УДК 004.312.22, 004.855.5, 519.688, 519.714.5, 003.26
DOI: 10.25559/SITITO.17.202102.295-307

Оригинальная статья

Обучение с подкреплением в задаче синтеза мажоритарных схем

С. И. Гуров*, Д. В. Золотарёв, А. И. Самбурский
ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова», г. Москва, Российская Федерация
119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1
* sgur@cs.msu.ru

Аннотация

В статье изложен подход к синтезу комбинационнологических схем с применением искусственных нейронных сетей (ИНС). Излагаемый метод ориентирован на использование перспективного базиса, использующего функцию большинства (булева функция от трёх аргументов, которая принимает значение «истина», если истинны хотя бы два из её входов). Такой выбор обусловлен возникающими нанотехнологиями, в которых представление элемента большинства наиболее осуществляется особенно просто. Класс применяемых ИНС – глубокие сети с подкреплением. Такие сети активно изучаются и применяются в последнее время. Известны примеры их эффективного использования для автоматической оптимизации логических схем. Предложенный в статье оригинальный синтеза метод с упрощения схем, реализующих разложение Шеннона по всем переменным соответствующей функции алгебры логики (ФАЛ). На больших схемах становится существенным использование некоторых простых, но действенных приёмов обучения агентов глубокой ИНС с подкреплением. Это позволяет распределить вычисления на несколько независимых подзадач, каждая из которых исследуется и выполняется агентами проще и быстрее. Описаны два алгоритма обучения с подкреплением для упрощения схем. Они обеспечивают решение конфликта *Exploration-Exploitation*, заключающегося в противоречии между исследованием среды для поиска оптимального эпизода и использованием информации об эпизоде, считающимся оптимальным на текущий момент времени. Представлены зависимостей параметров синтезированных схем от числа переменных ФАЛ и количества эпизодов обучения сети.

Ключевые слова: логический синтез схем, мажоритарная логика, искусственные нейронные сети, обучение с подкреплением

Финансирование: исследование выполнено за счет гранта Российского научного фонда, проект № 17-19-01645 (2020-2021) «Разработка методов и средств проектирования реконфигурируемых систем на кристалле повышенной надежности».

Авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Гуров, С. И. Обучение с подкреплением в задаче синтеза мажоритарных схем / С. И. Гуров, Д. В. Золотарёв, А. И. Самбурский. – DOI 10.25559/SITITO.17.202102.295-307 // Современные информационные технологии и ИТ-образование. – 2021. – Т. 17, № 2. – С. 295-307.

Original article

Reinforcement Learning in the Problem of Synthesis of Majority Schemes

S. I. Gurov*, D. V. Zolotarev, A. I. Samburskiy
Lomonosov Moscow State University, Moscow, Russian Federation
1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation
* sgur@cs.msu.ru

Abstract

The article presents an approach to the synthesis of combinational-logic circuits using artificial neural networks (ANNs). The presented method is focused on the use of a perspective basis using the majority function (a Boolean function of three arguments that takes the value "true" if at least two of its inputs are true). This choice is based on emerging nanotechnologies, where the majority element is most easily represented. The class of applied ANNs is deep networks with reinforcement. Such networks have been actively studied and applied in recent years. There are examples of their effective use for automatic logic circuits optimization. The original synthesis method proposed in the article with the simplification of circuits that implement the Shannon expansion in all variables of the corresponding function of the logic algebra (FLA). On large schemes, it becomes essential to use some simple but effective techniques for training deep ANN agents with reinforcement. This allows you to distribute calculations into several independent subtasks, each of which is explored and makes performing by agents quicker and easier. Two reinforcement learning algorithms for simplifying schemas are described. They provide a solution to the *Exploration-Exploitation* conflict, which is the contradiction between exploring the environment to find the optimal episode and using information about the episode considered optimal at the current time. The dependences of the parameters of the synthesized circuits on the number of FAL variables and the number of network training episodes are presented.

Keywords: logical synthesis, majority logic, artificial neural networks, reinforcement learning

Funding: The study was supported by a grant from the Russian Science Foundation, project No. 17-19-01645 (2020-2021) "Development of Methods and Tools of Design of Reconfigured Systems on Enhanced Reliability Crystal".

The authors declare no conflict of interest.

For citation: Gurov S.I., Zolotarev D.V., Samburskiy A.I. Reinforcement Learning in the Problem of Synthesis of Majority Schemes. *Sovremennye informacionnye tehnologii i IT-obrazovanie = Modern Information Technologies and IT-Education.* 2021; 17(2):295-307. DOI: https://doi.org/10.25559/SITITO.17.202102.295-307

Введение

Задача физической реализации логических функций является одним из важных направлений кибернетики. Вычислительные устройства состоят из больших интегральных схем и, соответственно, требуют эффективной реализации. Поэтому одним из серьёзных вопросов является синтез логических схем, содержащих минимально возможное число базовых элементов и/или оптимальных по другим параметрам.

Общеизвестно, что характеристики современных цифровых ИС в значительной степени зависит от возможностей средств их логического синтеза. В последнее время в этой области наблюдаются существенные сдвиги: применение методов искусственного интеллекта, использование нетрадиционных базисов и др.

Интерес к алгоритмам синтеза на основе функции большинства связан с возникающими нанотехнологиями, работающим на её основе [1-3]. *Функция большинства* $M(x, y, z)$ есть булева функция от трёх аргументов, которая принимает значение «истина», если истинны хотя бы два из её входов. Реализующий функцию большинства *мажоритарный элемент* обеспечивает функциональность основных булевых операций дизъюнкции и конъюнкции.

В программной реализации упрощение заключается в том, что исполнитель алгоритма может воспринимать эти операции равноценно через мажоритарный элемент. Например, законы дистрибутивности булевой алгебры взаимодвойственны, то есть симметрично выполняются для операций дизъюнкции и конъюнкции. В мажоритарном базисе эти тождества будут отличаться только на константы, поданные мажоритарным схемам. Поэтому программная реализация мажоритарных формул будет работать одинаково как с дизъюнкциями, так и с конъюнкциями. Также для мажоритарного базиса допустимы гибкие тождества для обработки схем, что так же положительно сказывается на простоте алгоритмической реализации. Это – ещё один аргумент в пользу реализации ФАЛ в мажоритарном базисе [4].

Чем сложнее функция (под сложностью функции можно понимать, например, число её существенных переменных), тем вычислительно труднее составить для неё схему и тем дольше и менее эффективно будут работать алгоритмы синтеза. Уже при 15-20 аргументах точные алгоритмы могут работать недопустимо долго. Выходом из данной ситуации является использование алгоритмов, строящие схемы близкие по некоторому критерию к оптимальным, но за приемлемое время. В данной статье рассматриваются реализации нескольких таких алгоритмов, а также исследуется вопрос повышения их эффективности с помощью методов обучения с подкреплением.

Создание оптимальной логической схемы можно подразделить на независимые этапы синтеза и упрощения. Точные методы решения поставленной задачи требуют выполнения полного перебора аргументов по очень большим множествам, как на первом, так и на втором из указанных этапов, что приводит к огромным затратам вычислительных ресурсов.

Одним из путей преодоления данного «проклятия размерности» является использование приближённых алгоритмов, не просто только прекращающих «слепой» перебор по истечении заданного числа шагов, но и его оптимизирующих. При такой оптимизации используются те или иные свойства конкретной схемы. Обучение с подкреплением, может служить основой для контроля за переборной частью алгоритмов. В статье будут описаны используемые алгоритмы синтеза и упрощения схем, в которых промежуточные результаты зависят только от найденных конфигураций параметров. Данные конфигурации можно получить перебором, однако ввиду упомянутого выявления закономерностей можно применить процедуру частичного интеллектуального поиска и выйти на хороший результат быстрее.

