множество телесных углов и индексировать переменные локальной функции переходов и функции вещания векторами из шаблона соседства и телесными углами из шаблона локаторов. При этом мы часто будем опускать в индексах внешние круглые скобки у векторов. Например, если $k = 2$, $n = 2$, $q = 2$ и $V = \{(-1, 0), (1, 0)\}$, $L = \{\Omega, (0, 1)\}$, то пример локальной функции переходов может выглядеть так: $\varphi = x_{-1,0} \& z_\Omega \vee x_{1,0} \& z_{0,1}$.

Если $\alpha \in \mathbb{Z}^k$, ν — телесный угол с вершиной в начале координат, то через $\nu(\alpha)$ обозначим телесный угол, полученный параллельным переносом угла ν в точку 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 (e, f), где e — произвольная функция, определенная на множестве $\mathbb{Z}^k$, принимающая значения из E_q, называемая *состоянием эфира*, f — произвольная функция, определенная на множестве $\mathbb{Z}^k$, принимающая значения из E_n и называемая *распределением состояний клеточного авто-*

мата с локаторами σ. Такую функцию можно интерпретировать как некую мозаику, возникающую в k-мерном пространстве в результате приписывания каждой точке с целочисленными координатами некоторого состояния из множества E_n и некоторого сигнала из множества E_q. Множество всевозможных состояний клеточного автомата с локаторами обозначим Σ.

Если $\alpha \in \mathbb{Z}^k$, (e, f) — состояние клеточного автомата с локаторами σ, то значение $e(\alpha)$ называем *сигналом ячейки* α, *определяемым состоянием* (e, f), а значение $f(\alpha)$ — *состоянием ячейки* α, *определяемым состоянием* (e, f). Для каждого $i \in \{1, \ldots, m\}$ значение

$$s_i(\alpha) = \sum_{\beta \in \nu_i(\alpha) \cap \mathbb{Z}^k} e(\beta) \tag{1}$$

называем *значением локатора* ν_i, *определяемым состоянием* (e, f). Здесь суммирование сигналов осуществляется с помощью определяющей операции + полугруппы E_q.

На множестве Σ определим *глобальную функцию переходов* Φ клеточного автомата с локаторами σ, полагая $\Phi(e, f) = (e', f')$, где $(e, f), (e', f') \in \Sigma$ и для любой ячейки $\alpha \in \mathbb{Z}^k$ выполняются тождества

$$f'(\alpha) = \varphi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)), \tag{2}$$

$$e'(\alpha) = \psi(f(\alpha), f(\alpha + \alpha_1), \ldots, f(\alpha + \alpha_{h-1}), s_1(\alpha), \ldots, s_m(\alpha)). \tag{3}$$

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 ψ и φ одинаково для всех ячеек.

Поведениями клеточного автомата с локаторами σ называем такие последовательности $(e_0, f_0), (e_1, f_1), (e_2, f_2), \ldots$ его состояний, для которых выполняется $(e_{i+1}, f_{i+1}) = \Phi(e_i, f_i)$ для всех $i = 0, 1, 2, \ldots$, причем (e_i, f_i) называется *состоянием клеточного автомата с локаторами* σ *в момент* i, а (e_0, f_0) также называется *начальным состоянием клеточного автомата с локаторами* σ.

Состояние клеточного автомата, у которого лишь конечное число ячеек находится в отличном от 0 состоянии и сигналы всех ячеек равны нулю, назовем *конфигурацией*. Множество конфигураций будем обозначать через Σ'.

Если задано некоторое состояние клеточного автомата, то ячейки, находящиеся в отличном от 0 состоянии, будем называть *активными.*

Рассмотрим несколько задач для клеточных автоматов и покажем, что эти задачи клеточными автоматами с локаторами решаются существенно проще.

3. Задача синхронизации стрелков

Задача синхронизации стрелков впервые предложена Дж. Майхиллом в 1957 году и опубликованная (с решением) в 1968 году Ф. Муром [10].

В этой задаче рассматривается одномерный клеточный автомат на $\mathbb{Z}^1$. Шаблон соседства равен $V = \{-1, 1\}$, т.е. каждая ячейка имеет двух соседей слева и справа. Множество состояний имеет как минимум три состояния: 0 — состояние покоя, 1 — солдат в начальном состоянии, 2 — огонь. Имеется ограничение на функцию переходов, что солдат в начальном состоянии, имея соседями таких же солдатов состояния не меняет, т.е. $\varphi(1, 1, 1) = 1$. Начальная конфигурация представляет собой непрерывный отрезок из r ячеек в состоянии 1 (солдаты), а все остальные ячейки находятся в состоянии покоя 0. Надо, чтобы в какой-то момент все активные ячейки перешли в состояние 2 одновременно (выстрелили).

Стандартное решение этой задачи описывает две волны состояний, распространяющиеся по ряду солдат, одна из которых движется в три раза быстрее другой. Быстрая волна отражается от дальнего края ряда и встречается с медленной в центре. После этого две волны разделяются на четыре, движущиеся в разные стороны от центра. Процесс продолжается, каждый раз удваивая число волн, пока длина отрезков ряда не станет равной 1. В этот момент все солдаты стреляют. Это решение требует $3r$ единиц времени для r солдат.

Клеточный автомат с локаторами σ_0, который решает задачу синхронизации стрелков, имеет следующий вид

$$\sigma_0 = (\mathbb{Z}^1, E_3, V = \{-1, 1\}, E_2, \vee, L = \{\Omega\}, \varphi, \psi),$$

где $\vee$ — дизъюнкция, взятая в качестве определяющей операции на полугруппе сигналов $E_2 = \{0, 1\}$, шаблон локаторов состоит из одного полного телесного угла Ω, функция вещания ψ принимает значение 1 только для самого левого солдата в начальном состоянии, т.е. $\psi(x_0, x_{-1}, x_1, z_\Omega) = x_0\bar{x}_{-1}\bar{z}_\Omega$, локальная функция переходов принимает значение 2 только, если ячейка находится в состоянии 1 и сигнал эфира

равен 1, и не меняет состояния во всех остальных случаях, т.е.

$$\varphi(x_0, x_{-1}, x_1, z_\Omega) = \max(2 \cdot ((x_0 = 1)\&(z_\Omega = 1) \vee (x_0 = 2)), 1 \cdot (x_0 = 1)).$$

Тем самым, задачу синхронизации стрелков можно решить за 2 такта с помощью трех состояний и двух сигналов вещания. И фактически это решение полностью соответствует решению, принятому среди военных: командир (самый левый солдат) командует "огонь" и вся команда стреляет.

4. Однонаправленное движение точки на луче

Задача однонаправленного движения точки на луче, предложенная и решенная в работах Е.Е.Титовой [11], состоит в следующем.

Множество ячеек представляет собой множество натуральных числе, т.е. луч направленный вправо. Каждая ячейка имеет двух соседей одного слева и одного справа. Часть состояний ячеек называются метками и считаются черными, остальные состояния считаются белыми. Правильными считаются конфигурации, когда на луче ровно одна черная ячейка, называемая точкой. Ячейка, соответствующая числу 1 (самая левая ячейка) не имеет соседа слева, и переменную, соответствующую состоянию соседа слева этой ячейки, будем воспринимать как управляющий вход, на который можем подавать любые управляющие воздействия. Описанное множество ячеек с одним управляющим входом будем называть *экраном*.

Формально *экраном* называется клеточный автомат

$$S = (\mathbb{N}, E_n, V = \{-1, 1\}, \varphi, M),$$

где n — число состояний ячейки клеточного автомата, а $\varphi : E_n^3 \to E_n$ — локальная функция переходов, M — множество меток, $M \subset E_n$, $0 \notin M$. Если ячейка находится в состоянии из M, то неформально считаем, что она окрашена в черный цвет, иначе она окрашена в белый цвет. Переменную x_{-1} локальной функции переходов ячейки, соответствующей числу 1, (*самой левой ячейки*) назовем *управляющим входом экрана* S.

Законом движения назовем бесконечную последовательность (сверхслово) из нулей и единиц. Если $F = f_1, f_2, f_3, \ldots$ — закон движения, то через $F(t)$ обозначим t-й элемент последовательности, т.е. $F(t) = f_t$.

Будем говорить, что *на экране S реализуется движение точки по закону F*, если выполняются следующие условия:

1) в некоторый момент времени в самой левой ячейке экрана появляется метка (до этого на экране нет меток), этот момент будем называть *моментом начала движения* или *началом движения*;

2) изменение позиции метки на экране в t-й момент от начала движения соответствует t-й букве в сверхслове F, а именно, если $F(t) = 0$, то в $(t+1)$-й момент метка остается в той же ячейке, где была в текущий момент, если $F(t) = 1$, то в $(t+1)$-й момент метка сдвинется на одну ячейку вправо, по сравнению со своим текущим положением;

3) в каждый момент времени после начала движения на экране есть ровно одна метка.

