С.И. Гуров[1], Д.В. Золотарев[2], А.И. Самбурский[3]

ИСКУСТВЕННЫЕ НЕЙРОННЫЕ СЕТИ И СИНТЕЗ ЛОГИЧЕСКИХ СХЕМ*

1. Введение

Появление новых технологий начинает оказывать существенное влияние на методы автоматизации проектирования электронных систем. Здесь актуальными остаются задачи уменьшения размеров схем, увеличения срока их службы, уменьшения глубины схемы, понижения вероятности смены состояния элемента и др.

Наиболее ярким примером новых подходов к традиционным задачам является появление алгоритмов синтеза оптимизации логических схем, использующих методы глубокого обучения искусственных нейронных сетей (ИНС). При этом логическая оптимизация рассматривается как детерминированный *марковский процесс принятия решений* (Markov decision process, MDP), к которой, с целью получения оптимальной схемы, применяются методы глубокого обучения с подкреплением.

Глубокое обучение - это подход к машинному обучению, основанный на ИНС, см., e. g. [1]. В последние годы развитие глубокого обучения произвело революцию в машинном обучении. В отличие от обычных нейронных сетей, *глубокие нейронные сети* (deep neural networks, DNN) объединяют множество скрытых слоев вместе, что позволяет выполнять более сложную обработку обучающих данных [2].

Среди всех типов ИНС особый интерес представляет *свёрточные нейронные сети* (convolutional neural networks, CNN), используемые, в первую очередь для обработки изображений [3]. На каждом своём уровне они используют фильтры, инвариантные к сдвигу и операцию свертки сигналов. В качестве нелинейной функции возбуждения обычно применяют *выпрямленную линейность* (rectified-linearity, ReLU), см. рис. 1. Указанные свойства CNN позволяют применять для их настройки эффективные методы градиентного спуска.

Успешным показал себя подход, объединяющий глубокое обучение и обучение с подкреплением [4]. Традиционные ИНС обучаются в рамках

[1]МГУ имени М. В. Ломоносова, e-mail: sgur@cs.msu.ru
[2]МГУ имени М. В. Ломоносова, e-mail: ub8caf@mail.ru
[3]МГУ имени М. В. Ломоносова, e-mail: sashasamb@yandex.ru
*Работа финансировалась грантом РНФ 17-19-01645 (2020-2021) «Разработка методов и средств проектирования реконфигурируемых систем на кристалле повышенной надежности».

парадигмы *обучения с учителем*, когда неизвестная зависимость между входом и выходом восстанавливается с помощью оценки (часто - бинарной), даваемых «учителем» данному выходу. При этом используется таблица прецедентов - объектов (входов) с известной классификацией.

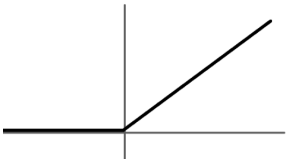

Рис. 1. Функция активации ReLU.

При *обучении с подкреплением* (reinforcement learning, RL) обучение проходит в результате взаимодействия *агента* и *внешней среды*.

В зависимости от предпринятого действия агент получает от внешней среды вознаграждение или штраф. Стремясь максимизировать своё вознаграждение, используя управляемый случайный поиск, агент выстраивает свою *политику* (стратегию), оптимизирующую его реакции на подаваемые входы. В результате последних исследований возникает специальный вид *глубоких ИНС с подкреплением* (DNN+RL) [4], [5].

2. Основные понятия. Состояния и переходы

Пример эффективного использования DNN с подкреплением для автоматической оптимизации логических схем описан в [6].

Напомним основные понятия и определения, связанные с задачами синтеза комбинационно-логических схем. Данные схемы будем представлять направленными ациклическими графами, вершины которых соответствуют логическим функциям, а направление дуг - естественному ходу вычислений. Входящие дуги связывают вершину либо с другими вершинами, либо с входящими переменными, либо с логическими константами 0/1. В логической схеме дуги разделяют на *прямые* и *инвертирующие*, изменяющие значение проходящего сигнала на противоположное. Две логические схемы называются э*квивалентными*, если они вычисляют одну и ту же логическую функцию. *Глубина* логической схемы есть максимальное количество вершин на пути в графе логической схемы, её *размер* - количество вершин в графе логической схемы.

Процесс оптимизации логической сети представляется как детерминированный марковский процесс принятия решений, который, в свою очередь, можно рассматривать как игру одного игрока с полной информацией. Такая игра описывается набором состояний *s* переходами от одного состояния в другое. Состояния игры соответствуют логическим схемам, а ходы *a* в игре - преобразованиям этих схем.

Ход или *преобразование* a: $s \to s'$ схемы s в эквивалентную ей схему s' называют *допустимым*. Множество допустимых ходов из состояния s обозначают $moves(s)$. Допустимость ходов зависит от конкретной логической схемы и типов разрешённых логических преобразований.

В большинстве работ, использующих DNN для оптимизации логических схем, последние представляются в виде *графов мажорирования/инвертирования* (majority inverter graphs, MIGs). Данные графы являются новой перспективной структурой для логического представления и эффективной реализации булевых функций [7], [8]. Указанные графы тесно связаны с медианной алгеброй, (конкретно с тернарной алгеброй большинства) [9]. Рассматриваемое представление схем позволяет описать полный и адекватный набор ходов-преобразований, позволяющих найти любую схему, эквивалентную данной. В двух следующих разделах представим результаты [6], иллюстрирующие плодотворность применения ИНС для целей синтеза схем.

Запись $s \overset{a}{\to} s'$ означает, применение $a \in moves(s)$ к схеме s преобразует её в схему s'. Для каждого состояния определяют *тождественное (единичное) преобразование* $s \overset{\varepsilon}{\to} s$, что приводит к непустоте набора разрешенных ходов. Теперь можно описать упомянутую игру одного игрока с полной информацией как конечную n-последовательность допустимых преобразований схем от исходной s_0 к заключительной s_n:

$$s_0 \overset{a_1}{\to} s_1 \overset{a_2}{\to} \dots s_{n-1} \overset{a_n}{\to} s_n.$$

Цена игры определяется *функцией оценки состоя*ния $score(s)$, которая показывает, насколько схема s соответствует критерию оптимизации.

Пусть, например, требуется оптимизировать размер логической схемы. Тогда значением $score(s)$ может быть величина, обратная размеру схемы. Это соответствует игре, цель которой - найти схему минимального размера, достижимую за n ходов.

3. Глубокое обучение для синтеза схем

Пусть MIG находится в состоянии s_t, и на шаге t преобразование a_t переводит его в новое состояние s_{t+1}. Пусть в результате выполнения этого преобразования схема оптимизируется, за что система получает вознаграждение $r(s_t, a_t)$. Это вознаграждение можно определить через разность значений функции оценки состояний: $r(s_t, a_t) = score(s_{t+1}) - score(s_t)$, что сокращённо записывают как $r_t = r(s_t, a_t)$.