Данная работа нацелена на изучение свойств алгоритмов обучения с подкреплением для повышения эффективности решения задач, требующих вычислительно трудоёмкой реализации [5]. Методы применяются к модельным задачам обработки функциональных схем логических функций: к упрощению и синтезу. В ходе работы описывается общая схема применяющихся алгоритмов и их альтернативных реализаций, указываются достоинства и недостатки применяющихся модификаций. Кроме того, обозначаются выявленные алгоритмические закономерности, позволяющие повысить эффективность рассматриваемых программ. Основная задача – сократить время работы трудоёмких алгоритмов, минимально потеряв точность их решения. Данный результат достигается как вводом параметров, регулирующих вычислительную сложность, так и эффектами обучаемости программы во время её работы.

Постановка задачи

В данной статье мы рассмотрим вопросы выявления основных закономерностей в работе алгоритмов обучения с подкреплением [6-9]. Целью является использование выявленных фактов для ускорения работы алгоритмов. Будут рассматриваться алгоритмы синтеза и оптимизации логических схем, реализующих двоичные функции [10]. Синтез подразумевает поиск минимальной функциональной структуры, моделирующей заданную булеву функцию. Упрощение в данном исследовании сводится к итеративному просмотру схемы с поиском возможности тождественных преобразований, снижающих значение выбранного функционала сложности схемы.

В работе обосновывается удобство использования мажоритарного базиса при вычислениях, а также приводятся базовые идеи и понятия методов обучения с подкреплением. Далее введённые понятия конкретизируются для каждой из двух указанных модельных задач, приводится описание используемых алгоритмов и выделяются закономерности, которые удалось определить. Так же описываются эффекты, которые были достигнуты при различных вариациях данных закономерностей.

Мажоритарный базис и его аксиоматика

Для представления схем в данной работе основным элементом является *мажоритарный* (подробнее см. [1,11,12]). По определению он принимает на вход 3 двоичных сигнала и реализует функцию большинства $M(x, y, z) = xy + yz + zx$ (+– сумма по $mod\ 2$, знак конъюнкции опускается). Данный элемент способен реализовывать операции дизъюнкции и конъюнкции: $M(x, y, 0) = xy,\ \ M(x, y, 1) = x \vee y$.

Мажоритарным будем называть базис, состоящий из функций

большинства , инверсии (отрицания) ‘ и булевых констант 0 и 1. Ввиду того, что мажоритарная функция обеспечивает функциональность дизъюнкции и конъюнкции, то данный базис является полным, то есть позволяет моделировать любые *функции алгебры логики* (ФАЛ). Мажоритарные схемы можно представлять функциональным деревом, в каждой вершине которого реализуется мажоритарная функция от тех трёх аргументов, которые соответствуют входящей в эту вершину тройку рёбер. На листьях данного дерева реализуются простейшие функции от одной переменной: константы или булева переменная. В корне дерева реализуется сама целевая ФАЛ. Приведём некоторые тождественные преобразования, применимые к простым мажоритарным подсхемам. Эти тождества могут использоваться для упрощения схем.

Следующие пять основных правил преобразований функции *M* большинства называют набором её *аксиом* и традиционно обозначают символом Ω [13].

Аксиома коммутативности Ω. *C*

$$M(x,y,z) = M(x,z,y) = M\ (z,y,x).$$

Аксиома ассоциативности Ω. *A*

$$M(x,w,M\ (y,w,z)) = M\ (z,w,M\ (y,w,x)).$$

Аксиома дистрибутивности Ω. *D*

$$M(x,y,M(u,v,z)) = M(M(x,y,u),M(x,y,v),M\ (x,y,z)).$$

Аксиома распространения инверсии Ω. *I*

$$M'(x,y,z) = M(x',y',z').$$

Аксиома мажорирования Ω. *M* представляется двумя равенствами:

$$M(x,x,y) = x, \qquad M(x,y,x') = y.$$

(первое выражает основной смысл понятия большинства, а второе свойство отношения «между»).

После введения операции $f_{x/y}$ замены переменной *x* на переменную *y* во всех её появлениях в *f*, становятся справедливыми также соотношения системы Ψ:

Релевантности Ω. *R*

$$M(x,y,z) = M(x,y,z_{x/y}).$$

Ассоциативность дополнения Ω. *C*

$$M(x,u,M(y,u',z)) = M(x,u,M(y,x,z)).$$

Подстановка Ω. *S*

$$M(x,y,z) = M\left(v,M\left(v',M_{v/u}(x,y,z),u\right),M\left(v',M_{v/u'},u'\right)\right).$$

Все данные тождества позволяют преобразовывать мажоритарные схемы на разном уровне сложности.

Обучение с подкреплением: основные понятия

В последние годы буквально революцию в машинном обучении произвело применение *глубокого обучения* искусственных нейронных сетей [14,15]. 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 [16]. При *обучении с подкреплением* (reinforcement learning, RL) обучение проходит в результате взаимодействия *агента* и *внешней среды* [17-19]. В зависимости от предпринятого действия агент получает от внешней среды вознаграждение или штраф. Стремясь максимизировать своё вознаграждение, агент использует управляемый случайный поиски выстраивает свою *политику* (стратегию), оптимизирующую его реакции на подаваемые входы. В результате последних исследований возникает специальный вид *глубоких ИНС с подкреплением* (DNN+RL) [20,21].

Алгоритмы обучения с подкреплением подразумевают наличие следующих конструктов:

- *агент* – исполнитель алгоритма;
- *среда* – область функционирования агента;
- система *наград* – правила сопоставления действиям агента числовой характеристики их полезности относительно выбранного критерия оптимальности; отрицательную награду называют *штрафом*.

В каждый момент времени агент находится в одном из состояний среды и может совершить одно из допустимых действий. За каждое действие среда назначает награду, которая впоследствии будет учитываться при повторном запуске агента в среду. Целью агента является решение двух задач: (1) предварительное исследование свойств среды для определения оптимальной последовательности действий в ней и (2) реализация найденной стратегии.

За основу архитектуры обучения агента мы взяли метод *Q-обучения (Q*-learning) [22]. В данном методе для обучения агента используется таблица *T* рейтингов (полезности действий*)*, задающая соответствие троек

[состояние] - [действие] - [полезность],

определяющее потенциальную награду при совершении действия в фиксированном состоянии. Полезность задаётся математическим ожиданием аккумулируемой награды, которую агент получит, совершив заданное действие. На каждом шаге работы в среде агент определяет наиболее подходящее действие по данной таблице.

По завершении работы агента, значения в таблице *T* обновляются в соответствии с вкладом каждого совершённого действия в успешность выполнения задачи. Это производится по правилу

$$\mathbb{E}[R|s_t,a_t] = r_t + \gamma\mathbb{E}[R|s_{t+1},a_{t+1}].$$

Здесь s_t, a_t и r_t — состояние агента, совершённое им действие и полученная награда в момент времени t состояния агента соответственно, R - общий «рейтинг оптимальности» пары [состояние] - [действие], γ — коэффициент дисконтирования, позволяющий учитывать степень близости действия к успешным/неуспешным завершениям работы программы (эффект *обратной инерции*), $\mathbb{E}[\bullet]$ символ математического ожидания. Полезность действия, видим, вычисляется через (а) непосредственную награду, полученную за её выполнение, и (б) аккумулятивную полезность всех дальнейших действий агента.

Агент итеративно запускается в среду, доходит до конца выполнения задачи или принудительно останавливается (при преодолении предела допустимых действий), после чего среда указанным выше образом пересчитывает награды за все совершённые действия и дополняет ими таблицы *T* полезности действий. В результате многократного повторения этих операций агент может выработать стратегию, которая позволит ему выполнить поставленную задачу. В процессе работы агент может частично ориентироваться на эти таблицы, и тем самым приближаться к оптимальной стратегии, со временем совершая больше полезных действий.