Экран S будем называть *универсальным для множества законов движения* $\mathcal{F}$, если для любого F из $\mathcal{F}$ существует такая управляющая последовательность, подаваемая на управляющий вход экрана, что на экране реализуется движение точки по закону F .

Через $\mathcal{F}^s$ обозначим множество таких законов движения F, в которых не встречается более чем s единиц подряд.

В работе [11] доказаны следующие теоремы

Теорема 1. *Для любого экрана S существует такой закон движения $F \in \{0,1\}^\infty$, что на экране S невозможно реализовать движение точки по закону F.*

Теорема 2. *Существует закон движения $F \in \{0,1\}^\infty$, движение по которому невозможно реализовать ни на каком экране S.*

Теорема 3. *Существует универсальный экран с $2s+2$ состояниями для множества законов движения $\mathcal{F}^s$.*

Вопрос описания множества всех реализуемых законов движения остается открытым, хотя в работе Г.В.Калачева и Е.Е.Титовой [12] сделаны существенные продвижения в этом направлении.

Приведем клеточный автомат с локаторами, который решает задачу однонаправленного движения точки на луче.

Рассмотрим следующий клеточный автомат с локаторами

$$\sigma_1 = (\mathbb{N}, E_2, V = \{-1, 1\}, E_2, \vee, L = \{\Omega\}, \varphi, \psi, M = \{1\}),$$

где $\vee$ — дизъюнкция, взятая в качестве определяющей операции на полугруппе сигналов $E_2 = \{0, 1\}$, шаблон локаторов состоит из одного

полного телесного угла Ω, множество меток состоит из одного символа 1, функция вещания ψ тождественно нулевая, локальная функция переходов принимает значение 1 только в двух случаях: если ячейка находится в состоянии 1 и сигнал эфира равен 0, или если ячейка слева находится в состоянии 1 и сигнал эфира равен 1, т.е. $\varphi(x_0, x_{-1}, x_1, z_\Omega) = x_0 \& \bar{z}_\Omega \vee x_{-1} \& z_\Omega$.

Будем считать, что переменная x_{-1} локальной функции переходов самой левой ячейки является управляющим входом. Кроме того, будем считать, что в качестве управляющих воздействий можно посылать в эфир сигналы из E_2.

Легко видеть, что, чтобы начать движение, надо на управляющий вход подать 1, а также послать 1 в эфир. В результате на экране в самой левой ячейке появится метка. Далее, чтобы реализовать закон движения F, надо в момент t посылать в эфир значение $F(t)$.

Тем самым, с помощью клеточных автоматов с локаторами можно реализовать любой закон движения, причем число состояний этого автомата равно 2, и мощность алфавита эфира равна 2.

Фактически с помощью сигналов в эфир мы даем команды точке, двигаться ей или стоять.

5. Построение кратчайшего пути

Задача построения кратчайшего пути для клеточных автоматов звучит следующим образом. В начальной конфигурации имеются только две ячейки в активном состоянии, которые назовем начальными точками. Кратчайший путь считается построенным, если с какого-то момента конфигурация становится стабильной, и активные ячейки этой конфигурации образуют кратчайший путь между начальными точками.

В работе [13] приводится адаптация традиционного решения этой задачи к клеточным автоматам. Такое решение предполагает наличие трех этапов в решении:

1) Распространение расширяющегося сигнала от одной из начальных точек. При расширении каждая ячейка запоминает откуда к ней пришел сигнал. Это позволит в дальнейшем осуществить обратный ход.

2) Когда волна достигает второй начальной точки осуществляется обратный ход, приводящий к первой точке, который и дает кратчайший путь.

3) Одновременно запускается волна очищения (приведения в состояние покоя) всех ячеек, кроме ячеек пути. Чтобы эта волна догнала расширяющуюся волну, расширяющаяся волна с первого этапа должна расширяться с половинной скоростью, а волна третьего этапа с единичной скоростью.

В работе [13] утверждается, что предложенный авторами автомат имеет 14 состояний. В этой работе не оценивается время построения пути, но не сложно понять, что оно не меньше, чем $6n$, где n — расстояние по Манхэттену между начальными точками. Здесь $2n$ тактов необходимо первому этапу, и $4n$ тактов третьему. Можно несколько ускорить процесс, пустив расширяющуюся волну из обоих начальных точек, но понятно, что все равно время построения пути будет пропорционально расстоянию между начальными точками.

Теперь рассмотрим решение этой задачи клеточными автоматами с локаторами.

Рассмотрим следующий клеточный автомат с локаторами

$$\sigma_2 = (\mathbb{Z}^2, E_2, V = \{(-1,0),(0,1),(1,0),(0,-1)\},\\ E_2, \vee, L = \{(-1,0),(0,1),(1,0),(0,-1)\}, \varphi, \psi),$$

где $\vee$ — дизъюнкция, взятая в качестве определяющей операции на полугруппе сигналов $E_2 = \{0,1\}$, шаблон соседства "крест", шаблон локаторов состоит из четырех лучей, направленных влево, вверх, вправо и вниз, функция вещания ψ принимает значение 1, если ячейка в состоянии 1 и имеет место один из четырех случаев: если все ее соседи в состоянии 0; у нее нет соседа сверху в состоянии 1 и верхний локатор получает сигнал 1; у нее нет соседа слева в состоянии 1 и левый локатор получает сигнал 1; у нее нет соседа справа в состоянии 1 и правый локатор получает сигнал 1; т.е.

$$\psi(x_0, x_{-1,0}, x_{0,1}, x_{1,0}, x_{0,-1}, z_{-1,0}, z_{0,1}, z_{1,0}, z_{0,-1}) =\\ = x_0(\bar{x}_{-1,0}\bar{x}_{0,1}\bar{x}_{1,0}\bar{x}_{0,-1} \vee \bar{x}_{0,1}z_{0,1} \vee \bar{x}_{-1,0}z_{-1,0} \vee \bar{x}_{1,0}z_{1,0});$$

локальная функция переходов принимает значение 1, если ячейка была в состоянии 1, или если одновременно пришли сигналы из эфира на одну из четырех пар локаторов: верхний и правый, верхний и левый, верхний и нижний, левый и правый, т.е.

$$\varphi(x_0, x_{-1,0}, x_{0,1}, x_{1,0}, x_{0,-1}, z_{-1,0}, z_{0,1}, z_{1,0}, z_{0,-1}) =\\ = x_0 \vee z_{0,1}z_{1,0} \vee z_{0,1}z_{-1,0} \vee z_{0,1}z_{0,-1} \vee z_{-1,0}z_{1,0}.$$

Покажем, что приведенный выше клеточный автомат с локаторами решает задачу построения кратчайшего пути.

В начальный (нулевой) такт на плоскости только две активные ячейки, которые называем начальными.

Рассмотрим различные случаи расположения начальных ячеек.

Случай 1. Начальные ячейки расположены на одной горизонтали. Левую начальную ячейку обозначим через A, а правую — через B.

Случай 1.1. Если начальные ячейки соседние, то эта пара ячеек и составляет кратчайший путь. Осталось заметить, что сигналы в эфир не появятся, следовательно, не появятся новые активные ячейки, и, значит, конфигурация останется стабильной.

Случай 1.2. Если начальные ячейки не соседние, то функция вещания каждой из начальных ячеек станет равной 1, поскольку у этих ячеек нет соседей. Следовательно, на такте 1 в эфир от ячеек A и B пойдет сигнал, и для всех ячеек между ячейками A и B левые и правые локаторы получат сигналы. Значит, локальные функции переходов этих ячеек примут значение 1. Следовательно, на такте 2 все ячейки между A и B перейдут в состояние 1. Кратчайший путь построен. Поскольку у всех ячеек есть соседи, функция вещания всех ячеек примет значение 0, и возникшая конфигурация останется стабильной.

Случай 2. Начальные ячейки расположены на одной вертикали. Этот случай доказывается аналогично случаю 1.

Случай 3. Начальные ячейки находятся в общем положении. Мысленно проведем через начальные ячейки вертикальные и горизонтальные линии. Получаем воображаемый прямоугольник со сторонами параллельными осям координат, в двух диагональных вершинах которого расположены начальные ячейки.

Случай 3.1. Одна начальная ячейка расположена в левом верхнем углу прямоугольника (обозначим ее через A), а вторая — в правом нижнем (обозначим ее через B).

Поскольку у начальных ячеек нет соседей, то функция вещания этих ячеек примет значение 1. Следовательно, на на такте 1 в эфир от ячеек A и B пойдет сигнал. Ячейка, расположенная в левом нижнем угле прямоугольника (обозначим ее через C) получит сигналы от верхнего и правого локаторов. Поэтому ее локальная функция переходов примет значение 1. Значит, на такте 2 ячейка C станет активной.

Случай 3.1.1. Ячейка C является соседней и для A, и для B. Следовательно, кратчайший путь построен. У всех ячеек есть соседи, следо-

вательно, сигналы в эфир больше подаваться не будут, и конфигурация останется стабильной.

