DOI 10.17587/issn.1684-6400

ISSN 1684-6400

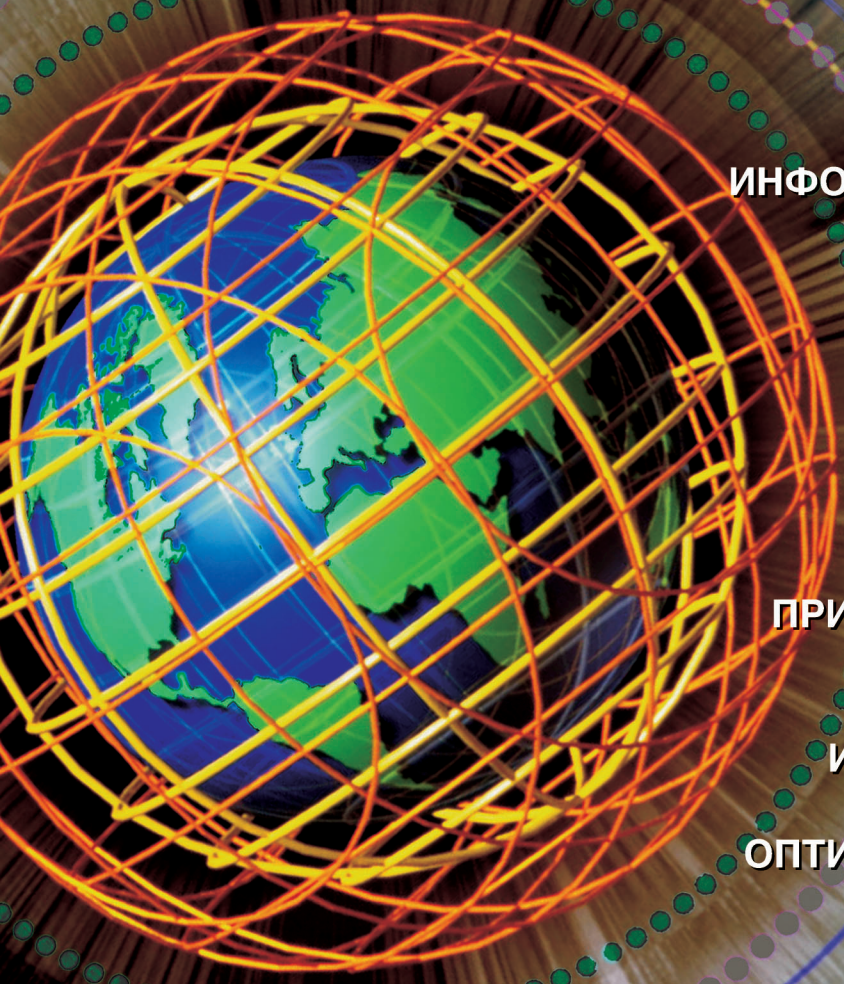
# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Том 25

2019

№ 11

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ



САПР

КОМПЬЮТЕРНАЯ ГРАФИКА

МЕТОДЫ ПРОГРАММИРОВАНИЯ

ОПЕРАЦИОННЫЕ СИСТЕМЫ И СРЕДЫ

ТЕЛЕКОММУНИКАЦИИ И ВЫЧИСЛИТЕЛЬНЫЕ СЕТИ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

НЕЙРОСЕТИ И НЕЙРОКОМПЬЮТЕРЫ

СТРУКТУРНЫЙ СИНТЕЗ

ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

ОПТИМИЗАЦИЯ И МОДЕЛИРОВАНИЕ

ИТ В ОБРАЗОВАНИИ

ГИС

Рисунки к статье О. А. Сосниной, А. Д. Филинских, Н. А. Ложкиной

## «АНАЛИЗ МЕТОДОВ СОЗДАНИЯ ВИРТУАЛЬНЫХ МОДЕЛЕЙ НЕТРИВИАЛЬНЫХ ФОРМ»

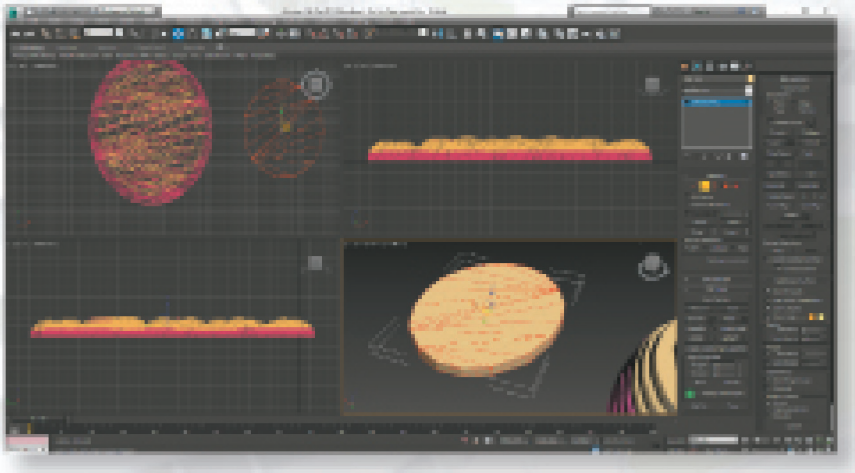


**Рис. 3. Файл формата *.stp**

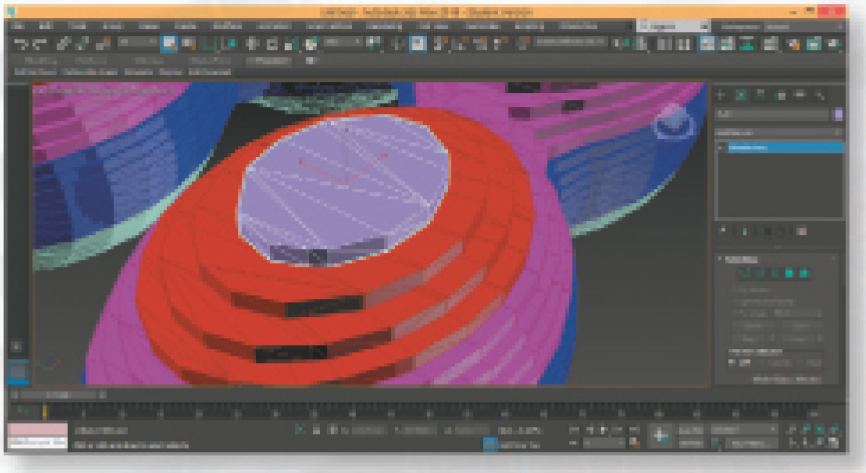


**Рис. 4. Файл формата *.sat**

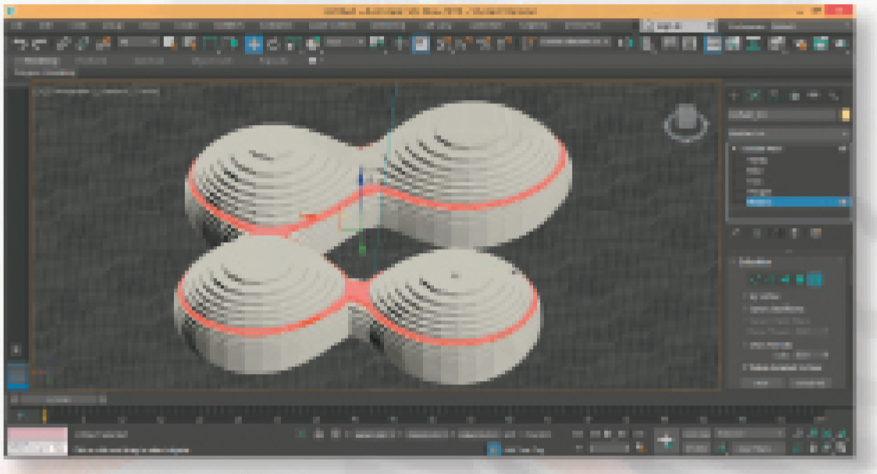


**Рис. 5. Модель, открытая в формате *.skp**

# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Том 25
2019
№ 11

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

**Издается с ноября 1995 г.** DOI 10.17587/issn.1684-6400

**УЧРЕДИТЕЛЬ**
**Издательство "Новые технологии"**


## СОДЕРЖАНИЕ




Информация о журнале доступна по сети Internet по адресу **http://novtex.ru/IT.**
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

# INFORMATION TECHNOLOGIES
# INFORMACIONNYE TEHNOLOGII

**Vol. 25**
**2019**
**No. 11**

## THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

**Published since November 1995**

**ISSN 1684-6400**


## CONTENTS




Complete Internet version of the journal at site: **http://novtex.ru/IT.**
According to the decision of the Higher Certifying Commission of the Ministry of Education of Russian Federation, the journal is inscribed in "The List of the Leading Scientific Journals and Editions wherein Main Scientific Results of Theses for Doctor´s or Candidate's Degrees Should Be Published"

УДК 004.94 DOI: 10.17587/it.25.643-649

**В. Н. Гридин**[1], науч. руководитель, д-р техн. наук, проф., e-mail: info@ditc.ras.ru,
**В. И. Анисимов**[1, 2], гл. науч. сотр., д-р техн. наук, проф., e-mail: vianisimov@inbox.ru,
[1] Центр информационных технологий в проектировании РАН,
[2] Санкт-Петербургский государственный электротехнический университет

# Организация параллельных вычислительных процессов в распределенных системах автоматизации схемотехнического проектирования[1]

*Рассматриваются методы построения распределенных систем автоматизированного проектирования на основе диакоптических подходов к декомпозиции моделируемой схемы. В статье рассматривается методика описания систем с помощью обобщенных сигнальных графов, которые отображают системы уравнений, записанных в обобщенной причинно-следственной форме. Такое описание помимо наглядного представления о структуре связей между подсхемами позволяет снять проблему выбора последовательности нумерации внутренних переменных подсхем и переменных узлов связи. Декомпозиция системы на базе диакоптического подхода позволяет реализовать эффективный вычислительный процесс, основанный на параллельной организации вычислений, когда формирование и обработка уравнений для отдельных подсхем выполняется на различных узлах распределенной сети. При этом достигается значительная экономия требуемого объема оперативной памяти и существенно снижается время, необходимое для решения задачи моделирования больших схем, что позволяет ощутимо повысить эффективность распределенной системы автоматизации схемотехнического проектирования.*

***Ключевые слова:*** *системы автоматизированного проектирования, диакоптика, моделирование систем, обобщенные сигнальные графы, распределенные системы, интернет-технологии*

## Введение

Внедрение в системы автоматизированного проектирования интернет-технологий позволяет обеспечить доступ к информационным ресурсам удаленных баз данных и организовать дистанционное взаимодействие распределенных коллективов пользователей САПР. Одним из направлений реализации таких технологий является построение систем автоматизированного проектирования на основе веб-сервисов, позволяющих приложениям взаимодействовать друг с другом независимо от платформы, на которой они развернуты, а также от языка программирования [1—4].

Использование интернет-технологий на основе веб-сервисов при разработке программного обеспечения распределенных систем автоматизации схемотехнического проектирования позволяет:

- перейти к концепции описания интерфейсов и взаимодействий на основе XML, объединяя любой тип приложения с другим приложением и предоставляя свободу изменения и развития с течением времени до тех пор, пока поддерживается соответствующий интерфейс;
- использовать более высокий уровень абстракции программного обеспечения, при котором оно может быть задействовано пользователями, работающими только на уровне бизнес-анализа;
- организовать взаимодействие между различными сервисами на любой платформе, написанными на любом языке программирования;
- учитывать слабосвязанность программного обеспечения, благодаря которой взаимодействие между приложениями сервиса не

[1] Работа выполняется в рамках темы № 0071-2019-0001.

нарушается каждый раз, когда меняется дизайн или реализация какого-либо сервиса;
- предоставлять существующему или унаследованному программному обеспечению сервисный интерфейс без изменения оригинальных приложений, давая им возможность полноценно взаимодействовать в сервисной среде;
- адаптировать существующие приложения к меняющимся условиям проектирования и потребностям заказчика.

Основным стандартом технологии веб-сервисов является протокол WSDL, который используется для создания описания интерфейса веб-сервиса. WSDL-документ описывает информационные блоки сервиса и включает в себя информацию об интерфейсе и методах, доступных для общего пользования, дает описание типов данных, использующихся при передаче в запросах и ответах по соответствующему протоколу, а также информацию о транспортном протоколе, который может быть использован для работы с сервисом, адресную информацию о местоположении описываемого сервиса. Следует отметить, что стандарт WSDL не зависит от какого-либо определенного протокола обмена XML-данными, но содержит встроенные средства поддержки протокола SOAP. Протокол SOAP — это межплатформенный стандарт, используемый для форматирования сообщений, которыми обмениваются веб-сервис и клиентское приложение. SOAP определяет XML-конверт для сообщений веб-сервиса, модель обработки и алгоритм кодирования информации перед ее отправкой. В технологии веб-сервисов используются два типа SOAP-сообщений. Первый тип является сообщением запроса, которое клиент направляет веб-сервису для инициализации выбранного веб-метода. Второй тип является ответным сообщением, которое веб-сервис возвращает клиенту.

При использовании универсально описанных интерфейсов появляется возможность использовать программные компоненты повторно, что позволяет снизить трудоемкость разработки САПР и осуществить более эффективное инвестирование в программное обеспечение. С учетом высокой стоимости программного обеспечения современных САПР указанная возможность имеет большое экономическое значение.

Главным достоинством распределенной архитектуры является возможность постоянной модификации программного обеспечения и расширения путем добавления новых, более эффективных методов. Единственным требованием для реализации таких возможностей является соблюдение неизменности выбранного при разработке интерфейса.

Переход к распределенной архитектуре имеет особенно большое значение для моделирования больших систем, когда часто возникают ситуации, при которых необходимо использовать вычислительные ресурсы, превосходящие имеющиеся в распоряжении пользователя возможности. Если моделируемая система имеет слабосвязанную иерархическую структуру, то наиболее эффективным способом организации вычислительных процессов является декомпозиция исходной системы на ряд подсхем с использованием диакоптического подхода [5—7]. Такой подход может существенно повысить эффективность вычислительных процессов и привести требуемые для решения задачи ресурсы в соответствие с реальными возможностями. При этом возможна организация параллельных вычислений в локальной или глобальной сети, где компьютер каждого узла сети осуществляет формирование и обработку данных, связанных только с отдельной подсистемой.

Вследствие декомпозиции схемы на слабосвязанные подсхемы обеспечивается существенное увеличение эффективности программного обеспечения по затратам требуемой оперативной памяти и по скорости решения задачи моделирования. Единственным ограничением для реализации декомпозиционного подхода к моделированию больших схем является требование отсутствия индуктивных связей между отдельными подсхемами, а также локализация внутри каждой подсхемы ее зависимых источников и управляющих этими источниками переменных.

В статье рассматривается методика организации параллельных вычислительных процессов в распределенных системах автоматизации схемотехнического проектирования на основе реализации диакоптического подхода к моделированию больших схем. Такой подход в сочетании с организацией параллельного вычислительного процесса позволяет существенно уменьшить время, необходимое для решения задач моделирования больших слабосвязанных систем, и тем самым значительно повысить эффективность распределенной системы автоматизированного проектирования.

### Моделирование больших схем на основе методов диакоптики

Диакоптический подход к моделированию больших схем является одним из видов перехода от сложной схемы к более простой, когда проводится разделение схемы на изолированные части. Сложная схема по определенным правилам расчленяется на некоторое число подсхем, и для каждой из подсхем в отдельности проводится требуемый анализ, общее решение получается путем соединения полученных частных решений для подсхем. При таком подходе не требуется составления полной системы уравнений, достаточно сформировать уравнения для ее элементарных подсхем, которые могут быть сделаны настолько малыми, насколько это практически целесообразно.

Для построения и преобразования описания моделируемой схемы на основании диакоптического подхода целесообразно использовать топологическое описание схемы с помощью обобщенных сигнальных графов, которые в отличие от сигнального графа отображают систему уравнений, записанных в обобщенной причинно-следственной форме [8, 9]. Обобщенный сигнальный граф содержит взвешенные вершины, среди которых могут быть и вершины с нулевым весом. Такое описание помимо наглядного представления о структуре связей между подсхемами позволяет снять проблему выбора последовательности нумерации внутренних переменных подсхем и переменных узлов связи.

Моделируемая схема может быть разбита на отдельные подсхемы, связанные между собой произвольным образом. Основным ограничением при таком разделении схемы является требование выполнения условий слабосвязанности отдельных подсхем, которые требуют отсутствия индуктивных связей между отдельными подсхемами, а также размещения внутри каждой подсхемы ее зависимых источников и управляющих этими источниками переменных. Кроме того, при математическом описании каждой подсхемы должно выполняться условие существования ее обратной матрицы.

При решении задачи декомпозиции исходной схемы для каждого узла моделируемой схемы следует ввести узловую потенциальную переменную, определяющую состояние этого узла, и из всех введенных узловых переменных $v_p$ образовать вектор узловых потенциалов $X_0 = V_0 = [..., v_p, ...]^{\mathrm{T}}$. В каждой подсхеме следует выделить внутренние переменные $x_i$ и составить для каждой подсхемы вектор внутренних переменных $X_k = [..., x_i, ...]^{\mathrm{T}}$.

Тогда уравнение для $k$-й подсхемы в обобщенной причинно-следственной форме может быть записано в следующем виде:

$$W_{kk}X_k = -W_{k0}X_0 - C_kS_k = 0, \quad k = \overline{1,m}, \qquad (1)$$

где $W_{kk}$, $W_{k0}$ — матрицы параметров $k$-й подсхемы; $S_k$ — вектор задающих источников $k$-й подсхемы; $C_k$ — матрица соединений задающих источников $k$-й подсхемы.

Аналогично можно записать в обобщенной причинно-следственной форме уравнение для узлов связи между подсхемами:

$$W_{00}X_0 = -\sum_{k=1}^{m} W_{0k}X_k - C_0S_0. \qquad (2)$$

Здесь $W_{00}$, $W_{0k}$ — матрицы параметров для узлов связи между подсхемами; $S_0$ — вектор задающих источников, подключенных к узлам связи; $C_0$ — матрица соединений задающих источников, подключенных к узлам связи.

Уравнениям (1) и (2) соответствует обобщенный сигнальный граф [1], приведенный на рис. 1.

Приведенная структура обобщенного сигнального графа наглядно демонстрирует соответствующую структуру математического описания слабосвязанной схемы, в которой взаимодействие между отдельными подсхемами реализуется только через узловые переменные связи.

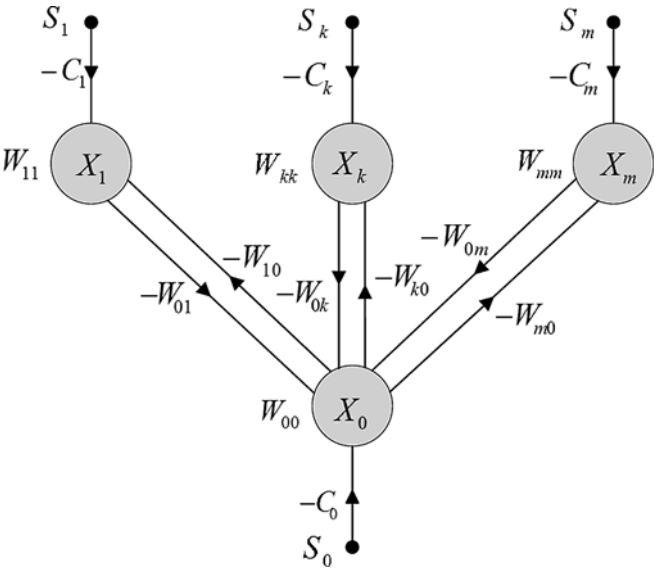



**Рис. 1. Обобщенный сигнальный граф слабосвязанной схемы**

Умножая уравнение (1) на обратную матрицу $W_{kk}^{-1}$, можно привести это уравнение к виду

$$X_k = -\bar{W}_{k0}X_0 - \bar{C}_k S_k, \quad k = \overline{1,m}, \qquad (3)$$

где $\bar{W}_{k0} = W_{kk}^{-1}W_{k0}$; $\bar{C}_k = W_{kk}^{-1}C_k$.

Преобразованной форме уравнения подсхем (3) соответствует преобразованный обобщенный сигнальный граф, приведенный на рис. 2.

Очевидно, что переход к такой структуре обобщенного сигнального графа возможен только при условии существования для всех подсхем обратной матрицы $W_{kk}^{-1}$, что требует выполнения условия неравенства нулю определителя $\Delta$ матрицы $W_{kk}$ для всех подсхем. Условия существования обратной матрицы $W_{kk}^{-1}$ для всех подсхем обычно выполняются, если в качестве отдельных подсхем выделять законченные функциональные блоки, каждый из которых выполняет вполне определенную задачу. Однако в исключительных случаях может понадобиться более детальный анализ структуры выделенных подсхем и значений их определителей, что особенно удобно выполнять топологическим путем на основании изучения обобщенных сигнальных графов отдельных подсхем.