Решение модельных задач. Оптимизация

В качестве одного из способов повышения эффективности алгоритмов для получения оптимальных логических схем рассмотрим сначала методы упрощения готовых схем, и обсудим, какими инструментами обучения с подкреплением их можно улучшить [23-25].

Будем считать, что уже задана некоторая схема из мажоритарных элементов, являющейся избыточно сложной. Она легко может быть получена использованием разложением Шеннона по первой существенной переменной функции

$f(x_1, \dots, x_n) = x_1 f(1, x_2, \dots, x_n) \vee x_1 f(0, x_2, \dots, x_n)$,

и рекурсивным продолжением разложения её подфункций.

Схема, полученная при реализации данного разложения, будет состоять из $3 \cdot (2^{n-1} - 1)$ мажоритарных элементов для всех n-арных функций, что во многих случаях сверхизбыточно (функция может иметь значительное количество несущественных фиктивных переменных). Немалая доля элементов сразу на момент синтеза будет вырожденной (такие элементы имеют два одинаковых аргумента). Реализация подобных преобразований, как и более сложных, возлагается на программу упрощения. Как выясняется, несмотря на такую иррациональность синтеза, именно в данном случае гораздо удобнее проводить изучение свойств алгоритмов упрощения, с целью найти наилучший метод.

Чтобы найти возможность упрощения, необходимо просмотреть всю схему целиком. К тому же нетривиальные преобразования схемы могут привести к необходимости проводить перебор заново, что значительно увеличивает время поиска преобразования.

Для упрощения этого процесса воспользуемся следующим приёмом. Введём в рассмотрение *абстрактный исполнитель команд* упрощения схемы. Будем считать, что он способен перемещаться по всей схеме, находясь в каждый момент времени в одном из её элементов. При выполнении условий на возможность локального упрощения в данном участке схемы, исполнитель может произвести это упрощение. Зададим для этого абстрактного исполнителя набор допустимых действий, состоящей из двух групп. Первая группа относится к перемещению по схеме: переход к следующему аргументу данного мажоритарного элемента. Вторая группа состоит из самих действий упрощения: исполнитель может провести проверку возможности одного из допустимых упрощений и при положительном результате проверки автоматически её произвести. Данные упрощения могут задаваться тождествами, описанными выше.

Теперь опишем поведение исполнителя. В каждый дискретный момент времени он находится в определённом элементе схемы и может выбрать одно из допустимых действий для перемещения или попытки упрощения. Исполнитель может выбрать то действие, приносящее наибольший выигрыш.

Для создания условий адекватного выбора создадим таблицу T соответствия действий в каждом мажоритарном элементе с рейтингом (полезностью) этого действия. Рейтинг соответствует уверенности исполнителя в возможности добиться положительного результата в дальнейшем при выполнении данного действия. Принципиальный момент заключается в том, что указанная таблица T обновляется динамически. Исполнитель может совершать ошибочные действия, однако при корректном задании отклика для каждого действия, вносящегося в таблицу, он со временем начнёт чаще и чаще совершать те действия, которые с большей вероятностью приведут его к цели, в чём и заключается его обучение. Исполнителю не требуется обходить всю схему целиком по нескольку раз, так как в процессе работы он на неё «настраивается» и может быстрее отыскать возможности для упрощения.

Каждому мажоритарному элементу сопоставим его кодированное представление в данной схеме. В результате получаем следующее соответствии терминов RL и нашей модели

RL	в предлагаемой модели
агент	абстрактный исполнитель команд
состояние агента	код элемента
среда	схема, представленная своими закодированными элементами
награда/штраф	численный аддитивный вклад действия агента в таблицу T

Основной алгоритм оптимизации (упрощения)

Агенту подаётся усложнённая схема, от него требуется её упростить. Базовым действием агента является команда перемещения или попытка локального упрощения схемы. Последовательность совершения базовых действий, фиксированная по их количеству, обозначим термином *эпизод*. Агент начинает эпизод с одного из допустимых состояний (можно, например, всегда начинать эпизод с корня схемы, так как до каждого потенциального узла с упрощением он находится на наименьшем «расстоянии» в среднем).

Во время эпизода агент, попадая в очередное состояние, может выбрать по таблице T рейтингов либо наилучшее действие, либо любое из оставшихся. Распределение вероятностей данных альтернатив зависит от дополнительного динамического параметра, обозначаемого далее , с возрастанием которого уменьшается вероятность выбора наилучшего действия.

Описанная стратегия является простейшим решением конфликта *ExplorationExploitation*, заключающегося в противоречии между исследованием среды для поиска оптимального (в данном случае, с точки зрения удачного упрощения) эпизода и использованием информации о эпизоде, считающимся оптимальным на текущий момент времени. Очевидно, что в течение обучения наилучший эпизод можно искать, как «похожий» на текущий оптимальный. Поэтому и существует необходимость, сохраняя свойства лучшего на настоящий момент эпизода, немного менять его структуру для нахождения более выигрышного.

За каждое совершённое действие агент может получить награду или штраф. Если агенту удаётся совершить упрощение, за соответствующее действие он получает награду. За любой другой исход агент получает небольшой штраф. Для того, чтобы агент не останавливался на месте (зацикливание), при сохранении состояния штраф увеличивается. Если агент завершает эпизод без упрощений, ему присуждается дополнительный штраф для исключения повторения похожей последователь-

ности действий в дальнейших эпизодах.

Если агент совершал в основном правильные действия, эпизод заканчивается упрощением схемы, иначе он обрывается по достижении заданного количества действий. В обоих случаях после завершения эпизода происходит обновление таблицы *T*. Начальный оптимальный эпизод может представлять случайный набор действий. Важно, что со временем агент начнёт переходить к более полезным состояниям за счёт обучения.

Интересное замечание заключается в том, что в данном случае одну из центральных ролей в корректной работе агента занимают штрафы. Само их существование является сильным условием того, что из-за разнобоя в действиях агент не будет оставаться вблизи стартового положения. Находясь там длительное время, суммарные штрафы настолько сильно возрастут, что агент будет вынужден переходить в новые состояния: такой переход в таблице *T* оценивается как , а оштрафованные состояния имеют отрицательный рейтинг. Данную закономерность в дальнейших описаниях будем называть *эффектом вытесняющих штрафов*.

Дальнейшим обобщением является серия эпизодов, называемая эпоха. Результатом одной эпохи является одно локальное упрощение схемы. Каждый эпизод эпохи является итерацией обучения агента. Последний завершённый эпизод как раз и должен соответствовать упрощению схемы.

Отметим, что среда агента, как и во многих других задачах, не является стационарной: упрощение схемы оптимальным эпизодом её ликвидирует, так как в обработанном участке схемы, естественно, нельзя произвести повторное упрощение. Это, в свою очередь, влечёт за собой неоднозначность таблицы *T*, которая помимо регулярных небольших изменений от эпизода к эпизоду должна радикально меняться от эпохи к эпохе. Делаем вывод: агента, как и всю программу, нельзя полностью настроить на всю схему сразу; необходимо настраивать его на каждую оптимизацию в отдельности, и реализовывать *Exploration-Exploitation* параллельно. С этим и связано использование деление на эпохи. Вдобавок к этому разделение на эпизоды и эпохи – ещё один способ контроля перебора, так как в них явным образом можно ограничить число последовательностей действий в эпохе и самих действий в эпизоде.

В ходе исследования было реализовано два варианта общих реализаций взаимодействия агента со средой: облегчённая (базовая) и модифицированная.