Суммирование получаемых вознаграждений приводит к упрощению

$$\sum_{t=0}^{n} r_t = \sum_{t=0}^{n-1} (score(s_{t+1}) - score(s_t)) = score(s_n) - score(s_0).$$

Поскольку оценка начального состояния s_0 общая для всех начинающихся с него последовательностей, для удовлетворения критерию оптимальности необходимо максимизировать значение указанной суммы. Максимизацию вознаграждения обычно выгоднее производить на данном шаге, не откладывая её на конечные ходы (их число ограничено значением n). Для учёта этого обстоятельства применяют процедуру дисконтирования вознаграждений с течением времени с применением *коэффициента дисконтирования по времени*, $0 < \lambda \leqslant 1$, в результате чего критерий оптимизации принимает вид

$$arg \max_{\pi} \mathbb{E}_{\pi} \left[\sum_{k=t}^{n} \lambda^k r_k \right].$$

Оптимизируемое выражение называется ожидаемой *наградой (доходом*), а математическое ожидание $\mathbb{E}$ берется по функционалу π, называемому *политикой*. Функционал политики с учётом текущего состояния присваивает каждому возможному ходу ту или иную вероятность; он зависит от параметра θ. Если ввести вероятность $\pi_\theta(a_t|s_t)$ применения преобразования a_t к состоянию s_t, то поиск оптимальной политики сведётся к поиску оптимального набора параметров θ.

Для определения политики естественно применить *метод градиента* (Policy Gradient, PG) [10]. В соответствии с PG перерасчёт параметра на каждом шаге осуществляется по правилу

$$\theta_{t+1} = \theta_t + \alpha\big(r_t \nabla \pi_\theta(a_t|s_t)\big),$$

где $\alpha > 0$ - скорость обучения. Формула (2) реализует интуитивно ясное условие: ход, приводящий к улучшению схемы подкрепляется пропорционально градиенту логарифмической вероятности шага, а ходы, ухудшающие схему не поощряются.

Конкретно в [6] функцию политики строит глубокая нейронной сеть на основе сетей свертки графов (Graph Convolution Networks, GCN) [11]. Здесь для каждой вершины MIG требуется получить информацию о целесообразности перехода к той или иной соседней вершине. Данная информация представляется вектором параметров, зависит от характеристик соседних вершин и получается с помощью серии L преобразований, соответствующих какому-либо количеству слоев свертки графа.

Слой l в сети GCN связан с $p \times d_l$-матрицей характеристик данной вершины графа. Каждая строка матрицы описывает одну из p вершин MIG

как d-мерный вектор признаков. Слой l следующим образом преобразует матрицу признаков:

$$F^{(l+1)} = \sigma\left(AF^{(l)}W^{(l)}\right),$$

где A - матрица смежности графа, а σ - нелинейная функция возбуждения (в нашем случае – ReLU). Использование матрицы смежности позволяет осуществлять передачу информации между соседними узлами. Конкретно, включение в рассмотрение соседей L-й степени имеет эффект расширения окрестности получения характеристик для конкретного узла. L называют размером *поля рецепторов*.

В начале процесса обучения DNN матрица признаков инициализируется с p-мерным вектором параметров каждого узла в графе. Таким образом, $d_0 = p$, что даёт единичную матрицу: $F^{(0)} = I$. Чтобы получить вероятности перехода к конкретной соседней вершине осуществляют определение информационных характеристик по всем слоям CGN, и затем анализируются два зависящих от неё слоя.

Набор θ параметров политики определяют параметры DNN. Начальные значения θ случайны, но в ходе игры они постоянно обновляются, стремясь к оптимальным. Обучение продолжается за n шагов-ходов, на каждом из которых выполняется процедура *выкатывания* (rollout). Эта процедура осуществляет виртуальное проигрывание (симуляцию) игры до конца, в результате вычисляется ожидаемая награда.

Из назначенного набора начальных состояний вручную выбираются начальные состояния, с которых можно начать развертывание. Они, например, могут соответствовать MIG, описывающим разложения Шеннона всех булевых функций с 3 входами. На каждой итерации набор начальных состояний дополняются путем случайной выборки MIG из истории развертывания.

4. Результаты оптимизации размера схем

Напомним, что существует всего 2^{2^n} булевых функций от n переменных. Булевы функции допускают разложение Шеннона: например, представление функции от 3-х переменны по переменной x, имеет вид

$$f(x, y, z) = xf(1, y, z) \vee \bar{x}f(0, y, z).$$

Набор данных описываемого алгоритма для функций от 3-х переменных есть $D = \{ (x_1, y_1), \dots, (x_{256}, y_{256}) \}$. Здесь где x_i - MIG, описывающее i-ю функцию от 3 переменных в форме разложения Шеннона, а y_i - оптимальное MIG-представление этой функции. Сначала с использованием пакета *CirKit* [12], [13] генерировались MIG оптимального размера и инициализировалась обучающая выборка $X = \{x_1, \dots, x_{256}\}$. Оптимальные схемы значения $Y = \{y_1, \dots, y_{256}\}$ использовались только для сравнения и оценки результатов обучения.

При проведении экспериментов было принято $n = 5$ число ходов в игре и выполнено по *100* итераций по обучающему набору выше. В результате было показано, что описываемый алгоритм класса DNN+RL способен находить оптимальные представления для всех (100%) функций с тремя входами.

Для функций от 4-х аргументов (где набор данных есть $D = \{(x_1, y_1), \dots, (x_{65536}, y_{65536})\}$) было достигнуто 83% потенциально возможного улучшения размера схем. При этом модель может быть обобщена на функции от большего числа переменных.

В результате показано, что методы машинного обучения могут быть применены к оптимизации логических схем. Становится возможным создания автоматической и универсальной процедуры оптимизации, которая дает результаты, близкие к результатам высокоспециализированных современных алгоритмов и даже превосходящие их.

5. Мажоритарный базис и его аксиоматика

В настоящее время базовыми элементами синтеза в противоположность традиционным AND/OR/NOT часто становятся функция большинства M и NOT. На основе этих элементов построены схемы, реализующие различные функции как простейшие (отображения картинки на экране), так и имеющие значительную трудоёмкость (взятие сложных интегралов, решения задач физического моделирования).