Случай 3.1.2. Ячейка C не является соседней для A, и является соседней для B. Тогда функция вещания ячейки C примет значение 1. Следовательно, на такте 3 в эфир пойдут сигналы от двух ячеек: A и C. Значит, все ячейки между A и C получат сигналы на свои верхние и нижние локаторы. Следовательно, их локальная функция переходов примет значение 1. Следовательно, на такте 4 все ячейки между A и C станут активными. Теперь у всех ячеек есть соседи, следовательно, сигналы в эфир больше подаваться не будут, и конфигурация останется стабильной.

Случай 3.1.3. Ячейка C является соседней для A, и не является соседней для B. Этот случай доказывается аналогично случаю 3.1.2.

Случай 3.1.4. Ячейка C не является соседней ни для A, ни для B. Тогда функция вещания ячейки C примет значение 1. Следовательно, на такте 3 в эфир пойдут сигналы от трех ячеек: A, B и C. Значит, все ячейки между A и C получат сигналы на свои верхние и нижние локаторы, а все ячейки между B и C получат сигналы на свои левые и правые локаторы. Следовательно, локальная функция переходов всех этих ячеек примет значение 1. Следовательно, на такте 4 все ячейки между A и C и между B и C станут активными. Кратчайший путь построен. У всех ячеек есть соседи, следовательно, сигналы в эфир больше подаваться не будут, и конфигурация останется стабильной.

Случай 3.2. Одна начальная ячейка расположена в левом нижнем углу воображаемого прямоугольника, а вторая — в правом верхнем. Этот случай доказывается аналогично случаю 3.1.

Таким, образом мы показали, что предлагаемый клеточный автомат с локаторами позволяет построить кратчайший путь не более чем за 4 такта. При этом у него только 2 состояния и 2 сигнала вещания.

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] Arrighi P., "An overview of Quantum Cellular Automata", *arXiv:1904.12956v2 [quant-ph] 6 Sep 2019*, September 9, 2019, 1–23.

[8] Li W., "Phenomenology of Non-local Cellular Automata", *Stat. Phys.l*, **68**:5/6 (1992), 829-882.

[9] Кудрявцев В. Б., Гасанов Э. Э., Подколзин А. С., *Теория интеллектуальных систем: в 4 кн. Книга четвертая. Теория автоматов.*, Издательские решения, Москва, 2018.

[10] Moore F.R., Langdon G.G., "A generalized firing squad problem", *Information and Control*, **12**:3 (March 1968), 212–220.

[11] Титова Е.Е., "Конструирование движущихся изображений клеточными автоматами", *Интеллектуальные системы*, **18**:1 (2014), 153–180.

[12] Калачев Г.В., Титова Е.Е., "О мере множества законов движения точки, реализуемых клеточными автоматами", *Интеллектуальные системы. Теория и приложения*, **22**:3 (2018), 105–125.

[13] Hochberger C., Hoffmann R., "Solving routing problems with cellular automata", *Proceedings of the Second Conference on Cellular Automata for Research and Industry*, Octber 1996, 89–98.

Celluar automata with locators

Gasanov E.E.

This article introduce new type of math object — cellular automaton with locators. It was created by implementing new functionality for automaton - broadcasting "on-air" signals and retrieving generalized "on-air" signal from all elementary automata. This article highlights some tasks whose solution will be greatly simplified by using cellular automaton with locators instead of traditional cellular automata.

Keywords: cellular automata, homogeneous structures, firing squad problem, motion picture design, constructing the shortest path.

Предугадывание сверхслов на отрезке

Маншилин О.Г.[1]

Автомат предугадывает символ входного сверхслова, если он выдаёт этот символ на выходе в предыдущий момент времени.

В работе вводится понятие степени предугадывания на отрезке. Исследуется вопрос о взаимосвязи предугадывания на отрезке и бесконечности. Получен результаты, позволяющие судить о степени предугадывания на отрезке по степени предугадывания на бесконечности и о степени предугадывания на бесконечности по степени предугадывания на отрезке.

Ключевые слова: предугадывающий автомат, автоматное предугадывание общерегулярных сверхслов, степень предугадывания на отрезке.

1. Введение

В этой статье рассматривается предвосхищение общерегулярных сверхсобытий над алфавитом $\{0, 1\}$. Ещё на заре развития теории автоматов было введено понятие предугадывания. Позже было показано, что идеально предугадываются только полностью периодические сверхслова [1]. Это ограничение можно обойти используя более сложную структуру автоматов [3].

Подобную задачу рассматривали не только с точки зрения конечных автоматов, но также с точки зрения теории вероятностей и машинного обучения [4]. Однако в нашей работе все сверхслова предугадываются конечными автоматами.

В работе Мастихиной А.А. было введено понятие частичного предугадывания сверхслов на бесконечности [2]. В этой статье введено понятие

[1] *Маншилин Олег Григорьевич* — магистр каф. высшей математики ф-та фундаментальных наук МГТУ им. Н.Э.Баумана, e-mail: manshilin.o@gmail.com.

Manshilin Oleg Grigorievich— graduate student, Bauman Moscow State University, Faculty of Fundamental Science, Chair of Higher Math.

частичного предугадывания сверхслов на отрезке. Автомат верно предугадывает некоторую долю входных символов на отрезке длиной l, которая называется степенью предугадывания на отрезке и определяется как нижний нижняя грань отношения угаданных символов на отрезке длиной l к l по всем возможным отрезкам длиной l в сверхслове.

Была дана оценка степени предугадывания на отрезке для конечных автоматов определённого вида. Была показана связь между предвосхищением сверхслов на отрезке и предвосхищением сверхслов на бесконечности для конечных автоматов произвольного вида.

В этой статье приведены примеры конечных автоматов, на которых явно видна связь между предвосхищением на бесконечности и предвосхищением на отрезке. Часть теорем, приведённых в этой статье, распространяется как на общерегулярные сверхслова так и на контекстно-свободные сверхслова.

Автор выражает благодарность А.А. Мастихиной за постановку задачи и помощь в работе.

2. Основные понятия и формулировка результатов.

В работе используются нижеприведённые понятия:

- $c_{\infty}^{\mathfrak{A}}$ - степень предугадывания автомата $\mathfrak{A}$ на бесконечности.
- $|a|$ - длина слова $a \in A^*$.
- $\alpha(n)$ - n-ый элемент слова или сверхслова α
- $pref(\alpha, n)$ - префикс слова или сверхслова α: $pref(\alpha, n) = \alpha(1)\alpha(2)...\alpha(n)$
- $a \cdot b$ - конкатенация слов a и b
- a^n - повторения слова a n раз.
- $\lfloor x \rfloor$ - округления действительного числа x до целочисленного в нижнюю сторону.

В работе рассматриваются конечные инициальные автоматы:

$$\mathfrak{A} = (A, Q, B, \varphi, \psi, q_0),$$

где A - входной алфавит, Q - множество состояний автомата, B - выходной алфавит, φ - функция переходов, ψ - функция выходов.

Если на вход автомату A подаётся сверхслово α, на выходе получается сверхслово y, а состояние автомата в момент времени t обозначается как q_t, то функционирование автомата задаётся системой:

$$\begin{cases} y(t) = \psi(\alpha(t), q_{t-1}) \\ q_t = \varphi(\alpha(t), q_{t-1}) \end{cases}$$

Выходное сверхслово автомата $\mathfrak{A}$ при подаче на него сверхслова α будем обозначать как $y_\alpha^{\mathfrak{A}}$. На протяжение всей статьи множество $\{0, 1\}$ рассматривается как входной и выходной алфавиты.

Мы говорим, что автомат $\mathfrak{A}$ частично угадывает сверхслово на бесконечности [2], если:

$$|y_{\alpha(i)}^{\mathfrak{A}} - \alpha(i+1)| < \infty \tag{1}$$

Степень предугадывания на бесконечности определяется как [2]:

$$c_\infty^\theta(\alpha) = 1 - \underline{\lim}_{t\to\infty} \frac{\sum_1^t |y_\alpha^\theta(i) - \alpha(i+1)|}{t} \tag{2}$$

Будем говорить, что автомат частично предугадывает сверхслово α в смысле отрезка, когда выполняется неравенство:

$$\sum_{i=N+n}^{N+n+t_1} |y_\alpha^{\mathfrak{A}}(i) - \alpha(i+1)| < t_1 | \text{для } \forall n \in \mathbb{N}, \tag{3}$$

где N - конечный префикс сверхслова α, t_1 - длина отрезка, на котором предугадывается сверхслово α. Это означает, что по прошествии некого конечного префикса на любом подслове длиной t_1 будет угадан по меньшей мере один символ.