Варианты реализации основного алгоритма и их сравнение

Базовый вариант. В данном облегчённом варианте происходит локальное обучение с фиксированным числом итераций. Это означает, что можно задать число действий, разрешённых для выполнения агента в течение одного эпизода и число самих эпизодов для определения наилучшей стратегии однократного упрощения схемы. В данном случае для удовлетворения баланса условий *Exploration-Exploitation* используется схема ε-*жадных* алгоритмов: внутри эпохи каждый эпизод запускается со своим параметром , причём каждый последующий эпизод запускается с уменьшенным значением этого параметра, что приводит к увеличению вероятности выбора оптимального (по мнению агента) действия и обеспечивает уточнение его стратегии. Была выбрана зависимость $\varepsilon=0{,}99^k$, где k – номер очередного эпизода, и тогда уже на 100-ом эпизоде вероятность выбора оптимального действия оказывается близкой к *70%*.

После проведённого обучения агент совершает набор из исключительно оптимальных действий, что приводит к улучшению оптимизируемой схемы. В случае, если лучшая стратегия оказалась при выбранных параметрах безрезультатной (схема остаётся прежней), задаётся число допускаемых подряд «неудачных» эпох, по истечении числа которого процесс останавливается.

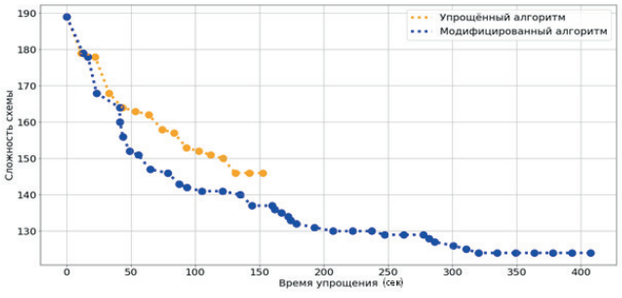
Так как после очередного улучшения схемы агент в соответствии с текущей таблицей *T* рейтингов настроен на последнее произведённое улучшение (что связано с проблемами нестационарности среды), таблица *T* очищается, и в следующую эпоху обучающая настройка агента происходит заново. Отметим, что данный способ перехода от эпохи к эпохе является сильно упрощённым и отстранённым от хорошего уровня глобального обучения агента. В то же время, данный метод «обнуления опыта» позволяет, по крайней мере, избежать проблем переполнения. Они связаны с тем, что преобразование очень больших схем ведёт к необходимости увеличивать число итераций обучения, что приводит к возможности безграничного роста или спада рейтинга одного из действий в определённом состоянии, и программа не может обрабатывать корректно числа таких порядков. Кроме того, что более важно, очищение таблицы рейтингов позволяет избавиться от проблемы некорректного обучения агента. Этот эффект может произойти из-за того, что агент проходил по эпизоду, близкому к оптимальному, но в конце не смог завершить его успешно, и по итогу в дальнейшем будет считать его неудачным, что негативно скажется на всём процессе.

Модифицированный вариант взаимодействия агента со средой обладает двумя принципиальными отличиями, позволяющими дополнительно повысить эффективность алгоритма, но, вместе с тем, и снизить устойчивость его работы.

Первое отличие заключается в замене фиксированного числа итераций эпохального обучения агента на быстрый перебор «до первой удачи». Каждая эпоха остаётся ограниченной сверху по числу содержащихся в ней эпизодов, но обрывается при первом найденном упрощающем эпизоде. Это ускоряет прохождение одной эпохи, хоть и приводит, возможно, не к самой лучшей стратегии упрощения, поскольку эксплуатируется первая найденная. Данная проблема частично решается в базовом варианте алгоритма, где Exploration позволяет дополнительно просматривать несколько вариантов упрощения и выбирать из них наилучший.

Второе отличие модифицированного метода заключается в том, что таблица *T* рейтингов, накопленная за эпоху, не обнуляется полностью, а лишь подвергается изменениям, в результате чего происходит сглаживание огромных значений. Дополнительный параметр случайности выбора оптимального действия может фиксироваться: ε=0. Это означает, что агент всегда будет выбирать оптимальное, по его мнению, действие. Это повышает подчинённость среды правилу вытесняющих штрафов, поскольку агент будет быстрее выбираться из окрестности начального состояния, но при этом не сможет свободно исследовать среду.

Данные условия обобщают обучение агента за пределы одной эпохи, что обеспечивает возможность ускорения работы алгоритма. В течение всего времени безрезультатного исследования агентом среды в поиске возможности её упрощения, заполняются таблицы *T* рейтингов. При этом агент начинает лучше определять неудачные действия, что позволяет ему упрощать схему эффективными «рывками». Данные рывки видны на приведённом ниже графике сравнения облегчённого и модифицированного алгоритмов, запущенных отдельно для упрощения одной схемы (см. Рис. 1).



Р и с. 1. Сравнение реализаций алгоритма упрощения
F i g. 1. Comparison of simplification algorithm implementations

Как видно, облегчённый алгоритм в целом проигрывает модифицированому. Отметим следующее свойство, следующее из приведённой зависимости: показатель «неудачности» любой эпохи (характеризующая, насколько поздно агент найдёт упрощающий эпизод) на ранних этапах работы программы прямо пропорционально влияет на объём накопленного опыта за эпоху, что выливается в повышение ускорения поиска упрощающего эпизода в последующие эпохи. Из-за этого график работы модифицированного алгоритма представляет такие периодически «скатывающиеся» кривые. Время его работы по упрощению до достижения сложности схемы базового алгоритма сокращается более, чем в два раза, а эффективность снижение общей сложности схемы – в полтора раза. Данная производительность объясняется глобальностью правила вытесняющих штрафов. Чем дальше агент продвигается по эпохам, тем больше имеет информации о неблагоприятных действиях и тем вероятнее успеет найти упрощающие эпизоды за предоставленные ему эпизодичные итерации.

Также имеются негативные последствия отсутствия обнуления таблиц рейтингов модифицированного алгоритма: в процессе обучения может произойти искажение информации об удачных эпизодах, и агент будет исследовать схему не там, где сможет найти упрощение. В итоге, модифицированный алгоритм с «испорченной» таблицей *T* не будет способен показать хорошие результаты:

Выявленные достоинства и недостатки каждого из алгоритмов оптимизации приведены в таблице.

Облегчённая реализация	***Модифицированная реализация***
+ Стабильно находит возможности упрощения + Имеет возможность выбора лучших видимых упрощений + Не подвержен проблеме нестационарности среды	+ При определённых условиях имеет возможность быстро находить возможности упрощения + Переводит неудачу эпизода в увеличение потенциала последующих + Настраивается на текущую схему + Выбор гиперпараметров обучения ограничивает только его перебор, но не замедляет его работу
- Ликвидирована постепенная настраиваемость на текущую задачу (отсутствие памяти) - Ошибка выбора гиперпараметра числа действий из архитектуры может сильно замедлить процесс	- Снижение стабильности упрощения (это может проявляться в том, что иногда этот алгоритм из-за «неудачных» эпизодов сохраняет в памяти искажённый контекст, что мешает ему в дальнейшем) - Теряет свою эффективность в случае сверхбыстрого изменения схемы (многократного увеличения показателя нестационарности среды)

Алгоритмы были реализованы на языке *Python* с использованием библиотеки *numpy* для векторных вычислений. Данная библиотека позволяет эффективно оперировать с картами Карно для задач синтеза и таблицей Т при оптимизации схем. Тесты проводились на ПК с процессором 3 GHzIntelCore i5.

Тестовые схемы и булевы функции выбирались случайной генерацией из множеств функций от *n* аргументов *n*=3,...,13, . Схемы выбирались с разной степенью функциональной «вырожденности» для того, чтобы проанализировать поведение алгоритма в различных по сложности задачах по разным по объёму схемам. При оптимизации методы быстро находили вырожденные участки, затем с замедлением упрощали более глубокие участки функционального дерева схемы. С ростом числа аргументов функций скорость упрощений экспоненциально снижалась, для временного преодоления чего и предлагаются рассматриваемые модификации основных алгоритмов. В случае из очень сложных схем ФАЛ, существенно зависящих от 7 переменных, облегчённый алгоритм, задействовав простейшие операции упрощения схем, работает около 2 минут и сокращает их сложность на 15-25% (от 189 до 150 элементов). Модифицированный алгоритм при корректной работе способен достигать этого результата вдвое быстрее, и при этом дополнительно может сократить сложность схемы ещё на 15% (до 120 элементов).