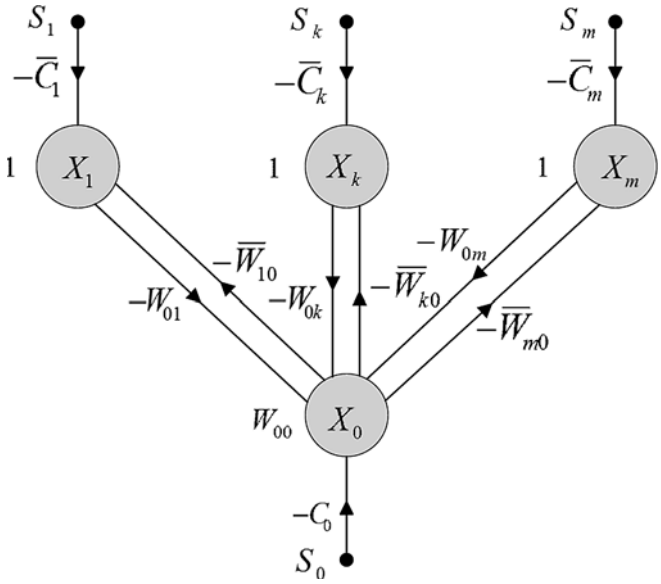



**Рис. 2. Обобщенный сигнальный граф преобразованной схемы**

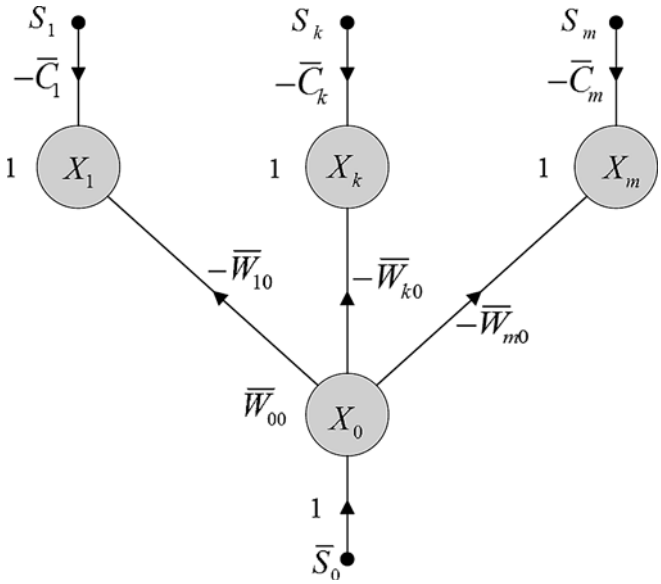



**Рис. 3. Заключительный этап топологического описания слабосвязанной схемы**

Для построения окончательного представления топологического описания задачи подставим уравнение (3) в уравнение (2), что дает

$$\bar{W}_{00}X_0 = \bar{S}_0, \qquad (4)$$

где

$$\bar{W}_{00} = W_{00} - \sum_{k=1}^{m} W_{0k}\bar{W}_{k0},$$

$$\bar{S}_0 = -C_0 S_0 + \sum_{k=1}^{m} W_{0k}\bar{C}_k S_k.$$

Уравнениям (3) и (4) соответствует обобщенный сигнальный граф, приведенный на рис. 3.

Таким образом, в результате проведенной декомпозиции могут быть определены все узловые переменные вектора $X_0 = V_0 = [..., v_p, ...]^T$, что позволяет найти на основании выражения (1) все внутренние переменные $X_k = [..., x_i, ...]^T$ подсхем моделируемой системы. При этом вместо решения общего уравнения системы $WX + S = 0$ используется обращение матриц $W_{kk}$ и $W_{00}$, порядок которых путем выбора соответствующего числа подсистем и узлов связи может быть сделан сколь угодно малым.

## Организация параллельных вычислительных процессов

Декомпозиция системы на базе диакоптического подхода весьма удобна для организации вычислительного процесса, основанного на параллельной организации вычислений, когда формирование и обработка уравнений для отдельных подсистем выполняется на различных компьютерах, работающих в узлах некоторой сети. Каждый из таких компьютеров должен

быть связан с центральным компьютером, осуществляющим решение уравнений связи на основе данных, полученных при обработке уравнений подсистем. Результат решения уравнений связей затем пересылается на компьютеры, реализующие работу с отдельными подсхемами, где и выполняется окончательный расчет всех переменных (рис. 4).

Структурная схема возможной организации параллельного вычислительного процесса приведена на рис. 5.

В соответствии с приведенной структурой организации параллельного вычислительного процесса процесс решения задачи разделяется на следующие этапы:

1. Для каждой $k$-й подсистемы на основе введенной информации на компьютере узла сети $N_k$ формируются массивы параметров блочных матриц $W_{kk}$, $W_{k0}$, $W_{0k}$, $W_{00}$, $S_0$, $S_k$.

2. Выполняется алгоритм Гаусса—Жордано [10—12] для каждой $k$-й подсистемы, в результате чего матрица $W_{kk}$ преобразуется в единичную диагональную матрицу, а матрицы $W_{k0}$ и $S_k$ преобразуются соответственно в $\bar{W}_{k0}$ и $\bar{S}_k$.

3. В каждом узле сети $N_k$ вычисляются поправки $\Delta_k W_{00} = \bar{W}_{k0} W_{0k}$ и $\Delta_k S_0 = W_{0k} \bar{S}_k$, которые после завершения обработки каждой подсистемы передаются на компьютер центрального узла $N_0$.

4. На компьютере центрального узла $N_0$ вычисляются матрицы $\bar{W}_{00} = W_{00} - \sum_{k=1}^{m} \Delta_k W_{00}$ и $\bar{S}_0 = S_0 - \sum_{k=1}^{m} \Delta_k S_0$, после чего на основе выражения (2) проводится расчет вектора узловых переменных $X_0$.

5. Значение вектора $X_0$ пересылается на все узлы сети $N_k$, где для каждой $k$-й подсистемы выполняется на основании выражения (1) расчет векторов $X_k$ внутренних переменных отдельных подсистем.

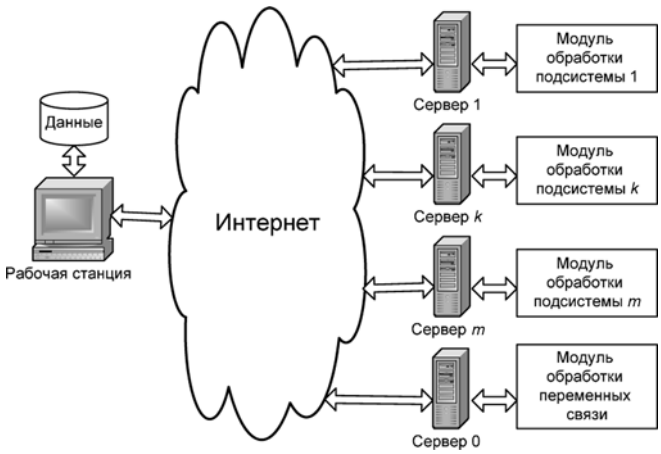



**Рис. 4. Архитектура распределенной системы автоматизированного проектирования**

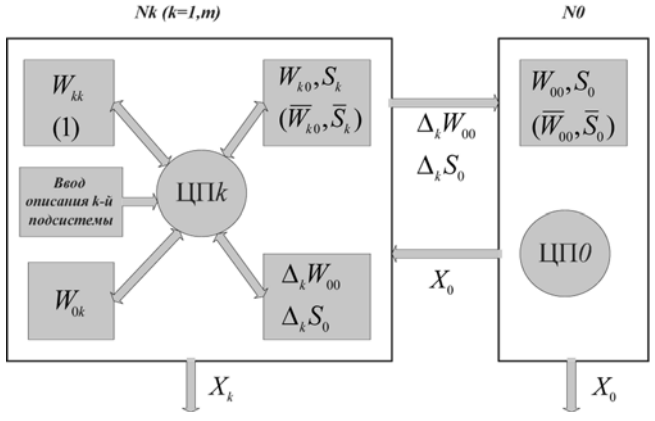



**Рис. 5. Организация параллельного вычислительного процесса**

Использование диакоптического подхода для моделирования больших систем на основе декомпозиции исходной системы на ряд подсхем позволяет существенно повысить эффективность вычислительных процессов при параллельной организации вычислений. При этом достигается значительная экономия требуемого объема оперативной памяти и существенно снижается время, необходимое для решения задачи моделирования больших систем, имеющих слабосвязанную иерархическую структуру.

Пусть $V_D$ — объем памяти, необходимой для расчета с использованием диакоптического подхода, $V$ — объем памяти, требуемый для расчета обычными методами. Если система разделена на $m$ подсхем, число переменных в каждой подсхеме равно $n_k$, а число переменных связи составляет $n_0$, то коэффициент экономии оперативной памяти $\alpha = V/V_D$ диакоптического подхода при организации параллельного вычислительного процесса можно определить выражением

$$\alpha = \frac{\left(n_0 + \sum_{k=1}^{m} n_k\right)^2}{(\max_k n_k + n_0)^2}.$$

Так, если система разделена на 10 одинаковых подсхем ($m = 10$), и число переменных в каждой подсхеме равно числу переменных связи $n_k = n_0$, то значение коэффициента экономии оперативной памяти $\alpha \approx 30$.

Эффективность параллельного вычислительного процесса по быстродействию может быть определена на основании соотношения

$$\beta = \frac{\left(n_0 + \sum_{k=1}^{m} n_k\right)^3}{\max_k n_k^3 + bn_0^2 \max_k n_k + n_0^3},$$

где $b = 1/3$ или $b = 1/2$.

Если моделируемая система разбита на 10 одинаковых подсхем ($m = 10$), и число переменных в каждой подсхеме равно числу переменных связи, т. е. $n_k = n_0$, то получим при переходе на параллельный вычислительный процесс повышение быстродействия вследствие использования диакоптического подхода в 570 раз.

Таким образом, рассмотренные методы моделирования систем на основе декомпозиции исходной системы на ряд подсхем и организации параллельного вычислительного процесса существенно уменьшают время, необходимое для решения задач моделирования больших слабосвязанных систем, и тем самым позволяют значительно повысить эффективность распределенной системы автоматизированного проектирования.

## Заключение

Использование диакоптического подхода для моделирования больших систем на основе декомпозиции исходной системы на ряд подсхем позволяет значительно повысить эффективность вычислительных процессов. Эффективность метода диакоптики проявляется наиболее полно при анализе сложных систем и сводится к сокращению затрат машинного времени, тем большему, чем сложнее анализируемая модель системы.

В статье рассмотрена методика описания систем с помощью обобщенных сигнальных графов, которые отображают систему уравнений, записанных в обобщенной причинно-следственной форме, и содержат взвешенные вершины, среди которых могут быть и вершины с нулевым весом. Такое описание помимо наглядного представления о структуре связей между подсхемами позволяет снять проблему выбора последовательности нумерации внутренних переменных подсхем и переменных узлов связи.

Декомпозиция системы на базе диакоптического похода весьма удобна для организации вычислительного процесса, основанного на параллельной организации вычислений, когда формирование и обработка уравнений для отдельных подсистем выполняется на различных компьютерах, работающих в узлах некоторой сети. При этом достигается значительная экономия требуемого объема оперативной памяти и существенно снижается время, необходимое для решения задачи моделирования больших систем, имеющих слабосвязанную иерархическую структуру, что позволяет значительно повысить эффективность распределенной системы автоматизированного проектирования.

### Список литературы


1. **Анисимов В. И., Гридин В. Н.** Методы построения схем автоматизированного проектирования на основе Интернет-технологий и компактной обработки разреженных матриц // Информационные технологии в проектировании и производстве. 2009. № 1. С. 3—7.

2. **Коваленко О. С., Курейчик В. М.** Обзор проблем и состояний облачных вычислений и серверов // Известия ЮФУ. Технические науки. 2012. № 7. С. 146—153.

3. **Гридин В. Н., Дмитревич Г. Д., Анисимов Д. А.** Построение схем автоматизированного проектирования на основе Web-технологий // Информационные технологии. 2011. № 5. С. 23—27.

4. **Анисимов Д. А.** Методы построения схем автоматизации схемотехнического проектирования на основе вебсервисов // Известия СПбГЭТУ "ЛЭТИ". 2012. № 10. С. 56—61.

5. **Крон Г.** Исследование сложных схем по частям — диакоптика / Пер с англ. М.: Наука, Главная редакция физико-математической литературы, 1972. 544 с.

6. **Хэпп Х.** Диакоптика и электрические цепи / Пер. с англ. М.: Мир, 1974. 342 с.

7. **Анисимов В. И., Тарасова О. Б., Алмаасали С. А.** Организация вычислительных процессов при моделировании схем на основе методов диакоптики // Информационные технологии в проектировании и производстве. 2013. № 4. С. 14—17.

8. **Мэзон С., Циммерман Г.** Электронные цепи, сигналы и схемы / Пер. с англ. М.: Изд. иностр. лит. 1963. 619 с.

9. **Анисимов В. И.** Топологический расчет электронных схем. Л.: Энергия, 1977. 238 с.

10. **Фаддеев Д. К., Фаддеева В. Н.** Вычислительные методы линейной алгебры. М.: Физматгиз, 1960. 656 с.

11. **Чуа Л. О., Лин П. М.** Машинный анализ электронных схем / Пер с англ. М.: Энергия, 1980. 640 с.

12. **Влах И., Сингхал К.** Машинные методы анализа и проектирования электронных схем / Пер. с англ. М.: Радио и связь, 1988. 560 с.

**V. N. Gridin**[1], Scientific Director, D. Sc., Professor, e-mail: info@ditc.ras.ru,
**V. I. Anisimov**[1, 2], Chief Researcher, D. Sc., Professor, e-mail: vianisimov@inbox.ru,
[1] Design information technologies Center Russian Academy of Sciences, Odintsovo, Russian Federation,
[2] Saint-Petersburg Electrotechnical University, Saint-Petersburg, Russian Federation

# Organization of Parallel Computing Processes in Distributed Automation Systems of Circuit Design

*Discusses methods to build distributed systems of the automated designing on the basis of by diakoptical approach to the decomposition of the simulated circuit. The article deals with the method of describing systems using generalized signal graphs, which display the system of equations written in generalized cause-and-effect form. This description, in addition to a visual representation of the structure of links between subcircuits, eliminates the problem of choosing the sequence of numbering of internal variables of subcircuits and variables of communication nodes. The decomposition of the system on the basis of the diacoptic approach allows to implement an effective computational process based on the parallel organization of calculations, when the formation and processing of equations for individual subcircuits is performed on different nodes of the distributed network. This achieves significant savings in the required amount of RAM and significantly reduces the time required to solve the problem of modeling large circuits, which can significantly improve the efficiency of distributed automation system circuit design.*

***Keywords:*** *computer-aided design, diacoptics, system modeling, generalized signal graphs, distributed systems, Internet technologies*

**Acknowledgements:** The work is carried out within the framework of topic No. 0071-2019-0001.

DOI: 10.17587/it.25.643-649

**References**

1. **Anisimov V. I., Gridin V. N.** *Informacionnye Tekhnologii v Proektirovanii i Proizvodstve*, 2009, no. 1, pp. 3—7 (in Russian).
2. **Kovalenko O. S., Kureychik V. M.** *Izvestiya YUFU. Tekhnicheskie nauki*, 2012, no. 7, pp. 146—153 (in Russian).
3. **Gridin V. N., Dmitrevich G. D., Anisimov D. A.** *Informacionnye Tekhnologii*. 2011, no. 5, pp. 23—27 (in Russian).
4. **Anisimov D. A.** *Izvestiya SPbGETU "LETI",* 2012, no. 10, pp. 56—61 (in Russian).
5. **Kron G.** A research of complex circuits in parts — a diakoptika, Moscow, Nauka, Glavnaya redakciya fiziko-matematicheskoj literatury, 1972, 544 p. (in Russian).
6. **Hepp H.** Diakoptika and electrical circuits, Moscow, Mir, 1974, 342 p. (in Russian).
7. **Anisimov V. I., Tarasova O. B., Almaasali S. A.** *Informacionnye Tekhnologii v Proektirovanii i Proizvodstve*, 2013, no. 4, pp. 14—17 (in Russian).
8. **Mezon S., Zimmerman G.** Electronic chains, signals and schemes, Moscow, Izd. inostr. lit., 1963, 619 p. (in Russian).
9. **Anisimov V. I.** Topological calculation of electronic circuits, Leningrad, Energiya, 1977, 238 p. (in Russian).
10. **Faddeev D. K., Faddeeva V. N.** Computing methods of a linear algebra. Gosudarstvennoe izdatel'stvo fiziko-matematicheskoj literatury, Moscow, 1960, 656 p. (in Russian).
11. **Chua L. O., Lin P. M.** Computerized analysis of electronic circuits, Moscow, Energiya, 1980, 640 p. (in Russian).
12. **Vlakh I., Singkhal To.** Machine methods of the analysis and design of electronic circuits, Moscow, Radio i svyaz', 1988, 560 p. (in Russian).

УДК 338.24 DOI: 10.17587/it.25.650-657

**А. Д. Цвиркун,** д-р. техн. наук, зав. лабораторией, e-mail: tsvirkun@ipu.ru,
**В. В. Топка,** канд. техн. наук, ст. науч. сотр., e-mail: topka3@mail.ru,
ФГБУН Институт проблем управления им. В. А. Трапезникова РАН, Москва

# Равномерное распределение невозобновимого ресурса в крупномасштабных инновационных проектах

*Рассматривается задача равномерного распределения по минимаксному критерию невозобновимого ресурса по работам инновационного проекта. Применяется лексикографический метод упорядочивания данных минимаксных критериев. Для решения задачи по одному критерию разработан жадный алгоритм отыскания максимального пути на ациклическом орграфе с двойными весами на дугах, имеющий квадратичную по числу дуг вычислительную трудоемкость. Полученное решение позволяет построить календарный план проекта с четырьмя видами временных резервов при неточных исходных данных.*

***Ключевые слова:*** *крупномасштабный инновационный проект, календарное планирование, показатель надежности работ проекта, жадный алгоритм, максимальный путь с двойными весами на дугах*

## Введение

В имеющихся программных системах управления проектами существуют модули, выполняющие оценку таких параметров, как риск стоимости и риск расписания. Вместе с тем для моделирования неопределенности и оценки риска проектов может быть предложен подход на основе понятий теории надежности.

В теории надежности показатель надежности сети определяется вероятностью ее связности. Для связности требуется, чтобы нашелся хотя бы один путь, связывающий начало и конец сети. Однако в управлении проектами в сети типа PERT для надежного выполнения проекта необходимо выполнение всех работ, входящих в проект. В этом случае ставится задача оценки не только связности начала и конца сети, но и полноты реализации проекта.

В информационных системах управления проектами количественная оценка риска расписания (Risk of Schedule) и риска стоимости (Risk of Cost) выполняется на основе моделирования методом Монте-Карло.

Предлагаемый подход позволяет выполнить анализ риска исполнения (Risk of Performance) [1] проекта, аппарат для которого не разработан. Оценка риска исполнения, т. е. того факта, выполнена ли требуемая миссия по завершении работы, иными словами, достигнуты ли заданные тактико-технические характеристики (ТТХ), основана на кумулятивной вероятности благоприятного/неблагоприятного исходов. Вероятность успеха проекта — это вероятность того, что при исполнении данного проекта отказ не произойдет, а вероятность безотказного исполнения принята за показатель надежности выполнения. Преимущество предлагаемого подхода состоит в том, что он дает инструмент для работы в неисследованной области оценивания риска исполнения проекта.

Для анализа параметра риска исполнения проекта в данной работе будем учитывать показатели надежности работ и надежности проекта в целом. В этих целях инновационный проект рассматривается как сложная техническая система, в которой в выполнении проекта задействованы машины, механизмы, оборудование с участием людей, обеспеченных необходимыми ресурсами.

Для работ инновационных проектов с высокой степенью неопределенности для получения характеристик случайных величин их выполнения — вероятности достижения цели, т. е. для построения функции распределения такой случайной величины, можно использовать эм-

пирические данные, или данные, полученные в результате активного эксперимента, в котором в результате моделирования методом Монте-Карло получена функция распределения вероятности технического успеха работ — степень достижения заданных ТТХ данной разработки в виде монотонно возрастающей с насыщением функции затраченных ресурсов [2].

Вероятность выполнения работы к некоторому времени характеризует возможность достижения определенных технических показателей результата работы при условии, что предшествующие работы, обеспечивающие ее начало, выполнены. Эта вероятность зависит от объема тех или иных израсходованных ресурсов, которые имеют стоимостное выражение.