Введём степень предугадывания в смысле отрезка как:

$$c_{t_1}^\theta(\alpha) = \sup_{N\in\mathbb{N}} \inf_{n\in\mathbb{N}} \{1 - \frac{\sum_{N+n}^{N+n+t_1} |y_\alpha^\theta(i) - \alpha(i+1)|}{t_1}\} \tag{4}$$

Множество сверхслов называется частично предугадываемым на бесконечности, если существует такой конечный автомат, на котором степень предугадывания каждого сверхслова множества строго больше нуля.

Множество сверхслов R называется частично предугадываемым на отрезке длиной t_1, если существует такой конечный автомат, для которого степень предугадывания на любом произвольном подслове длиной t_1 на каждом сверхслове множества R строго больше нуля.

Автомат $\mathfrak{A}$ предугадывает произвольное множество сверхслов $S \subset \{0,1\}^\infty$ на бесконечности со степенью θ, если для $\forall \alpha, \alpha \in S, c_\infty^{\mathfrak{A}} \geqslant \theta$.

Автомат $\mathfrak{A}$ предугадывает произвольное множество сверхслов $S \subset \{0,1\}^\infty$ на отрезке длиной t_1 со степенью θ, если для $\forall \alpha, \alpha \in S, c_{t_1}^{\mathfrak{A}} \geqslant \theta$

Регулярное событие над алфавитом A определяется как:

- $\emptyset, \{a\}, a \in A$ - регулярные события.
- Пусть R_1, R_2 - регулярные события. Тогда $R_1 \cdot R_2, R_1 \cup R_2$ и R_1^* тоже являются регулярными. R_1^* является множеством всех возможных конечных итераций события R_1.

Общерегулярное событие над алфавитом A определяется как:

- Если R_1 - регулярное событие над алфавитом A, то R_1^∞ - общерегулярное событие над этим же алфавитом. R_1^∞ является сверхитерацией события R_1.
- Если R_1 - регулярное событие над алфавитом A, а R_2 - общерегулярное событие над алфавитом A, то $R_1 \cdot R_2$ - общерегулярное событие над алфавитом A.
- Если R_1, R_2 -общерегулярные сверхсобытия над алфавитом A, то $R_1 \cup R_2$ - общерегулярное сверхсобытие над алфавитом A.

Конечный автомат $\mathfrak{A}$ принимает множество сверхслов R_1^∞ с помощью множеств состояний $M = \{M_1, \ldots, M_t\}$, если автомат проходит все состояния из одного $M_i \in M$ бесконечное число раз и проходит оставшиеся состояния конечное число раз.

В любом автомате можно выделить сильно связные компоненты $Q_1, ..., Q_k$, то есть совокупность состояний, достижимых друг из друга $\forall q^{'}, q^{''} \in Q_i, i = 1, ..., k, \exists \alpha \in \{0,1\}^*$, т.ч. $\varphi(q^{'}, \alpha) = q^{''}$.

Назовём сильно связную компоненту замкнутой, если из неё не достижима никакая другая сильно связная компонента. Очевидно, что хотя бы одна замкнутая сильно связная компонента существует.

Тупиковым множеством будем называть такую сильно связную компоненту $Q^{'}$, что $\forall Q^{''} \in 2^{Q^{'}}, Q^{''} \not\subseteq M$.

Будем обозначать множество состояний, из которых самое близкое тупиковое множество [2] достигается за l рёбер как $Q(l)$

Теорема 1. *Имеется конечный автомат θ, принимающий множество сверхслов R. Пусть у этого автомата присутствуют тупиковые множества и максимальная длина пути до тупикового множества равна n_1. Подобному автомату можно задать выходную функцию таким образом, что его степень предугадывания множества R на отрезке длиной l будет больше или равна $\frac{\lfloor l/n_1 \rfloor}{l}$.*

Теорема 2. *Степень предугадывания конечного автомата на бесконечности всегда больше степени предугадывания конечного автомата на отрезке.*

Теорема 3. *Если у конечного автомата $\mathfrak{A}$ степень предугадывания на бесконечности строго больше нуля, то найдётся такая длина отрезка l, что степень предугадывания автомата $\mathfrak{A}$ на отрезке длиной l будет строго больше нуля.*

3. Вывод теорем

3.1. Вывод степени предугадывания на отрезке для предугадывающего автомата, полученного из принимающего

Теорема 1. *Имеется конечный автомат θ, принимающий множество сверхслов R. Пусть у этого автомата присутствуют тупиковые множества и максимальная длина пути до тупикового множества равна n_1. Подобному автомату можно задать выходную функцию таким образом, что его степень предугадывания множества R на отрезке длиной l будет больше или равна $\frac{\lfloor l/n_1 \rfloor}{l}$.*

Доказательство. Зададим функцию выхода автомату θ следующим образом:

- **1-ый шаг:**

  Для всех состояний из множества $Q(1)$ определим выходные функции следующим образом: если для состояния $q \in Q(1)$ и входного символа $x \in \{0,1\}$ справедливо что $\varphi(q,x) \in T$, то $f(q) = \overline{x}$.

- **i-ый шаг:**

  Для всех состояний из множества $Q(i)$ определим выходные функции следующим образом: если для состояния $q \in Q(i)$ и входного символа $x \in \{0,1\}$ справедливо что $\varphi(q,x) \in Q(i-1)$, то $f(q) = \overline{x}$

Для получения степени предугадывания на отрезке построим наихудшую последовательность символов для предугадывания. Ради этой цели начнём строить последовательность из состояния $q_0 \in Q(n_1)$. В состоянии q_0 мы можем либо угадать следующий символ и остаться во множестве состояний $Q(n_1)$, либо не угадать и перейти в состояние $Q(n_1 - 1)$. Верное предугадывание гарантируется лишь во множестве состояний $Q(1)$. Действительно, если из каждого множества $Q(i)$ автомат будет переходить во множество $Q(i-1)$, то он перейдёт во множество $Q(1)$ за n_1 шагов из состояния q_0. Наихудшим переходом из множества $Q(1)$ является переход во множество $Q(n_1)$. При переходе во множество $Q(n_1)$ автомат может ошибиться $n_1 - 1$ раз подряд в наихудшем случае, когда как при переходе во множество $Q(i), i < n_1$ автомат может ошибиться $i - 1$ раз подряд в наихудшем случае. Это означает, что при переходе во множество $Q(i), i < n_1$, автомат будет реже ошибаться чем при переходе во множество $Q(n_1)$, а значит степень предугадывания при таком построении не будет являться минимальной. При прохождении n_1 символов обязательно угадывается лишь один символ из них. Это соответствует тому, что при прохождении l символов гарантированно будет угадано лишь $\lfloor l/n_1 \rfloor$ символов. Теорема доказана. □

На Рис. 1 показан пример конечного автомата, который предугадывает множество сверхслов $(0(11 \cup 10)^*0)^\infty$ со степенью 0.5 на бесконечности, но с меньшей степенью на отрезке любой длины.

Рассмотрим слово $(00)^\infty$. Степень предугадывания на бесконечности этого сверхслова автоматом, изображённым на рис. 1, равняется 0.5. Степень предугадывания такого слова на отрезках длиной кратной двум также равняется 0.5.

Теперь рассмотрим слово $(010000)^\infty$. Степень предугадывания этого сверхслова автоматом, показанным на рис. 1, равняется 0.5 на бесконечности. Степень предугадывания данного автомата на отрезке длиной в 3 символа равняется $\frac{1}{3}$. Степень предугадывания данного автомата на отрезке длиной в 4 символа равняется $\frac{1}{4}$.

На Рис. 2 показан пример конечного автомата, который предугадывает множество сверхслов $(01^*00^*1)^\infty$ со степенью $0.(3)$ на бесконечности и на отрезках длиной кратной трём.

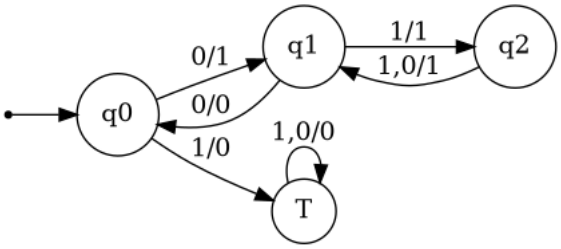


Рис. 1. Пример конечного автомата, у которого степени предугадывания на отрезке и бесконечности не совпадают.

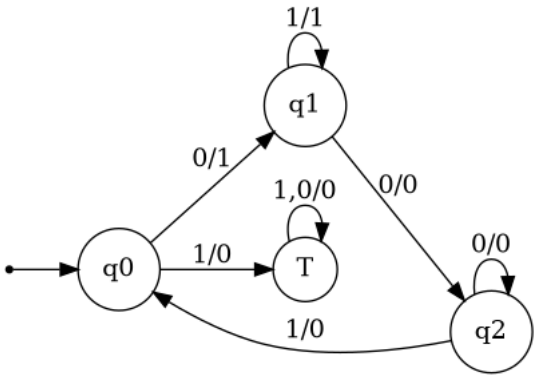


Рис. 2. Пример конечного автомата, у которого степени предугадывания на отрезке и бесконечности совпадают.

