stage. This procedure converges in a finite number of stages. Such lexicographic disintegration is a decomposition procedure and is suitable for planning and scheduling large-scale projects. Practical calculations for the convex problem in the initial statement have been performed with the help of ready-made software, and the obtained greedy solution of the problem-consequence has theoretical significance in graph theory.

Keywords: *large-scale innovation project, even allocation of resource, risk of performance, reliability index of the project's activities, concave reliability function, inaccurate schedule, inaccurate slacks, greedy algorithm, maximum path with double weights on arcs*

DOI: 10.17587/it.25.650-657

References

1. **The Owner's** Role in Project Risk Management, Washington, D. C., The National Academies Press, 2005, available at: www.nap.edu.
2. **Elkjaer M.** *Int. J. Project Management,* 2000, vol. 18, no. 2, pp. 139—147.
3. **Rabbani M., Tavakkoli Moghaddam R., Jolai F., Ghorbani H. R.** *IJE Transactions A. Basics,* 2006, vol. 19, no. 1, pp. 55—66.
4. **Tsvirkun A. D., Akinfiyev V. K., Konovalov Ye. N.** Modeling and management of innovative programs in large-scale technical systems, Moscow, Preprint / ICS, 1991 (in Russian).
5. **Topka V. V.** *Automation and Remote Control*, 2012, vol. 73, no. 7, pp. 1173—1180 (in Russian).
6. **Burkov V. N., Zalozhnev A. Yu., Novikov D. A.** Graph theory in the management of organizational systems, Moscow, Sinteg, 2001, 124 p. (in Russian).
7. **Christofides N.** Graph theory: an algorithmic approach. Computer science and applied mathematics, Academic Press, 1975.
8. **Karzanov A. V.** On the minimum mean weight sections and cycles of an oriented graph, In: Qualitative and approximate methods for investigating operator equations, Yaroslavl, Yar SU, 1985, pp. 72—83 (in Russian).
9. **Topka V. V.** *Journal of Computer and Systems Sciences International*, 2014, vol. 53, no. 6, pp. 877—895 (in Russian).
10. **Topka V. V.** Dialogue integrated system for the management and design of research and development (DISUPIR 3.2). Certificate of state registration of computer programs No. 2015617277. The date of state registration in the Register of computer programs July 6, 2015 (in Russian).
11. **Topka V. V.** The lexicographic solution of the two-criteria problem of project planning with a restriction on the indicator of its reliability, Izv. RAS. TiSU, 2014, no. 6, pp. 105—123 (in Russian).

УДК 004.3.02 DOI: 10.17587/it.25.657-662

А. Д. Иванников, д-р техн. наук, гл. науч. сотр., e-mail: ADI@ippm.ru,
Институт проблем проектирования в микроэлектронике РАН, Зеленоград, Москва

Теоретические основы выбора множества отладочных тестов проектов цифровых систем на основе алфавита выполняемых функций[1]

Рассматриваются цифровые системы управления объектами, функционирование которых может быть представлено как последовательность выполнения функций из конечного алфавита. Последовательности выполняемых функций представлены как их произведения, показано, что они образуют частичную полугруппу. При отладке проектов цифровых систем методом моделирования для проверки правильности проекта используются отладочные тесты проектов, которые должны наиболее полно проверить правильность выполнения всех функций проектируемой системой. Рассмотрены методы составления и модификации разработчиком перечня выполняемых функций наиболее удобным для проверки образом. Кроме того, рассматривается разбиение каждой функции цифровой системы на подфункции в целях проверки правильности функционирования различных режимов аппаратного обеспечения и ветвей программ. Формализованно описывается последовательность действий разработчика при формировании множества отладочных тестов цифровых систем.

Ключевые слова: *моделирование цифровых систем, отладка проектов, отладочные тесты, алфавит функций цифровой системы*

Введение

При разработке проектов цифровых систем на основе средств микроэлектроники важным этапом является проверка правильности проекта методом моделирования на ЭВМ [1—4]. Это позволяет выявить ошибки и неточности проекта цифровой системы и исключить необходимость перевыполнения проекта при обнаружении ошибок в уже изготовленной системе. Процесс выявления и устранения ошибок как в проекте

[1] Работа выполнена при поддержке гранта РФФИ № 17-07-00683.

технических средств, так и в программном обеспечении цифровой системы называется *отладкой проекта* цифровой системы [5].