При увеличении числа аргументов ФАЛ возрастает и время выполнения программы. На функциях от 11 и более аргументов становится вычислительно трудно совершать преобразования. Несмотря на это, при задании гиперпараметров

трудоёмкости исследования среды можно учесть это и предоставить агенту большее число возможных действий в течение одного эпизода, то есть допустима регуляризация сложности вычислений.

На очень больших схемах становится существенным использование некоторых простых, но действенных приёмов обучения агентов, что позволяет распределить вычисления на несколько независимых подзадач, каждая из которых исследуется и выполняется агентами проще и быстрее. Обсудим их ниже.

Основные свойства алгоритмов упрощения

Кратко опишем два основных свойства применения методов, позволяющие повысить скорость работы программы. Оба свойства заключаются в необходимости снижения сложности обучения одного агента, распределив его возможности группе из нескольких агентов. Снижение возможностей агента, с одной стороны, уменьшают его функционал, но, с другой – позволяют гораздо эффективнее решить его подзадачу.

1. Разделение агентов по допустимым упрощениям. При реализации алгоритмов упрощения следует создавать несколько экземпляров агентов и каждому из них ставить в соответствие только небольшую часть допустимых действий по упрощению. Во-первых, это позволит ему настраиваться на конкретные типы оптимизаций, что снизит неудачные проверки подсхем для упрощений. Во-вторых, снижается размер таблиц T, и агент может быстрее пройти по возможным вариантам действий и найти оптимальное.

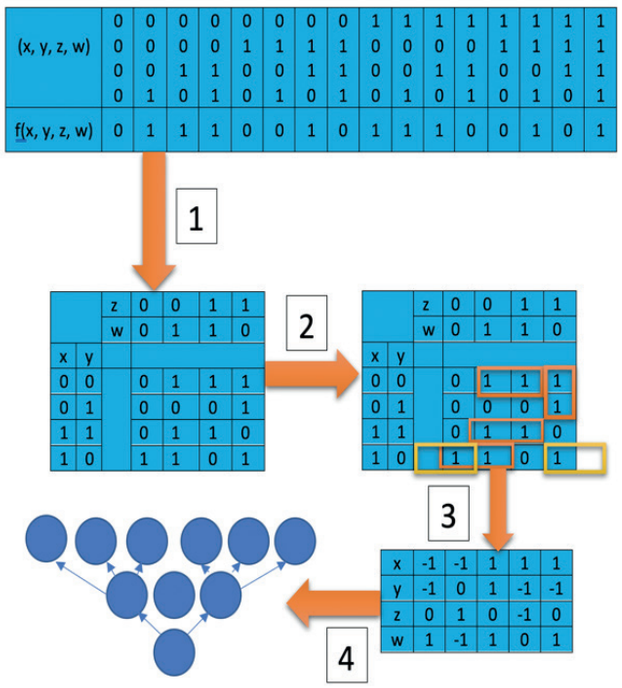
2. Dropout частей схемы, состоящий в разделении схемы на несколько независимых частей для разных независимых агентов. Эффект получается тот же, но в данном случае, как дополнительное улучшение, появляется возможность распараллеливания работы агентов: каждый из них до конца работы программы должен исследовать только свою подсхему.

Данные подходы можно комбинировать, повышая эффективность метода. Это особенно важно при оптимизации больших схем, для которых базовый алгоритм будет работать неприемлемо долго. В среднем, разделяя действия по упрощению схемы можно получить прирост в скорости обучения не менее 20% на ФАЛ от 10 переменных. С увеличением числа аргументов реализуемой функции этот эффект будет усиливаться.

При разбиении схемы на подсхемы и распараллеливании обучения на каждой из них коэффициенту ускорения программы пропорционален количеству подсхем. Вместе с этим, необходимо затрачивать больше вычислительных ресурсов для обучения нескольких групп агентов этим способом.

Алгоритм синтеза схем

Повышение числа существенных переменных реализуемой ФАЛ приводит к экспоненциальному росту времени обработки полученной схемы. Естественным облегчением решения задачи упрощающих алгоритмов является предварительное упрощение функциональной схемы, прямо в момент синтеза, либо более эффективный (по сложности получаемой схемы) выборочный синтез. В данной работе рассматривается последний из этих двух подходов.



Алгоритм состоит из двух этапов: перевода формулы в ДНФ и дополнительного снижения сложности. На рисунке приведена визуализация алгоритма построения ФАЛ функции по шагам (шаги 1-3 первый этап алгоритма, шаг 4 второй).

1. По таблице истинности строится её карта Карно.
2. По карте Карно строится (приближённо) минимальное покрытия характеристического множества .
3. По найденному покрытию строится ДНФ функции .
4. Осуществляется перевод ДНФ в более компактную формулу.

Первый этап – задача здесь заключается в построении «приблизительно» тупиковой дизъюнктивной нормальной формы ДНФ, что можно удобно реализовать, используя карту Карно. Приблизительность получаемой ДНФ заключается в том, что ни одну *элементарную конъюнкцию* (ЭК) нельзя выбросить из ДНФ, чтобы та по-прежнему корректно реализовывала целевую ФАЛ, но в то же время нет гарантий того, что получаемые ЭК являются минимально возможными. На этом этапе применяются простые алгоритмы обучения с подкреплением, что позволяют сделать перебор контролируемым и более быстрым, чем полный перебор.

Конкретно использовались два алгоритма, оба используют в качестве агента исполнитель команд регулирования размеров граней сектора, покрывающего фиксированную единицу. Если в прошлом разделе от агента требовалось самое быстрое нахождение любого решения задачи (локального упрощения), и, в связи с этим, подходила тактика вытесняющих штрафов, то в этом разделе ситуация обратная: некоторое решение известно – сектор карты Карно, состоящий из одной единицы, что, вероятно, является не наибольшим покрытием. В данном случае агент должен ориентироваться на наградную составляющую рейтинга действий, чтобы найти наибольший сектор, покрывающий единицу. Состояниями являются конфигурации

размеров сектора. Агенту предоставляется некоторое фиксированное число действий по изменению размера сектора (эпизод), и он должен найти то состояние, попадая в которое, он получает наибольший выигрыш. Оптимальной стратегией (оптимальным эпизодом) будем считать последовательность действий, совершив которую, агент попадает в наиболее вознаграждаемое состояние.

Первый алгоритм основывается на процессе эволюции решения. Все эпизоды начинаются с самого обширного состояния, которое охватывает всю карту Карно в один сектор, и заключаются в сокращении размеров сектора и заканчиваются на первом найденном секторе из единиц. Если полученное состояние соответствует большему показателю карты очков, то пройденный эпизод и это состояние объявляются оптимальными. Так происходит заданное число раз. Агент придерживается текущей выигрышной стратегией и при этом совершает исследование в поисках более оптимального покрытия. В течение эпизодов вероятность выбора случайного действия занижается и агент сходится к итоговой оптимальной стратегии.

Второй алгоритм производит более быстрый поиск и после сокращения размеров сектора начинает его расширять до момента включения в сектор первого нуля карты Карно. В данном случае он так же сохраняет учёт текущей оптимальной стратегии, но теперь совершает действия для того, чтобы как можно сильнее расходиться с ней. Это может приводить к поиску более оптимальных секторов и расширению исследования среды. Сложность перебора в данных задачах контролируются параметрами, задающими число эпизодов, производящих поиск оптимального сектора.

Второй этап состоит в построении мажоритарной схемы по полученному множеству ЭК. Кратко опишем используемый жадный алгоритм решения этой задачи.