Система логических функций, напомним, называется полной, если любую логическую функцию можно выразить через функции системы. Мажоритарным базисом будем называть следующую полную систему: $\{M, \neg, 1, 0\}$, где M – функция большинства от 3-х переменных, $\neg$ – отрицание.

x	y	z	$M(x, y, z)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Для мажоритарного базиса справедлива аксиоматическая система **Ω**, которую используют для осуществления преобразований формул, реализующих данную функцию [7]:

$$\Omega \begin{cases} \textit{Распространение инверсии } \Omega.I \\ \overline{M(x,y,z)} = M(\overline{x},\overline{y},\overline{z}) \\ \textit{Мажорирование } \Omega.M \\ \begin{cases} \textit{при } x = y\colon M(x,y,z) = x = y \\ \textit{при } x = \overline{y}\colon M(x,y,z) = z \end{cases} \\ \textit{Ассоциативность } \Omega.A \\ M\big(u,x,M(u,y,z)\big) = M\big(u,y,M(u,z,x)\big) \\ \textit{Дистрибутивность } \Omega.D \\ M\big(u,v,M(x,y,z)\big) = M(M(u,v,x),M(u,v,y),z) \\ \textit{Коммутативность } \Omega.C \\ M(x,y,z) = M(y,x,z) = M(z,y,x) \end{cases}$$

Введём операцию $f_{x/y}$ замены переменной x на y во всех её появлениях в формуле f, тогда для мажоритарного базиса также справедлива система Ψ [7]:

$$\Psi \begin{cases} \textit{Релевантность } \Psi.R \\ M(x,y,z) = M\big(x,y,z_{x/\overline{y}}\big) \\ \textit{Ассоциативность дополнения } \Psi.C \\ M\big(u,x,M(\overline{u},y,z)\big) = M\big(u,x,M(x,y,z)\big) \\ \textit{Подстановка } \Psi.S \\ M(x,y,z) = M\left(u,M\left(\overline{u},v,M_{u/v}(x,y,z)\right),M\left(\overline{u},\overline{v},M_{u/\overline{v}}(x,y,z)\right)\right) \end{cases}$$

Будем рассматривать задачу оптимизации размера логической схемы.

7. Оптимизации схем в мажоритарном базисе: предлагаемый метод

Задача построения алгоритма оптимизации сводится к выбору системы эквивалентных преобразований и определению последовательности применения этих правил.

В 2014-м году группа швейцарских учёных предложила оптимизировать логические схемы в мажоритарном базисе с помощью преобразований систем **Ω** и Ψ [6], [7]. Для достижения любой эквивалентной логической схемы в мажоритарном базисе достаточно преобразований системы **Ω** (она является полной). Но также вводится система Ψ для сокращения количества требуемых преобразований. Для

оптимизации размера логической схемы предлагается циклически применять следующей последовательности правил:

$$\Omega.M_{L\to R}(y);\ \Omega.D_{R\to L}(y);$$
$$\Omega.A(y);\ \Psi.C(y);$$
$$\Psi.R(y);\ \Psi.S(y);$$
$$\Omega.M_{L\to R}(y);\ \Omega.D_{R\to L}(y);$$

Нижний индекс показывает в каком направлении – прямом или обратном нужно применять преобразования. Первая и последняя строчки отвечают за уменьшения размера схемы, а вторая и третья строчки отвечают за изменение формы схемы чтобы обеспечить возможность последующего сокращения размера схемы. Эта группа учёных также предложила алгоритм для оптимизации глубины схемы, алгоритм заключается в циклическом применении следующей последовательности правил:

$$\Omega.M_{L\to R}(y);\ \Omega.D_{L\to R}(y);\ \Omega.A(y);\ \Psi.C(y);$$
$$\Omega.A(y);\ \Psi.C(y);$$
$$\Psi.R(y);\ \Psi.S(y);$$
$$\Omega.M_{L\to R}(y);\ \Omega.D_{L\to R}(y);\ \Omega.A(y);\ \Psi.C(y);$$

Здесь вторая и третья строчки также отвечают за изменение формы схемы для выхода из локальных минимумов, в то время как первая и последняя строчки уже выполняют роль уменьшения глубины. Циклы в этих алгоритмах предлагается выполнять фиксированное количество раз для ограничения времени выполнения алгоритма.

Обзор известных методов синтеза и оптимизации схем в мажоритарном базисе дан в [8] и [9].

Существующие методы используют фиксированную последовательность преобразований, что может быть не оптимально. Ниже мы предлагаем метод, который будет принимать решение о применении того или иного преобразования в зависимости от структуры логической схемы. Также применение одного преобразования сразу ко всей частям схемы может быть не оптимальным, поэтому предлагается создать возможность алгоритму выбирать преобразование для каждого участка.

Для реализации алгоритма используются рекурсивные нейронные сети. В ИНС *прямого распространения* сигналы распространяются последовательно, без возврата на предыдущие слои сети, что возможно в рекурсивных нейронных сетях. *Рекурсивная нейронная сеть* (Recursive neural network, RvNN) – тип DN, сформированный применением одной и той же нейронной сети рекурсивно через структуру в виде дерева, чтобы сделать скалярное или структурированное предсказание над входными данными переменного размера через обход дерева в топологическом

порядке. Далее в работе рекурсивно применяемую нейронную сеть будем называть *рекурсивным блоком*.

Опишем идею метода. Пусть уже зафиксирован набор эквивалентных преобразований применимых к логической схеме. Тогда нейронная сеть для каждого узла должна определить, какое преобразование (в том числе тождественное) может быть к нему применено. Так как размеры схемы могут меняться в процессе оптимизации, то логичней всего использовать рекуррентную или рекурсивную нейронную сеть. Из этих двух архитектур для задачи подходит последняя, так как она естественным образом учитывает топологию схемы. Поскольку для эффективности решения задачи требуется знать информацию о поддеревьях данного узла, эта информация будет возвращаться при рекурсивном вызове RvNN, также как и решение по задаче выбора преобразования к данному узлу.

Информацию, не относящуюся к решению задачи выбора преобразования, будем называть *вектор-представлением* логической схемы. Мы рассматриваем логические схемы в мажоритарном базисе, узлы которой реализуют функцию большинства от трёх аргументов, поэтому на каждом узле RvNN должна быть в наличии информация о трёх вектор-представлениях поддеревьев.

Пусть N – длина вектор-представления, M – количество различных преобразований и K– число передаваемых параметров узла. Тогда рекурсивный блок будет иметь $3 \cdot (N + M) + K$ входных и $N + M$ выходных значений. Работа предлагаемого алгоритма состоит в повторении определённое количество раз следующих шагов.

1. Рекурсивное нахождение для каждой вершины следующего преобразования.
2. Применение выбранного преобразования к вершинам, начиная от корня дерева.

Нахождение преобразования на 1-м шаге заключается в вычислении RvNN, где изначально вычисляется ответ для всех поддеревьев, а потом с помощью рекурсивного блока, и для самого узла.

Предложенная идея допускает любой рекурсивный блок с необходимым количеством входных и выходных параметров, и так как функция большинства коммутативна, то предложим рекурсивный блок коммутативный по входным векторам, отвечающим за поддеревья.

Поскольку функции большинства коммутативна, становиться возможным представить используем рекурсивный блок, как состоящий из двух нейронных сетей. В этом блоке первая сеть принимает вектор длины $N + M$ и возвращает вектор длины H, а вторая - принимает вектор длины $H + K$ и возвращает вектор длины $N + M$. Первая нейронная сеть отвечает за обработку информации поддеревьев и применяется к векторам,

полученным от каждого поддерева. Полученные три вектора размера H поэлементно складываются и делятся на 3, и вектор этих средних значений вместе с информацией о текущем узле передаётся во вторую нейронную сеть для составления вектор-представления.