Во время такой отладки на программную модель цифровой системы подаются некоторые тестовые последовательности — отладочные тесты, вызывающие моделирование выполнения функций цифровой системы. При этом осуществляется контроль соответствия моделируемых состояний внутренних регистров цифровой системы и сигналов на ее выводах требуемым. В случае обнаружения каких-либо отклонений от требуемых реакций в текст программного обеспечения или в схему технических средств вносятся необходимые коррекции, и процесс повторяется. В данной работе мы предполагаем, что разработчику известны требуемые значения внутренних состояний и выходных сигналов цифровой системы, и он имеет возможность отслеживать эти данные в процессе моделирования. Мы предполагаем также, что разработчик может определить те неточности в программном обеспечении и технических средствах, которые являются причиной неправильного функционирования модели отлаживаемой цифровой системы. В настоящей работе рассматривается вопрос о выборе некоторого набора входных тестовых последовательностей, подаваемых на модель отлаживаемой цифровой системы в целях проверки ее функционирования.

В связи с тем что выполнение какой-либо функции цифровой системой может быть инициировано не только входным сигналом, подаваемым на систему, но и самой цифровой системой, в качестве аргументов функционирования цифровых систем будем рассматривать входные взаимодействия, включающие как входные сигналы системы, так и ее выходные сигналы управления обменом [1].

Последовательность выполняемых функций как полугруппа

В работе [5] показано, что функционирование цифровых систем управления объектами может быть представлено как последовательность выполняемых функций, каждая из которых задается подачей на цифровую систему некоторого входного взаимодействия. Иными словами, каждая цифровая система выполняет некоторую последовательность функций из конечного алфавита **K**, причем выполнение каждой функции вызывается одним из входных взаимодействий определенного класса. Входные взаимодействия, задающие выполнение цифровой системой функции $k \in \mathbf{K}$, могут различаться значениями данных, временными соотношениями между отдельными сигналами в допустимых пределах, могут иметь и другие различия.

Состав набора функций **K** определяется разработчиком на основании технического задания, которое обычно содержит описание функционирования цифровой системы управления с той или иной степенью подробности, но не содержит полной формальной спецификации. Разработчик цифровой системы на основе понимания общей идеи, а также с учетом знания предыдущих подобных разработок выбирает набор выполняемых цифровой системой функций, которые бы полностью обеспечивали ее требуемое функционирование.

Следует отметить, что выделение набора функций цифровой системы является в определенной степени процессом субъективным. Прежде всего могут быть рассмотрены функции различного объема. Так, если цифровая система определяет какие-то действия на основании данных, поступающих с нескольких датчиков, то одной из выполняемых функций может быть опрос датчиков. Вместе с тем опрос каждого датчика может рассматриваться как отдельная функция. Может быть также выделена некоторая обобщенная функция, например, "вычерчивание отрезка прямой" для некоторой цифровой системы управления чертежным автоматом, а могут быть выделены более детализированные функции: "вычерчивание сплошного отрезка прямой", "вычерчивание штрихового отрезка прямой", "вычерчивание штрихпунктирного отрезка прямой". Может быть также выделена некоторая функция цифровой системы, содержание выполнения которой зависит от состояния цифровой системы в конкретный момент времени.

Рассматривая последовательное выполнение функций k_i и k_j как операцию умножения $\cdot$, а последовательность $k_i \cdot k_j$ — как произведение, можно сказать, что на множестве последовательностей выполняемых функций **F** определена полугруппа, в общем случае частичная.

Рассмотрим полугруппу $\langle \mathbf{F}, \cdot \rangle$, в общем случае частичную, с конечным множеством **K** порождающих элементов k.

В наиболее общем случае произведение $f' \cdot f''$ определено, если f' и f'' могут быть представлены как $f' = k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k'$, $f'' = k'' \cdot k_{j_2} \cdot \ldots \cdot k_{j_l}$, и если существуют произведения

$k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k''$, $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k'' \cdot k_{j_2}$, ..., ..., $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k' \cdot k'' \cdot k_{j_2} \cdot \ldots \cdot k_{j_l}$. Таким образом, вопрос о существовании произведения $f' \cdot f''$ сводится к вопросу о существовании произведения $f \cdot k$, где $f \in \mathbf{F}$, $k \in \mathbf{K}$.