Развиваемый в статье подход направлен на разработку количественных методов оценки и оптимизации характеристик проекта с учетом критерия равномерного распределения невозобновимого (складируемого) ресурса при ограничении на показатель надежности проекта.

## Показатели надежности работ и проекта в целом

Ранее для описания выполнения инновационных проектов использовались целочисленные, линейные и некоторые нелинейные модели [3]. Будем моделировать [4] зависимость вероятности технического успеха (показателя надежности) работ проекта от затраченного однородного невозобновимого ресурса $u$ двухпараметрическими степенными вогнутыми функциями вида

$$p_j(u_j) = \frac{u_j^{\alpha_j}}{u_j^0} \in [\varepsilon, 1],\ \varepsilon \to +0; \qquad u_j \in [\varepsilon, (u_j^0)^{1/\alpha_j}],\ j = 1, \ldots, n, \tag{1}$$

где $0 < \alpha_j < 1$ — параметр формы, $u_j^0 > 0$ — параметр масштаба. Пусть задан ациклический ориентированный граф $G(E, \Gamma)$, где $E$ — множество вершин, соответствующих работам проекта (представление *Activity-on-Node*), а $\Gamma \subseteq E \times E$ — множество дуг, задающих отношения частичного порядка непосредственного технологического предшествования работ. Сеть $G(E, \Gamma)$ имеет одну конечную и одну начальную вершины, $n$ — конечная вершина (последний номер среди всех вершин проекта), и фиктивную вершину 0, означающую запуск проекта.

Таким образом, имеем ациклический ориентированный граф $G(E, \Gamma)$, в вершинах которого задан показатель надежности разработки, удовлетворяющий соотношению (1). Будем задавать надежность проекта вероятностью его технического успеха (степенью достижения заданных ТТХ проекта) к определенному допустимому сроку, т. е. оценкой надежности проекта в самом неблагоприятном случае. В теории надежности такая оценка определяется слабейшим звеном, т. е. наихудшей из технологических цепочек из вершины нижнего уровня в конечную вершину

$$P_n(u) = \min_{\mu_i} \prod_{(j,k)\in\mu_i} w_{jk} \frac{u_j^{\alpha_j}}{u_j^0} \in [\varepsilon, 1],\ \varepsilon \to +0; \qquad \alpha_j \in (0, 1),\ w_{jk} \in [e_0, 1],\ u_j^0 > 0;\ \forall \mu_j, \tag{2}$$

где $w_{jk} \in [\varepsilon_0, 1]$, $\varepsilon_0 > 0$ — коэффициент надежности дуги $(j, k)$ (входящей в матрицу путей по дугам сети $G(E, \Gamma)$: $(l_{j,k})$, $(j, k) \in \mu_i$) передачи результата $j$-й работы посредством дуги $(j, k)$ для выполнения $k$-й работы (подобно передаточной функции соединения звеньев в теории автоматического управления). Через $\mu_i$ обозначен путь с № $i = 1, \ldots, m$ на сети из начальной вершины в конечную вершину $n$. При решении задач по планированию проекта в условиях степенной модели [5] не учитывается временна́я переменная. В статье будем рассматривать детерминированную сетевую модель проекта с дизъюнктивными входными дугами ИЛИ или с конъюнктивными входными дугами И, обладающими тем свойством, что для показателя надежности проекта выполняется оценка снизу (2) [5]. Таким образом, будем говорить о $p_i$ из (1) как о показателе надежности работы, а о $P_n$ из (2) — как об оценке показателя гарантированной надежности выполнения проекта. В модели для представления сетевых ограничений будем использовать в качестве характеристической функции системы ограничений матрицу путей (по дугам) сети $G(E, \Gamma)$: $(l_{j,k})$, $(j, k) \in \mu_i$, которая строится на основе списка дуг сети. В сетевой модели проекта $G(E, \Gamma)$ каждый путь $\mu_i = (0, 1, \ldots, j, k, \ldots, n)$ из начальной вершины в конечную однозначно представляется строкой специально построенной матрицы путей (по дугам) $(l_{j,k})$, $(j, k) \in \mu_i$, в которой ее элементы — дуги — стоят в $i$-й строке только в том случае, когда $(j, k)$-я дуга принадлежит $i$-му пути $(j, k) \in \mu_i$. Всем остальным случаям, когда $(j, k) \notin \mu_i$, соответствуют пустые клетки.

### Задача последовательного равномерного распределения невозобновимого ресурса по критерию Чебышева

Пусть $u = (u_1, u_2, ..., u_n)$ — вектор невозобновимого ресурса, выделяемого для выполнения всех работ проекта $G(E, \Gamma)$. В качестве критерия распределения ресурсов будем использовать равномерный, или чебышевский, критерий в виде

$$\max_{j \in E} u_j \to \inf_{u \in U}, u_j \geqslant 0,\ j = 1, \ldots, n. \tag{3}$$

Задачу вида (3) принято называть задачей дискретного минимакса.

Данная задача дискретного минимакса рассматривается в виде задачи гладкой условной минимизации:

$$u^0 \to \inf_{(u^0, u_j) \in U},$$

$$U = \begin{cases} u_j \leqslant u^0,\ \forall j = 1, \ldots, n; \\ \prod\limits_{(j,k) \in \mu_i} w_{jk} \dfrac{u_j^{\alpha_j}}{u_j^0} \geqslant p_0 \in [\varepsilon, 1], \forall \mu_i, i = 1, \ldots, m; \\ u_j \in U_0 = [\varepsilon, (u_j^0)^{1/\alpha_j}], j = 1, \ldots, n. \end{cases} \tag{4}$$

Логарифмируя вторую группу ограничений задачи и принимая во внимание первую, получим

$$u^0 \to \min_{(u^0, u_j) \in U}, \tag{5}$$

$$U = \begin{cases} \ln u^0 \geqslant \dfrac{\sum\limits_{(j,k) \in \mu_i} \ln u_j^0 - \sum\limits_{(j,k) \in \mu_i} \ln w_{jk} + \ln p_0}{\sum\limits_{(j,k) \in \mu_i} \alpha_j}; \\ u_j \in U_0. \end{cases} \tag{6}$$

Пусть ограничения задачи совместны, и здесь и в дальнейшем $U \neq \varnothing$. В силу неравенства (6), следствия исходной постановки, учитывая сетевую специфику задачи, будем искать ее решение в виде максимального пути с двойными весами на дугах, когда оптимальное решение задачи (5), (6) реализуется на пути (возможно, множестве путей) со значением $\lambda^0$ таким, что

$$\lambda^0 = \max_{\mu_i} \frac{\sum\limits_{(j,k) \in \mu_i} \ln u_j^0 - \sum\limits_{(j,k) \in \mu_i} \ln w_{jk} + \ln p_0}{\sum\limits_{(j,k) \in \mu_i} \alpha_j} = \max_{\mu_i} \frac{\sum\limits_{(j,k) \in \mu_i} \sigma_{jk}}{\sum\limits_{(j,k) \in \mu_i} \alpha_j}.$$

При этом $\min\limits_{(u^0, u_j) \in U} u^0 = \exp \lambda^0$. Поэтому необходимо найти множество максимальных путей с двойными весами на дугах в ациклическом ориентированном графе $G(E, \Gamma)$.

$$\mu^{0*} = \operatorname{Arg} \max_{\mu_i} \lambda(\mu_i),$$

$$u_j^* = u^0 = \exp \lambda^0 = \exp \lambda(\mu^{0*}),\ j \in \mu^{0*}.$$

Будем полагать, что параметры задачи таковы, что здесь и в дальнейшем условие $u \in U_0$ выполняется. Получили *ресурсный критический путь* с максимальным расходом невозобновимого ресурса. Пусть $\mu^{0*} = \operatorname{Arg} \max\limits_{\mu_i} \lambda(\mu_i)$ найдено. Распределение ресурсов по минимаксному критерию продолжаем последовательно, до тех пор пока на каком-то этапе $q$ ($q$ — индикатор множества) не получим область определения задачи $E = E^q \cup ... \cup E^r \cup ... \cup E^0$ и

$$u_j^{q*} = u^q = \exp \lambda^q(\mu^{q*}) \to \text{fixed},$$

$$E^q = \left\{ j \middle| j \in \mu^{q*} \setminus \bigcup_0^{q^*-1} \mu^r \right\} \to \text{fixed}, E = \bigcup_0^q E^r \tag{7}$$

(fixed — зафиксированные значения).

Общая задача решается последовательной оптимизацией оставшегося подграфа по минимаксному критерию на каждом этапе, как в лексикографическом методе упорядочивания решения задач многокритериальной оптимизации. Данная процедура сходится за конечное число этапов. Такое лексикографическое разложение, по сути, представляет собой декомпозиционную процедуру и подходит для планирования проектов большой размерности.

Для нахождения пути максимальной эффективности в работе [6, с. 13] предложен алгоритм переборного типа (с экспоненциальной трудоемкостью), сводящийся к отысканию максимального пути в сети. В работе [7, с. 192] для отыскания в графе с двойными весами цикла с минимальным значением используется алгоритм обнаружения в графе цикла отрицательного веса (при условии, что для всех циклов сумма весов в знаменателе положительна). Решение данной задачи с двойными весами с помощью предложенного алгоритма требует $O\left(|E|^3 \log \frac{1}{\eta}\right)$ операций, где $\eta$ — погрешность (слабополиномиальный алгоритм), $|E|$ — число вершин в сети. В работе [8] предложен общий метод решения задачи о среднем в линейных пространствах. Из него в качестве частного случая следует сильнополиномиальный ал-

горитм нахождения минимального в среднем контура в слабосвязном ориентированном графе, имеющий трудоемкость $O(|E|^3|\Gamma|)$. Для решения задачи отыскания максимального пути с двойными весами в ациклическом орграфе $G(E, \Gamma)$ предлагается жадный эвристический алгоритм, для которого справедлива, как будет показано ниже, оценка $O(|\Gamma|^2)$ его вычислительной трудоемкости. Его преимущество по сравнению с приведенными выше алгоритмами состоит в более низкой трудоемкости.

### Алгоритм отыскания максимального пути в орграфе с двойными весами на дугах

Построение множества путей $M = \{\mu_s\}$, с весами $\sigma_{ji}$ и $\alpha_j$ на дугах $l_{ji}$ $(j, i) \in \Gamma$, орграфа $G(E, \Gamma)$ покрывающего все его вершины $E$.

*Начальный шаг.* Просмотренные дуги $L := \varnothing$, вершины $E' := \varnothing$, номер пути $s := 1$.

1. *Выбор дуги на множестве непросмотренных дуг* $\tilde{\Gamma} = \Gamma \backslash L$. *В* ациклическом орграфе $G(E, \Gamma)$ выберем дугу $l_{ji}$, $(j, i) \in \Gamma$, оптимальную п локальному критерию

$$l_{ji} = \operatorname{Arg}\max_{(j,i)\in\Gamma} \frac{\sigma_{ji}}{\alpha_j}.$$

В массив $L$ заносим дугу $l_{ji}$: $L := l_{ji}$.

2. *Построение пути.*

2.1. *Добавление новых дуг.*

Построим путь $\mu_s$ из начальной вершины $j = 1$: $\Gamma_j^{-1} = \varnothing$ в конечную вершину $j = n$: $\Gamma_j = \varnothing$ по локальному критерию, т. е. для дуги $l_{ji}$ выбираются смежные дуги $l_{kj}$ и $l_{it}$ из условий

$$l_{kj} = \operatorname{Arg}\max_{k\in\Gamma_j^{-1}} \frac{\sigma_{kj} + \sum\limits_{(j,i)\in L} \sigma_{ji}}{\alpha_k + \sum\limits_{j\in L} \alpha_j};$$

$$l_{it} = \operatorname{Arg}\max_{t\in\Gamma_i} \frac{\sum\limits_{(j,i)\in L} \sigma_{ji} + \sigma_{it}}{\sum\limits_{j\in L} \alpha_j + \alpha_i},$$

которые добавляются к имеющейся дуге $L := l_{ji}$, образуя путь $\mu_s := \{...l_{kj} \cup L \cup l_{it} ...\}$ с двойными весами, где $\sum\limits_{(j,i)\in L} \sigma_{ji}$ и $\sum\limits_{j\in L} \alpha_j$ — значения суммарных весов для пути, найденного на предыдущем этапе. Положив $\Delta_\mu^s := 1$, для построенного таким образом пути $\mu_s$ определим величину $\lambda^s(\mu_s)$, $s = 1, 2, ...$:

$$\max_{\mu} \frac{\sum\limits_{(j,i)\in\mu} \sigma_{ji}}{\sum\limits_{(j,i)\in\mu} \alpha_j} \Delta_\mu^s = \lambda^s(\mu_s).$$

Список просмотренных дуг — $L := \{...l_{kj} \cup L \cup l_{it} ...\}$; для просмотренных вершин — $E^s = \{j | j \in \mu_s\}$; $E' := E' \cup E^s$.

2.2. *Недопущение зацикливания алгоритма на просмотренных путях.*

Для того чтобы не допустить зацикливания алгоритма вдоль просмотренного пути $\mu_s$, в дальнейшем будем вычислять значение индикатора $\Delta_\mu^{s+1}$. Если для непросмотренных вершин $E \backslash E' \neq \varnothing$, то пусть $k \in E \backslash E'$. Положим

$$\delta_k = \begin{cases} 1, \text{ если } k \in E', \\ 0, \text{ если } k \in E \backslash E'. \end{cases}$$

$$\text{Тогда } \begin{cases} \prod\limits_{k\in\mu} \delta_k = 1, \forall k \in E'; \\ \prod\limits_{k\in\mu} \delta_k = 0, \exists k_0 \in \mu : k_0 \in E \backslash E'. \end{cases}$$

В этом случае положим

$$\Delta_\mu^{s+1} = \begin{cases} -\infty, \text{ если } \prod\limits_{k\in\mu} \delta_k = 1, \\ 1, \text{ если } \prod\limits_{k\in\mu} \delta_k = 0, \end{cases}$$

и тогда в дальнейшем последовательно будем определять

$$\max_{\mu} \frac{\sum\limits_{(j,i)\in\mu} \sigma_{ji}}{\sum\limits_{(j,i)\in\mu} \alpha_j} \Delta_\mu^{s+1} = \lambda^{s+1}(\mu_{s+1}),$$

$\lambda^{s+1}(\mu_{s+1}) > -\infty$ только для новых путей $\mu_{s+1}$, не совпадающих с $\{\mu_s\}$, $s = 1, 2, ...$ .

3. *Цикл.* Если для непросмотренных дуг $\tilde{\Gamma} \neq \varnothing$, то цикл по $s := s + 1$ и переход к п. 1. Иначе: просмотр построенного множества путей $\{\mu_{s+1}\}$ и определение оптимального пути(-ей) с двойными весами $\mu^{0*} = \operatorname{Arg}\max_s \lambda^s(\mu_s)$, $s = 1,2$. Для первого этапа (5), (6) задачи: $\lambda^0 = \lambda(\mu^{0*})$, а для последующих — $\lambda^r = \lambda(\mu^{r*})$, $r = 1, ..., q$. В алгоритме просмотр дуг для построения множества путей осуществляется от $\Gamma$ до $\varnothing$, что позволяет избежать полного перебора путей, а введение счетчика $\Delta_\mu^s \neq -\infty$ для путей, имеющих хотя бы одну непросмотренную дугу, устраняет зацикливание алгоритма на одном и том же старом пути.

Изложенный алгоритм проходил обсуждение на международной конференции MLSD'2016 [9], где положительно оценен и рекомендован к включению в состав разрабатываемой диалоговой интегрированной системы управления и проектирования исследований и разработок — ДИСУПИР [10].

Трудоемкость алгоритма оценивается следующим образом. Выбор исходной локально-опти-

Таблица 1

**Нахождение первого максимального пути**

| Номер пути | Значения с двойными весами на путях для $\ln u^0$ | Номер пути | Значения с двойными весами на путях для $\ln u^0$ |
|---|---|---|---|
| 1 | **4,0628** | 11 | 2,2212 |
| 2 | 2,3828 | 12 | 2,2322 |
| 3 | 2,7416 | 13 | 2,2010 |
| 4 | 2,3891 | 14 | 2,5093 |
| 5 | 2,9082 | 15 | 2,3978 |
| 6 | 2,1519 | 16 | 2,2619 |
| 7 | 2,3016 | 17 | 2,7578 |
| 8 | 2,1389 | 18 | 3,1007 |
| 9 | 2,8276 | 19 | 2,3034 |
| 10 | 2,5899 | 20 | 2,2053 |

мальной дуги $l_{ji}$ требует $|\Gamma|$ операций, построение пути $\mu$ из начальной вершины в конечную — еще порядка $D|\Gamma|$ операций, где $D$ — максимальная степень вершин графа $G(E, \Gamma)$. Построение всех таких путей для каждой из $|\Gamma|$ дуг повторяется порядка $O[(|\Gamma| + D|\Gamma|)|\Gamma|]$ раз плюс последующий отбор $\lambda$-оптимального пути, потребует порядка $|\Gamma|$ операций. Поэтому трудоемкость составляет $O[(|\Gamma| + D|\Gamma|)|\Gamma| + |\Gamma|] \sim O(|\Gamma|^2)$ операций. А для полного вершинного покрытия потребуется повторить такой алгоритм порядка $|\Gamma|$ раз, тогда всего потребуется $O(|\Gamma|^3)$ операций.

Для наглядности проиллюстрируем (табл. 1) нахождение первого максимального пути с двойными весами на дугах путем просчета всех имеющихся путей и выбора из них максимального.

Таблица 2

**Верхняя граница *ub* для $\ln u^0$**

| № $j$ | Верхняя граница *ub* для $\ln u^0$ | Параметр формы $\alpha_j$ | Параметр масштаба $u_j^0$ |
|---|---|---|---|
| 0 | | 0,3000 | 0,5100 |
| 1 | 4,3944 | 0,2500 | 0,6000 |
| 2 | 4,1650 | 0,2200 | 0,5000 |
| 3 | 0 | 0,2400 | 0,5000 |
| 4 | 0 | 0,2500 | 0,5000 |
| 5 | 0 | 0,2800 | 0,4800 |
| 6 | 4,3565 | 0,2300 | 0,4780 |
| 7 | 0 | 0,2330 | 0,4750 |
| 8 | 0 | 0,2350 | 0,4600 |
| 9 | 0 | 0,2380 | 0,4550 |
| 10 | 0 | 0,2400 | 0,4500 |
| 11 | 0 | 0,3000 | 0,4480 |
| 12 | 0 | 0,3200 | 0,4460 |
| 13 | 0 | 0,3500 | 0,4450 |
| 14 | 0 | 0,3800 | 0,4420 |
| 15 | 0 | 0,4000 | 0,4400 |
| 16 | 0 | 0,4200 | 0,4300 |
| 17 | 0 | 0,4300 | 0,4200 |
| 18 | 0 | 0,4400 | 0,4000 |
| 19 | 0 | 0,4420 | 0,3800 |
| 20 | 0 | 0,4450 | 0,3500 |
| 21 | 0 | 0,4460 | 0,3200 |
| 22 | 0 | 0,4480 | 0,3000 |
| 23 | 0 | 0,4500 | 0,2400 |
| 24 | 0 | 0,4550 | 0,2380 |
| 25 | 0 | 0,4600 | 0,2350 |
| 26 | 0 | 0,4750 | 0,2330 |
| 27 | 0 | 0,4780 | 0,2300 |
| 28 | 0 | 0,4800 | 0,2800 |
| 29 | 0 | 0,5000 | 0,2500 |
| 30 | 4,4742 | 0,2800 | 0,7000 |
| 31 | 0 | 0,5000 | 0,3200 |
| 32 | 4,1759 | 0,3000 | 0,7000 |

Таблица 3

**Численное решение задачи (4) в целом**

| № $j$ | $\ln u_j$ | Ограничения $c$ | Верхняя *ub* | Форма $\alpha_j$ | Параметр масштаба $u_j^0$ |
|---|---|---|---|---|---|
| 0 | 3,12 | 25,000 | 3,1203 | 0,300 | 0,510 |
| 1 | 4,39 | 4,1021 | 4,3944 | 0,250 | 0,600 |
| 2 | 4,16 | 3,7607 | 4,1650 | 0,220 | 0,500 |
| 3 | 3,81 | 3,4393 | 3,8179 | 0,240 | 0,500 |
| 4 | 3,66 | 3,2914 | 3,6652 | 0,250 | 0,500 |
| 5 | **0,29** | 0,0444 | 3,1267 | 0,280 | 0,480 |
| 6 | **0,29** | 0,0444 | 4,3565 | 0,230 | 0,478 |
| 7 | **0,29** | 0,0394 | 3,7124 | 0,233 | 0,475 |
| 8 | **0,29** | 0,0444 | 3,5443 | 0,235 | 0,460 |
| 9 | **0,29** | 0,1276 | 3,4537 | 0,238 | 0,455 |
| 10 | **0,29** | 0,0444 | 3,3789 | 0,240 | 0,450 |
| 11 | **0,29** | 0,0444 | 2,6883 | 0,300 | 0,448 |
| 12 | **0,29** | 0,0444 | 2,5063 | 0,320 | 0,446 |
| 13 | 1,75 | 1,4512 | 2,2850 | 0,350 | 0,445 |
| 14 | **0,29** | 0,0444 | 2,0868 | 0,380 | 0,442 |
| 15 | **0,29** | 0,0444 | 1,9711 | 0,400 | 0,440 |
| 16 | **0,29** | 0,0444 | 1,8225 | 0,420 | 0,430 |
| 17 | 1,24 | 0,9752 | 1,7254 | 0,430 | 0,420 |
| 18 | 1,57 | 1,2756 | 1,5753 | 0,440 | 0,400 |
| 19 | **0,29** | 0,0444 | 1,4522 | 0,442 | 0,380 |
| 20 | 1,25 | 0,9652 | 1,2576 | 0,445 | 0,350 |
| 21 | **0,29** | 0,0307 | 1,0538 | 0,446 | 0,320 |
| 22 | 0,90 | 0,6127 | 0,9051 | 0,448 | 0,300 |
| 23 | 0,40 | 0,1128 | 0,4052 | 0,450 | 0,240 |
| 24 | 0,38 | 0,0899 | 0,3823 | 0,455 | 0,238 |
| 25 | 0,35 | 0,0582 | 0,3506 | 0,460 | 0,235 |
| 26 | 0,32 | 0,0291 | 0,3215 | 0,475 | 0,233 |
| 27 | **0,2924** | 0 | 0,2924 | 0,478 | 0,230 |
| 28 | 0,7010 | 0,4086 | 0,7010 | 0,480 | 0,280 |
| 29 | 0,4463 | 0,1539 | 0,4463 | 0,500 | 0,250 |
| 30 | 4,4742 | 4,1818 | 4,4742 | 0,280 | 0,700 |
| 31 | 0,9400 | 0,6476 | 0,9400 | 0,500 | 0,320 |
| 32 | 4,1759 | 3,8835 | 4,1759 | 0,300 | 0,700 |

Тестовая сеть, состоящая из 32 вершин-работ, взята из известной библиотеки сетевых моделей проектов — http://www.om-db.wi.tum.de/psplib/, с матрицей "столбцы — вершины" × × "строки — пути по вершинам", которая задает характеристическую функцию системы ограничений. Параметры заданы в последних двух столбцах табл. 2 и табл. 3, принято $p_0 = 0{,}97$.