Положительные стороны предложенной модели заключаются в следующем. Во-первых, в выборе преобразования учитывается информация о логической схеме (топология, индивидуальная информация каждого узла, коммутативность узлов). Во-вторых, при различных преобразованиях проводится сокращение количества проходов по логической схеме.

8. Реализация алгоритма

Предложенный метод был реализован на языке Python 3 с применением библиотек NumPy иTorch.

Логическая схема была представлена в виде ссылочной древесной структуры, где вершина является переменной, константой 0/1 или является узлом и тогда она хранит ссылки на трёх *потомков* (в противоположность распространению сигнала в сети). Каждая вершина хранит необходимую информацию о глубине и размере своего поддерева, возможном инвертировании выхода и др.

В качестве правил преобразования была взята система Ω. Длина N вектор-представления была принята равной 17, из которой 16 элементов вычислялись нейросетью, а последний отвечает за операцию отрицания. При обучении рекурсивной нейронной сети ставилась задача получить по логической схеме вектор-представление и вектор для выбора преобразования.

Опишем реализованную структуру рекурсивного блока, который, как описано выше, реализован двумя сетями. Обе сети являются двухслойными и формально осуществляют преобразования векторов.

В первой нейронной сети входной, промежуточный и выходной векторы имеют длины $21\ (= 17 + 4)$, 30 и 50 соответственно. После получения данных от 3-х потомков выполняется их осреднение. Далее к осреднённому вектору с приписыванием значение, указывающее на тип выхода узла (прямой/инверсный) и полученный вектор передаётся во вторую сеть.

Входной, промежуточный и выходной векторы второй сети имеют длины 51, 30 и 20. Первые 16 позиций выходного вектора с приписанным значением типа выхода возвращаются рекурсивным блоком как вектор-представление. От остальных 4-х позиций берётся величина модуля, от которого вычисляется функция *softmax* (преобразует вектор-аргумент в вектор такой же длины, где каждая координата есть вещественное число в интервале [0,1] и сумма координат равна 1; конкретно выбрано

преобразование «нормированное значение экспоненты значения координаты»). Наконец, составляется вектор абсолютных значений полученных величин.

9. Обучение сети, результаты и выводы

В общем случае нет информации о том какое преобразование эффективней применять к данной схеме. Однако можно сравнить успешность различных алгоритмов на определённом наборе схем. Это даёт возможность воспользоваться генетическими алгоритмами обучения сети.

Поскольку оптимизируется размер $S(y)$ схемы y, за *функцию приспособленности* в генетических алгоритмах была принята функция

$$W = \frac{S(y) + 1}{S(y^*) + 1} - 1,$$

где y – логическая схема до оптимизации, y^* - она же после оптимизации. За особь в генетическом алгоритме возьмём набор весов нейронных сетей рекурсивного блока.

Генетический алгоритм работает следующим образом:

1. Случайным образом генерируется набор особей
$$P^0 = \{p_1^0 \dots p_m^0\},$$
где p_j^i – j-я особь i-ого поколения.
2. Проводится оценка приспособленности особей
$$F^i = \{ W(p_1^i) \dots W(p_m^i) \},$$
где $W(p)$ – вычисление функции приспособленности на заданном наборе тестов для особи p.
3. Выход, если пройдено определённого количества поколений.
4. Иначе генерируем новое поколение P^{i+1} и переходим к шагу 2

Генерация нового поколения происходила следующим образом:

- добавляется не изменённые лучшие 10% особей текущего поколения;
- добавляются случайным образом изменённые лучшие 10% особей текущего поколения;
- 40% нового поколения заполняется случайным образом сгенерированными особями;
- остальные 40% есть результат скрещивания двух случайно выбранных особей из лучших 10% и 60% особей текущего поколения.

Скрещивание особей в данной работе реализовано как взятие среднего арифметического параметров.

Обучение проводилось на тестовой выборке, состоящей из схем, в которых достаточно применить ровно одно правило, и ещё 4-х схем в которых нужно применять последовательность правил.

После обучения было проведено тестирование на схемах различного исходного размера. В результате было замечено, что при таком подходе среднее значение приспособленности лучших трёх особей быстро выходит на плато, после чего долго держится на одном значении:

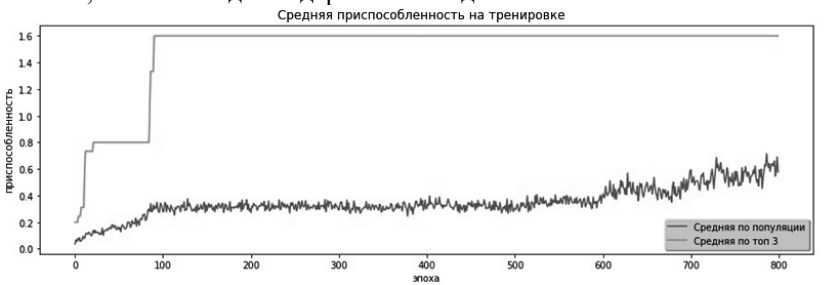

Рис. 2 Динамика средней оценки приспособленности

Это происходит из-за того, что набор параметров (гены) более приспособленных особей быстро распространяются, а нахождение новой хорошо приспособленной особей случайно. При этом обученные особи умели применять только несколько или даже одно правило. Поэтому применение генетических алгоритмов для обучения нейронной сети «с нуля» не является оптимальным. Также важен набор схем, на котором происходит тренировка. Он должен включать схемы, в полной мере показывающие способности нейросети, иначе в ходе обучения предложенным генетическим алгоритмом эти способности могут быть утрачены. В связи с указанными особенностями, перед использованием генетических алгоритмов, для большей эффективности предлагается обучить несколько особей совершать различные действия, после чего включить их в начальное поколение.

Заключение

В данной работе были рассмотрены возможности и результаты применения ИНС для синтеза логических схем. Показана перспективность данного подхода. Рассмотрены оптимизация схем в мажоритарном базисе, предложен метод применения нейронных сетей для построения алгоритма оптимизации размера логических схем.

Литература

1. *Chua L., Roska T.* The CNN Paradigm. // IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 1993, vol. 40, № 3, pp. 147-156.