Достаточно сложным является вопрос о начальном состоянии цифровой системы и подмножестве функций, которые могут быть выполнены при нахождении системы в этом состоянии. Одним из возможных вариантов решения этого вопроса является принятие за начальное состояние системы того состояния, в котором она находится при включении питания. При этом вопрос о существовании произведения $f \cdot k$, где $f \in \mathbf{F}$, $k \in \mathbf{K}$, можно рассматривать при представлении f как $k_{i_1} \cdot k_{i_2} \cdot \ldots \cdot k_{i_n}$, где n — число функций в f.

Множество **F** есть множество слов в конечном алфавите **K**, которое может быть задано конечным инициальным автоматом без выходов

$$\mathbf{A} = (\mathbf{K}, \mathbf{X}, x_н, \varphi),$$

где **K** — множество входных символов; **X** — множество состояний автомата **A**; $x_н$ — начальное состояние, $x_н \in \mathbf{X}$; φ: $\mathbf{K} \times \mathbf{X} \to \mathbf{X}$ — частичное переходное отображение.

Слова множества **F** могут заканчиваться любым символом из **K**. В связи с этим любое состояние из **X** может являться заключительным. Каждому состоянию x, $x \in \mathbf{X}$, соответствует подмножество $\mathbf{F}^x$ ($\mathbf{F}^x \subset \mathbf{F}$) слов, заканчивающихся в состоянии x.

Автомат **A** задает все возможные последовательности функций, выполняемые цифровой системой, т. е. множество допустимых слов **F**. Будем называть автомат **A** автоматом функций. Спецификации на входные взаимодействия цифровой системы могут определяться заданием множества входных взаимодействий, соответствующих каждой функции k, $k \in \mathbf{K}$ [1, 5], и автомата функций **A**.

Модификация алфавита функций цифровой системы

При составлении отладочного набора тестов для проверки правильности функционирования проектируемой цифровой системы было бы гораздо удобнее, если бы существование произведения $k_i \cdot k_j$ не зависело бы от предыстории выполняемых цифровой системой функций. Другими словами, желательно, чтобы рассмотрение вопроса о существовании произведения $f \cdot k$, где $f \in \mathbf{F}$, $k \in \mathbf{K}$, был бы заменен на вопрос существования произведения $k_i \cdot k_j$, где $k_i \in \mathbf{K}$, $k_j \in \mathbf{K}$.

Итак, пусть разработчик составил начальное конечное множество функций $\mathbf{K}_н$, выполняемых цифровой системой управления, причем функционирование проектируемой цифровой системы можно представить как множество некоторых последовательностей этих функций. Рассмотрим всевозможные пары k_i, k_j, где $k_i \in \mathbf{K}_н$, $k_j \in \mathbf{K}_н$, с точки зрения существования их произведений. Выделим те пары, для которых существование произведения (возможность выполнения цифровой системой этих функций последовательно) зависит от последовательности выполненных ранее функций, другими словами, от внутреннего состояния цифровой системы. В связи с тем что число внутренних состояний цифровой системы управления всегда конечно, вторую функцию k_j пары можно разбить на конечное число подфункций $k_j^1, k_j^2, \ldots, k_j^n$ таким образом, что существование произведений $k_i \cdot k_j^l$, $l \in \{1, \ldots, n\}$ не зависит от предыстории функционирования цифровой системы, причем в большинстве случаев такое разделение на подфункции имеет явный физический смысл [6].

Модифицированному алфавиту функций $\mathbf{K}_м$ соответствует автомат функций $\mathbf{A}_м$. Для него тестирование правильности выполнения допустимых последовательностей функций осуществляется набором тестовых последовательностей, содержащих все переходы автомата $\mathbf{A}_м$. Разделение функций k_j на подфункции и осуществлялось для того, чтобы такой алгоритм выбора тестовых последовательностей был возможен.

Формирование множества отладочных тестов для проверки правильности выполнения отдельной функции

Рассмотрим множество входных взаимодействий $\mathbf{M}_k$, инициирующих выполнение цифровой системой функции $k \in \mathbf{K}_м$. Несмотря на то что все входные взаимодействия множества инициируют выполнение одной и той же функции k, ее выполнение может осуществляться по-разному с использованием различающихся в какой-то степени алгоритмов и, соответственно, различных режимов работы аппаратного обеспечения и различных ветвей программного обеспечения. Так, напри-

мер, при выполнении функции "вычерчивание прямой" цифровой системой управления чертежным автоматом может осуществляться вычерчивание сплошной, штриховой или штрихпунктирной линии, вертикальной, горизонтальной или наклонной линии, вычерчивание со сменой или без смены пера.