3.2. Вывод степени предугадывания на бесконечности из степени предугадывания на отрезке

Теорема 2. *Степень предугадывания конечного автомата на бесконечности всегда больше степени предугадывания конечного автомата на отрезке.*

Доказательство. Выпишем определение степени предугадывания на отрезке.

$$c^{\theta}_{t_1}(\alpha) = \sup_{N\in\mathbb{N}} \inf_{n\in\mathbb{N}} \{1 - \frac{\sum_{N+n}^{N+n+t_1} |y^{\theta}_{\alpha}(i) - \alpha(i+1)|}{t_1}\},$$

где N - длина конечного префикса, $t_1 \in \mathbb{N}$ - длина отрезка, на котором производится предугадывание.

Раскроем infinum.

$$S_{t_1} = \sum_{1}^{t1} |y_\alpha^{\mathfrak{A}}(i) - \alpha(i+1)| \leqslant t_1 - t_1 c_{t_1}^{\mathfrak{A}}(\alpha) \tag{5}$$

Выпишем определение степени предугадывания на бесконечности и раскроем предел.

$$c_\infty^{\mathfrak{A}}(\alpha) = 1 - \underline{\lim}_{t\to\infty} \frac{\sum_1^t |y_\alpha^{\mathfrak{A}}(i) - \alpha(i+1)|}{t}$$

$$c_\infty^{\mathfrak{A}}(\alpha) = 1 - \underline{\lim}_{t\to\infty} \frac{\sum_1^N |y_\alpha^{\mathfrak{A}}(i) - \alpha(i+1)| + \sum_{N+1}^{t/t_1} S_{t_1}}{t} =$$
$$1 - \underline{\lim}_{t\to\infty} \frac{(\frac{t}{t1} - N - 1)S_{t_1}}{t} = 1 - \frac{S_{t_1}}{t_1}$$

Заменим S_{t_1} неравенством (5).

$$c_\infty^{\mathfrak{A}}(\alpha) = 1 - \frac{S_{t_1}}{t_1} \geqslant 1 - \frac{t_1 - t_1 c_{t_1}^{\mathfrak{A}}(\alpha)}{t_1} = 1 - 1 + c_{t_1}^{\theta}(\alpha) = c_{t_1}^{\mathfrak{A}}(\alpha)$$

$$c_\infty^{\mathfrak{A}} \geqslant c_{t_1}^{\mathfrak{A}}$$

Теорема доказана. □

Замечание: В доказательстве не использовалось никаких уникальных свойств конечных автоматов, вследствие чего **теорема 2** распространяется как на конечные автоматы, так и на произвольные детерминированные автоматы.

На Рис. 3 приведён пример конечного автомата, который угадывает контекстно-свободное сверхслово $(0^n1^n)^\infty$ со степенью $\frac{1}{2}$ как на бесконечности, так и на отрезке, длина которого строго больше или равна двум.

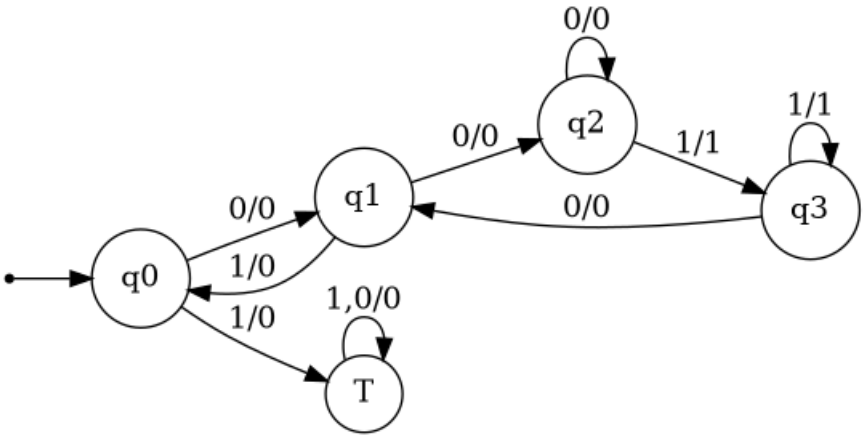


Рис. 3. Пример конечного автомата, который предугадывает контекстно свободный язык как на бесконечности, так и на отрезке.

3.3. Вывод степени предугадывания на отрезке из степени предугадывания на бесконечности для произвольного конечного автомата

Теорема 3. *Если у конечного автомата $\mathfrak{A}$ степень предугадывания на бесконечности строго больше нуля, то найдётся такая длина отрезка l, что степень предугадывания автомата $\mathfrak{A}$ на отрезке длиной l будет строго больше нуля.*

Доказательство. Покажем, что для любого конечного автомата $\mathfrak{A}$, который предугадывает множество сверхслов R_1 с нулевой степенью на отрезке любой длины, можно построить сверхслово из множества R_1, для которого $c_\infty^{\mathfrak{A}} = 0$. Возьмём конечный автомата $\mathfrak{A}$, предугадывающий множество сверхслов R_1. Допустим, что его степень предугадывания равна нулю для любой конечной длины отрезка: $c_l^{\mathfrak{A}} = 0$. Вместе с предугадывающим конечным автоматом $\mathfrak{A}$ рассмотрим принимающий конечный автомат $\mathfrak{B}$. Подавая одни и те же символы на вход конечных автоматов $\mathfrak{A}$ и $\mathfrak{B}$, сопоставим между собой пути в автомате $\mathfrak{A}$ и пути в автомате $\mathfrak{B}$. Все циклы автомата $\mathfrak{B}$ будем брать соответствующими одному конкретному сверхслову, принадлежащему R_1, степень предугадывания которого равна нулю на любой длине отрезка.

В начальный момент автомат $\mathfrak{A}$ будет находится в состоянии $q_{k_0}^{\mathfrak{A}}$, а автомат $\mathfrak{B}$ - в состоянии $q_{k_0}^{\mathfrak{B}}$. Будем подавать на оба автомата такую по-

следовательность символов α, на которой конечный автомат $\mathfrak{A}$ не предугадывает ни единого символа. Обозначим длину α как l: $|\alpha| = l$. Такую последовательность можно найти любой длины благодаря предположению, что степень предугадывания автомата $\mathfrak{A}$ на отрезке любой длины равна нулю. Получим последовательности состояний $\theta_1^{\mathfrak{A}} = q_{k_0}^{\mathfrak{A}} q_{k_1}^{\mathfrak{A}} q_{k_2}^{\mathfrak{A}} ... q_{k_l}^{\mathfrak{A}}$ для автомата $\mathfrak{A}$ и последовательность состояний $\theta_1^{\mathfrak{B}} = q_{k_0}^{\mathfrak{B}} q_{k_1}^{\mathfrak{B}} q_{k_2}^{\mathfrak{B}} ... q_{k_l}^{\mathfrak{B}}$ для автомата $\mathfrak{B}$, где $q_{k_i} = \varphi(pref(\theta_1, i), \alpha(i))$. Различные q_{k_i} могут обозначать одно и то же состояние. Подберём такую длину последовательности α и такое число $n \in \mathbb{N}, n < l$, что как $\theta_2^{\mathfrak{A}} = q_{k_n}^{\mathfrak{A}} q_{k_{n+1}}^{\mathfrak{A}} ... q_{k_l}^{\mathfrak{A}}$ так и $\theta_2^{\mathfrak{B}} = q_{k_n}^{\mathfrak{B}} q_{k_{n+1}}^{\mathfrak{B}} ... q_{k_l}^{\mathfrak{B}}$ будут являться циклами. Будем обозначать как α_n такую подпоследовательность слова α, при подаче которой на автоматы $\mathfrak{A}$ и $\mathfrak{B}$ они проходят циклы $\theta_2^{\mathfrak{A}}$ и $\theta_2^{\mathfrak{B}}$ соответственно. Такое n и слово α всегда возможно подобрать ввиду того, что слово α с нулевой степенью предугадывания можно найти произвольной длины, ввиду нулевой степени предугадывания на отрезке любой длины. Цикл же в любом конечном автомате всегда можно получить подав на него достаточно большое количество символов. В наихудшем случае он последовательно пройдёт все свои состояния, а затем перейдёт в одно из ранее пройденных состояний, чем образует цикл. Следующим шагом будем повторять циклы $\theta_2^{\mathfrak{A}}$ и $\theta_2^{\mathfrak{B}}$. При таком построение возможно два варианта:

- Сверхслово $pref(\alpha, n)(\alpha_n)^\infty$ проходит все состояния, которыми конечный автомат принимает множество сверхслов R_1. При подаче такого сверхслова на предугадывающий конечный автомат $\mathfrak{A}$ он будет бесконечно проходить цикл $\theta_2^{\mathfrak{A}}$, на котором он по построению не предугадывает ни одного символа. При этом сверхслово $pref(\alpha, n)(\alpha_n)^\infty$ принадлежит множеству R^∞. Вследствие этого степень предугадывания сверхслов $pref(\alpha, n)(\alpha_n)^\infty$ на бесконечности равняется нулю: $c_\infty^{\mathfrak{A}} = 0$.