Тогда для (5) имеем $u^0 = 1{,}3396$.

Если решать рассматриваемую задачу (4) в целом как задачу математического программирования готовыми программными средствами, то получим численное решение, представленное в табл. 3.

Для $w_{jk} \in [\varepsilon_0, 1]$, $\varepsilon_0 > 0$ — коэффициента надежности дуги $(j, k)$ (из матрицы путей по дугам сети $G(E, \Gamma)$: $(l_{j,k})$, $(j, k) \in \mu_i$) передачи результата $j$-й работы посредством дуги $(j, k)$ для выполнения $k$-й работы (подобно передаточной функции соединения звеньев в теории автоматического управления) имеем следующую таблицу значений (табл. 4).

Таблица 4

**Коэффициенты надежности дуг**

| $\Gamma^0$ | Дуги | Значение $w_{jk}$ |
|---|---|---|
| 1 | (5,20) | 0,95 |
| 2 | (11,20) | 0,95 |
| 3 | (18,20) | 0,95 |
| 4 | (16,22) | 0,95 |
| 5 | (17,22) | 0,95 |
| 6 | (18,22) | 0,95 |
| 7 | (10,25) | 0,95 |
| 8 | (15,25) | 0,95 |
| 9 | (20,25) | 0,95 |

Для всех остальных дуг сети

$$w_{jk} = 1 \ \forall w_{jk} \in \Gamma^1 = \Gamma \backslash \Gamma^0.$$

## Определение календарного плана при неточных исходных данных

Поскольку значения параметров модели календарного планирования

$$\alpha_j \in (0, 1),\ w_{jk} \in [\varepsilon_0, 1],\ \varepsilon_0 > 0,\ u_j^0 > 0,$$
$$j, k = 1, ..., n,\ (j, k) \in \Gamma \subset G(E, \Gamma)$$

известны с определенной погрешностью, алгоритм распределения ресурсов, а также физический объем работ $v$ [руб·ч] и найденный в результате решения рассмотренной задачи объем невозобновимого ресурса, имеющего стоимостное выражение $u$ [руб], являются приближенными, то временны́е параметры проекта — продолжительности выполнения работ $t_j^* = v_j / u_j^{r*}$, $r = 0, 1, ..., q$; $j = 1, ..., n$, и резервы работ определяются путем обработки чисел, известных с погрешностью. Полученное решение задачи (3), (7) вместе с надежностными ограничениями — вектор

$$u^* = \{u_j^{r*}\},\ u_j^{r*} > 0,\ j = 1, \ldots, n,$$
$$r = 0, 1, \ldots, q,\ \{r\} \subset \{j\}, q \ll n$$

таково, что (опуская индекс $r$)

$$u_j^* = \{u_j^*(\delta_0) | u_j^*(\delta_0) \in [[u_j^*] + \Delta u_j^*]\}. \qquad (8)$$

Обозначая неточно заданные величины времени как $t(\delta)$, с помощью действий над приближенными числами [11] можно получить приближенные параметры оптимального календарного плана. Для этого вычисляем за один проход в прямом и обратном направлениях детерминированное расписание проекта на основе полученных оптимальных (с приведенной погрешностью) продолжительностей работ проекта $t_j^*(\delta_0)$, стартуя с начальных работ проекта. Метод критического пути с величинами, заданными неточно (8), с помощью действий над приближенными числами [11] позволяет рассчитать также и поздние сроки выполнения работ проекта обратным проходом по сети, начиная от установленной даты завершения проекта (вычисленной путем прямого прохода по сети), а также позволяет вычислить временны́е резервы работ (англ. slacks) [11]: общий (полный) резерв, свободный (локальный) резерв, безопасный резерв, независимый резерв.

## Заключение

В статье рассматривается детерминированная сетевая модель проекта, в которой для показателя надежности проекта в целом справедлива его оценка снизу в виде слабейшего звена — наихудшей технологической цепочки из начальной вершины проекта в конечную. Для показателя надежности работы-вершины проекта используется степенная вогнутая модель в зависимости от затраченного невозобновимого (складируемого) ресурса. Особенностью модели являются взвешенные дуги сети, которые задают коэффициент надежности дуги $(j, k)$, передачи результата $j$-й работы посредством дуги $(j, k)$ для выполнения $k$-й работы.

В этих условиях рассматривается задача календарного планирования проектов на основе последовательно применяемого минимаксного

критерия по типу лексикографического упорядочивания критериев в задачах многокритериальной оптимизации. Такое лексикографическое разложение, по сути, представляет декомпозиционную процедуру и подходит для планирования проектов большой размерности. Для решения задачи равномерного распределения невозобновимого ресурса разработан жадный алгоритм отыскания максимального пути с двойными весами на дугах ациклического орграфа, с квадратичной по числу дуг вычислительной трудоемкостью, который позволяет получить *ресурсный критический путь* с максимальным расходом невозобновимого ресурса. После этого определяется календарный план проекта, включающий его временной критический путь при неточных данных. Рассмотренный алгоритм был эффективно использован при решении ряда практических задач по управлению проектами. Предложенная модель и эффективный метод ее решения дают существенный вклад в анализ малоисследованного параметра Risk of Performance в интересах планирования крупномасштабных инновационных проектов. Практические расчеты для рассматриваемой выпуклой задачи (4) можно выполнять с помощью готовых программных средств, а полученное жадное решение задачи-следствия (5), (6) имеет теоретическое значение в теории графов. Представленный в статье материал создает теоретические основы для дальнейшего развития функционального наполнения диалоговой интегрированной системы управления и проектирования исследования и разработок [10].

**Список литературы**


1. **The Owner's** Role in Project Risk Management. Washington, D. C.: The National Academies Press, 2005.

2. **Elkjaer M.** Stochastic Budget Simulation // Intern. J. Project Management. 2000. Vol. 18, N. 2. P. 139—147.

3. **Rabbani M., Tavakkoli Moghaddam R., Jolai F. and Ghorbani H. R.** A Comprehensive Model for R&D Project Portfolio Selection with Zero-One Linear Goal-Programming // IJE Transactions A. Basics. 2006. Vol. 19, N. 1. P. 55—66.

4. **Цвиркун А. Д., Акинфиев В. К., Коновалов Е. Н.** Моделирование и управление инновационными программами в крупномасштабных технических системах. М.: Препринт ИПУ, 1991.

5. **Топка В. В.** Минимизация времени и стоимости с учетом показателя надежности в дизъюнктивной модели проекта // А и Т. 2012. № 7. С. 78—88.

6. **Бурков В. Н., Заложнев А. Ю., Новиков Д. А.** Теория графов в управлении организационными системами. М.: Синтег, 2001. 124 с.

7. **Кристофидес И.** Теория графов. Алгоритмический подход. М.: Мир, 1978. 432 с.

8. **Карзанов А. В.** О минимальных по среднему весу разрезах и циклах ориентированного графа // Качественные и приближенные методы исследования операторных уравнений. Ярославль: ЯрГУ, 1985. С. 72—83.

9. **Топка В. В., Цвиркун А. Д., Юркевич Е. В.** Жадный алгоритм в календарном планировании инновационных проектов // Тр. 9-й Междунар. конф. "Управление развитием крупномасштабных систем" (MLSD' 2016, Москва). М.: ИПУ РАН, 2016. Т. 1. С. 216—227.

10. **Топка В. В.** Диалоговая интегрированная система управления и проектирования исследований и разработок (ДИСУПИР 3.2). Свидетельство о государственной регистрации программы для ЭВМ № 2015617277. Дата государственной регистрации в Реестре программ для ЭВМ 06 июля 2015 г.

11. **Топка В. В.** Лексикографическое решение двухкритериальной задачи планирования проекта при ограничении на показатель его надежности // Изв. РАН. ТиСУ. 2014. № 6. С. 105—123.



**A. D. Tsvirkun,** Head the Department, Chief Researcher Scientist, e-mail: tsvirkun@ipu.ru,
**V. V. Topka,** Senior Researcher Scientist, e-mail: topka3@mail.ru,
Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russian Federation


# Even Allocation of Non-Renewable Resource in Large-Scale Innovative Projects


*The problem of even allocation according to the minimax criterion of a non-renewable resource of an innovative project's activities is considered. A lexicographic method of ordering these minimax criteria is used. To solve the problem — one criterion — developed a greedy algorithm for finding the maximum path on an acyclic digraph with double weights on arcs. The obtained solution allows to build a project resources plan and schedule with four types of temporary slacks with inaccurate initial data. The proposed approach allows performing an analysis of the project Risk of Performance parameter, for which the apparatus for its analysis has not been developed. These include the risks that the project when complete fails to perform as intended or fails to meet the mission or business requirements that generated the justification for the project. Performance risks are based on the probability distribution function of a favorable / unfavorable outcome. For these: the probability of a project's success is the probability that a failure will not occur during the execution of this project, and the probability of failure-free execution is taken as a measure of reliability of performance. By virtue of the inequality of the problem, the consequence of the initial statement, taking into account the network specificity, we will seek solution of that problem-consequence in the form of a maximal path with double weights or an λ — optimal path. The complexity of the algorithm outlined is evaluated as* $O[(|\Gamma| + D|\Gamma|)|\Gamma| + |\Gamma|] \sim O(|\Gamma|^2)$ *operations. And for a complete vertex coverage, you need to repeat this algorithm* $|\Gamma|$ *more than once, then all* $O(|\Gamma|^3)$ *operations will be required. The general problem is solved by successive optimization of the remaining subgraph by the minimax criterion at each*

*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, $x_н \in \mathbf{X}$; $\varphi$: $\mathbf{K} \times \mathbf{X} \to \mathbf{X}$ — 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 $\mu$ 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).

нообразна, как и область самого машинного обучения.

Целью данной статьи является систематизация информации о различных методах оптимизации, которые нашли широкое применение в машинном обучении, и о новых менее известных подходах. Обзор включает краткое описание метода обратного распространения ошибки — одного из методов обучения нейронных сетей, метода оптимизации на основе градиентного спуска и возникающих при их использовании проблем сходимости. В последующих разделах статьи рассмотрены методы: градиентный спуск с импульсом, быстрый градиентный метод Нестерова, AdaGrad, RMSprop, AdaDelta, Adam, AdaMax, Nadam, AMSGrad, ND-Adam, NosAdam, Padam и Yogi.

## 1. Обратное распространение ошибки

Чтобы определить место и роль математической оптимизации в машинном обучении, сначала необходимо рассмотреть один из методов обучения нейронных сетей: метод обратного распространения ошибки (англ. backpropagation) — градиентный метод обновления весов многослойных нейронных сетей прямого распространения (многослойных персептронов) [1—3], представляющий собой итеративный процесс минимизации ошибки работы многослойного персептрона в целях получения выхода, близкого к желаемому.

При обучении нейронной сети методом обратного распространения ошибки предполагается, что для каждого входного вектора задан желаемый выходной вектор. Данные векторы образуют обучающую пару, а для обучения используется множество пар. Метод обратного распространения ошибки включает в себя прямой и обратный проходы по всем слоям сети. Прямой проход предполагает подачу на входной слой нейронной сети входного вектора сигналов и дальнейшее его распространение от слоя к слою. В результате формируется вектор выходных сигналов, который вычитается из желаемого результата. Полученная ошибка распространяется в направлении, обратном прямому распространению сигналов, т. е. от выходного слоя к входному, при этом весовые коэффициенты нейронов (параметры сети) корректируются. Для возможности применения метода обратного распространения ошибки передаточная (активационная) функция нейронов должна быть дифференцируема.

Следует отметить, что при использовании метода обратного распространения ошибки процесс обучения может требовать больших временных затрат, также процесс обучения может фактически остановиться. Так как в алгоритме обучения применяется разновидность градиентного спуска, то одной из проблем является наличие точек, в которых процесс обучения останавливается, хотя рядом может находиться оптимальное решение. Такой точкой может быть локальный минимум или седловая точка (стационарная для заданной функции, однако не являющаяся экстремумом, значение функции является максимальным в одном измерении и минимальным в другом).

Кроме того, необходимо уделять внимание длине шага обучения. Доказано, что метод обратного распространения ошибки обладает сходимостью при условии, что коррекция весовых коэффициентов бесконечно мала, однако при малом шаге обучения скорость сходимости будет также мала, а время обучения будет стремиться к бесконечности. При очень большой длине шага процесс обучения оказывается неустойчивым, сходимость может не достигаться, возникает паралич сети.

## 2. Метод градиентного спуска

Рассмотрев основные принципы метода обратного распространения ошибки, можно сделать вывод, что каждый параметр сети вносит свой вклад в ошибку. Процесс обратного распространения ошибки напрямую не корректирует весовые коэффициенты, а только перемещает информацию с выхода сети. За изменение весов отвечают другие методы, решающие задачу оптимизации. Базовым методом является градиентный спуск. Применяются разные подходы к реализации градиентного спуска для коррекции параметров сети:

1) стохастический градиентный спуск: параметры сети меняются после расчета ошибки для каждой обучающей пары;

2) пакетный (англ. batch) градиентный спуск: параметры сети корректируются после прохождения всех обучающих пар;

3) градиентный спуск с использованием мини-пакетов: множество обучающих пар разбивается на группы.

Для корректировки параметров используется одна из разновидностей метода градиентного спуска. Задачу обычного градиентного спуска можно сформулировать в следующем виде [4, 5]:

$$x_{i+1} = x_i - \lambda_i \nabla F(x_i), \quad (1)$$

где $i$ — номер итерации обучения; $x$ — веса (параметры) сети; $F(x)$ — целевая функция или функция потерь; $\lambda$ — коэффициент, влияющий на значение шага обучения. Значение $\lambda$ может быть задано константой (обучение с постоянным шагом) или может меняться на каждой итерации (например, в методе наискорейшего спуска [4]).

Представленные в явном виде формулы для обновления параметров современных многослойных сетей могут выглядеть крайне сложно, так как каждый нейрон связан с множеством других нейронов и зависит от них. Возникает ряд проблем:

1) сложная и неоднородная поверхность функции потерь, которая может зависеть от миллионов параметров. Имеется большое число локальных минимумов и седловых точек;

2) зависимость от длины шага обучения (долгое обучение, зависание в локальных минимумах или седловых точках либо постоянная неустойчивость);

3) проблема разреженных данных. Ожидаемые значения могут отсутствовать в наборе, а значимые признаки могут встречаться редко, но учет всех редких признаков может привести к переобучению.

В теории оптимизации существуют методы второго порядка (Ньютона [6] и др.), способные найти оптимальное решение при сложной поверхности функции потерь, однако число параметров значительно увеличивает вычислительную сложность. Чтобы воспользоваться методом второго порядка, необходимо для функции $F(x)$ вычислить матрицу Гессе и обратную ей (для метода Ньютона).Отдельно сто́ит выделить квазиньютоновские методы, которые основаны на приближенных выражениях для матрицы Гессе (например, метод Бройдена—Флетчера—Гольдфарба—Шанно (BFGS) [7]).

В данной статье будут рассмотрены различные вариации метода градиентного спуска, которые позволяют справиться с названными проблемами без значительного увеличения вычислительной сложности и не прибегать к использованию вторых производных.

### 3. Импульс и быстрый градиентный метод Нестерова

Быстрый градиентный метод Нестерова (англ. Nesterov Accelerated Gradient, NAG) является одним из методов, которые ускоряют работу градиентного спуска в соответствующем направлении и уменьшают колебания в областях с локальным минимумом [8].

В подобных методах (Momentum [9]) вводится понятие импульса. Идея накопления импульса проста: если некоторое время происходит движение в определенном направлении, то в будущем некоторое время также следует двигаться в этом направлении. Для реализации данного подхода можно хранить $N$ последних изменений каждого параметра и на каждой итерации считать среднее. Для экономии памяти используется экспоненциальная фильтрация:

$$E[\nabla F(x)]_{i+1} = \alpha E[\nabla F(x)]_i + (1-\alpha)\nabla F(x_{i+1}), \quad (2)$$

где $0 \leqslant \alpha \leqslant 1$ — коэффициент сохранения экспоненциального скользящего среднего $E$ последовательности величин (градиентов). Значение коэффициента $\alpha$ обычно берется близким к 1.

Задача оптимизации с учетом импульса (классический подход):

$$x_{i+1} = x_i + \mu_i v_i - \lambda_i \nabla F(x_i), \quad (3)$$

где $v$ вычисляется по следующей формуле:

$$v_{i+1} = \mu_i v_i - \lambda_i \nabla F(x_i), \quad (4)$$

где $0 \leqslant \mu \leqslant 1$ — коэффициент сохранения импульса (эквивалент $\alpha$). При этом коэффициент $(1 - \mu)$ может быть учтен в значении $\lambda$.

В методе Нестерова также используется идея экстраполяции (предсказания), и градиент вычисляется в другой точке:

$$x_i = y_i - \lambda_i \nabla F(y_i), \quad (5)$$

$$y_{i+1} = x_i + \mu_i (x_i - x_{i-1}). \quad (6)$$

Важно отметить, что минимизация выполняется относительно параметров, обозначенных $y$, но искомым вектором параметров нейронной сети также является $x$.

Коэффициент $\mu$ может быть константой, близкой к 1. Также встречаются варианты, использующие обновление $\mu$:

$$\mu_i = (a_i - 1)/a_{i+1}, \quad a_{i+1} = (1 + \sqrt{4a_i^2 + 1})/2; \quad (7)$$

$$\mu_i = i/(i+3); \quad (8)$$

$$\mu_i = 1 - 3/(5+i). \quad (9)$$

### 4. Метод AdaGrad (Adaptive Gradient)

Метод AdaGrad предполагает подстройку шага обучения для каждого параметра сети [10]. Иначе говоря, скорость обучения адаптируется, выполняя значительные обновления для редких признаков и малые обновления для часто встречающихся. Считается сравнительно простым методом оптимизации, хорошо подходит для обучения на разреженных данных.