При этом все множество входных взаимодействий может быть представлено как объединение непересекающихся подмножеств, каждое из которых инициирует выполнение специфического варианта функции k. При выборе множества отладочных тестов желательно предусмотреть проверку правильности выполнения каждого такого специфического варианта.

Разработчик для каждой функции формирует набор признаков $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_n$ и алфавиты их значений. Пусть имеется конечное множество признаков $\mathcal{L}_1$, $\mathcal{L}_2$, ..., $\mathcal{L}_n$, каждый из которых может принимать конечное множество значений $\mathbf{Z}_1^*, \mathbf{Z}_2^*, \ldots, \mathbf{Z}_n^*$. Тогда взаимосвязь признаков можно представить двудольным графом $\mathcal{G}(\mathcal{V}, \mathcal{E})$ Множество вершин этого графа есть $\mathcal{V} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\} \cup \left\{\bigcup_{i=1}^{n} \mathbf{Z}_i^*\right\}$. Одна группа вершин представляет множество признаков, а другая группа — возможные значения каждого признака. Множество ребер $\mathcal{E} = \mathcal{E}^1 \cup \mathcal{E}^2$, $\mathcal{E}^1 \cap \mathcal{E}^2 = \varnothing$. Ребра множества $\mathcal{E}^1$ соединяют вершину $\mathcal{L}_i$ с вершиной z, если $z \in \mathbf{Z}_i^*$. Ребра множества $\mathcal{E}^2$ соединяют вершину z_i^j, где $z_i^j \in \mathbf{Z}_i^*$, с вершиной $\mathcal{L}_i$, если признак $\mathcal{L}_i$ определен для входного взаимодействия μ в том случае, если для этого входного взаимодействия $\mathcal{L}_i = z_i^j$.

Множество отладочных тестов считается полным, если для любого признака $\mathcal{L}_i$, $i = 1, ..., n$, и $z \in \mathbf{Z}_i^*$ в множестве отладочных тестов найдется по крайней мере одно входное взаимодействие, для которого $\mathcal{L}_i = z$. Минимальным полным множеством отладочных тестов назовем такое полное множество тестов, число тестов в котором минимально. Задача составления минимального полного множества отладочных тестов для проверки выполнения каждой функции состоит в выборе сочетаний значений признаков для каждого отладочного теста множества.

В работе [7] показано, что минимальная мощность минимального полного множества отладочных тестов есть $\max|\mathbf{Z}_i^*|$, $i = 1, ..., n$, где n — число признаков $\mathcal{L}_1$, ..., $\mathcal{L}_n$; максимальная мощность минимального полного множества отладочных тестов есть $\sum_{i=1}^{n}(|\mathbf{Z}_i^*| - 1) + 1$.

Последовательность действий для формирования множества тестовых примеров для отладки проектов цифровых систем

Напомним, что мы рассматриваем цифровые системы, функционирование которых может быть представлено как последовательность выполнения функций из некоторого конечного алфавита. Практический опыт подтверждает такое предположение, хотя в ряде случаев удобнее рассматривать параллельное и не всегда независимое выполнение нескольких последовательностей функций цифровой системой. Рассмотрим последовательность действий для формирования множества тестовых примеров для отладки проектов таких цифровых систем.

1. Разработчик формирует начальный конечный по мощности перечень функций, выполняемых цифровой системой, а именно алфавит $\mathbf{K}$.

2. Для каждой функции $k_i \in \mathbf{K}$ разработчик определяет, какие функции $k_j \in \mathbf{K}$, $j = 1, ..., n$, где n — число элементов в алфавите $\mathbf{K}$, могут выполняться после k_i. При этом для каждого k_i выделяется подмножество $\{k_{j_1}, \ldots, k_{j_m}\} \subset \{k_{i_1}, \ldots, k_{i_n}\}$, для каждого элемента k_{j_l} которого существование произведения $k_i \cdot k_{j_l}$ зависит от некоторых условий, т. е. некоторой предыдущей последовательности функций или, по-другому, внутреннего состояния цифровой системы. Затем каждая функция k_{j_l} разбивается на подфункции $k_{j_l}^1, \ldots, k_{j_l}^q$, причем если выполнение функции k_{j_l} инициируется одним из входных взаимодействий множества $\mathbf{M}_{j_l}$, то это множество разбивается на непересекающиеся подмножества $\mathbf{M}_{j_l}^r$, $r = 1, ..., q$, и $\mathbf{M}_{j_l} = \bigcup_{r=1}^{q} \mathbf{M}_{j_l}^r$. Таким образом, формируется новый также конечный перечень функций, выполняемых цифровой системой, а именно модифицированный алфавит $\mathbf{K}_{\text{мод}}$. При дальнейшем описании будем рассматривать этот модифицированный алфавит как алфавит функций, выполняемых цифровой системой, и будем обозначать его $\mathbf{K}$.