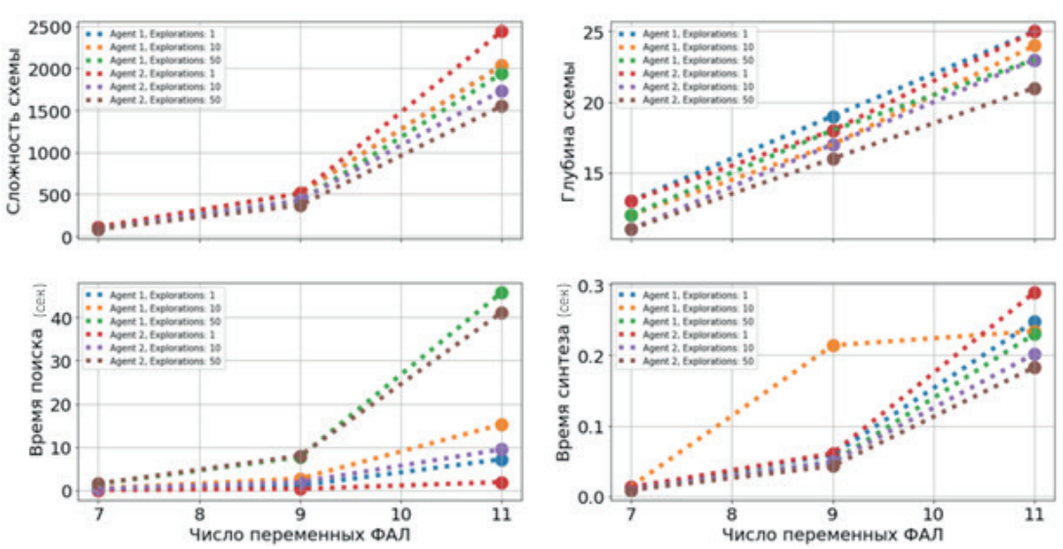
1. Из обрабатываемого множества ЭК выделить наибольшее подмножество ЭК, содержащих общую переменную; повторить для всех оставшихся ЭК.
2. Распределить полученные множества равномерно по числу входящих в них ЭК на две группы и соединить данные группы через операцию дизъюнкции. Впоследствие данные группы будут сами преобразованы в мажоритарные подсхемы, а в корне данной схемы будет реализована дизъюнкция всех ЭК исходного множества.
3. Если в группе остались разные множества ЭК (которые не содержат одну общую переменную), то повторить для этого множества шаг 1.
4. Если в группе осталось только одно множество ЭК (все содержат как минимум общую букву), то распределить через конъюнкции равномерно по глубине все общие буквы, а над оставшейся группой ЭК (не имеющих одной общей переменной), выполнить шаг 1.
5. Для всех оставшихся множеств и групп повторять шаги 1-4.

В результате получаем более компактную формулу, реализующую заданную ФАЛ, как по сложности, так и по глубине моделируемых схем.

Результаты синтеза

Описанные алгоритмы были запрограммированы и протестированы. По результатам работы программы можно проанализировать успешность двух указанных стратегий в зависимости от предоставленных эпизодов для исследования среды (см. Рис. 2).

Как видно, если заданное число эпизодов велико (50), то более хорошие результаты показывает второй алгоритм, ориентирующийся на отличия от текущей оптимальной стратегии. При снижении числа допустимых эпизодов стратегии начинают показывать сравнимые результаты.



Р и с. 2. Сравнение стратегий исследования среды

F i g. 2. Comparison of environmental exploration strategies

В данной задаче поиска оптимального решения выигрышной является стратегия, подразумевающая принудительное расширение исследования среды без настраивания на текущую оптимальную конфигурацию.

Исследование зависимостей параметров синтезированных схем от числа , переменных ФАЛ и количества эпизодов обучения сети RL показывает:

- Сложность полученных оптимальных секторов для покрытия в карте Карно и времена как их поиска, так и общего синтеза схем экспоненциально, а глубина линейно растут в зависимости от (что и следовало ожидать).
- Количество эпизодов обучения сети RL существенно влияет только на время поиска оптимального сектора покрытия в карте Карно (уменьшение на порядок с увеличением от 10 до 100), заметно сокращает сложность синтезированной схемы, но, естественно, увеличивает время синтеза.

Таким образом, методы обучения с подкреплением применимы и для модельной задачи синтеза схем и позволяют регулировать баланс между вычислительной трудоёмкостью и эффективностью результата (в плане сложности получаемой схемы).

Заключение

В работе были рассмотрены алгоритмы обучения с подкреплением применительно к задачам обработки логических схем и на основе этого сформированы базовые методы повышения эффективности подходов. Также были реализованы различные стратегии поведения агентов для указанных задач упрощения и синтеза, проведён анализ преимуществ данных методов и указаны их основные алгоритмические свойства. Получен механизм регулировки соотношения оптимальности решения и его вычислительной трудоёмкости.

Список использованных источников

[1] Soeken, M. Exact synthesis of majority inverter graphs and its applications / M. Soeken, L. G. Amarú, P.-E. Gaillardon, G. De Micheli. – DOI 10.1109/TCAD.2017.2664059 // IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems. – 2017. – Vol. 36, issue 11. – Pp. 1842-1855.

[2] Cai, R. A Majority Logic Synthesis Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits / R. Cai, O. Chen, A. Ren, N. Liu, C. Ding, N. Yoshikawa, Y. Wang. – DOI 10.1145/3299874.3317980 // Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI'19). – Association for Computing Machinery, NY, USA, 2019. – Pp. 189-194.

[3] Balasubramanian, P. Mathematical estimation of logical masking capability of majority/minority gates used in nanoelectronic circuits / P. Balasubramanian, R. T. Naayagi. – DOI 10.1109/CIRSYSSIM.2017.8023173 // 2017 International Conference on Circuits, System and Simulation (ICCSS). – IEEE Press, London, UK, 2017. – Pp. 1823.

[4] Amarú, L. Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient Logic Optimization / L. Amarú, P. E. Gaillardon, G. De Micheli. – DOI 10.1145/2593069.2593158 // Proceedings of the 51st Annual Design Automation Conference (DAC'14). – Association for Computing Machinery, New York, NY, USA, 2014. – Pp. 1-6.

[5] Гуров, С. И. Искусственные нейронные сети и синтез логических схем / С. И. Гуров, Д. В. Золотарёв, А. И. Самбурский // Прикладная математика и информатика: труды факультета ВМК МГУ имени В. М. Ломоносова. – М.: ООО «МАКС Пресс», 2021. – № 68. – С. 75-87. – URL: https://www.elibrary.ru/item.asp?id=47319217 (дата обращения: 18.03.2021).

[6] Devraj, A. M. Fundamental Design Principles for Reinforcement Learning Algorithms / A. M. Devraj, A. Bušić, S. Meyn. – DOI 10.1007/978-3-030-60990-0_4 // Handbook of Reinforcement Learning and Control. Studies in Systems, Decision and Control; ed. by K. G. Vamvoudakis, Y. Wan, F. L. Lewis, D. Cansever. – Vol. 325. – Springer, Cham, 2021. – Pp. 75-137.

[7] Vafashoar, R. Applications of Cellular Learning Automata and Reinforcement Learning in Global Optimization / R. Vafashoar, H. Morshedlou, A. Rezvanian, M. R. Meybodi. – DOI 10.1007/978-3-030-53141-6_4 // Cellular Learning Automata: Theory and Applications. Studies in Systems, Decision and Control. – Vol. 307. – Springer, Cham, 2021. – Pp. 157-224.

[8] Yonekura, K. Framework for design optimization using deep reinforcement learning / K. Yonekura, H. Hattori. – DOI 10.1007/s00158-019-02276-w // Structural and Multidisciplinary Optimization. – 2019. – Vol. 60, issue 4. – Pp. 1709-1713.