2. *Nair V., HintonG. E.* Rectified linear units improve restricted Boltzmann machines // Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807-814.
3. *Le Cun Y., Bengio Y.* et al. Convolution networks for images, speech, and time series. / The Handbook of Brain Theory and Neural Networks, 1995, vol. 3361, no. 10.
4. *Silver D., Huang A., Maddison C.J., Guez A. et al.* Mastering the Game of Go with Deep Neural Networks and Tree Search. // Nature, 2016, vol. 529, no. 7587, pp. 484-489.
5. *Mnih V., Kavukcuoglu K., Silver D. et al.* Human-level control through deep reinforcement learning. // Nature, 2015, vol. 518, no. 7540, pp. 529-533.
6. *Haaswijk W., Collins E., Seguin B., Soeken M., Kaplan F., Susstrunk S., Micheli G.* Deep Learning for Logic Optimization Algorithms. // 2018 IEEE International Symposium on Circuits and Systems (ISCAS). 1-4. 10.1109/ISCAS.2018.8351885.
7. *AmarŮ L., Gaillardon P.E., De Michel G.* Majority-inverter graph: A novel data-structure and algorithms for efficient logic optimization. // Proceedings of the 51st Annual Design Automation Conference. ACM. DAC'14, June 1-5 2014, San Francisco, CA, USA, 2014, pp. 194:1-194:6. http://dx.doi.org/10.1145/2593069.2593158.
8. *Гуров С.И.* Мажоритарная алгебра для синтеза комбинационно-логических схем II. Обзор. // Таврический вестник информатики и математики. Международное научно-теоретическое издание. Симферополь: Крымский федеральный университет имени В. И. Вернадского.- 2020, № 3 (48). - С. 59-76.
9. *Гуров С.И.* Мажоритарная алгебра для синтеза комбинационно-логических схем I. Обзор. // Таврический вестник информатики и математики. Международное научно-теоретическое издание. Симферополь: Крымский федеральный университет им. В. И. Вернадского.- 2020, № 2 (47). - С. 40-61.
10. *Sutton R.S., Mcallester D.A., Singh S.P., Mansour Y. et al.* Policy gradient methods for reinforcement learning with function approximation. // NIPS, 1999, vol. 99, pp. 1057–1063.
11. *Kipf T.N. and Welling M.* Semi-supervised classification with graph convolutional networks. 2016, arXiv preprint arXiv:1609.02907.
12. *Soeken M. CirKit,* https://github.com/msoeken/cirkit.
13. *Soeken M., AmarŮ L., Gaillardon, P.-E., De Micheli G.* Exact synthesis of majority-inverter graphs and its applications // IEEE Transactions on CAD, 2017.

Н. Е. Александрова[1], *Д. С. Романов*[2]

НИЖНЯЯ ОЦЕНКА ДЛИНЫ ЕДИНИЧНОГО ДИАГНОСТИЧЕСКОГО ТЕСТА ОТНОСИТЕЛЬНО ВСТАВОК ЭЛЕМЕНТА СЛОЖЕНИЯ ПО МОДУЛЮ 2*

Введение и основные определения

Схемы из функциональных элементов (СФЭ) — одна из классических моделей управляющих систем без памяти. Анализ функционирования СФЭ при возникновении в них неисправностей традиционно осуществляется с помощью тестового подхода, предложенного С. В. Яблонским и И. А. Чегис [38].

Под вставкой m-входового элемента E в схему S понимается следующая операция [7]: в S выбираются m не обязательно различных функциональных элементов (ФЭ) $E^{(1)}$, $E^{(2)}$, ..., $E^{(m-1)}$, $E^{(m)}$ так, что никакой элемент среди $E^{(1)}$, $E^{(2)}$, ..., $E^{(m-1)}$ не лежит ни на каком ориентированном пути от $E^{(m)}$ к выходу схемы (однако, некоторые из элементов $E^{(1)}$, $E^{(2)}$, ..., $E^{(m-1)}$ или даже все могут совпадать с элементом $E^{(m)}$), все дуги, исходящие из $E^{(m)}$, заменяются на дуги с теми же концами, исходящие из E (при этом, если элемент $E^{(m)}$ был выходным элементом схемы, то в результате вставки выходным оказывается элемент E, а элемент $E^{(m)}$ перестает быть выходным), и добавляются m дуг с концами в E, исходящих из $E^{(1)}$, $E^{(2)}$, ..., $E^{(m)}$. О такой вставке элемента E в дальнейшем будем говорить как о вставке элемента E от элементов $E^{(1)}$, $E^{(2)}$, ..., $E^{(m-1)}$ под элемент $E^{(m)}$.

Пусть $F(x_1, x_2, \ldots, x_n) = (f_1, f_2, \ldots, f_N)$ — система булевых функций (БФ) n переменных, S — реализующая ее схема из функциональных элементов (СФЭ) в базисе B. Пусть на СФЭ S мог подействовать

[1]Младший консультант ООО «ГлоуБайт Аналитические решения», e-mail: alexandrova-nat@mail.ru.

[2]Доцент факультета ВМК МГУ имени М. В. Ломоносова, e-mail: romanov@cs.msu.ru.

*Статья опубликована при финансовой поддержке Минобрнауки РФ в рамках реализации программы Московского центра фундаментальной и прикладной математики по соглашению №075-15-2019-1621, а также проекта «Математические модели глобального информационно-энергетического баланса на основе циклов управления природно-антропогенными системами (на примере системы “загрязнение среды — здоровье населения”)» (код FSZZ-2020-0002).

источник неисправностей U, способный видоизменять схему конечным числом способов. Множество T входных наборов называется проверяющим тестом для схемы S относительно источника неисправностей U тогда и только тогда, когда для любой схемы S', полученной действием U на S, справедливо: если S' реализует некоторую систему БФ F', неравную F, то найдется набор α из T такой, что $F'(\alpha) \neq F(\alpha)$. Множество T входных наборов называется диагностическим тестом для схемы S относительно источника неисправностей U тогда и только тогда, когда для любых двух схем S' и S'', полученных действием U на S, справедливо: если S' и S'' реализуют некоторые неравные друг другу системы БФ F' и F'' соответственно, то найдется набор α из T такой, что $F'(\alpha) \neq F''(\alpha)$. Число наборов в тесте T называется его длиной $L(T)$. Под поломкой будем понимать неисправность, связанную с изменением или возникновением ровно одного элемента или входа схемы. Неисправность, не меняющую распределение сигналов в схеме на всяком входном наборе, назовем тривиальной. К тривиальным неисправностям отнесем и такую вставку элемента G под элемент G', при которой на выходе элемента G реализуется та же функция от входных переменных схемы, что и на выходе элемента G' в отсутствие неисправностей. Тест относительно одиночных (соответственно всевозможных кратных) поломок называется единичным (соответственно полным). Схема S называется неизбыточной (тестопригодной) относительно источника неисправностей U тогда и только тогда, когда при любой нетривиальной одиночной поломке (соответственно при любой нетривиальной неисправности), вызванной действием источника U, реализуемая неисправной схемой система БФ оказывается неравной системе, реализуемой схемой S при отсутствии неисправностей. Под длиной $L_B^{\mathrm{dt}}(U,S)$ (соответственно $L_B^{\mathrm{dn}}(U,S)$) минимального проверяющего (соответственно диагностического) теста относительно источника неисправностей U для СФЭ S в базисе B понимается минимум величины $L(T)$ по всем проверяющим (соответственно диагностическим) тестам T для S относительно U. Под длиной минимального проверяющего (диагностического) теста относительно источника неисправностей U для системы БФ $F(x_1,\ldots,x_n)$, реализованной с помощью СФЭ в базисе B, понимается величина $L_B^{\mathrm{dt}}(U,F)$ (соответственно $L_B^{\mathrm{dn}}(U,F)$), равная минимуму величины $L_B^{\mathrm{dt}}(U,S)$ (соответственно $L_B^{\mathrm{dn}}(U,S)$) по всем реализующим F неизбыточным СФЭ S с n входами в базисе B. Аналогично определим величину $\hat{L}_B^{\mathrm{dn}}(U,F)$ как минимум величины $L_B^{\mathrm{dn}}(U,S)$ по всем реализующим F СФЭ S с n входами в базисе B. Функцией Шеннона длины проверяющего (диагностического) теста для реализованной с помощью СФЭ в базисе B булевой функции f относительно источника неисправностей U называется величина $L_B^{\mathrm{dt}}(U,n)$ (соответственно $L_B^{\mathrm{dn}}(U,n)$), равная максимуму по всем БФ f, существенно