3. Для полученного алфавита функций $\mathbf{K}$ и частичной полугруппы $\langle \mathbf{F}, \cdot \rangle$ с конечным множеством $\mathbf{K}$ порождающих элементов k строится граф переходов $\mathbf{G(V, E)}$ автомата функций $\mathbf{A}_\text{ф}$. В графе $\mathbf{G(V, E)}$ каждая вершина $v \in \mathbf{V}$ помечена одним из элементов k алфавита $\mathbf{K}$, т. е. представляет все слова $\mathbf{F}^k \subset \mathbf{F}$, заканчивающиеся элементом k алфавита функций. Исключение представляет собой вершина $v_{\text{нач}}$, соответствующая пустому слову и представляющая

состояние, в котором цифровая система оказывается после включения питания. Множество ребер $e_{ij} \in \mathbf{E}$ графа $\mathbf{G(V, E)}$ определяет возможные произведения $k_i \cdot k_j$. Иными словами, из каждой вершины, помеченной как k_i, ребра выходят в вершины k_j, помеченные функцией алфавита $\mathbf{K}$, которая может быть выполнена после функции k_i, т. е. условием наличия ребра e_{ij} является существование произведения $k_i \cdot k_j$.

Одним из возможных методов задания графа $\mathbf{G(V, E)}$ является перечень пар (k_i, k_j), произведение которых существует. Такой перечень удобно задавать в виде таблицы, в которой для каждой функции k_i как первого элемента пары указываются вторые элементы пары k_l^i, $l = 1, ..., q_i$. В начале таблицы указываются все функции, выполнение которых возможно из начального состояния $x_н$, соответствующего включению питания системы.

Вариант задания перечня пар (k_i, k_j), произведение которых существует

Первый элемент пары k_i, $i = 1, ..., n$, или начальное состояние $x_н$	Вторые элементы пары k_j
$x_н$	$k_{j_1^н}$ ⋮ $k_{j_{q_н}^н}$
k_1	$k_{j_1^1}$ ⋮ $k_{j_{q1}^1}$
⋮	⋮
k_n	$k_{j_1^n}$ ⋮ $k_{j_{qn}^n}$

Набор отладочных тестов должен включать последовательное выполнение всех заданных в таблице пар функций.

4. Как указывалось выше, выполнение одной и той же функции цифровой системы может осуществляться по разному, с использованием различных ветвей программного обеспечения и различных режимов аппаратных средств. Поэтому выполнение каждой функции при выполнении набора отладочных тестов должно осуществляться с различными наборами признаков $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_n$. Выбранные различные сочетания признаков определяют число необходимых выполнений функции k_i.

Заключение

Таким образом, в соответствии с изложенными теоретическими основами формирования множества тестовых примеров для отладки проектов цифровых систем необходимо, чтобы:

- система тестов содержала все произведения функций k_i, k_j, заданные в таблице;
- кратное выполнение всех функций k с заданными наборами признаков $\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_n$, т. е. выполнение каждой функции k_i столько раз, сколько наборов признаков для нее определено.

Разработка практического алгоритма формирования минимального набора отладочных тестов проектов цифровых систем на основе проведенного теоретического исследования является текущей задачей.

Список литературы

1. **Иванников А. Д., Стемпковский А. Л.** Формализация задачи отладки проектов цифровых систем // Информационные технологии. 2014. № 9. С. 3—10.

2. **Lin Yi-Li, Su Alvin W. Y.** Functional Verification for SoC Software / Hardware Co-Design: From Virtual Platform to Physical Platform // 2011 IEEE International SOC Conference (SOCC). P. 201—206.

3. **Shi Jin, Liu Weichao, Jiang Ming** el al. Software Hardware Co-Simulation and Co-Verification in Safety Critical System Design // 2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT). P. 71—74.

4. **Кащеев Н. И., Пономарев Д. М., Подъяблонский Ф. М.** Построение тестов цифровых схем с использованием обобщенной модели неисправностей и непрерывного подхода к моделированию // Вестник Нижегородского университета им. Н. И. Лобачевского. 2011. № 3 (2). С. 72—77.