Обозначим параметр сети $x_{i,k}$, где $k$ — номер параметра. Так как каждый параметр обновляется отдельно, то

$$x_{i+1,k} = x_{i,k} - \lambda \nabla F(x_{i,k}). \qquad (10)$$

Рассматриваемый метод использует предобусловливание. Вводится диагональная матрица предобусловливания, основанная на внешнем произведении векторов:

$$G_i = \operatorname{diag}\left( \sum_{j=1}^{i} \nabla F(x_j) \cdot \nabla F(x_j)^{\mathrm{T}} \right), \qquad (11)$$

где каждый диагональный $kk$-й элемент является суммой квадратов градиентов.

Запишем правило обновления $k$-го параметра:

$$x_{i+1,k} = x_{i,k} - \frac{\lambda}{\sqrt{G_{i,kk}}} \nabla F(x_{i,k}). \qquad (12)$$

На практике, чтобы избежать деления на ноль, в формулу добавляют коэффициент $\varepsilon$ (равный, например, $10^{-8}$):

$$x_{i+1,k} = x_{i,k} - \frac{\lambda}{\sqrt{G_{i,kk} + \varepsilon}} \nabla F(x_{i,k}). \qquad (13)$$

Учитывая, что введенная матрица содержит значения для всех параметров $x_i$, можно представить $G_i$ в виде вектора, и общее правило обновления параметров сети записать с использованием поэлементного произведения ($\odot$) вектора скорректированных коэффициентов и вектора градиентов:

$$x_{i+1} = x_i - \frac{\lambda}{\sqrt{G_i + \varepsilon}} \odot \nabla F(x_i). \qquad (14)$$

*Замечание:* в ходе дальнейшего изложения при умножении векторов будет иметься в виду поэлементное произведение векторов.

Преимуществом AdaGrad является автоматическая подстройка скорости обучения, нет необходимости в точном подборе $\lambda$. Недостаток заключается в увеличении суммы квадратов градиентов в процессе обучения. В итоге скорость обучения становится бесконечно малой.

### 5. Метод RMSProp (Root Mean Square Propogation)

Метод RMSProp был предложен Джеффом Хинтоном в его онлайн-лекции [11]. Работа метода направлена на устранение стремительного и монотонного уменьшения скорости обучения с использованием AdaGrad.

Для описания работы метода воспользуемся формулой экспоненциального фильтра (2), заменив градиент $g = \nabla F(x)$ квадратом градиента $g^2$. Знаменатель в формуле (14) меняется на вектор

$$RMS[g]_i = \sqrt{E[g^2]_i + \varepsilon}. \qquad (15)$$

При этом автор фиксировал значение коэффициента сохранения экспоненциального скользящего среднего $\alpha = 0{,}9$.

### 6. Метод AdaDelta

AdaDelta — модификация метода AdaGrad, которая также позволяет исправить недостаток этого метода [12]. RMSprop и AdaDelta были разработаны независимо друг от друга в одно время, что было связано с необходимостью радикально уменьшить время обучения при использовании AdaGrad.

Вместо накопления квадратов градиента за все время (по текущую $i$-ю итерацию) предложено ограничивать историю окном фиксированного размера $w$. Таким образом, за большое число итераций скорость обучения не замедлится.

AdaDelta отличается от RMSprop числителем в формуле (14). Запишем следующие выражения:

$$\Delta x_i = -\frac{\lambda}{RMS[\nabla F(x)]_i} \nabla F(x_i); \qquad (16)$$

$$E[\Delta x^2]_i = \alpha E[\Delta x^2]_{i-1} + (1-\alpha)\Delta x_i^2; \qquad (17)$$

$$RMS[\Delta x]_i = \sqrt{E[\Delta x^2]_i + \varepsilon}. \qquad (18)$$

Тогда правило обновления параметров сети (14) принимает вид

$$x_{i+1} = x_i - \frac{RMS[\Delta x]_{i-1}}{RMS[\nabla F(x)]_i} \odot \nabla F(x_i). \qquad (19)$$

Для AdaGrad, RMSProp и AdaDelta не нужно очень точно подбирать темп обучения $\lambda$. Для AdaDelta этот коэффициент не надо задавать ввиду его отсутствия в правиле обновления параметров сети, а для RMSProp рекомендуется $\lambda = 10^{-3}$. Коэффициент $\alpha$ обычно советуют оставить равным 0,9 или 0,95.

### 7. Метод Adam (Adaptive moment estimation)

Adam также является методом оптимизации, который адаптивно подбирает темп обучения для каждого параметра [13]. Как отмечают авторы, он сочетает в себе преимущества методов AdaGrad и RMSProp. Скорости обновления параметров адаптируются на основе экспоненциального скользящего среднего значений градиентов (первый момент градиента) $g_i = \nabla F(x_i)$:

$$m_i = \alpha_1 m_{i-1} + (1 - \alpha_1) g_i \tag{20}$$

и экспоненциального скользящего среднего квадратов градиентов (второй момент градиента):

$$v_i = \alpha_2 v_{i-1} + (1 - \alpha_2) g_i^2. \tag{21}$$

Авторы Adam отмечают, что оценки моментов могут быть смещены к нулю, так как изначально инициализируются нулями. Особенно это проявляется на начальных итерациях обучения и при гиперпараметрах $\alpha_1$ и $\alpha_2$, близких к 1 (рекомендованные авторами значения: $\alpha_1 = 0{,}9$, $\alpha_2 = 0{,}999$). Для решения проблемы вычисляются скорректированные оценки первого и второго моментов:

$$\widehat{m}_i = m_i / (1 - \alpha_1^i), \quad \hat{v}_i = v_i / (1 - \alpha_2^i). \tag{22}$$

Правило обновления параметров сети имеет следующий вид:

$$x_i = x_{i-1} - \lambda \widehat{m}_i / (\sqrt{\hat{v}_i} + \varepsilon), \tag{23}$$

где гиперпараметры $\lambda = 10^{-3}$, $\varepsilon = 10^{-8}$ (предложены авторами Adam).

Авторами эмпирически доказано, что данный метод устойчив и хорошо подходит для широкого круга задач оптимизации в машинном обучении.

### 8. Метод AdaMax

AdaMax является модификацией метода Adam, предложенной в той же статье [13]. Отмечается, что правило обновления параметров в Adam основано на $L2$-норме текущего и прошлых значений градиентов и предлагается сформулировать правило на основе $Lp$-нормы. Формула для $v_i$ принимает вид:

$$\begin{aligned} v_i &= \alpha_2^p v_{i-1} + (1 - \alpha_2^p) |g_i|^p = \\ &= (1 - \alpha_2^p) \sum_{j=1}^{i} \alpha_2^{p(i-j)} |g_i|^p . \end{aligned} \tag{24}$$

Теперь пусть $p$ стремится к бесконечности:

$$\begin{aligned} u_i &= \lim_{p \to \infty} (v_i)^{1/p} = \\ &= \max(\alpha_2^{i-1} |g_1|, \alpha_2^{i-2} |g_2|, \ldots, \alpha_2 |g_{i-1}|, |g_i|) = \\ &= \max(\alpha_2 u_{i-1}, |g_i|). \end{aligned} \tag{25}$$

Вычисление второго момента $v_i$ заменяется вычислением экспоненциально взвешенной бесконечной нормы $u_i$. Правило обновления параметров сети принимает следующий вид:

$$x_i = x_{i-1} - \lambda \widehat{m}_i / u_i, \tag{26}$$

где $\lambda = 2 \cdot 10^{-3}$.

Авторы утверждают, что хотя для больших $p$ метод и становится нестабилен, но для значения $p$, стремящегося к бесконечности, метод дает положительные результаты.

### 9. Метод Nadam (Nesterov-accelerated Adam)

Метод Adam использует идеи RMSprop для адаптации скорости обучения и в то же время обладает импульсной компонентой. Классический подход на основе импульса уступает быстрому градиентному методу Нестерова (NAG). Исходя из этого был предложен метод Nadam, объединяющий в себе Adam и NAG [14]. Чтобы воспользоваться идеей NAG в методе Adam, необходимо изменить в нем импульсную компоненту $m_i$.

Во-первых, модифицируется NAG так, чтобы импульсная компонента напрямую участвовала в обновлении параметров сети. Слагаемые на основе импульса и градиента зависят от текущего градиента:

$$m_i = \alpha_{1,i} m_{i-1} + \lambda_i g_i; \tag{27}$$

$$x_i = x_{i-1} - (\alpha_{1,i+1} m_i + \lambda_i g_i). \tag{28}$$

Во-вторых, модифицируется метод Adam, и скорректированная оценка первого момента вычисляется следующим образом:

$$\widehat{m}_i = \frac{\alpha_{1,i+1} m_i}{1 - \prod_{j=1}^{i+1} \alpha_{1,j}} + \frac{(1-\alpha_{1,i}) g_i}{1 - \prod_{j=1}^{i} \alpha_{1,j}}. \qquad (29)$$

Заметим, что были учтены возможные изменения гиперпараметров $\alpha_1$ и $\lambda$ в зависимости от итерации, потому что их постепенное увеличение или уменьшение часто помогает работе метода. Однако в экспериментальных исследованиях использовались фиксированные значения: $\alpha_1 = 0{,}975$, $\lambda = 2 \cdot 10^{-3}$. Исходя из этого перепишем (29):

$$\widehat{m}_i = \frac{\alpha_1 m_i}{1 - \alpha_1^{i+1}} + \frac{(1-\alpha_1) g_i}{1 - \alpha_1^{i}}. \qquad (30)$$

Вычисления $v_i$ и $\hat{v}_i$ не меняются и соответствуют формулам (21) и (22), $\alpha_2 = 0{,}999$. Правило обновления параметров сети соответствует виду (23).

Результаты исследований, представленные в работе [14], свидетельствуют о том, что Nadam повышает скорость сходимости и качество обучаемых сетей.

### 10. Методы AMSGrad, ND-Adam и NosAdam

Рассмотренные адаптивные методы оптимизации стали популярными при обучении нейронных сетей, но на практике было замечено, что в некоторых случаях, например, при обучении глубоких нейронных сетей, в задачах машинного перевода [15] и распознавания объектов [16] они не могут сходиться к оптимальному решению и проигрывают стохастическому градиентному спуску с импульсом.

В качестве причины плохой сходимости и ухудшения обобщающей способности нейронных сетей было названо экспоненциальное скользящее среднее квадрата градиента. С одной стороны, экспоненциальная фильтрация позволяла избежать падения скорости обучения, но с другой, такая краткосрочная память о градиентах оказалась проблемой, поскольку были приведены примеры, когда именно последние несколько значений градиента значительно ухудшали сходимость.

Для преодоления разрыва между стохастическим градиентным спуском и Adam с точки зрения обобщающей способности был предложен метод ND-Adam (Normalized Direction-preserving Adam) [17]. Во-первых, скорость обучения адаптируется для каждого вектора параметров сети, а не для каждого параметра, чтобы сохранить направление градиента. Во-вторых, каждый вектор нормализуется с помощью оптимизации параметров на основе сферы. Эмпирически показано, что производительность метода действительно может быть близка к стохастическому градиентному спуску с импульсом.

Авторы AMSGrad [18] предложили наделить метод долгосрочной памятью, чтобы повысить не только сходимость, но и производительность. Подобной идее следовали и авторы метода NosAdam (Nostalgic Adam) [19]. В NosAdam градиентам с прошлых итераций устанавливаются веса больше, чем для более новых градиентов. Ускорение сходимости объясняется тем, что оценка второго момента градиента $v_i$ "забывается" медленнее, чем в Adam.

Наибольшее распространение получил метод AMSGrad, основное отличие которого от метода Adam в том, что для нормализации скользящего среднего значения градиента используется максимальное значение квадрата градиента $v_i$, а не его скользящее среднее (либо скорректированная оценка).

Значения $m_i$ и $v_i$, как и в Adam, вычисляются согласно (20) и (21), но скорректированная оценка $v_i$ вычисляется иначе:

$$\hat{v}_i = \max(\hat{v}_{i-1}, v_i). \qquad (31)$$

Запишем правило обновления параметров сети для AMSGrad:

$$x_{i+1} = x_i - \lambda_i m_i / \sqrt{\hat{v}_i}. \qquad (32)$$

В результате, даже если $v_{i-1,k} > g_{i,k}^2 > 0$, то скорость обучения не возрастает значительно, как это происходит у Adam (на практике это отрицательно сказывалось на работе метода). Эксперименты показали лучшую производительность AMSGrad на наборах CIFAR-10 и MNIST. В свою очередь, авторы NosAdam эмпирически показали превосходство своего метода для некоторых задач машинного обучения.

### 11. Методы Padam и Yogi

Одними из последних методов, предназначенных для улучшения работы адаптивных методов оптимизации при обучении глубоких нейронных сетей, являются методы Padam (Partially adaptive momentum estimation method) [20] и Yogi [21].

Авторы Padam рассматривают проблему ухудшения обобщающей способности при использовании адаптивных методов, причем речь идет не только о методе Adam, но и о AMSGrad. Фактически Padam модифицирует AMSGrad добавлением дополнительного гиперпараметра $\rho \in (0; 1/2]$ в правило обновления параметров сети (32):

$$x_{i+1} = x_i - \lambda_i m_i / \hat{v}_i^{\rho}. \quad (33)$$

Авторы называют $\rho$ частично адаптивным гиперпараметром, причем при $\rho = 1/2$ метод повторяет AMSGrad, а при $\rho \to 0$ приближается к стохастическому градиентному спуску с импульсом. Рекомендованное значение: 1/8.

Метод Yogi является модификацией Adam. Авторы Yogi, во-первых, говорят о достижении лучшей сходимости при увеличении размера мини-пакетов с обучающими парами. Во-вторых, предлагают модификацию вычисления второго момента градиента:

$$v_i = v_{i-1} - (1 - \alpha_2)\text{sign}(v_{i-1} - g_i^2)g_i^2. \quad (34)$$

Оцениваемые моменты $m_i$ и $v_i$ не корректируются. В остальном метод повторяет оригинальный Adam:

$$x_{i+1} = x_i - \lambda_i m_i / (\sqrt{v_i} + \varepsilon). \quad (35)$$

Некоторое улучшение достигнуто методом Padam при обучении на наборах CIFAR-10 и CIFAR-100 моделей сетей ResNet и VGGNet. Для метода Yogi предоставлены обширные исследования для многих современных нейронных сетей (ResNet, DenseNet и др.). Показано, что Yogi достигает схожих с Adam или лучших результатов без особой настройки гиперпараметров.

## Заключение

Были рассмотрены нашедшие применение в машинном обучении методы оптимизации первого порядка, такие как градиентный спуск с импульсом, быстрый градиентный метод Нестерова, AdaGrad, RMSprop, AdaDelta, Adam и его модификации, AMSGrad и др. Выбор наиболее подходящего метода нередко зависит от решаемой задачи и обучающего набора данных.

Исследования в данной области продолжаются, выявляются проблемы процесса обучения нейронных сетей. На основе анализа причин возникновения проблем предлагаются модификации методов оптимизации, а также другие методы первого и второго порядков. Можно отметить, что градиентный спуск, импульс и метод Нестерова являются базовыми для применяемых в обучении сетей методов AdaGrad, Adam и др., при этом подстройка скорости обучения на каждой итерации выполняется для каждого параметра отдельно. В более поздних работах отмечается ухудшение сходимости и обобщающей способности, связанное с использованием экспоненциального скользящего среднего (краткосрочная память о градиентах). Такие методы, как AMSGrad, NosAdam, Padam, направлены на решение данной проблемы и используют преимущества как Adam, так и стохастического градиентного спуска.

### Список литературы


1. **Галушкин А. И.** Нейронные сети: основы теории. М.: Горячая линия-Телеком, 2010. 480 с.
2. **Werbos P. J.** Beyond regression: New tools for prediction and analysis in the behavioral sciences: Ph. D. thesis. Cambridge: Harvard University, 1974. 453 p.
3. **Rumelhart D. E., Hinton G. E., Williams R. J.** Learning Internal Representations by Error Propagation // Parallel Distributed Processing. 1986. Vol. 1. P. 318—362.
4. **Коршунов Ю. М.** Математические основы кибернетики: Учеб. пособие для вузов. 3-е изд. М.: Энергоатомиздат, 1987. 496 с.
5. **Глебов Н. И., Кочетов Ю. А., Плясунов А. В.** Методы оптимизации: Учеб. пособие. Новосибирск: Изд-во НГУ, 2000. 105 с.
6. **Mordecai A.** Nonlinear Programming: Analysis and Methods. New York: Dover Publications, 2003. 544 p.
7. **Fletcher R.** Practical methods of optimization. 2nd edition. New York: John Wiley & Sons, 1987. 456 p.
8. **Нестеров Ю. Е.** Алгоритмическая выпуклая оптимизация: дис. на соискание ученой степени доктора физ.-мат. наук. М.: МФТИ, 2013. 367 с.
9. **Qian N.** On the momentum term in gradient descent learning algorithms // Neural Networks: The Official Journal of the International Neural Network Society. 1999. Vol. 12 (1). P. 145—151.
10. **Duchi J., Hazan E., Singer Y.** Adaptive Subgradient Methods for Online Learning and Stochastic Optimization // Journal of Machine Learning Research. 2011. Vol. 12. P. 2121—2159.
11. **Hinton G. E.** Neural Networks for Machine Learning, lecture 6e // Department of Computer Science at the University of Toronto, 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (дата обращения: 15.04.19).
12. **Zeiler M. D.** ADADELTA: An Adaptive Learning Rate Method // arXiv:1212.5701. 2012. P. 1—6.
13. **Kingma D. P., Ba J. L.** Adam: A Method for Stochastic Optimization // Proc. International Conf. on Learning Representations. San Diego, 2015. P. 1—15.
14. **Dozat T.** Incorporating Nesterov Momentum into Adam // Proc. International Conf. on Learning Representations. San Juan, 2016. P. 1—4.
15. **Johnson M., Schuster M., Le Q. V., Krikun M. et al.** Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation // arXiv:1611.04558. 2016. P. 1—17.

16. **Huang G., Liu Z., Weinberger K. Q., Van der Maaten L.** Densely Connected Convolutional Networks // Proc. IEEE Conf. on Computer Vision and Pattern Recognition. Honolulu, 2017. P. 2261—2269.

17. **Zhang Z., Ma L., Li Z., Wu C.** Normalized Direction-preserving Adam // arXiv:1709.04546. 2017. P. 1—12.

18. **Reddi S. J., Kale S., Kumar S.** On the Convergence of Adam and Beyond // Proc. International Conf. on Learning Representations. Vancouver, 2018. P. 1—23.

19. **Huang H., Wang C., Dong B.** Nostalgic Adam: Weighing more of the past gradients when designing the adaptive learning rate // arXiv:1805.07557. 2018. P. 1—12.

20. **Chen J., Gu Q.** Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks // arXiv:1806.06763. 2018. P. 1—18.

21. **Zaheer M., Reddi S. J., Sachan D. et al.** Adaptive Methods for Nonconvex Optimization // Proc. 32nd Conf. on Neural Information Processing Systems. Montreal, 2018. P. 1—11.

**M. D. Ershov,** Ph. D. Student, e-mail: ershov.m.d@rsreu.ru,
Ryazan State Radio Engineering University, Ryazan, 390005, Russian Federation

# First-Order Optimization Methods in Machine Learning

*The problems arising in the training of multilayer neural networks of direct distribution due to the disadvantages of the gradient descent method are considered. A review of first-order optimization methods which are widely used in machine learning and less well-known methods is performed. The review includes a brief description of one of training methods for neural networks: the backpropagation method (also known as the backward propagation of errors). A separate section is devoted to the gradient descent optimization method and convergence problems arising from the use of the backpropagation method with gradient descent. The review considers the following first-order optimization methods with adaptive learning rate: gradient descent with momentum, Nesterov accelerated gradient method (NAG), AdaGrad, RMSprop, AdaDelta, Adam, AdaMax, Nadam, AMSGrad, ND-Adam, NosAdam, Padam, and Yogi. The features of each method and the problems of their use in practice are described. It can be noted that gradient descent, momentum and NAG are basis for the AdaGrad, Adam and other methods used in machine learning. In addition, the learning rate adjustment is performed for each parameter separately at each iteration of the neural network training. In later works the deterioration of convergence and generalization ability is described, which is associated with the use of the exponential moving average (a short-term memory of gradients). Such methods as AMSGrad, NosAdam, Padam are aimed at solving this problem and take advantage of both Adam and the stochastic gradient descent.*

***Keywords:*** *neural networks, machine learning, deep learning, optimization, gradient descent, adaptive learning rate*

**Acknowledgements:** The studies were carried out with financial support from the scholarship of the President of the Russian Federation to young scientists and graduate students (SP-2578.2018.5).

DOI: 10.17587/it.25.662-669

**References**

1. **Galushkin A. I.** Neural networks: theoretical basis, Moscow, Goryachaya liniya-Telekom, 2010, 480 p. (in Russian).