- Сверхслово $pref(\alpha, n)(\alpha_n)^\infty$ не проходит все состояния, которыми конечный автомат $\mathfrak{B}$ принимает множество R_1. Тогда будем строить требуемое сверхслово таким образом:

  $$pref(\alpha, n)\alpha_n\beta\alpha_n\alpha_n\beta...(\alpha_n)^m\beta...,$$

  где β является словом по которому как принимающий так и предугадывающий автоматы проходят цикл. Слово β подбирается таким образом, чтобы при его подаче автомат $\mathfrak{B}$ прошёл все оставшиеся

состояния, необходимые для принятия сверхслова, а также конечное количество произвольных принимающих состояний. Такое слово β всегда можно подобрать ввиду конечного количества состояний у предугадывающего и принимающего автоматов. Сверхслово $pref(\alpha, n)\alpha_n\beta\alpha_n\alpha_n\beta...(\alpha_n)^m\beta...$ по построению принадлежит множеству R_1. Его степень предугадывания на бесконечности равняется нулю, вследствие стремления m к бесконечности и абсолютной непредугадываемости цикла $\theta_2^{\mathfrak{A}}$.

Раз можно построить по меньшей мере одно свехслово из множества со степенью предугадывания на бесконечности, равной нулю, то степень предугадывания на бесконечности у всего множества равняется нулю. Теорема доказана. □

Утверждение 1. *Для контекстно-свободных языков из предугадывания на бесконечности не следует предугадывание на отрезке.*

Приведём пример: Возьмём конечный автомат, выдающий на любой входной символ константу ноль. Попробуем предугадать им сверхслово $(0^n1^n)^\infty$. Очевидно, что степень предугадывания такого сверхслова этим автоматом будет равняться 0.5. Однако подслово 1^n может иметь любую конечную длину и более того, оно повторяется в сверхслове. Из-за повторений подслова его невозможно пропустить, выбросив конечное число символов из начала сверхслова. Получается, что насколько большой отрезок мы не брали бы - всегда можно найти такую последовательность 1^n, что её длина будет больше нежели длина отрезка, на котором мы предугадываем сверхслово. Подобный отрезок может повторяется бесконечное число раз в сверхслове. Из этого следует, что степень предугадывания на отрезке у такого автомата равняется нулю.

Итого мы получили пару автомат - сверхслово, у которой степень предугадывания на бесконечности равняется 0.5, а степень предугадывания на отрезке равняется нулю.. Данное наблюдение доказывает утверждение.

Список литературы

[1] Вереникин А.Г., Гасанов Э.Э., “Об автоматной детерминизации множеств сверхслов”, *Дискретная математика*, **18**:2 (2006), 84–97.

[2] Мастихина А.А., “Критерий частичного предвосхищения общерегулярных свехсобытий”, *Дискретная математика*, **23**:4 (2011), 103–114.

[3] Tim Smith, “Prediction of Infinite Words with Automata”, 2016, arXiv: https://arxiv.org/abs/1603.02597.

[4] S. E. Marzen, J. P. Crutchfield, “Probabilistic Deterministic Finite Automata and Recurrent Networks, Revisited”, 2019, arXiv: https://arxiv.org/abs/1910.07663.

Prediction of superword segments
Manshilin O.G.

The automaton predicts the character of the input sequence, if it outputs this character at the previous time.

The paper introduces the concept of the degree of prediction on a superword segments. The question of the relationship between prediction on superword segments and prediction on superwords is investigated.

We obtained results that allow us to judge the degree of prediction on superword segments if we know degree of prediction on superword and vice versa.

Keywords: predicting automaton, prediction of superwords with automatons, prediction degree on finite subsequency.

Разрешимость задачи определения порядка линейного автомата

Муравьев Н.В.[1]

Рассматривается задача определения порядка линейного автомата. Доказан критерий конечности порядка линейного автомата, позволяющий решать задачу алгоритмически. Дана верхняя оценка на порядок линейного автомата.

Ключевые слова: конечные автоматы, линейные автоматы, порядок в полугруппе.

1. Введение

В 2017 году Pierre Gillibert доказал [1], что задача определения порядка элемента в автоматной группе (группе, порожденной конечным автоматом) алгоритмически неразрешима. Позже этот результат был усилен Bartholdi и Митрофановым [2]. В частности из него следует, что, начиная с некоторого n, задача определения порядка элемента неразрешима и в группе всех обратимых автоматов AS_n. Представляет интерес нахождение таких классов автоматов, для которых задача разрешима.

Одним из важнейших классов конечных автоматов с выходом является класс линейных автоматов. Данное семейство автоматов имеет большое значение как с теоретической, так и с практической точек зрения. На сегодняшний день существует большое количество книг [3] и статей [7, 8], посвященных этой тематике.

Ранее Алешин С.В. показал [5], что в группе одномерных линейных автоматов над полем из двух элементов автомат имеет конечный порядок тогда и только тогда, когда его переходы безусловны. В данной работе

[1] *Муравьев Никита Валерьевич* — студент каф. математической теории интеллектуальных систем мех.-мат. ф-та МГУ, e-mail: ne-ki-tos@yandex.ru .

Muravev Nikita Valerevich — student, Lomonosov Moscow State University, Faculty of Mechanics and Mathematics, The Department of Mathematical Theory of Intellectual Systems.

этот результат будет обобщен на линейные автоматы любой размерности над произвольным конечным полем. Более того, будет выведена верхняя оценка на порядок линейного автомата, зависящая от его размерности и поля, над которым он линеен.

2. Базовые определения и утверждения

Определение 2.1. Абстрактным инициальным конечным автоматом (далее автомат) называется шестерка $(\Sigma, Q, \Omega, \phi, \psi, q_0)$, где

Σ - конечное непустое множество, называемое входным алфавитом,
Q - конечное непустое множество, называемое множеством состояний,
Ω - конечное непустое множество, называемое выходным алфавитом,
$\phi : \Sigma \times Q \to Q$ - функция переходов,
$\psi : \Sigma \times Q \to \Omega$ - функция выходов,
$q_0 \in Q$ - начальное состояние.

Определение 2.2. Заметим, что всякий автомат U с начальным состоянием q_0 задает отображение $f_{U_{q_0}} : \Sigma^* \to \Omega^*$, определенное индуктивно и называющееся автоматной функцией:

1) $f_{U_{q_0}}(\varepsilon) = \varepsilon$, где ε - пустое слово
2) $a \in \Sigma \Rightarrow f_{U_{q_0}}(a) = \psi(a, q_0)$
3) $a_0 a \in \Sigma^*, a_0 \in \Sigma \Rightarrow f_{U_{q_0}}(a_0 a) = \psi(a_0, q_0) f_{U_{\phi(a_0, q_0)}}(a)$.

Определение 2.3. Автоматы, задающие одинаковые автоматные функции, называются эквивалентными.

Определение 2.4. Два состояния автоматов (двух разных или одного и того же) называются эквивалентными, если инициальные автоматы с началом в этих состояниях эквивалентны.

Определение 2.5. Автоматы, между множествами состояний которых существует биекция, переводящая состояния в эквивалентные им, называются изоморфными.

Известна следующая теорема [4]:

Теорема 1. *Для каждого автомата существует единственный с точностью до изоморфизма эквивалентный автомат с наименьшим возможным количеством состояний.*

Такой автомат называется минимальным или приведенным для исходного автомата.

В дальнейшем мы зачастую будем отождествлять понятия инициального автомата U_{q_0} и его автоматной функции $f_{U_{q_0}}$, если это не будет вызывать путаницы.

При совпадении входного и выходного алфавитов множество обратимых автоматных функций (обратимых автоматов) образует группу относительно суперпозиции, которую мы будем обозначать AS_n, где n - мощность входного-выходного алфавита.

Если отказаться от требования обратимости, то мы получим полугруппу с единицей (моноид) автоматных функций. Этот моноид обозначим AP_n, где n - мощность входного-выходного алфавита.

Очевидно, что при разных алфавитах одной и той же мощности соответствующие группы (моноиды) автоматов изоморфны, так что данное обозначение корректно.

Определение 2.6. Каноническими уравнениями автомата $U = (\Sigma, Q, \Omega, \phi, \psi, q_0)$ называется следующая система уравнений:

$$\begin{cases} q(t+1) = \phi(q(t), x(t)) \\ y(t) = \psi(q(t), x(t)) \\ q(0) = q_0, \end{cases}$$

где $t \in \mathbb{N} \cup \{0\}, q(t) \in Q, x(t) \in \Sigma, y(t) \in \Omega$.