5. **Иванников А. Д., Стемпковский А. Л.** Математическая модель отладки проектов сложных цифровых систем и микросистем на основе представления последних в виде семейства стационарных динамических систем // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2014. Часть II. С. 123—128.

6. **Иванников А. Д.** Формирование отладочного набора тестов для проверки функций цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2017. Т. 18. № 12. С. 795—801.

7. **Иванников А. Д.** Автоматизация генерации отладочных тестов для проектов цифровых систем управления объектами // Мехатроника, автоматизация, управление. 2018. Т. 19. № 12. С. 770—776.

A. D. Ivannikov, Dr. of Tech. Sc., e-mail: ADI@ippm.ru,
Institute for Design Problems in Microelectronics of Russian Academy of Sciences

The Theoretical Basis for the Selection of Design Debugging Tests Set for Digital Systems Based on the Alphabet of Functions Performed

The paper considers control digital systems, the functioning of which can be represented as a sequence of functions from the finite alphabet. The sequences of functions performed are presented as their products, it is shown that they form a partial semigroup. When debugging projects of digital systems using the simulation method, to verify the correctness of the project, project debugging tests are used, which should most fully verify the correctness of the performance of all functions by the designed system. The methods of compiling and modifying the list of digital system functions by the developer in the way, which is most convenient for verification, are presented. In addition, the breakdown of each digital system function into subfunctions is considered in order to verify the correct functioning of various hardware modes and program branches. The action sequence of a designer while digital systems debugging tests set developing is formally described.

Keywords: *digital systems modeling, design debugging, debugging tests, digital system functions alphabet*

Acknowledgements: This work was supported by the RFBR grant No. 17-07-00683.

DOI: 10.17587/it.25.657-662

References

1. **Ivannikov A. D., Stempkovsky A. L.** *Informacionnye Tekhnologii*. 2014, no. 9, pp. 3—10 (in Russian).
2. **Lin Yi-Li, Su Alvin W. Y.** *2011 IEEE International SOC Conference (SOCC)*, 2011, pp. 201—206.
3. **Shi Jin, Liu Weichao, Jiang Ming,** el al. *2013 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, 2013, pp. 71—74.
4. **Kasheev N. I., Ponomarev D. M., Podyablonsky F. M.** *Vestnik Nijegorodskogo Universiteta*, 2011, no. 3 (2), pp. 72—77 (in Russian).
5. **Ivannikov A. D., Stempkovsky A. L.** Complex Digital Systems and Microsystems Design Debugging Mathematic Model on the Basis of Stationary Dynamic System Family Presentation, *Problemi Rasrabotki Perspektivnih Mikro- I Nanoelectronnih Sistem*, 2014, part II, pp. 123—128 (in Russian).
6. **Ivannikov A. D.** *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2017, vol. 18, no. 12, pp. 795—801 (in Russian).
7. **Ivannikov A. D.** *Mekhatronika, Avtomatizatsiya, Upravlenie*, 2018, vol. 19, no. 12, pp. 770—776 (in Russian).

УДК 004.85 DOI: 10.17587/it.25.662-669

М. Д. Ершов, аспирант, e-mail: ershov.m.d@rsreu.ru,
Рязанский государственный радиотехнический университет, г. Рязань

Методы оптимизации первого порядка в машинном обучении[1]

Рассмотрены проблемы, возникающие при обучении многослойных нейронных сетей прямого распространения из-за недостатков метода градиентного спуска. Выполнен обзор методов оптимизации первого порядка, которые нашли широкое применение в машинном обучении, и менее известных методов. Обзор включает стохастический градиентный спуск, быстрый градиентный метод Нестерова и различные методы с адаптацией скорости обучения. Описаны особенности каждого метода и проблемы их использования на практике.

Ключевые слова: *нейронные сети, машинное обучение, глубокое обучение, оптимизация, градиентный спуск, адаптивная скорость обучения*

Введение

Методы оптимизации являются востребованными при решении различных научных и инженерных задач, на практике возникают новые проблемы оптимизации, и их сложность растет. Одной из основных составляющих машинного обучения также является математическая оптимизация. С точки зрения машинного обучения основная цель оптимизации заключается в минимизации или максимизации значения математической функции (целевой функции, функции потерь или функции стоимости). Методы оптимизации составляют совершенно отдельную область исследований, которая почти столь же раз-

[1] Исследования выполнены при финансовой поддержке стипендии Президента Российской Федерации молодым ученым и аспирантам (СП-2578.2018.5).