2. **Werbos P. J.** Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph. D. thesis, Cambridge, Harvard University, 1974, 453 p.

3. **Rumelhart D. E., Hinton G. E., Williams R. J.** Learning Internal Representations by Error Propagation, *Parallel Distributed Processing*, 1986, vol. 1, pp. 318—362.

4. **Korshunov Yu. M.** Mathematical basics of cybernetics. Textbook for universities, 3rd edition, Moscow, Energoatomizdat, 1987, 496 p. (in Russian).

5. **Glebov N. I., Kochetov Yu. A., Plyasunov A. V.** Optimization methods. Study guide, Novosibirsk, Publishing House of NSU, 2000, 105 p. (in Russian).

6. **Mordecai A.** Nonlinear Programming: Analysis and Methods, New York, Dover Publications, 2003, 544 p.

7. **Fletcher R.** Practical methods of optimization6 2nd edition, New York, John Wiley & Sons, 1987, 456 p.

8. **Nesterov Yu. E.** Algorithms for convex optimization. Doctor of phys.-mat. sciences thesis, Moscow, MIPT, 2013, 367 p. (in Russian).

9. **Qian N.** On the momentum term in gradient descent learning algorithms, *Neural Networks: The Official Journal of the International Neural Network Society*, 1999, vol. 12, no. 1, pp. 145—151.

10. **Duchi J., Hazan E., Singer Y.** Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2121—2159.

11. **Hinton G. E.** Neural Networks for Machine Learning, lecture 6e, Department of Computer Science at the University of Toronto, 2012. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (date of access: 15.04.19).

12. **Zeiler M. D.** ADADELTA: An Adaptive Learning Rate Method, *arXiv*:1212.5701, 2012, pp. 1—6.

13. **Kingma D. P., Ba J. L.** Adam: A Method for Stochastic Optimization, *Proc. International Conf. on Learning Representations. San Diego*, 2015, pp. 1—15.

14. **Dozat T.** Incorporating Nesterov Momentum into Adam, *Proc. International Conf. on Learning Representations. San Juan*, 2016, pp. 1—4.

15. **Johnson M., Schuster M., Le Q. V., Krikun M. et al.** Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation, *arXiv*:1611.04558, 2016, pp. 1—17.

16. **Huang G., Liu Z., Weinberger K. Q., Van der Maaten L.** Densely Connected Convolutional Networks, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition. Honolulu*, 2017, pp. 2261—2269.

17. **Zhang Z., Ma L., Li Z., Wu C.** Normalized Direction-preserving Adam, *arXiv*:1709.04546, 2017, pp. 1—12.

18. **Reddi S. J., Kale S., Kumar S.** On the Convergence of Adam and Beyond, *Proc. International Conf. on Learning Representations. Vancouver*, 2018, pp. 1—23.

19. **Huang H., Wang C., Dong B.** Nostalgic Adam: Weighing more of the past gradients when designing the adaptive learning rate, *arXiv*:1805.07557, 2018, pp. 1—12.

20. **Chen J., Gu Q.** Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks, *arXiv*:1806.06763, 2018, pp. 1—18.

21. **Zaheer M., Reddi S. J., Sachan D. et al.** Adaptive Methods for Nonconvex Optimization, *Proc. 32nd Conf. on Neural Information Processing Systems. Montreal*, 2018, pp. 1—11.

УДК 004.896 DOI: 10.17587/it.25.670-678

**Б. К. Лебедев,** д-р техн. наук, проф., e-mail: lebedev.b.k@gmail.com,
**О. Б. Лебедев,** канд. техн. наук, доц., e-mail: lebedev.ob@mail.ru,
Южный федеральный университет, Ростов-на-Дону,
**О. А. Пурчина,** ассистент, e-mail: knagna_olga@inbox.ru,
Донской государственный технический университет, Ростов-на-Дону

# Модифицированный роевой алгоритм "муравьиное дерево" в задаче диверсификации трассировочных ресурсов[1]

*Предлагается модифицированный роевой алгоритм "муравьиное дерево" (trees ant colony optimization (T-ACO)). Отличием от канонической структуры является то, что в качестве графа поиска решений используется дерево решений. Агентом на графе поиска решений создается не маршрут, а дерево, которое по своей структуре совпадает с представлением решения задачи трассировки в надъячеечной области. Это исключает применение дополнительных трансформаций в процессе декодирования решений, допускающих интерпретацию решений в виде деревьев, и позволяет отбросить множество "нелегальных" решений, что приводит к повышению качества полученных решений. Использование процедуры трассировки в надъячеечной области на основе муравьиного алгоритма позволяет разгрузить канал на 15...20 %. Временная сложность алгоритма зависит от числа вершин графа n, числа агентов y, числа итераций l и определяется как $O(n^2ly)$.*

***Ключевые слова:*** *оптимизация, алгоритм "муравьиное дерево", САПР СБИС, диверсификация трассировочных ресурсов, надъячеечная область, планарный эскиз топологии*

## Введение

Одной из наиболее важных задач, которые решаются при проектировании топологии СБИС, является канальная трассировка.

Канал представляет собой область, ограниченную двумя линейками контактов: верхней и нижней [1]. На область трассировки наносится опорная ортогональная сеть, по линиям которой проходят трассы. Горизонтальные линии называют магистралями. Вертикальные линии проходят через контакты. При канальной трассировке каждая цепь, связывающая эквипотенциальные выводы, представляется в виде набора горизонтальных и вертикальных фрагментов (участков). В общем случае задача трассировки в канале (ЗКТ) рассматривается как задача размещения множества горизонтальных участков на множестве магистралей с соблюдением ограничений.

Существует множество трассировщиков, некоторые из которых позволяют получать почти оптимальные решения [1—5]. Однако по-прежнему проблема минимизации области канальной трассировки является актуальной. Дефицит трассировочного ресурса часто приводит к тому, что в процессе детальной трассировки проводники прокладываются в обход загруженных областей, что приводит к увеличению задержек в соединениях и уменьшению быстродействия СБИС в целом. Более того, дефицит трассировочного ресурса при недостаточно качественной глобальной трассировке может привести к невозможности 100%-ной детальной трассировки. Оценкой ***трассируемости*** коммутационной среды (КС) является соотношение между ресурсами КС и ресурсами, требуемыми для реализации соединений. Чем больше протрассированных соединений для КС с заданными размерами, или чем меньше размеры КС, требуемые для 100%-ной трассировки, тем выше трассируемость. Ресурсы КС определяются размерами КС.

Ресурсы, требуемые для реализации соединений, определяются конфигурацией и областью распространения соединений в КС, т. е. геометрическими параметрами соединений (координатами выводов, связываемых цепями).

В работе задача повышения трассируемости решается путем диверсификации трассировочных ресурсов, заключающейся в использовании дополнительных областей, примыкающих к каналам снизу и сверху, с трассировочным слоем, который располагается над ячейками "over-the-cell", как показано на рис. 1. Тогда часть соединений может быть перенаправлена из канала в надъячеечные области (НЯО), что

[1]Работа выполнена при финансовой поддержке РФФИ (грант № 18-07-00737а).

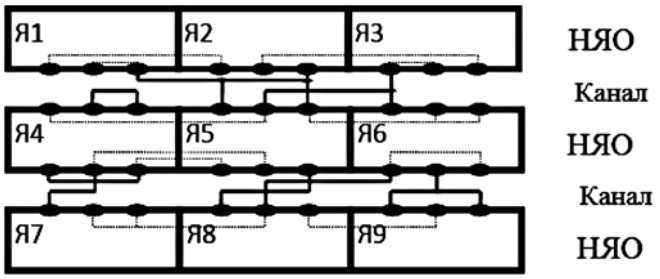



**Рис. 1. Структура коммутационного поля**

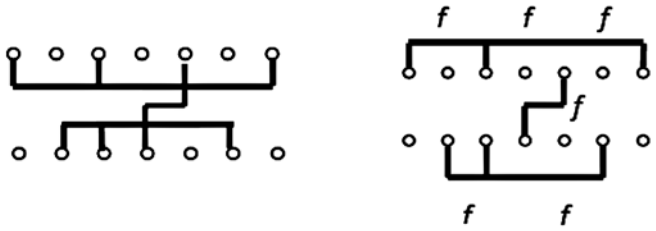



**Рис. 2. Разбиение многовыводной цепи на фрагменты**

приведет к сокращению числа магистралей, которые используются в канале [4, 5, 7, 8]. Перенаправленные соединения трассируются в одном слое. Целью надъячеечного трассировщика является создание максимального планарного эскиза топологии в НЯО, это должно позволить минимизировать плотность канала [4, 5].

Среди существующих алгоритмов трассировки в надъячеечной области [2] наиболее распространены: генетические алгоритмы (ГА) (авторы: B. M. Goni, T. Arslan, B. Torton); алгоритм на основе методов динамического программирования (МДП) (авторы: N. Holmes, N. Sherwani, M. Sarrafzadeh). В связи с тем что сложность и размерность решаемых задач в наше время резко возросли, МДП редко используется, поскольку они очень трудоемки. В свою очередь по тем же причинам в ГА резко возросла трудоемкость декодирований и усилились проблемы, связанные с получением "легальных" решений. Помимо этого, возникают проблемы надъячеечной трассировки в областях с ломаными границами, поэтому разработка алгоритма трассировки в НЯО, который мог бы отвечать современным требованиям, является актуальной научной задачей.

Основные процедуры при решении задачи надъячеечной трассировки:

1. Разбиение многовыводных цепей на двухтерминальные соединения.

2. Формирование наборов двухтерминальных соединений (ДТС) — кандидатов для размещения в надъячеечной области.

3. Формирование ориентированного графа поиска решений $G$ в виде дерева, задающего бинарное отношение вложения между каждой парой вершин, соответствующих ДТС.

4. Формирование полного графа $R$ пересечений, отражающего наличие неустранимого пересечения между каждой парой ДТС.

5. Выделение в графе $G$ ориентированного решающего дерева $T_w = (V_w, U_w)$, такого что между каждой парой вершин ДТС отсутствует отношение неустранимого пересечения.

Прежде всего, формируется набор $F = \{f_i | i = 1, 2, ..., m\}$ двухтерминальных соединений, которые являются кандидатами на перевод из канала в НЯО. Каждая из многовыводных цепей (рис. 2) на каждой из сторон канала разбивается на двухвыводные цепи (фрагменты), которые связывают контакты в линейке.

Пусть $l_i$ обозначает левый край фрагмента $f_i$, тогда $r_i$ будет правым краем, а $x(l_i)$ и $x(r_i)$ будут координатами краев фрагмента $f_i$ по оси $x$, которая распространяется вдоль линейки контактов.

Если $x(l_j) < x(l_i) < x(r_j)$, то между фрагментами $f_i$ и $f_j$ существует неустранимое пересечение (рис. 3). Формируется граф неустранимых пересечений $CR = (V, E)$, вершина $v_i \in V$ соответствует фрагменту $f_i$, т. е. $f_i = \Gamma(v_i)$, вершины $v_i$ и $v_j$ связаны ребром $e_k = (v_i, v_j)$, когда $f_j$ пересекается с $f_i$.

Будем считать, что $f_i$ вкладывается в $f_j$, в случае если:

- $x(l_j) < x(l_i)$, $x(r_j) > x(r_i)$;
- $f_i$ и $f_j$ размещаются в надъячеечной области без пересечений, т. е. $f_j$ будет выше $f_i$ (рис. 4).

Если $x(r_i) < x(l_j)$, то фрагменты $f_i$ и $f_j$ не пересекаются и могут размещаться в надъячеечной области без взаимопересечения.

Построим ориентированный граф поиска решений $G = (V, U)$, вершина $v_i \in V$ соответствует фрагменту $f_i$ ($f_i = G(v_i)$).

Вершины $v_i$ и $v_j$ связаны ребром, направленным от $v_i$ к $v_j$ только тогда, когда $f_j$ вкладывается в $f_i$. Вершина $v_r$, соответствующая верхней границе НЯО, включена в состав $V$. Вершина ($v_r$) будет связана с каждой вершиной $V \backslash v_r$, потому что любой из фрагментов вкладывается в НЯО, которая ограничена верхней границей.

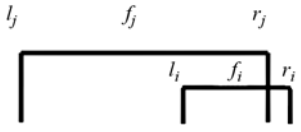



**Рис. 3. Неустранимое пересечение**

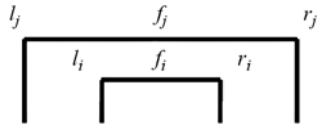



**Рис. 4. Вложение цепей**

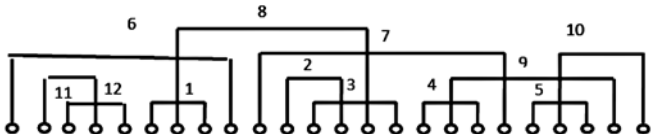



**Рис. 5. Пример схемы**

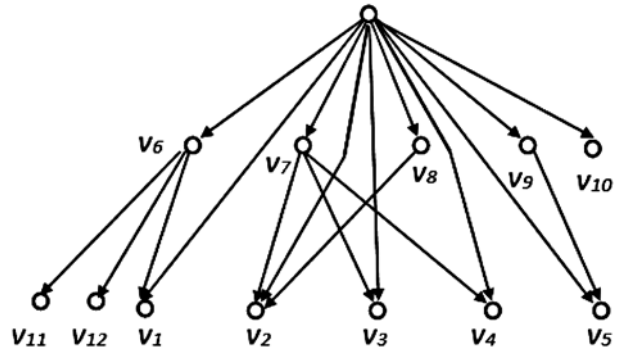



**Рис. 6. Граф поиска решений *G***

На рис. 5 изображена схема, а на рис. 6 соответствующий данной схеме граф поиска решений (ГПР) *G*.

Среди множества фрагментов $F_i$, вкладываемых в $f_i$, существует подмножество $F_{is} \subset F_i$, $G^{-1}(F_{is}) = B_{is} \subset B_i$, такое что никакие два фрагмента $f_k \in F_{is}$ и $f_j \in F_{is}$ не пересекаются. Назовем данное подмножество $F_{is}$ *s*-м вложением фрагмента $f_i$, а подмножество $B_{is}$ — *s*-м вложением вершины $v_i$. Множеств таких, как $F_{is}$, может быть несколько, $\cup F_{is} = F_i$.

## 1. Постановка задачи

Имеется набор $F = \{f_i | i = 1, 2, ..., m\}$ двухтерминальных соединений (фрагментов), которые являются кандидатами на перевод из канала в НЯО. На базе набора $F$ построен ориентированный граф поиска решений $G = (V, U)$ в виде дерева, отражающий бинарное отношение вложения, и полный граф неустранимых пересечений $CR = (V, E)$. Каждому из фрагментов $f_i$ соответствует стоимость $\varepsilon_i$. Необходимо в ГПР $G$ выделить подграф $T_w = (V_w, U_w)$, $T_w \subset G$, который обладает следующими свойствами [6, 9]:

- $T_w$ представляет собой дерево с корневой вершиной $v_r$;
- $B_{iw}$ — это множество дочерних вершин вершины $v_i \in V_w$, $B_{iw} \subset V_w$;
- для любой $v_i \in V_w$ множество $B_{iw}$ является вложением вершины $v_i$, иными словами, фрагменты множества $F_{iw} = G(B_{iw})$ являются непересекающимися и вкладываемыми во фрагмент $f_i$.

Дерево $T_w \in G$ со свойствами, которые перечислены выше, по факту задает способ планарной укладки в надъячеечной области фрагментов, которые соответствуют вершинам множества $V_w \subset V$. Будем называть данный подграф решающим деревом (РД). На рис. 7 приводится вариант РД $T_1$ с планарными укладками фрагментов, которые соответствуют вершинам РД.

Допустим, что каждому из фрагментов $f_i$ соответствует стоимость $\varepsilon_i$, тогда РД $T_w = (V_w, U_w)$ соответствует суммарная стоимость:

$$\psi_w = \Sigma_i(\varepsilon_i),\ i | v_i \in V_w.$$

Параметр $\psi_w$ принимается в качестве критерия оптимизации. Цель оптимизации состоит в максимизации параметра $\psi_w$.

Определим рекурсивно уровень $\alpha_{wi}$ фрагмента $f_i$ (вершины $v_i$), входящего в состав РД $T_w$ [9]. Вершина $v_i \in V_w$, не содержащая вложенных в нее вершин, имеет нулевой уровень ($\alpha_{wi} = 0$). Пусть $\mu_{wi}$ — максимальный уровень среди вершин, вложенных в $v_i$, следовательно, уровень $v_i$ равен $\mu_{wi} + 1$ ($\alpha_{wi} = \mu_{wi} + 1$).

Отметим, что вершина $v_i$ (фрагмент $f_i$) в разных РД может иметь разный уровень $\alpha_i$. Назовем блоком $t_{wi}$ поддерево РД $T_w$, имеющего своим корнем вершину $v_i \in V_w$. Уровень блока $t_{wi}$ равен уровню вершины $v_i \in V_w$, которая входит в РД $T_w$. Будем считать, что блок $t_{wi}$ входит в состав блока $t_{wj}$ в случае, если в РД $T_w$ корневая вершина $v_i$ блока $t_{wi}$ является дочерней вершиной корневой вершины $v_j$ блока $t_{wj}$ и $\alpha_{wj} \geqslant \alpha_{wi} + 1$.

Блоки $t_{wi}$ и $t_{wj}$ будем считать непересекающимися лишь в том случае, когда $v_i$ и $v_j$ не пересекаются друг с другом.

Пусть множеству вершин $B_{iw}$, являющихся вложением вершины $v_i$, соответствует множество блоков $Q_{iw}$.

Установим стоимость $\omega_{wi}$ блока $t_{wi}$. Стоимость $\omega_{wi}$ блока $t_{wi}$, располагающегося на ненулевом уровне ($\alpha_{wi} \neq 0$), будет эквивалентна суммарной стоимости множества блоков $Q_{iw}$, входящих в состав блока $t_{wi}$, и стоимости $\varphi_i$ вершины $v_i$:

$$\omega_{wi} = \varepsilon_i + \Sigma j(\omega_{wj}),\ j | v_j \in B_{wi}.$$

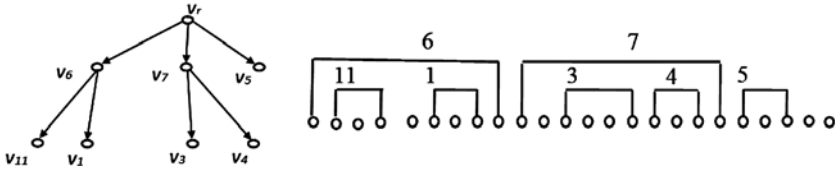



**Рис. 7. Пример решающего дерева**

Вершины $v_i \in V_w$, располагающиеся на нулевом уровне, играют роль блока нулевого уровня, его стоимость равна стоимости $v_i$; $\omega_{wi} = \varepsilon_i$.

Поэтому $\psi_w = \omega_{wr}$ — стоимость блока $t_{wr}$ с корнем в вершине $v_r$, равна стоимости РД $T_w$. В свою очередь, $t_{wr}$ и $T_w$ будут соответствовать друг другу. Чтобы установить число магистралей $m_w$, используемых для укладки РД $T_w$, воспользуемся значением уровня $\alpha_{wr}$ корневой вершины ($m_w = \alpha_{wr}$).

## 2. Однослойная трассировка в НЯО на основе комбинаторной оптимизации "муравьиное дерево"

В работах [6—10] исследователями было предложено описание и проведен анализ имеющихся на сегодняшний день алгоритмов однослойной трассировки. Для построения наиболее эффективного современного алгоритма, соответствующего всем требованиям, прибегают к использованию различных инновационных подходов и технологий. Сейчас активно разрабатывается одно из научных направлений, включающее в себя различные математические методики, построение которых осуществляется непосредственно на базе принципов природных механизмов принятия решений [11].

Для создания РД применяется структура комбинаторной оптимизации. Она получила название "*trees ant colony optimization* (T-ACO)". Эта структура создается на идеях адаптивного поведения муравьиной колонии [12, 13] и предоставляет возможность выполнить синтез древа [14]. Оптимизационная задача, имеющая вид структуры T-ACO, заключается в следующем:

- построение ГПР в виде дерева;
- создание на ГПР роем агентов допустимых деревьев решений;
- выбор лучшего решения.