зависящим от n переменных, величины $L_B^{\mathrm{dt}}(U,f)$ (соответственно $L_B^{\mathrm{dn}}(U,f)$). Введем еще слабую функцию Шеннона $\hat{L}_B^{\mathrm{dn}}(U,n)$ длины диагностического теста для реализованной с помощью СФЭ в базисе B булевой функции f относительно источника неисправностей U как максимум по всем БФ f, существенно зависящим от n переменных, величины $\hat{L}_B^{\mathrm{dn}}(U,f)$.

Для произвольной булевой функции $f(x_1,x_2,...,x_n)$ через $\mathscr{L}_B(f)$ будем обозначать количество функциональных элементов в минимальной по числу элементов СФЭ $\hat{S}$ в базисе B, реализующей функцию f (т. е. невзвешенную сложность реализации f с помощью СФЭ в базисе B).

Через $U_1^{\oplus}$ (соответственно через $U_1^{\sim}$) обозначим источник одиночных вставок двухвходовых элементов сумм по модулю 2 (соответственно элементов эквивалентности).

Имеется достаточно много результатов, в которых длины тестов относительно константных или инверсных неисправностей или слипаний на входах или выходах элементов в СФЭ для любых булевых функций ограничены сверху константами (см., например, работы [3, 1, 2, 12, 31, 4, 5, 8, 9, 11, 32, 33, 34, 21, 35, 36, 37, 23, 24, 25, 26, 27, 28, 29, 30]), — в ряде работ используется константное число дополнительных входов или выходов в схемах или многозначность. Известные же растущие с числом переменных нижние оценки функций Шеннона длин тестов для СФЭ получались либо в ситуациях, когда явно допускались константные или инверсные неисправности на входах схем (что обусловлено результатами работ [6, 17, 18, 19, 20]), либо когда такие неисправности на входах схем моделировались поломками инцидентных входам схем элементов [10, 22].

В настоящей работе доказывается линейно растущая с числом переменных нижняя оценка функции Шеннона длины единичного диагностического теста относительно одиночных вставок элемента сложения по модулю 2 в СФЭ, причем рост нижней оценки не связан тем или иным образом с неисправностями на входах схем.

Формулировки и доказательства основных результатов

Оказывается, имеет место следующее утверждение.

Теорема 1. *Пусть B — функционально полный базис, n — натуральное число, $f(x_1,\ldots,x_n)$ — произвольная булева функция, тождественно не равная булевой переменной. Тогда имеет место неравенство* $\hat{L}_B^{\mathrm{dn}}(U_1^{\oplus},f) \geq \log_2 \mathscr{L}_B(f)$.

Доказательство. Пусть S — любая реализующая $f(\tilde{x}^n)$ СФЭ (с одним выходом) над базисом B. Отметим, что вставка элемента сложения по модулю 2 от элемента схемы, на выходе которого реализуется константа ноль, является тривиальной неисправностью. Рассмотрим случай неисправности, имеющей вид вставки нового элемента E сложения по

модулю 2 под выходной элемент E' схемы S так, что на один вход элемента E подается выход элемента E', а на другой вход элемента E подается выход какого-то элемента E'' схемы S (возможно, совпадающего с E'). При этом элемент E оказывается единственным выходным элементом неисправной схемы S', а сама эта схема, очевидно, реализует булеву функцию $f(x_1,\ldots,x_n)\oplus g(x_1,\ldots,x_n)$, где $g(x_1,\ldots,x_n)$ — функция от входных переменных схемы, реализуемая в вершине схемы S, соответствующей элементу E''. Нетрудно заметить, что количество $\mu(S)$ попарно неравных функций неисправности, которые могут быть получены в результате таких вставок элемента в схему S, не меньше, чем число $\mathscr{L}_B(f)$ функциональных элементов в минимальной по числу элементов СФЭ $\hat{S}$, реализующей функцию f. Докажем этот факт от противного. Пусть выполняется неравенство $\mu(S)<\mathscr{L}_B(f)$. Для каждой пары элементов E_1 и E_2 некоторой СФЭ, на выходах которых реализуются равные функции от входных переменных схемы, можно выполнить следующие действия. Заметим, что один из этих двух элементов (допустим, E_1) не лежит ни на каком ориентированном пути, соединяющем другой элемент (E_2) с выходом схемы S. Перенесем начала всех дуг, исходящих от выхода элемента E_2, к выходу элемента E_1, не меняя концов этих дуг. При этом, если элемент E_2 был выходным элементом исходной схемы, сделаем вместо него E_1 выходным элементом получаемой схемы. В этой новой схеме последовательно удалим все функциональные элементы, не являющиеся выходными и не имеющие исходящих из их выходов дуг. Получим СФЭ, в которой меньше, чем в исходной, пар элементов с равными реализованными на их выходах функциями, но при этом реализующую ту же булеву функцию, что и исходная схема. Повторяя такие действия последовательно для всех пар элементов схемы S, на выходах которых реализуются равные функции, до тех пор, пока подобных пар не останется, мы сможем получить по схеме S новую СФЭ $\check{S}$, реализующую функцию f и имеющую число элементов меньшее, чем $\mathscr{L}_B(f)$, что приводит к противоречию с определением величины $\mathscr{L}_B(f)$. Использованное здесь перестроение схемы S в схему $\check{S}$ называется приведением схемы S (см., напр., [15, с. 95]). А поскольку $\mu(S)\geq\mathscr{L}_B(f)$, с очевидностью получаем: $\hat{L}_B^{\mathrm{dn}}(U_1^{\oplus},f)\geq\log_2\mathscr{L}_B(f)$. Теорема доказана.

Следствие 1. *В условиях теоремы* 1 *если существует неизбыточная относительно источника неисправностей* $U_1^{\oplus}$ *СФЭ в базисе* B, *реализующая* f, *то имеет место неравенство* $L_B^{\mathrm{dn}}(U_1^{\oplus},f)\geq\log_2\mathscr{L}_B(f)$.