Ясно, что эта система задает функцию $f_{U_{q_0}}$, а именно

$$f_{U_{q_0}}(x(0)x(1)x(2)...) = y(0)y(1)y(2)... \quad \forall x(t) \in \Sigma, t \in \mathbb{N} \cup \{0\}.$$

Определение 2.7. Абстрактный инициальный автомат $(\Sigma, Q, \Omega, \phi, \psi, q_0)$ назовем линейным над конечным полем F_m (где m - мощность поля), если его множество состояний Q, входной Σ и выходной Ω алфавиты есть подмножества конечномерных векторных пространств над F_m, а канонические уравнения имеют следующий вид:

$$\begin{cases} q(t+1) = Aq(t) + Bx(t) \\ y(t) = Dq(t) + Lx(t) \\ q(0) = q_0, \end{cases}$$

где $q(t) \in Q, x(t) \in \Sigma, y(t) \in \Omega$; A, B, D, L - линейные операторы между соответствующими пространствами.

Ввиду изоморфизма конечномерных линейных пространств одинаковой размерности над общим полем, в дальнейшем будем всегда полагать, что линейные пространства имеют вид F_m^n, где n - размерность.

Лемма 1. *После минимизации линейного над F_m автомата мы получаем автомат, изоморфный некому линейному над тем же полем.*

Доказательство. Пусть мы склеили два эквивалентных состояния линейного автомата. Если два состояния эквивалентны, то они реализуют одну и ту же функцию, а значит их разность реализует ту же функцию, что и нулевой вектор (нулевое состояние). Получается, что если состояния q_1 и q_2 эквивалентны, то и состояния $q, q + k(q_1 - q_2)$ эквивалентны для любых $q \in Q, k \in F_m$. А значит мы можем спроецировать пространство состояний на любую гиперплоскость (подпространство коразмерности один), не параллельную $q_1 - q_2$, (это линейное отображение) и домножить слева линейные операторы в функции переходов на оператор проецирования, получив автомат, эквивалентный исходному. Повторяем данную процедуру, пока не придем к приведенному автомату. Получили линейный минимальный автомат, эквивалентный исходному. □

Данная лемма показывает, что при изучении линейных автоматов мы можем ограничиться лишь минимальными (приведенными) автоматами.

Далее все автоматы считаются приведенными.

Заметим, что обратимость линейного автомата эквивалентна обратимости оператора L из его канонических уравнений на линейной оболочке $\langle\Sigma\rangle$.

Лемма 2. *Суперпозиция линейных над F_m автоматов есть линейный над F_m автомат.*

Доказательство. Пусть имеется линейный автомат G с каноническими уравнениями

$$\begin{cases} q(t+1) = Aq(t) + Bx(t) \\ y(t) = Dq(t) + Lx(t) \\ q(0) = q_0 \end{cases}$$

и линейный автомат G' над тем же полем, чей выходной алфавит совпадает с входным алфавитом автомата G и заданный каноническими уравнениями

$$\begin{cases} q'(t+1) = A'q'(t) + B'x(t) \\ y(t) = D'q'(t) + L'x(t) \\ q'(0) = q'_0. \end{cases}$$

Тогда ясно, что их суперпозиция $G \circ G'$ задается следующими каноническими уравнениями:

$$\begin{cases} \begin{pmatrix} q'(t+1) \\ q(t+1) \end{pmatrix} = \begin{pmatrix} A' & 0 \\ BD' & A \end{pmatrix} \begin{pmatrix} q'(t) \\ q(t) \end{pmatrix} + \begin{pmatrix} B' \\ BL' \end{pmatrix} x(t) \\ y(t) = \begin{pmatrix} LD' & D \end{pmatrix} \begin{pmatrix} q'(t) \\ q(t) \end{pmatrix} + LL'x(t) \\ \begin{pmatrix} q'(0) \\ q(0) \end{pmatrix} = \begin{pmatrix} q'_0 \\ q_0 \end{pmatrix}. \end{cases}$$

То есть суперпозиция есть линейный автомат. □

Теорема 2. *Обратный автомат к линейному эквивалентен линейному над тем же полем.*

Доказательство. Пусть имеется линейный обратимый автомат G с каноническими уравнениями

$$\begin{cases} q(t+1) = Aq(t) + Bx(t) \\ y(t) = Dq(t) + Lx(t) \\ q(0) = q_0. \end{cases}$$

Докажем, что обратным к нему будет автомат G' с со следующими каноническими уравнениями:

$$\begin{cases} q'(t+1) = (A - BL^{-1}D)q'(t) + BL^{-1}x(t) \\ y(t) = -L^{-1}Dq'(t) + L^{-1}x(t) \\ q'(0) = q_0, \end{cases}$$

где L^{-1} - обратный оператор к сужению L на $\langle\Sigma\rangle$, произвольно продолженный на все пространство.

Рассмотрим их суперпозицию $G \circ G'$:

$$\begin{cases} \begin{pmatrix} q'(t+1) \\ q(t+1) \end{pmatrix} = \begin{pmatrix} A - BL^{-1}D & 0 \\ -BL^{-1}D & A \end{pmatrix} \begin{pmatrix} q'(t) \\ q(t) \end{pmatrix} + \begin{pmatrix} BL^{-1} \\ BL^{-1} \end{pmatrix} x(t) \\ y(t) = \begin{pmatrix} -D & D \end{pmatrix} \begin{pmatrix} q'(t) \\ q(t) \end{pmatrix} + LL^{-1}x(t) \\ \begin{pmatrix} q'(0) \\ q(0) \end{pmatrix} = \begin{pmatrix} q_0 \\ q_0 \end{pmatrix}. \end{cases} \Rightarrow$$

$$\overset{q(t)=q'(t)}{\Rightarrow} \begin{cases} \begin{pmatrix} q'(t+1) \\ q(t+1) \end{pmatrix} = \begin{pmatrix} (A - BL^{-1}D)q(t) \\ (A - BL^{-1}D)q(t) \end{pmatrix} + \begin{pmatrix} BL^{-1} \\ BL^{-1} \end{pmatrix} x(t) \\ y(t) = x(t) \\ \begin{pmatrix} q'(0) \\ q(0) \end{pmatrix} = \begin{pmatrix} q_0 \\ q_0 \end{pmatrix}. \end{cases}$$

То есть G' - обратный к G. □

Следствие 2.1. *Множество линейных над F_m автоматов с совпадающими входными-выходными алфавитами Σ образует моноид относительно суперпозиции.*

Обозначим этот моноид LP_Σ.

Следствие 2.2. *Множество обратимых линейных над F_m автоматов с совпадающими входными-выходными алфавитами Σ образует группу относительно суперпозиции.*

Обозначим эту группу LS_Σ.

3. Определение порядка автомата в моноиде LP_Σ

Без ограничения общности везде далее считаем, что размерности линейных оболочек алфавитов и линейных оболочек множеств состояний совпадают с размерностями соответствующих векторных пространств, подмножествами которых они являются.

Определение 3.1. Порядком элемента в полугруппе будем называть мощность подполугруппы, порожденной данным элементом.

Определение 3.2. С каждым линейным автоматом G

$$\begin{cases} q(t+1) = Aq(t) + Bx(t) \\ y(t) = Dq(t) + Lx(t) \\ q(0) = q_0 \end{cases}$$

свяжем два формальных ряда:

$$M_G(z) = \sum_{v=0}^{\infty} DA^v Bz^{v+1} + L,$$

$$S_G(z) = \sum_{v=0}^{\infty} DA^v q_0 z^v.$$

Назовем их передаточной функцией и сдвигом соответственно.

Для произвольного поля K обозначим $Frac(K[z])$ поле частных кольца многочленов над K от переменной z.

Следующие две леммы являются обобщениями известных результатов [4]. Для сокращения изложения мы опустим их доказательства.

Лемма 3. *Для любого линейного автомата G над полем F_m его передаточная функция $M_G(z)$ есть линейный оператор над полем $Frac(F_m[z])$*

Лемма 4. *Сопоставим каждому слову $x = x_0x_1x_2x_3... \in \Sigma^\infty$ формальный ряд*

$$x(z) = \sum_{v=0}^{\infty} x_v z^v,$$

А каждому слову $y = y_0y_1y_2y_3... \in \Omega^\infty$ - формальный ряд

$$y(z) = \sum_{v=0}^{\infty} y_v z^v.$$

Тогда для любого линейного автомата G и любых $x \in \Sigma^\infty, y \in \Omega^\infty$

$$y = G(x) \Leftrightarrow y(z) = M_G(z)x(z) + S_G(z).$$

Лемма 4 позволяет нам переформулировать задачи для линейных автоматов в терминах передаточных функций и сдвигов. Например верна

Лемма 5. *Пусть G есть линейный над F_m автомат, тогда его порядок конечен тогда и только тогда, когда $\exists k, n \in \mathbb{N} : M_G^k(z) = M_G^n(z), k \neq n$.*

Доказательство. По определению порядок автомата конечен, когда конечна порожденная им полугруппа. То есть в бесконечной последовательности $G, G^2, G^3, ...$ есть лишь конечное число различных элементов, а значит $\exists k, n \in \mathbb{N} : G^k = G^n, k \neq n$. Из леммы 4 и замечания о том, что линейные оболочки алфавитов совпадают с содержащими их векторными пространствами, следует, что по автомату однозначно определяется его передаточная функция, а значит

$$G^k = G^n \Rightarrow M_G^k(z) = M_G^n(z).$$

Теперь докажем, что из существования таких различных $k, n \in \mathbb{N}$, что $M_G^k(z) = M_G^n(z)$, следует конечность порядка автомата G. Если $M_G^k(z) = M_G^n(z)$, то $M_G^{k+l}(z) = M_G^{n+l}(z)$ для любого $l \in \mathbb{N}$. Получается, что последовательность передаточных функций степеней автомата периодическая с периодом $|n-k|$, а последовательность сдвигов степеней автомата имеет вид

$$S_G(z),\ (M_G(z)+I)S_G(z),\ (M_G^2(z)+M_G(z)+I)S_G(z),\ ...$$

где I - единичная матрица.