[9] Yan, D. Deep reinforcement learning with credit assignment for combinatorial optimization / D. Yan, J. Weng, S. Huang, C. Li, Y. Zhou, H. Su, J. Zhu. – DOI 10.1016/j.patcog.2021.108466 // Pattern Recognition. – 2022. – Vol. 124. – Pp. 108466.

[10] Butyrlagin, N. V. Logical Synthesis and Circuitry of Digital Current Circuits: Polynomial Approach / N. V. Butyrlagin, N. I. Chernov, N. N. Prokopenko, V. Yugay. – DOI 10.1109/TELFOR48224.2019.8971153 // 2019 27th Telecommunications Forum (TELFOR). – IEEE Press, Belgrade, Serbia, 2019. – Pp. 1-4.

[11] Гуров, С. И. Мажоритарная алгебра для синтеза комбинационно-логических схем. Обзор / С. И. Гуров // Таврический вестник информатики и математики. – 2020. – № 2(47). – С. 39-60. – URL: https://www.elibrary.ru/item.asp?id=44600613 (дата обращения: 18.03.2021). – Рез. англ.

[12] Гуров, С. И. Мажоритарная алгебра для синтеза комбинационно-логических схем. Обзор. часть II / С. И. Гуров // Таврический вестник информатики и математики. – 2020. – № 3(48). – С. 59-76. – URL: https://www.elibrary.ru/item.asp?id=46301092 (дата обращения: 18.03.2021). – Рез. англ.

[13] Amarú, L. A Sound and Complete Axiomatization of Majority-n Logic / L. Amarú, P.-E. Gaillardon, A. Chattopadhyay, G. De Micheli. – DOI 10.1109/TC.2015.2506566 // IEEE Transactions on Computers. – 2016. – Vol. 65, issue 9. – Pp. 2889-2895.

[14] Chua, L. The CNN Paradigm / L. Chua, T. Roska. – DOI 10.1109/81.222795 // IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. – 1993. – Vol. 40, issue 3. – Pp. 147156.

[15] Haaswijk, W. Deep Learning for Logic Optimization Algorithms / W. Haaswijk, E. Collins, B. Seguin, M. Soeken, F. Kaplan, S. Susstrunk, G. De Micheli. – DOI 10.1109/ISCAS.2018.8351885 // 2018 IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE Press, Florence, Italy, 2018. – Pp. 14.

[16] Nair, V. Rectified Linear Units Improve Restricted Boltzmann Machines / V. Nair, G. E. Hinton // Proceedings of the 27th International Conference on Machine Learning (ICML-10). – Haifa, Israel, 2010. – Pp. 807814. – URL: https://icml.cc/Conferences/2010/papers/432.pdf (дата обращения: 18.03.2021).

[17] Mazyavkina, N. Reinforcement learning for combinatorial optimization: A survey / N. Mazyavkina, S. Sviridov, S. Ivanov, E. Burnaev. – DOI 10.1016/j.cor.2021.105400 // Computers & Operations Research. – 2021. – Vol. 134. – Article 105400.

[18] Möller, F. J. D. A Reinforcement Learning Based Adaptive Mutation for Cartesian Genetic Programming Applied to the Design of Combinational Logic Circuits / F. J. D. Möller, H. S. Bernardino, L. B. Gonçalves, S. S. R. F. Soares. – DOI 10.1007/978-3-030-61380-8_2 // Intelligent Systems. BRACIS 2020. Lecture Notes in Computer Science; ed. by R. Cerri, R. C. Prati. – Vol. 12320. – Springer, Cham, 2020. – Pp. 18-32.

[19] Zhu, K. Exploring Logic Optimizations with Reinforcement Learning and Graph Convolutional Network / K. Zhu, M. Liu, H. Chen, Z. Zhao, D. Pan. – DOI 10.1145/3380446.3430622 // Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20). – Association for Computing Machinery, New York, NY, USA, 2020. – Pp. 145-150.

[20] Silver, D. Mastering the game of Go with deep neural networks and tree search / D. Silver, A. Huang, C. Maddison [и др.]. – DOI 10.1038/nature16961 // Nature. – 2016. – Vol. 529. – Pp. 484-489.

[21] Mnih, V. Human-level control through deep reinforcement learning / V. Mnih, K. Kavukcuoglu, D. Silver [и др.]. – DOI 10.1038/nature14236 // Nature. – 2015. – Vol. 518. – Pp. 529-533.

[22] Machine Learning Proceedings 1991 / Ed. by L. A. Birnbaum, G. C. Collins. – DOI 10.1016/C2009-0-27661-6 // Proceedings of the Eighth International Conference. – Morgan Kaufmann, Evanston, Illinois, 1991. – 661 p.

[23] Brudermueller, T. Making Logic Learnable With Neural Networks / T. Brudermueller, D. L. Shung, A. J. Stanley, J. Stegmaier, S. Krishnaswamy // arXiv:2002.03847. – 2020. – URL: https://arxiv.org/abs/2002.03847 (дата обращения: 18.03.2021).

[24] Somayaji, N. S. K. Prioritized Reinforcement Learning for Analog Circuit Optimization With Design Knowledge / N. S. K. Somayaji, H. Hu, P. Li. – DOI 10.1109/DAC18074.2021.9586189 // 2021 58th ACM/IEEE Design Automation Conference (DAC). – IEEE Press, San Francisco, CA, USA, 2021. – Pp. 1231-1236.

[25] Coello, C. A. Ant Colony System for the Design of Combinational Logic Circuits / C. A. Coello, R. L. G. Zavala, B. M. García, A. H. Aguirre. – DOI 10.1007/3-540-46406-9_3 // Evolvable Systems: From Biology to Hardware. ICES 2000. Lecture Notes in Computer Science; ed. by J. Miller, A. Thompson, P. Thomson, T. C. Fogarty. – Vol. 1801. – Springer, Berlin, Heidelberg, 2000.– Pp. 21-30.

Поступила 18.03.2021; одобрена после рецензирования 11.05.2021; принята к публикации 24.05.2021.

Об авторах:

Гуров Сергей Исаевич, старший научный сотрудник, доцент кафедры математических методов прогнозирования, факультет вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), кандидат физико-математических наук, доцент, **ORCID: https://orcid.org/0000-0001-5486-1357**, sgur@cs.msu.ru

Золотарёв Дмитрий Витальевич, студент факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0002-5349-2147**, ub8caf@mail.ru

Самбурский Александр Ильич, студент факультета вычислительной математики и кибернетики, ФГБОУ ВО «Московский государственный университет имени М. В. Ломоносова» (119991, Российская Федерация, г. Москва, ГСП-1, Ленинские горы, д. 1), **ORCID: https://orcid.org/0000-0003-1188-0229**, sashasamb@yandex.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

References

[1] Soeken M., Amarú L.G., Gaillardon P.-E., De Micheli G. Exact synthesis of majority inverter graphs and its applications. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*. 2017; 36(11):1842-1855. (In Eng.) DOI: https://doi.org/10.1109/TCAD.2017.2664059

[2] Cai R., Chen O., Ren A., Liu N., Ding C., Yoshikawa N., Wang Y. A Majority Logic Synthesis Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. *Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI'19).* Association for Computing Machinery, NY, USA; 2019. p. 189-194. (In Eng.) DOI: https://doi.org/10.1145/3299874.3317980

[3] Balasubramanian P., Naayagi R.T. Mathematical estimation of logical masking capability of majority/minority gates used in nanoelectronic circuits. *2017 International Conference on Circuits, System and Simulation (ICCSS)*. IEEE Press, London, UK; 2017. p. 1823. (In Eng.) DOI: https://doi.org/10.1109/CIRSYSSIM.2017.8023173

[4] Amarú L., Gaillardon P.E., De Micheli G. Majority-Inverter Graph: A Novel Data-Structure and Algorithms for Efficient

Logic Optimization. *Proceedings of the 51st Annual Design Automation Conference (DAC'14).* Association for Computing Machinery, New York, NY, USA; 2014. p. 1-6. (In Eng.) DOI: https://doi.org/10.1145/2593069.2593158

[5] Gurov S.I., Zolotarev D.V., Samburskiy A.I. *Iskusstvennye nejronnye seti i sintez logicheskih shem* [Artificial neural networks and synthesis of logic circuits]. In: Ed. by A. S. Krylov. *Applied Mathematics and Computer Science: Proceedings of the CMC MSU Faculty.* MAKS Press, Moscow; 2021. No. 68. p. 75-87. Available at: https://www.elibrary.ru/item.asp?id=47319217 (accessed 18.03.2021). (In Russ.)