Задача роя агентов заключается в построении на графе $G = (V, U)$ путем последовательного выбора множества вершин $V_w \subset V$ маршрута $M_w$, обладающего следующими свойствами. Маршрут $M_w$ является упорядоченной последовательностью подмаршрутов $M_w = \{M_{wi} | i = 1, 2, ..., m\}$, $\cup_i(M_{wi}) = M_w$. Подмаршрут $M_{wi}$ является упорядоченной последовательностью множества взаимно непересекающихся вершин $V_{wi} \subset V_w$, $\cup_i(V_{wi}) = V_w$. Вершины множества $V_{wi}$ соответствуют множеству фрагментов, размещаемых на $i$-й магистрали НЯО. Число подмаршрутов соответствует числу магистралей в НЯО. Подграф графа $G = (V, U)$, включающий множество вершин $V_w \subset V$, фактически является РД $T_w = (V_w, U_w)$, задающим способ планарной укладки фрагментов в надъячеечной области.

## 3. Построение модифицированного графа поиска решений

В качестве графа поиска решений авторы остановили свой выбор на модифицированном ГПР $G = (V, U)$. Основное отличие модернизированного графа поиска решений заключается в том, что в графе отсутствуют ребра, соответствующие замыканию транзитивных маршрутов вложения. Если $v_j$ вкладывается в $v_i$, то в графе $G$ существует только один маршрут единичной длины, связывающий $v_i$ и $v_j$ ребром, направленным от $v_i$ к $v_j$. Разработка графа решений начинается от основной вершины $v_r$. Данная вершина соответствует верхней точке надъячеечной зоны [14]. Изначально с учетом полученных результатов в ходе проведенного анализа схемы проводится построение множества вершин $V_1$. Любая из таких вершин может входить в состав только основной вершины $v_r$, в нашем примере на рис. 8 это $V_1 = \{v_6, v_8, v_7, v_9, v_{10}\}$. $v_r$ связана ребрами со всеми вершинами множества $V_1$. Затем формируется множество $V_2$. Любая из вершин множества $V_2$ может вкладываться только в состав некоторых вершин $V_1$. Другими словами, любая из вершин множества $V_2$ может быть дочерней вершиной только у некоторых вершин $V_1$. Между парами $v_i \in V_1$ и $v_j \in V_2$ формируются ребра, имеющие направление от $v_i$ к $v_j$, если $f_j$ вкладывается в состав $f_i$. На шаге $n$ с учетом полученных результатов в ходе проведенного анализа схемы проводится построение множества вершин $V_n$. Любая вершина множе-

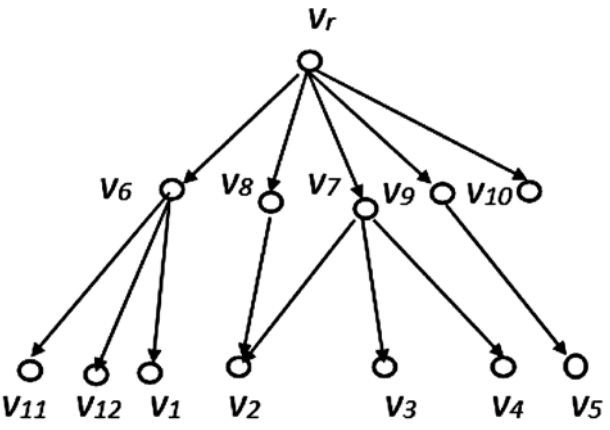



**Рис. 8. Граф поиска решений**

ства $V_n$ может вкладываться только в некоторые вершины множества $V_{n-1}$. Такая процедура осуществляется до тех пор, пока в графе $G$ не будут отображены все фрагменты схемы. Граф решений, созданный в качестве примера, отображен на рис. 8. Таким образом, ребра связывают вершины двух соседних уровней.

Результатом работы алгоритма муравьиного дерева (T-ACO) является РД $T_r$ с наилучшей оценкой на графе решений $G$. Поиск решения (решающего дерева $T_r$) сводится к построению маршрута с вышеописанными свойствами на базе графа решений $G$, который имеет фиксированную структуру, с использованием вспомогательного графа решений $GT$ с трансформирующейся структурой. Исходная структура графа $GT$ идентична структуре графа $G$. В качестве исходных данных формируется граф пересечений $CR = (V, E)$, между парами вершин $v_i \in E$ и $v_j \in E$ создается ребро, если фрагменты $f_i$ и $f_j$ пересекаются.

Поиск решений задачи осуществляет коллектив агентов $A = \{a_k | k = 1, 2, ..., n_k\}$. Модели поведения агентов связаны с распределением феромона на вершинах графа $G$, используемого в качестве хранилища коллективной эволюционной памяти. Первоначально на всех вершинах графа $G$ откладывается равное, небольшое, количество феромона $Q/v$, здесь $v = |U|$, параметр $Q$ задан априори. Поиск решений является итерационным процессом. На каждой итерации алгоритма муравьиного дерева каждый агент $a_k$ формирует свое решение задачи на трансформирующемся графе решений $GT$, используя в качестве хранилища коллективной памяти граф $G$. Интерпретацией решения является РД $T_k \subset G$, которому будет соответствовать эскиз планарной трассировки в НЯО. Итерация $l$ состоит из трех этапов.

## 4. Построение агентом решения (маршрута $M_k$ и соответствующего решающего дерева $T_k \subset G$)

Первый этап каждой итерации заключается в нахождении каждым агентом популяции своего решения (маршрута $M_k$ и дерева $T_k$) посредством последовательной трансформации исходного вспомогательного графа решений $GT = (V, U)$. Второй этап — агенты откладывают феромон на вершинах графа поиска решений $G$. Третий этап состоит в испарении феромона на вершинах графа поиска решений $G$. В работе применяется циклический "ant-cycle"-метод муравьиной системы, при этом феромон откладывается агентами на вершинах графа решений $G$ после того, как будут сформированы все решения популяции.

Решение находится в процессе последовательного просмотра вершин дерева решений.

Формирование агентом $a_k$ маршрута $M_k$ и соответствующего РД $T_k \subset G$ осуществляется путем последовательного распределения вершин $V$ графа $G$ по магистралям, начиная с первой, нумерация магистралей — снизу вверх. После выбора очередной магистрали формируется для размещения на ней множество $SK$ вершин-кандидатов.

При заполнении первой магистрали в состав множества кандидатов $SK_1$ входят только дочерние вершины графа $GT(0)$. Для ГПР (рис. 8) $SK_1 = \{v_{11}, v_{12}, v_1, v_2, v_3, v_4, v_5, v_{10}\}$.

С помощью процедуры $RAZM$ агент $a_k \in A$ формирует максимальное по стоимости подмножество $SK_1^* \subset SK_1$ взаимно-непересекающихся вершин множества $SK_1$ для размещения на первой магистрали: $SK_1^* \subset \{v_{11}, v_1, v_3, v_4, v_5\}$.

На графе пересечений $CR = (V, E)$ определяется множество $D_1$ вершин, пересекающиеся с вершинами множества $SK_1^*$. Для нашего примера $D_1 = \{v_{12}, v_8, v_2, v_9, v_{10}\}$.

После этого выполняется трансформация графа $GT(0)$ в граф $GT(1)$ путем удаления из $GT(0)$ подмножеств вершин $D_1$ и $SK_1^*$ со всеми инцидентными этим вершинам ребрами графа $GT(0)$.

Далее в графе $GT(1)$ формируется множество вершин $SK_2$, в состав которого входят только дочерние вершины графа $GT(0)$, и осуществляется переход к заполнению магистрали 2, $\gamma = 2$. Для нашего примера $SK_2 = \{v_6, v_7\}$.

При заполнении магистрали $\gamma$ в множество кандидатов $SK_\gamma$ входят только дочерние вершины графа $GT(\gamma - 1)$. С помощью процедуры $RAZM$ агент формирует максимальное по стоимости подмножество $SK_\gamma^*$ взаимно-непересекающихся вершин множества $SK_\gamma$ для размещения в магистрали $\gamma$.

На графе пересечений $CR = (V, E)$ определяется множество вершин $D_\gamma$, пересекающиеся с вершинами множества $SK_\gamma^*$.

Далее проводится трансформация графа $GT(\gamma - 1)$ в граф $GT(\gamma)$ путем удаления из $GT(\gamma - 1)$ подмножества вершин $D_\gamma$ и $SK_\gamma^*$ со всеми инцидентными этим вершинам ребрами графа $GT(\gamma - 1)$. Формируется множество вершин $SK_{\gamma+1}$ и выполняется переход к формированию уровня $\gamma = \gamma + 1$.

Распределение агентом $a_k$ вершин по магистралям прекращается после того, как в ре-

зультате трансформаций граф $GT(\gamma)$ станет пустым ($GT(\gamma) = \varnothing$) и, следовательно, $SK_{\gamma+1} = \varnothing$. Обозначим $V_k(l)$ — множество вершин графа $G$, распределенных агентом $a_k$ по $\gamma$ магистралям на итерации $l$. РД $T_k(l) \subset G$ формируется следующим образом. Множество вершин $V_k(l) \subset G$ графа $G$ помечаются. Затем из графа удаляются все непомеченные вершины со всеми инцидентными этим вершинам ребрами графа $G$. Оставшаяся часть графа $G$ является решающим деревом $T_k(l) \subset G$.

После этого подсчитывается суммарная стоимость $\psi_k(l)$ множества вершин $V_k(l)$, распределенных агентом по магистралям, которая является оценкой решения.

Рассмотрим процедуру *RAZM*. Упорядочим множество вершин $SK_\gamma$ — кандидатов для размещения на магистрали по возрастанию координат левых концов фрагментов, соответствующих этим вершинам. Построим вектор $P$. Пусть $P(v_i)$ — порядковый номер $v_i$ в векторе $P$. Тогда $\forall ij\{x(l_i) < x(l_j)\} \rightarrow \{P(v_i) < P(v_j)\}$.

Упорядочим вершины множества $SK_\gamma$ в линейку в том порядке, в каком соответствующие им фрагменты расположены в векторе $P$.

Формирование агентом $a_k \in A$ максимального по стоимости подмножества $SK^*_\gamma \subset SK_\gamma$ взаимно-непересекающихся вершин множества $SK_\gamma$ для размещения на $\gamma$-м уровне осуществляется последовательно (пошагово) путем последовательного просмотра элементов упорядоченного списка $SK_\gamma$.

На шаге $t$ в памяти агента $a_{\tau k}$ хранится номер заполняемой магистрали $\gamma$, множество $SK_\gamma$, список $SK^*_\gamma(t)$ вершин, уже включенных в формируемый уровень, $v_z$ — последняя вершина (фрагмент $f_z$), включенная в список $SK^*_\gamma(t)$.

Агент просматривает множество всех свободных на данном шаге вершин $SK_\gamma \backslash SK^*_\gamma(t)$, размещенных в списке $SK_\gamma$ справа от $v_z$, и выделяет из него подмножество вершин $Z_\gamma(t) \subset SK_\gamma \backslash SK^*_\gamma(t)$, удовлетворяющих *условиям размещения* фрагментов в текущей магистрали.

*Условия размещения*:

а) каждая вершина $v_i \in Z_\gamma(t)$ может быть размещена на текущей заполняемой магистрали без пересечений с уже размещенными вершинами множества $SK^*_\gamma(t)$;

б) если $f_i \in Z_\gamma(t)$ поместить в магистраль, то не существует фрагмента $f_j \in Z_\gamma(t)$, который может быть размещен в оставшейся свободной зоне заполняемой магистрали между фрагментами $f_z$ и $f_i$ без нарушения ограничений и конфликтов (пересечений) с $f_i$.

Соблюдение рассмотренных условий объясняется следующим образом. Поскольку заполнение магистралей фрагментами проводится последовательно с использованием "безвозвратной" стратегии, нет смысла оставлять на магистрали не заполненными (пустыми) области, в которые могут быть помещены некоторые фрагменты.

Для каждой, отвечающей *условиям размещения*, вершины $f_i \in Z_\gamma(t)$ рассчитываются три параметра $\varphi_i$, $\lambda_i$, $\varepsilon_i$:

$\varphi_i$ — суммарное количество феромона, скопившегося на вершине $v_i$;

$\lambda_i$ — расстояние между окончанием последнего в заполняемой магистрали фрагмента $f_z$ и началом фрагмента $f_i$, размещенных в одной магистрали. Если $f_i$ располагается в начале магистрали, то $d_i$ — расстояние между началом магистрали и началом $f_i$;

$\varepsilon_i$ — стоимость фрагмента $f_i$.

Интегральная стоимость $Q_{iz}(t)$ связи вершины $f_i \in SK^*_\gamma(t)$ $c$ $f_z \in SK^*_\gamma(t)$ определяется при мультипликативной свертке по формуле

$$Q_{iz}(t) = (\varphi_i^\alpha (l-1)\varepsilon_i^\beta)/(\lambda_i^\delta + 1), \qquad (1)$$

а при аддитивной свертке — по формуле

$$Q_{iz}(t) = \alpha\varphi_i(l-1) + (\delta/(\lambda_i + 1) + \beta\varepsilon_i, \qquad (2)$$

где $\alpha$, $\beta$, $\delta$ — управляющие параметры, которые подбираются экспериментально.

Вероятность $P_i(t)$ включения вершины $f_i \in Z_\gamma(t)$ в формируемый маршрут $SK^*_\gamma(t)$ определяется следующим соотношением:

$$P_i(t) = Q_{iz}(t)/\theta, \text{ где } \theta = \Sigma_i(Q_{iz}(t)),\ i|f_i \in Z_\gamma(t). \qquad (3)$$

Рассмотрим эвристические соображения, положенные в основу формул (1) и (2). Чем больше $\varphi_i(l-1)$, тем больше вероятность того, что вершина $f_i$ входит в состав оптимального маршрута. Чем меньше $\lambda_i$, тем выше плотность заполнения магистрали, что способствует уменьшению числа магистралей, необходимого для размещения фрагментов.

Учет в выражениях (1) и (2) параметра $\varepsilon_i$ способствует максимизации стоимости подмножества $SK^*_\gamma \subset SK_\gamma$ взаимно-непересекающихся вершин множества $SK_\gamma$ для размещения на $\gamma$-й магистрали, что косвенным образом способствует повышению эффективности поиска и минимизации целевой оценки — числа магистралей.

Агент с вероятностью $P_i(t)$ выбирает одну из вершин $f_i \in Z_\gamma(t)$, которая размещается на $\gamma$-й магистрали. После этого множество размещенных вершин уже обозначается как $SK^*_\gamma(t+1)$, а $f_i$ исключается из множества $Z_\gamma(t)$. При $\alpha = 0$ наиболее вероятен выбор ближайшего к $f_z$ фрагмента $f_i$, т. е. алгоритм становится подобным алгоритму "левого конца". При $\beta = 0$ вероятность выбора практически зависит только от количества феромона на ребре, что приводит к субоптимальным решениям. Соотношение между этими величинами находится экспериментально.

После формирования всеми агентами деревьев решений на вершинах ГПР откладывается феромон. Каждый агент откладывает на вершинах $V_k(l) \subset V$ феромон в количестве $\tau_k(l)$, пропорциональном оценке $\psi_k(l)$ РД:

$$\tau_k(l) = c\psi_k(l), \quad (4)$$

где $c$ — числовой коэффициент.

Обозначим $\varphi_i(l)$ — общее количество феромона, скопившегося на вершине $v_i \in V$ в ГПР $G$ после выполнения агентами второго этапа $l$-й итерации.

На третьем этапе итерации на вершинах графа $R^*_\tau(F, E)$ в соответствии со следующей формулой происходит испарение феромона.

$$\tau_k(l) = \tau_k(l)(1 - \rho), \quad (5)$$

где $\rho$ — коэффициент обновления.

Последние действия на итерации связаны с нахождением лучшего решения, которое запоминается. После этого выполняется следующая итерация. Временная сложность этого алгоритма зависит от числа вершин графа $n$, числа муравьев $y$, числа итераций $l$ и определяется как $O(n^2ly)$.

## 5. Исследование алгоритма трассировки соединений в приканальной надъячеечной области

Целью эксперимента было определение числа итераций муравьиного алгоритма, которое с большой вероятностью (близкой к единице) гарантировало бы нахождение оптимального решения.

Вначале были проведены исследования, направленные на поиск значения управляющих параметров, которые смогли бы обеспечить наибольшую эффективность работы НЯО-МД. Результаты для муравьиного алгоритма получены при следующих количественных значениях:

- агенты — 100;
- итерации — 120;
- начальное отложение феромона — 1000;
- количество феромона, откладываемого агентом на решении — 100;
- коэффициент испарения феромона — 0,7.

График зависимости трудоемкости алгоритма трассировки в НЯО от числа цепей изображен на рис. 9.

Во всех решенных примерах получены оптимальные результаты. Трудоемкость алгоритма: $O(n^2)$.

В таблице приведены значения для десяти тестовых примеров, полученные трассировщиком без использования НЯО, и результаты, полученные трассировщиком с использованием разработанного муравьиного алгоритма трассировки в НЯО.

Успех работы алгоритма зависит от правильной настройки параметров управления. Источ-

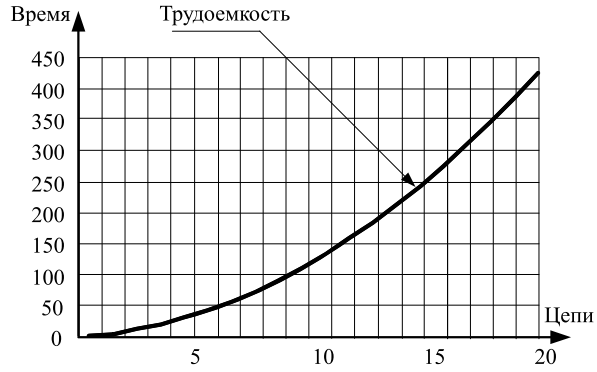



**Рис. 9. Зависимость трудоемкости алгоритма трассировки в НЯО от числа цепей**

**Результаты экспериментальных исследований**

| № примера | Без НЯО (число магистралей) | С НЯО (число магистралей) | Процент разгрузки канала |
|---|---|---|---|
| 1 | 14 | 12 | 14,3 % |
| 2 | 12 | 10 | 16,7 % |
| 3 | 14 | 9 | 35,7 % |
| 4 | 20 | 16 | 20 % |
| 5 | 24 | 21 | 12,5 % |
| 6 | 25 | 22 | 12 % |
| 7 | 29 | 24 | 17,2 % |
| 8 | 32 | 26 | 20,8 % |
| 9 | 33 | 26 | 21,2 % |
| 10 | 35 | 28 | 25 % |

ником усовершенствования работы муравьиного алгоритма может стать использование более сложных модифицированных стратегий выбора альтернатив при прокладке маршрутов.

Проведен синтез ряда примеров с заранее известными оптимальными значениями целевых функций для анализа точности решений, получаемых с использованием данного алгоритма.

При этом вероятность нахождения оптимальных решений составляет 0,93, а общая оценка временной сложности находится в пределах $O(n^2)...O(n^3)$.

## Заключение

В работе задача повышения трассируемости решается путем диверсификации трассировочных ресурсов, заключающейся в использовании дополнительных областей, примыкающих к каналам снизу и сверху, с трассировочным слоем, который располагается над ячейками — "over-the-cell". Тогда часть соединений может быть перенаправлена из канала в НЯО, что приведет к сокращению числа магистралей, которые используются в канале. Перенаправленные соединения трассируются в одном слое. Целью надъячеечного трассировщика является создание максимального планарного эскиза топологии в НЯО, это должно позволить минимизировать плотность канала.

В целях построения эскиза однослойной трассировки использована структура комбинаторной оптимизации "муравьиное дерево". В качестве графа поиска решений использован упорядоченный модифицированный граф решений $G^* = (V, U)$, его построение осуществляется путем рекурсивного определения возможного вложения фрагментов друг в друга. Отличием от канонической структуры муравьиного алгоритма является то, что муравьем на графе поиска решений создается не маршрут, а дерево, которое по своей структуре совпадает с представлением решения. Это исключает применение дополнительных трансформаций в процессе декодирования решений, допускающих интерпретацию решений в виде деревьев, и позволяет отбросить множество "нелегальных" решений, что приводит к повышению качества полученных решений. Использование процедуры трассировки в НЯО позволяет разгрузить канал на 15...20 %. Временная сложность этого алгоритма зависит от числа вершин графа $n$, числа агентов $y$, числа итераций $l$ и определяется как $O(n^2ly)$.

**Список литературы**


1. **Немудров В., Мартин Г.** Системы-на-кристалле. Проектирование и развитие. М.: Техносфера, 2004. 157 с.

2. **Bozhko A. N., Karpenko A. P.** Computer-aided Subassembly Generation // Proceedings of the Vth International workshop "Critical infrastructures: Contingency management, Intelligent, Agent-based, Cloud computing and Cyber security" (IWCI 2018). 2018. DOI:10.2991/iwci-18.2018.2.

3. **Божко А. Н., Родионов С. В.** Методы искусственного интеллекта в автоматизированном проектировании процессов сборки // Наука и образование. МГТУ им. Н. Э. Баумана. Электрон. журн. ЭЛ № ФС 77-48211. 2016. № 8. DOI: 10.7463/0816.0844719.