Хорошо известно, что для почти всех булевых функций $f(\tilde{x}^n)$ (то есть для стремящейся к единице доли булевых функций n переменных) имеет место асимптотическое равенство $\mathscr{L}_B(f(\tilde{x}^n))=\rho_B\cdot\frac{2^n}{n}\cdot(1+o(1))$, где ρ_B есть минимум по всем неодновходовым элементам базиса B

величин, обратных к уменьшенному на 1 числу входов элемента [16]; при этом функция Шеннона сложности СФЭ $\mathscr{L}_B(n)$, равная максимуму по всем булевым функциям $f(\tilde{x}^n)$ от n переменных величин $\mathscr{L}_B(f(\tilde{x}^n))$, ведет себя асимптотически как $\mathscr{L}_B(n) = \rho_B \cdot \frac{2^n}{n} \cdot (1 + o(1))$ ([16], уточнения оценок функции Шеннона сложности СФЭ см. в работах [13, 14]). При сличении приведенных фактов с учетом теоремы 1 оказывается доказанным следующее утверждение.

Теорема 2. *Пусть B — произвольный функционально полный базис. При $n \to \infty$ доля булевых функций $f(x_1,\ldots,x_n)$, для которых имеет место неравенство $\hat{L}_B^{\mathrm{dn}}(U_1^{\oplus}, f) \geq n - \log_2 n + \log_2 \rho_B + o(1)$, стремится к единице.*

Следствие 2. *В условиях теоремы 2 имеет место неравенство $\hat{L}_B^{\mathrm{dn}}(U_1^{\oplus}, n) \geq n - \log_2 n + \log_2 \rho_B + o(1)$.*

В работе [7] (в рамках введенного в настоящей статье определения неизбыточной относительно источника неисправностей $U_1^{\oplus}$ СФЭ) для всякой булевой функции фактически было доказано существование реализующей эту функцию неизбыточной схемы относительно источника $U_1^{\oplus}$ в каждом из базисов $B' = \{x \,\&\, y, x \oplus y, x \sim y\}$, $B'' = \{x \vee y, x \oplus y, x \sim y\}$. Поэтому справедливо следующее утверждение.

Теорема 3. *При $n \to \infty$ для любого базиса B из множества $\{B', B''\}$ имеет место неравенство $L_B^{\mathrm{dn}}(U_1^{\oplus}, n) \geq n - \log_2 n + o(1)$.*

Заметим, что в силу принципа двойственности аналогичные утверждения для двойственных базисов имеют место в случае одиночной вставки двухвходового элемента эквивалентности. В частности, из теоремы 3 вытекает следующее усверждение.

Теорема 4. *При $n \to \infty$ для любого базиса B из множества $\{B', B''\}$ имеет место неравенство $L_B^{\mathrm{dn}}(U_1^{\sim}, n) \geq n - \log_2 n + o(1)$.*

Литература

1. *DasGupta S., Hartmann C. R. P., Rudolph L. D.* Dual-mode logic for function-independent fault testing // IEEE Trans. Comput. 1980. Vol. C-29, No. 11. Pp. 1025–1029.
2. *Dubrova E. V., Muzio J. C.* Generalized Reed-Muller canonical form for a multiple-valued algebra // Multiple-Valued Logic. 1996. Vol. 1. Pp. 104–109.
3. *Inose H., Sakauchi M.* Synthesis of automatic fault diagnosable logical circuits by function conversion method // Proc. First USA-Japan Computer Conf. 1972. Pp. 426–430.
4. *Pan Zh.* Fault detection for testable realizations of multiple-valued logic functions // The 12th Asian Test Symposium (ATS 2003). IEEE, 2003. Pp. 242–247.

5. *Pan Zh.* Circuit testable design and universal test sets for multiple-valued logic functions // Journal of Electronics (China). 2007. Vol. 24, No. 1. Pp. 138–144.

6. *Reddy S. M.* Easily testable realization for logic functions // IEEE Trans. Comput. 1972. Vol. 21, Iss. 1. Pp. 124–141.

7. *Александрова Н. Е., Романов Д. С.* О длине единичного проверяющего теста относительно вставок элементов, не сохраняющих константу // Прикладная математика и информатика. Вып. 64. М.: МАКС Пресс, 2020. С. 64–78. Перевод: *Aleksandrova N. E., Romanov D. S.* The length of a single fault detection test for constant-nonpreserving element insertions // Computational Mathematics and Modeling. 2020. Vol. 31, Iss. 4. Pp. 484–493. DOI: 10.1007/s10598-021-09510-5.

8. *Бородина Ю. В.* О синтезе легкотестируемых схем в случае однотипных константных неисправностей на выходах элементов // Вестн. Моск. ун-та. Сер. 15. Вычисл. матем. и киберн. 2008. № 1. С. 40–44. Перевод: *Borodina Yu. V.* Synthesis of easily-tested circuits in the case of single-type constant malfunctions at the element outputs // Moscow University Computational Mathematics and Cybernetics. 2008. Vol. 32, No. 1. Pp. 42–46. DOI: 10.3103/S0278641908010068.

9. *Бородина Ю. В.* О схемах, допускающих единичные тесты длины 1 при константных неисправностях на выходах элементов // Вестн. Моск. ун-та. Сер. 1. Матем. Мех. 2008. Т. 63, № 5. С. 49–52. Перевод: *Borodina Yu. V.* Circuits admitting single-fault tests of length 1 under constant faults at outputs of elements // Moscow University Mathematics Bulletin. 2008. Vol. 63, No. 5. Pp. 202–204. DOI: 10.3103/S0027132208050069.

10. *Бородина Ю. В.* Нижняя оценка длины полного теста в базисе $\{x \mid y\}$ // Вестн. Моск. ун-та. Сер. 1. Матем. Мех. 2015. Т. 70, № 4. С. 49–51. Перевод: *Borodina Y. V.* Lower estimate of the length of the complete test in the basis $\{x \mid y\}$ // Moscow Univ. Math. Bull. 2015. Vol. 70. Pp. 185–186. DOI: 10.3103/S0027132215040063.

11. *Бородина Ю. В., Бородин П. А.* Синтез легкотестируемых схем в базисе Жегалкина при константных неисправностях типа “0” на выходах элементов // Дискретная математика. 2010. Т. 22, вып. 3. С. 127–133. Перевод: *Borodina Yu. V., Borodin P. A.* Synthesis of easily testable circuits over the Zhegalkin basis in the case of constant faults of type 0 at outputs of elements. 2010. Vol. 20, No. 4. Pp. 441–449. DOI: 10.1515/dma.2010.027.

12. *Коваценко С. В.* Синтез легкотестируемых схем в базисе Жегалкина для инверсных неисправностей // Вестн. Моск. ун-та. Сер. 15. Вычисл. матем. и киберн. 2000. № 2. С. 45–47.

13. *Ложкин С. А.* Асимптотические оценки высокой степени точности для сложности реализации булевских функций схемами из функциональных элементов // Труды II Международной конференции «Дискретные модели в теории управляющих систем», Красновидово, (23–28 июня 1997 г.). М.: Диалог-МГУ, 1997. С. 37–39.

14. *Ложкин С. А.* Асимптотические оценки высокой степени точности для сложности управляющих систем. Докторская диссертация по специальности 01.01.09 – Дискретная математика и математическая кибернетика (физ.-мат. науки). М.: МГУ имени М.В. Ломоносова, 1997.

15. *Ложкин С. А.* Лекции по основам кибернетики : Учебное пособие. М.: Изд. отдел ф-та ВМиК МГУ, 2004. 256 с.