[6] Devraj A.M., Bušić A., Meyn S. Fundamental Design Principles for Reinforcement Learning Algorithms. In: Ed. by K. G. Vamvoudakis, Y. Wan, F. L. Lewis, D. Cansever. *Handbook of Reinforcement Learning and Control. Studies in Systems, Decision and Control.* 2021; 325:75-137. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-030-60990-0_4

[7] Vafashoar R., Morshedlou H., Rezvanian A., Meybodi M.R. Applications of Cellular Learning Automata and Reinforcement Learning in Global Optimization. Cellular Learning Automata: Theory and Applications. *Studies in Systems, Decision and Control.* 2021; 307:157-224. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-030-53141-6_4

[8] Yonekura K., Hattori H. Framework for design optimization using deep reinforcement learning. *Structural and Multidisciplinary Optimization*. 2019; 60(4):1709-1713. (In Eng.) DOI: https://doi.org/10.1007/s00158-019-02276-w

[9] Yan D., Weng J., Huang S., Li C., Zhou Y., Su H., Zhu J. Deep reinforcement learning with credit assignment for combinatorial optimization. *Pattern Recognition*. 2022; 124:108466. (In Eng.) DOI: https://doi.org/10.1016/j.patcog.2021.108466

[10] Butyrlagin N.V., Chernov N.I., Prokopenko N.N., Yugay V. Logical Synthesis and Circuitry of Digital Current Circuits: Polynomial Approach. *2019 27th Telecommunications Forum (TELFOR).* IEEE Press, Belgrade, Serbia; 2019. p. 1-4. (In Eng.) DOI: https://doi.org/10.1109/TELFOR48224.2019.8971153

[11] Gurov S.I. Majority Algebra for the Synthesis of Combinational Logic Schemes. Review. *Taurida Journal of Computer Science Theory and Mathematics*. 2020; (2):39-60. Available at: https://www.elibrary.ru/item.asp?id=44600613 (accessed 18.03.2021). (In Russ., abstract in Eng.)

[12] Gurov S.I. Review of Works on the use of Majority Boolean Algebra for the Synthesis of Combinational Logic Schemes. *Taurida Journal of Computer Science Theory and Mathematics*. 2020; (3):59-76. Available at: https://www.elibrary.ru/item.asp?id=46301092 (accessed 18.03.2021). (In Russ., abstract in Eng.)

[13] Amarú L., Gaillardon P.-E., Chattopadhyay A., De Micheli G. A Sound and Complete Axiomatization of Majority-*n* Logic. *IEEE Transactions on Computers*. 2016; 65(9):2889-2895. (In Eng.) DOI: https://doi.org/10.1109/TC.2015.2506566

[14] Chua L., Roska T. The CNN Paradigm. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*. 1993; 40(3):147156. (In Eng.) DOI: https://doi.org/10.1109/81.222795

[15] Haaswijk W., Collins E., Seguin B., Soeken M., Kaplan F., Susstrunk S., De Micheli G. Deep Learning for Logic Optimization Algorithms. *2018 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE Press, Florence, Italy; 2018. p. 14. (In Eng.) DOI: https://doi.org/10.1109/ISCAS.2018.8351885

[16] Nair V., Hinton G.E. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10).* Haifa, Israel; 2010. p. 807814. Available at: https://icml.cc/Conferences/2010/papers/432.pdf (accessed 18.03.2021). (In Eng.)

[17] Mazyavkina N., Sviridov S., Ivanov S., Burnaev E. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*. 2021; 134:105400. (In Eng.) DOI: https://doi.org/10.1016/j.cor.2021.105400

[18] Möller F.J.D., Bernardino H.S., Gonçalves L.B., Soares S.S.R.F. A Reinforcement Learning Based Adaptive Mutation for Cartesian Genetic Programming Applied to the Design of Combinational Logic Circuits. In: Ed. by R. Cerri, R. C. Prati. *Intelligent Systems. BRACIS 2020. Lecture Notes in Computer Science*. 2020; 12320:18-32. Springer, Cham. (In Eng.) DOI: https://doi.org/10.1007/978-3-030-61380-8_2

[19] Zhu K., Liu M., Chen H., Zhao Z., Pan D. Exploring Logic Optimizations with Reinforcement Learning and Graph Convolutional Network. *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20).* Association for Computing Machinery, New York, NY, USA; 2020. p. 145-150. (In Eng.) DOI: https://doi.org/10.1145/3380446.3430622

[20] Silver D., Huang, A., Maddison, C., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*. 2016; 529:484-489. (In Eng.) DOI: https://doi.org/10.1038/nature16961

[21] Mnih V., Kavukcuoglu K., Silver D., et al. Human-level control through deep reinforcement learning. *Nature*. 2015; 518:529-533. (In Eng.) DOI: https://doi.org/10.1038/nature14236

[22] Machine Learning Proceedings 1991. In: Ed. by L. A. Birnbaum, G. C. Collins. *Proceedings of the Eighth International Conference*. Morgan Kaufmann, Evanston, Illinois; 1991. 661 p. (In Eng.) DOI: https://doi.org/10.1016/C2009-0-27661-6

[23] Brudermueller T., Shung D.L., Stanley A.J., Stegmaier J., Krishnaswamy S. Making Logic Learnable With Neural Networks. arXiv:2002.03847. 2020. Available at: https://arxiv.org/abs/2002.03847 (accessed 18.03.2021). (In Eng.)

[24] Somayaji N.S.K., Hu H., Li P. Prioritized Reinforcement Learning for Analog Circuit Optimization With Design Knowledge. *2021 58th ACM/IEEE Design Automation Conference (DAC).* IEEE Press, San Francisco, CA, USA; 2021. p. 1231-1236. (In Eng.) DOI: https://doi.org/10.1109/DAC18074.2021.9586189

[25] Coello C.A., Zavala R.L.G., García B.M., Aguirre A.H. Ant Colony System for the Design of Combinational Logic Circuits. In: Ed. by J. Miller, A. Thompson, P. Thomson, T. C. Fogarty. *Evolvable Systems: From Biology to Hardware. ICES 2000. Lecture Notes in Computer Science*. 2000; 1801:21-30. Springer, Berlin, Heidelberg. (In Eng.) DOI: https://doi.org/10.1007/3-540-46406-9_3

Submitted 18.03.2021; approved after reviewing 11.05.2021;
accepted for publication 24.05.2021.

About the authors:

Sergey I. Gurov, Senior Researcher, Associate Professor of the Department of Mathematical Methods of Forecasting, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), Ph.D. (Phys.-Math.), Associate Professor, **ORCID: https://orcid.org/0000-0001-5486-1357**, sgur@cs.msu.ru

Dmitry V. Zolotarev, student of the Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0002-5349-2147**, ub8caf@mail.ru

Alexander I. Samburskiy, student of the Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University (1 Leninskie gory, Moscow 119991, GSP-1, Russian Federation), **ORCID: https://orcid.org/0000-0003-1188-0229**, sashasamb@yandex.ru

All authors have read and approved the final manuscript.