Из чего следует, что она тоже периодическая с периодом $p \cdot |n-k|$, где p - характеристика поля F_m. Но если последовательности передаточных функций и сдвигов периодические, то по лемме 4 и последовательность степеней автомата периодическая. □

Введем некоторые понятия, которые нам понадобятся в дальнейшем, и напомним несколько алгебраических свойств конечных полей.

Определение 3.3. Размерностью линейного автомата будем называть размерность линейной оболочки его входного-выходного алфавита.

Для любого простого p и любых натуральных m, n, таких что $m|n$, в поле F_{p^n} содержится ровно одно подполе порядка p^m.

Из этого следует, что имеет место следующая цепочка включений

$$F_p \subseteq F_{p^{2!}} \subseteq F_{p^{3!}} \subseteq ... \subseteq F_{p^{n!}} \subseteq ...$$

Известно, что корни многочлена порядка n над полем F_m лежат в поле F_{m^n}.

Из этого следует, что $F_{p^\infty} = \bigcup_{k=1}^{\infty} F_{p^{k!}}$ является алгебраическим замыканием поля F_{p^n} при любом натуральном n и простом p.

Теперь мы готовы доказать основную теорему.

Теорема 3. *Порядок линейного над F_m автомата конечен тогда и только тогда, когда коэффициенты характеристического многочлена его передаточной функции есть константы из поля F_m.*

Доказательство. По лемме 5 порядок линейного над F_m автомата G конечен тогда и только тогда, когда конечен порядок его передаточной функции $M_G(z)$.

Пусть порядок n-мерного автомата G конечен. Тогда конечен порядок его передаточной функции. По лемме 3 передаточная функция есть линейный оператор над полем $Frac(F_m[z])$, а значит она линейный оператор и над полем $Frac(F_{p^\infty}[z])$. Порядок оператора конечен, а значит конечны и порядки по умножению его собственных значений (ведь при возведении матрицы в степень собственные значения тоже возводятся в степень). Следовательно собственные значения, лежащие в алгебраическом замыкании поля $Frac(F_{p^\infty}[z])$, имеют конечные порядки. Но тогда они либо нули, либо корни из единицы. А все корни из единицы в поле $Frac(F_{p^\infty}[z])$ лежат в F_{p^∞}. Это позволяет переписать уравнение на собственные значения

$$\sum_{i=0}^{n}(a_{i,k}z^k + ...a_{i,1}z + a_{i,0})\lambda^i = 0,\ a_{i,j} \in F_m, k \in \mathbb{N} \cup \{0\}$$

в виде системы

$$\begin{cases} a_{n,k}\lambda^n + ... + a_{1,k}\lambda + a_{0,k} = 0 \\ ... \\ a_{n,0}\lambda^n + ... + a_{1,0}\lambda + a_{0,0} = 0. \end{cases}$$

Получили систему линейных уравнений над полем F_m. Как было замечено ранее, ее решения лежат в поле F_{m^n}. Следовательно собственные значения передаточной функции лежат в поле F_{m^n}. Но тогда и коэффициенты характеристического многочлена лежат в F_{m^n}. С другой стороны коэффициенты характеристического многочлена есть произведения и суммы элементов матрицы $M_G(z)$, то есть они принадлежат полю $Frac(F_m[z])$. Итого коэффициенты характеристического многочлена для передаточной функции $M_G(z)$ лежат в $Frac(F_m[z]) \cap F_{m^n} = F_m$. Что и требовалось показать.

Осталось доказать теорему в другую сторону. Если коэффициенты характеристического многочлена передаточной функции автомата есть константы из поля F_m, то порядок автомата конечен.

Обозначим эти коэффициенты $M_0, ..., M_{n-1}$ и запишем характеристический многочлен:

$$\lambda^n + M_{n-1}\lambda^{n-1} + ... + M_0.$$

По теореме Гамильтона-Кэли характеристический многочлен аннулирует свою матрицу, то есть

$$M_G(z)^n + M_{n-1} \cdot M_G(z)^{n-1} + ... + M_0 = 0.$$

Данное равенство можно домножить на $M_G(z)^{s-n}$ для любого $s > n$ и получить

$$M_G(z)^s + M_{n-1} \cdot M_G(z)^{s-1} + ... + M_0 \cdot M_G(z)^{s-n} = 0.$$

Таким образом мы выразили s-ю степень (при $s > n$) передаточной функции автомата через линейную комбинацию с константными коэффициентами предыдущих n степеней. Следовательно всякая степень передаточной функции есть линейная комбинация с константными коэффициентами первых n степеней передаточной функции (начиная с нулевой). Таких комбинаций конечное число, ведь их коэффициенты принадлежат конечному полю F_m. А значит и порядок передаточной функции конечен. Откуда следует конечность порядка всего автомата. □

Следствие 3.1. *Порядок n-мерного линейного над F_m автомата G не превышает $p(m^n - 1)$, где p - характеристика поля F_m.*

Доказательство. При доказательстве предыдущей теоремы было показано, что степень передаточной функции $M_G(z)$ есть линейная комбинация с константными коэффициентами первых n степеней передаточной функции. Так как коэффициенты принадлежат полю F_m, всего таких комбинаций m^n. Однако заметим, что комбинация из одних нулей возможна тогда и только тогда, когда передаточная функция нильпотентна, и в таком случае ее порядок не превышает n. Если же она не нильпотентна, мы имеем лишь $m^n - 1$ возможных комбинаций. Итого порядок передаточной функции не превышает $max\{m^n - 1, n\} = m^n - 1$.

Однако автомат определяется не только своей передаточной функцией, но и сдвигом. Сдвиг автомата G^n имеет вид

$$(M_G^{n-1}(z) + ... + M_G(z) + I)S_G(z).$$

Если k, l наименьшие натуральные числа, для которых $M_G^k(z) = M_G^l(z), \quad k < l,$ то либо $(M_G^{l-1}(z) + ... + M_G^k(z))S_G(z) = 0$ и

порядок автомата совпадает с числом различных степеней его передаточной функции, либо $(M_G^{l-1}(z) + ... + M_G^k(z))S_G(z) \neq 0$ и порядок автомата равен $l - 1 + (p - 1)(l - k)$. Заметим, что $l - 1$ есть порядок передаточной функции $M_G(z)$ и $l - k \leq l - 1$. То есть порядок автомата не превышает $p(m^n - 1)$. □

В заключение автор выражает благодарность своему научному руководителю Бабину Д.Н. за постановку задачи и ценные указания и замечания по ходу работы.

Список литературы

[1] P. Gillibert, "An automaton group with undecidable order and Engel problems", *preprint, available online at arxiv.org/abs/1710.09733*, 2017.

[2] L. Bartholdi, I. Mitrofanov, "The word and order problems for self-similar and automata groups", *preprint, available online at arxiv.org/abs/1710.10109*, 2017.

[3] Гилл А., *Линейные последовательностные машины*, "Наука", Москва, 1974.

[4] Кудрявцев В.Б., Алешин С.В., Подколзин А.С., *Введение в теорию автоматов*, "Наука", Москва, 1985.

[5] Алешин С.В., *Алгебраические системы автоматов.*, "МАКС Пресс", Москва, 2016.

[6] Винберг Э.Б., *Курс алгебры (2-е изд.)*, "Факториал Пресс", Москва, 2001.

[7] Бабин Д.Н., "Автоматы с линейными переходами", *Интеллектуальные системы. Теория и приложения*, **23**:3 (2019), 87-95.

[8] Часовских А.А., "О полноте в классе линейных автоматов", *Математические вопросы кибернетики*, 1995, №3, 140–166.

Decidability of the order problem for linear automata
Muravev N.V.

We consider the order problem for linear automata. A finite order criterion for linear automata is presented that provides an algorithm solving this problem. An upper bound of linear automata orders is proved.

Keywords: finite automata, linear automata, order in semigroup.

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: 10.06.2020
Дата выхода: 25.06.2020
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).