16. *Лупанов О. Б.* Об одном методе синтеза схем // Изв. ВУЗ. Радиофизика. 1958. Т. 1, № 1. С. 120–140.

17. *Носков В. Н.* Диагностические тесты для входов логических устройств // Дискретный анализ. Вып. 26. Новосибирск: ИМ СО АН СССР, 1974. С. 72–83.

18. *Носков В. Н.* О сложности тестов, контролирующих работу входов логических схем // Дискретный анализ. Вып. 27. Новосибирск: ИМ СО АН СССР, 1975. С. 23–51.

19. *Носков В. Н.* О длинах минимальных единичных диагностических тестов, контролирующих работу входов логических схем // Методы дискретного анализа в синтезе управляющих систем. Вып. 32. Новосибирск: ИМ СО АН СССР, 1978. С. 40–51.

20. *Погосян Г. Р.* О проверяющих тестах для логических схем. М.: ВЦ АН СССР, 1982. 57 с.

21. *Попков К. А.* О точном значении длины минимального единичного диагностического теста для одного класса схем. Препринт № 74 ИПМ им. М. В. Келдыша РАН. М.: ИПМ им. М. В. Келдыша РАН, 2015. 20 с.

22. *Попков К. А.* Нижние оценки длин полных диагностических тестов для схем и входов схем // Прикладная дискретная математика. 2016. № 4(34). С. 65–73.

23. *Попков К. А.* Единичные проверяющие тесты для схем из функциональных элементов в базисе «конъюнкция-отрицание» // Прикладная дискретная математика. 2017. № 38. С. 66–88.

24. *Попков К. А.* О точном значении длины минимального единичного диагностического теста для одного класса схем // Дискретн. анализ и исслед. опер. 2017. Т. 24, № 3. С. 80–103. Перевод: *Popkov K. A.* On

the exact value of the length of the minimal single diagnostic test for a particular class of circuits // J. Appl. Ind. Math. 2017. Vol. 11. Pp. 431–443. DOI: 10.1134/S1990478917030140.

25. *Попков К. А.* Полные диагностические тесты длины два для схем при инверсных неисправностях функциональных элементов. Препринты ИПМ им. М. В. Келдыша. 2017. № 103. М.: ИПМ им. М. В. Келдыша РАН, 2017. 10 с.

26. *Попков К. А.* Полные проверяющие тесты длины два для схем при произвольных константных неисправностях элементов // Дискретн. анализ и исслед. опер. 2018. Т. 25, № 2. С. 62–81. Перевод: *Popkov K. A.* Complete fault detection tests of length 2 for logic networks under stuck-at faults of gates // J. Appl. Ind. Math. 2018. Vol. 12. Pp. 302–312. DOI: 10.1134/S1990478918020102.

27. *Попков К. А.* Синтез легкотестируемых схем при произвольных константных неисправностях на входах и выходах элементов. — Препринты ИПМ им. М. В. Келдыша. 2018. № 149. — М.: ИПМ им. М. В. Келдыша РАН, 2018. — 32 с.

28. *Попков К. А.* Короткие единичные тесты для схем при произвольных константных неисправностях на выходах элементов // Дискретная математика. 2018. Т. 30, вып. 3. С. 99–116. Перевод: *Popkov K. A.* Short single tests for circuits with arbitrary stuck-at faults at outputs of gates // Discrete Mathematics and Applications. 2019. Vol. 29, No. 5. Pp. 321–333. DOI: 10.1515/dma-2019-0030.

29. *Попков К. А.* Метод построения легко диагностируемых схем из функциональных элементов относительно единичных неисправностей // Прикладная дискретная математика. 2019. № 46. С. 38–57.

30. *Попков К. А.* Короткие полные проверяющие тесты для схем из двухвходовых функциональных элементов // Дискретн. анализ и исслед. опер. 2019. Т. 26, № 1. С. 89–113. Перевод: *Popkov K. A.* Short complete fault detection tests for logic networks with fan-in two // J. Appl. Ind. Math. 2019. Vol. 13. Pp. 118–131. DOI: 10.1134/S1990478919010137.

31. *Редькин Н. П.* Единичные проверяющие тесты для схем при инверсных неисправностях элементов // Матем. вопросы киберн. Вып. 12. М.: Физматлит, 2003. С. 217–230.

32. *Романов Д. С.* О синтезе схем, допускающих полные проверяющие тесты константной длины относительно произвольных константных неисправностей на выходах элементов // Дискретная математика. 2013. Т. 25, вып. 2. С. 104–120. DOI: 10.4213/dm1239. Перевод: *Romanov D. S.* On the synthesis of circuits admitting complete fault detection test sets

of constant length under arbitrary constant faults at the outputs of the gates // Discrete Mathematics and Applications. 2013. Vol. 23, Iss. 3–4. Pp. 343–362. DOI: 10.1515/dma-2013-024.

33. *Романов Д. С.* Метод синтеза легкотестируемых схем, допускающих единичные проверяющие тесты константной длины // Дискретная математика. 2014. Т. 26, вып. 2. С. 100–130. Перевод: *Romanov D. S.* Method of synthesis of easily testable circuits admitting single fault detection tests of constant length // Discrete Mathematics and Applications. 2014. Vol. 24, No. 4. Pp. 227–251. DOI: 10.1515/dma-2014-0021.

34. *Романов Д. С.* О синтезе схем, допускающих полные проверяющие тесты константной длины относительно инверсных неисправностей на выходах элементов // Вестн. Моск. ун-та. Сер. 15. Вычисл. матем. и киберн. 2015. № 1. С. 30–37. Перевод: *Romanov D. S.* Synthesis of circuits admitting complete checking tests of constant length under inverse faults at outputs of elements // Moscow University Computational Mathematics and Cybernetics. 2015. Vol. 39, No. 1. Pp. 26–34. DOI: 10.3103/S0278641915010057.

35. *Романов Д. С.* Метод синтеза неизбыточных схем в базисе Жегалкина, допускающих единичные диагностические тесты длины один // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2015. № 4. С. 38–54.

36. *Романов Д. С., Романова Е. Ю.* Метод синтеза неизбыточных схем, допускающих короткие единичные диагностические тесты при константных неисправностях на выходах элементов // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. 2016. № 2 (38). С. 87–102. DOI: 10.21685/2072-3040-2016-2-8.

37. *Романов Д. С., Романова Е. Ю.* Метод синтеза неизбыточных схем, допускающих единичные проверяющие тесты константной длины // Дискретная математика. 2017. Т. 29, № 4. С. 87–105. Перевод: *Romanov D. S., Romanova E. Yu.* A method of synthesis of irredundant circuits admitting single fault detection tests of constant length // Discrete Mathematics and Applications. 2019. Vol. 29, No. 1. Pp. 35–48. DOI: 10.1515/dma-2019-0005

38. *Чегис И. А., Яблонский С. В.* Логические способы контроля электрических схем // Труды МИАН СССР. 1958. Т. 51. С. 270–360.