4. **Карпенко А. П., Волосатова Т. М., Маничев В. Б., Грошев С. В., Жук Д. М., Божко А. Н., Мартынюк В. А., Пивоварова Н. В., Трудоношин В. А., Норенков И. П.** Основы автоматизированного проектирования. М.: ООО "Научно-издательский центр ИНФРА-М", 2015. 329 с.

5. **Литвинов Е. И., Жихарев Г. Ю., Шагурин И. И.** Возможности использования виртуальных платформ для верификации RTL-моделей сложнофункциональных блоков в составе "Систем-на-кристалле" // Проблемы разработки перспективных микро- и наноэлектронных систем — 2014. Сборник трудов. М.: ИППМ РАН, 2014. Часть 2. С. 51—56.

6. **Быков С. О.** Автоматизация проектирования сетей-на-кристалле со специализированной топологией // Проблемы разработки перспективных микро- и наноэлектронных систем — 2014. Сборник трудов. М.: ИППМ РАН. 2014. Часть 2. С. 21—24.

7. **Казеннов Г. Г.** Основы проектирования интегральных схем и систем. М.: Бином. Лаборатория знаний, 2005. 295 с.

8. **Гарбулина Т., Лялинская О. В., Хватов В. М.** Повышение эффективности проектирования интегральных схем на ПЛИС с ограниченными трассировочными ресурсами // Проблемы разработки перспективных микро- и наноэлектронных систем — 2016. Сборник трудов. М.: ИППМ РАН. 2016. № 1. С. 165—171.

9. **Лебедев О. Б., Пурчина О. А.** Роевой алгоритм трассировки в приканальной надъячеечной области // Известия Южного федерального университета. Технические науки. Ростов-на-Дону: Изд-во ЮФУ, 2015. № 6, С. 125—138.

10. **Лебедев Б. К., Лебедев О. Б., Пурчина О. А.** Роевой алгоритм повышения плотности топологии СБИС // Сборник научных трудов VIII Международной научно-практической конференции "Интегрированные модели и мягкие вычисления в искусственном интеллекте". Т. 2. М.: Изд-во Физматлит. 2015. С. 624—634.

11. **Карпенко А. П.** Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой: учеб. пособие. М: Издательство МГТУ им. Н. Э. Баумана, 2014. 448 с.

12. **Dorigo M., Stützle T.** Ant Colony Optimization. MIT Press. Cambridge MA, 2004. 187 p.

13. **Лебедев О. Б.** Модели адаптивного поведения муравьиной колонии в задачах проектирования. Таганрог: Изд-во ЮФУ. 2013. 199 с.

14. **Лебедев Б. К., Лебедев О. Б., Пурчина О. А.** Однослойная трассировка в приканальной надъячеечной области на основе алгоритма муравьиного дерева // Труды конгресса по интеллектуальным системам и информационным технологиям "IS—IT'14". Науч. изд. в 4-х томах. М.: ФИЗМАТЛИТ. Т. 1. 2014. С. 58—69.

15. **Рыженко Н. В., Быков С. А., Сорокин А. А.** Трассировка битовых элементов памяти с автоматическим построением ограничений на границах ячеек // Проблемы разработки перспективных микро- и наноэлектронных систем — 2016. Сб. тр. М.: ИППМ РАН. 2016. № 1. С. 144—150.

**B. K. Lebedev,** Doctor of Engineering Sciences, Professor, e-mail: lebedev.b.k@gmail.com,
**O. B. Lebedev,** Candidate of Engineering Sciences, Associate Professor, e-mail: lebedev.ob@mail.ru,
Southern Federal University, Rostov-on-Don, 344006, Russian Federation,
**O. A. Purchina,** Assistant, e-mail: knagna_olga@inbox.ru,
Don State Technical University, Rostov-on-Don, 344000, Russian Federation

# Modified Swarm Algorithm "Ant Tree" in the Problem of Diversification of Tractor Resources

*A modified swarm algorithm (trees ant colony optimization (T-ACO)) is proposed. The difference from the canonical structure is that the decision tree is used as a decision search graph. An agent for the decision search graph does not create a route, but a tree, which in its structure coincides with the representation of the solution to the trace problem in the "over-the-cell" region. This eliminates the use of additional transformations in the decoding process of decisions that allow interpretation of decisions in the form of trees and allows you to discard a lot of "illegal" solutions, which leads to an increase in the quality of the solutions obtained. Using the tracing procedure in "over-the-cell" based on the ant algorithm allows you to unload the channel by 15—20 %. The time complexity of the algorithm depends on the number of vertices n of the decision search graph, the number of agents y, the number of iterations l, and is defined as O ($n^2$ly).*

***Keywords:*** *optimization, "ant tree" algorithm, CAD VLSI, diversification of trace resources, "over-the-cell" area, planar sketch of topology*

**Acknowledgements:** This work was supported by the Russian Foundation for Basic Research (Grant № 18-07-00737a).

DOI: 10.17587/it.25.670-678

**Reference**

1. **Nemudrov V., Martin G.** System-on-a-chip. Design and development, Moscow, Technosphere 2004, 157 p. (in Russian).

2. **Bozhko A. N., Karpenko A. P.** Computer-aided Subassembly Generation // *Proceedings of the Vth International workshop "Critical infrastructures: Contingency management, Intelligent, Agent-based, Cloud computing and Cyber security" (IWCI 2018),* 2018, DOI:10.2991/iwci-18.2018.2.

3. **Bozhko A. N., Rodionov S. V.** Methods of artificial intelligence in computer-aided design of the assembly process, *Nauka i obrazovaniye. MGTU im. N. E. Baumana. Elektron. zhurn. № FS 77-48211,* 2016, № 8. DOI: 10.7463 / 0816.0844719 (in Russian).

4. **Karpenko A. P., Volosatov T. M., Manichev V. B., Groshev S. V., Zhuk D. M., Bozhko A. N., Martynyuk V. A., Pivovarova N. V., Trudonoshin V. A., Norenkov I. P.** Fundamentals of computer-aided design: a tutorial, Moscow, Scientific Publishing Center INFRA-M, 2015, 329 p. (in Russian).

5. **Litvinov E. I., Zhikharev G. Yu., Shagurin I. I.** Possibilities of using virtual platforms for verification of RTL-models of complex functional units in the "Systems on a chip", *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem — 2014, Sbornik trudov*, 2014, no. 2, pp. 51—56 (in Russian).

6. **Bykov S. O.** Automation of network-on-chip design with a specialized topology, *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem — 2014. Sbornik trudov,* 2014, no. 2, pp. 21—24 (in Russian).

7. **Kazennov G. G.** Fundamentals of design of integrated circuits and systems, Moscow, Binom, Laboratory of knowledge, 2005, 295 p. (in Russian).

8. **Garbulina T., Lyalinskaya O. V., Khvatov V. M.** Improving the efficiency of designing integrated circuits on a FPGA with limited trace resources, *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem* — 2016*, Sb. trudov,* 2016, no. 1, pp. 165—171 (in Russian).

9. **Lebedev O. B., Purchina O. A.** The swarm tracing algorithm in the near-canal supracellular region, *Izvestiya Yuzhnogo federal'nogo universiteta. Tekhnicheskiye nauki*, 2015, no. 6, pp. 125—138 (in Russian).

10. **Lebedev B. K., Lebedev O. B., Purchina O. A.** Swarm algorithm for increasing the density of the VLSI topology, *Sbornik nauchnykh trudov VIII Mezhdunarodnoy nauchno-prakticheskoy konferentsii "Integrirovannyye modeli i myagkiye vychisleniya v iskusstvennom intellekte"*, 2015, vol. 2, pp. 624—634 (in Russian).

11. **Karpenko A. P.** Modern search engine optimization algorithms. Algorithms inspired by nature: a tutorial, Moscow, Publishing House MSTU N. E. Bauman, 2014, 448 p. (in Russian).

12. **Dorigo M., Stützle T.** Ant Colony Optimization, MIT Press, Cambridge MA, 2004, 187 p.

13. **Lebedev O. B.** Models of adaptive behavior of an ant colony in design problems, Rostov-on-Don, Publishing House of Southern Federal University, 2013, 199 p. (in Russian).

14. **Lebedev B. K., Lebedev O. B., Purchina O. A.** Single-layer tracing in the canal supracellular area based on the ant tree algorithm, *Trudy kongressa po intellektual'nym sistemam i informatsionnym tekhnologiyam "IS—IT'14",* 2014, vol. 1, pp. 58—69 (in Russian).

15. **Ryzhenko N. V., Bykov S. A., Sorokin A. A.** Tracing of bit memory elements with automatic construction of constraints on cell borders, *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem* — 2016. *Sbornik trudov,* 2016, no.1, pp. 144—150 (in Russian).

УДК 004.925.83 + 004.438 DOI: 10.17587/it.25.679-681

**О. А. Соснина,** канд. техн. наук, доц., e-mail: o.a.sosnina@mail.ru,
**А. Д. Филинских,** канд. техн. наук, доц., e-mail: alexfil@yandex.ru,
**Н. А. Ложкина,** магистрант, e-mail: nataly9951@gmail.com,
Нижегородский государственный технический университет им. Р. Е. Алексеева

# Анализ методов создания виртуальных моделей нетривиальных форм

*Параметрическая архитектура, которая включает в себя и проектирование, и дизайн, невозможна без параметрического программного моделирования. Одной из программ, позволяющих создавать здания в параметрическом стиле, является "Grasshopper". В данной статье описано моделирование в этой программе на примере одного из самых ярких зданий параметрической архитектуры — бизнес-центра "Galaxy Soho", построенного в Пекине по проекту Захи Хадид.*

***Ключевые слова:*** *параметрическая архитектура, параметризм, Grasshopper, 3ds max, 3D-моделирование, Заха Хадид*

## Введение

В настоящий момент существует множество программ для создания виртуальных трехмерных моделей и реалистичной визуализации. Затраты времени для создания определенных моделей различны, так как программы имеют различный функционал. В данной статье рассматривается метод моделирования в двух различных программах — "Grasshopper" [1] и "3ds Max" [2] — в целях сокращения времени на получение фотореалистичной визуализации. Одной из важнейших задач в этом случае является правильная передача данных между двумя средами моделирования.

Процесс моделирования в "Grasshopper" — это создание алгоритма методом связывания нужных компонентов друг с другом. Компоненты содержат входящие и выходящие параметры, изменяя значения которых, мы можем перестраивать весь объект в целом.

Отличие генеративного моделирования от традиционного заключается в том, что мы создаем логику, по которой можно получить модель при различных исходных данных. Создав алгоритм один раз, мы можем использовать и редактировать его в будущем.

## Моделирование в "Grasshopper"

В основе "Galaxy Soho" [3] лежат шары, соединенные друг с другом. Для описания подобной формы в "Grasshopper" используется компонент "MetaBall". С помощью сопутствующих компонентов создается схема, частично отображающая форму будущего здания (рис. 1, 2).

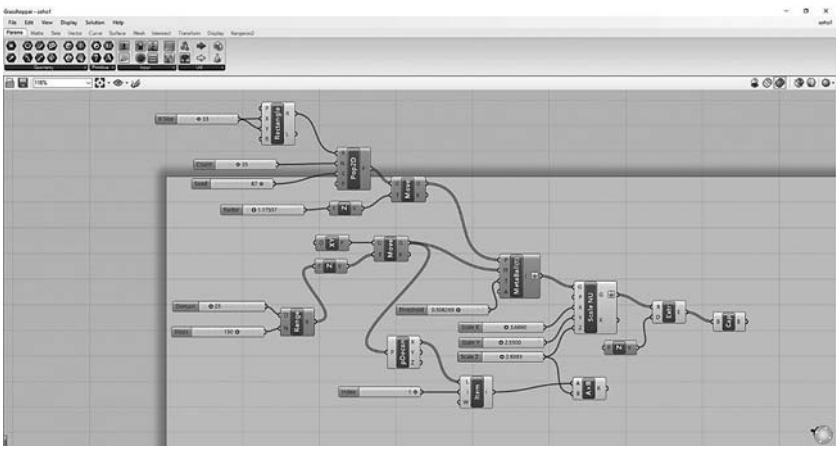


**Рис. 1. Схема компонентов, отражающая модель "Galaxy Soho"**

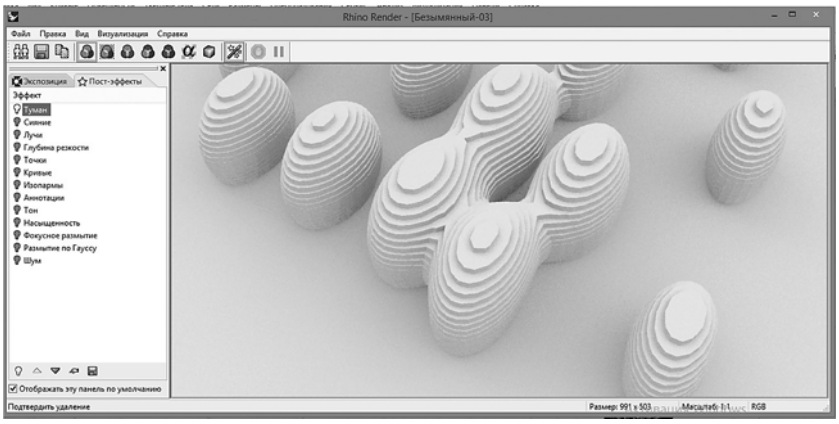



**Рис. 2. Модель, подготовленная для передачи в "3ds max"**

| Формат | Число полигонов | Число ребер | Размер файла, Кбайт |
|---|---|---|---|
| *.stp | 8136 | 24 408 | 6379 |
| *.dwg | — | 10 315 | 1626 |
| *.obj | 10 654 | 31 962 | 9141 |
| *.sat | 9518 | 21 453 | 9929 |
| *.dxf | — | 12 069 | 6779 |
| *.dae | 18 156 | 54 468 | 2506 |
| *.skp | 18 146 | 54 438 | 1810 |
| *.stl | 64 238 | 192 714 | 3608 |
| *.wrl | 18 558 | 55 674 | 4108 |

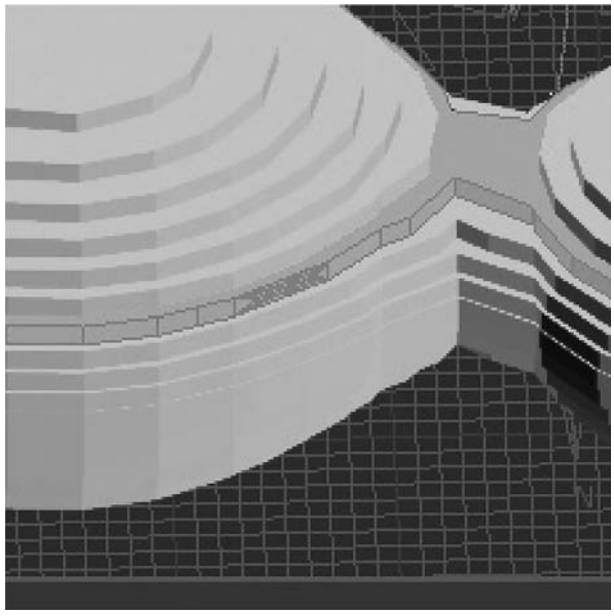


**Рис. 6. Повреждения сетки**

Для создания фотореалистичной визуализации целесообразно передать модель в программу моделирования "3ds max" и продолжить работу.

### Передача геометрии в "3ds max"

Объекты из "Rhino 3D" [4] (с плагином "Grasshopper") можно экспортировать в очень высоком качестве. Параметр, отвечающий за сглаженность модели (и, соответственно, за число полигонов), редактируется при создании модели. При очень высоком качестве число полигонов резко возрастает, и на экспорт требуется большое количество времени. При экспорте в большинство форматов компьютеру не хватало ресурсов, и он "зависал". Поэтому было решено несущественно снизить качество модели, при этом число полигонов и время на экспорт значительно уменьшилось.

Объекты можно экспортировать в 31 формат. Все они дают разные результаты. Главное их отличие в числе получаемых полигонов и ребер, а также в сетке объекта. При проведении эксперимента из 31 формата в "3ds max" открылись лишь 9. Не открылись следующие форматы: *3dm, 3mf, amf, cd, x, emf, kmz, gts, lwo, inc, udo, ply, pov, raw rib, svg, svgbak, slc, vda, gdf, wmf, x3dv, zpr, msg, x_t, xgl, mtl.* Файл формата **.fbx* загружался очень долго, и компьютер "зависал". Результат открытия и информация о полигонах указаны в таблице.

Исходя из результатов исследования можно сделать вывод, что для дальнейшего редактирования тела лучше использовать файл формата **.sat* или **.stp.* Но объекты данных форматов импортируются как "Body Object", вследствие чего они имеют слабый функционал редактирования. При конвертации объекта в "Editable Poly" сетка объекта ломается (рис. 3, 4, см. вторую сторону обложки), и число полигонов возрастает до 31 914.

Формат **.skp* имеет 18 146 полигонов и 54 438 ребер, но при передаче файла из "Rhino" в "3ds max" сохраняется сетка. Она сохраняется не полностью, имеет некоторые повреждения; но поскольку эти повреждения находятся в местах, где редактирование проводиться не будет, их можно не учитывать (рис. 5, см. вторую сторону обложки, рис. 6).

### Моделирование в "3ds max"

После того как была получена информация о сетке и числе полигонов объекта, можно приступить к его дальнейшему редактированию. Создаем недостающие связи между шарами (рис. 7, см. третью сторону обложки).

На данном этапе модель можно сгладить командой "MeshSmooth". Как видно на рис. 8 (см. третью сторону обложки), модель приобретает повреждения, неприемлемые для дальнейшей работы. Поэтому оставим модель *несглаженной*, и продолжим работу над ее редактированием.

За моделированием следует этап текстурирования. Далее создадим окружение для более высокой реалистичности рендера (рис. 9, см. третью сторону обложки). Для создания домов на заднем плане используем плагин "Greeble" [5]. Также используем плагин "Forest Pack" [6]: он позволяет сгенерировать траву, камни и несколько видов деревьев и кустарников.

Для создания освещения используется встроенная в "3ds max" система "DayLight System". Она позволяет создать освещение, близкое к естественному, учитывая при этом географическое положение объекта и время суток, в которое сделана съемка. После того как сцена готова к визуализации, устанавливаем камеры, корректируем их; устанавливаем настройки рендера и осуществляем сам рендеринг. Результат визуализации представлен на рис. 10 (см. третью сторону обложки).

### Заключение

В результате проделанной работы было выявлено, что при создании объектов нетривиальных форм разделение процесса моделирования на две программы, "GrassHopper" и "3ds max", существенно сокращает время работы. Это объясняется тем, что создавать нестандартные архитектурные формы быстрее и проще в программе "GrassHopper", а фотореалистичные рендеры — в "3ds max". При этом качество полученной модели получается высоким.

#### Список литературы


1. **GrassHopper.** URL: https://archi.place/program/grasshopper/ (дата обращения 03.02.2019).
2. **Autodesk 3ds max.** URL: https://www.autodesk.ru/products/3ds-max/overview (дата обращения 03.02.2019).
3. **Archi.ru:** Комплекс Galaxy SOHO. URL: https://archi.ru/projects/world/6129/kompleks-galaxy-soho (дата обращения 03.02.2019).
4. **Rhinoceros.** URL: https://www.rhino3d.com/ (дата обращения 03.02.2019).
5. **Greeble.** URL: https://en.wikipedia.org/wiki/Greeble (дата обращения 03.02.2019).
6. **Forest Pack.** URL: https://www.itoosoft.com/ru/ (дата обращения 03.02.2019).
7. **Барчугова Е. В.** Параметризм как направление современной проектной деятельности. URL: https://www.marhi.ru/AMIT/2013/4kvart13/barchugova/barchugova.pdf (дата обращения 03.02.2019).
8. **Официальный** сайт Zaha Hadid Architects. URL: http://www.zaha-hadid.com/ (дата обращения 03.02.2019).



**O. A. Sosnina,** Assistant Professor, e-mail: o.a.sosnina@mail.ru,
**A. D. Filinskikh,** Assistant Professor, e-mail: alexfil@yandex.ru,
**N. A. Lozhkina,** Master's Student, e-mail: nataly9951@gmail.com,
Nizhny Novgorod State Technical University n. a. R. E. Alekseev


## An Analysis of Methods for Creating Virtual Models of Complex Forms


*Parametric architecture, which includes engineering and design, is impossible without parametric software modeling. Grasshopper is a program that permits creating buildings in the parametric style. This article describes modeling in Grasshopper as exemplified by one of the most prominent buildings of parametric architecture — the Galaxy Soho business center, built in Beijing upon Zaha Hadid's design.*



***Keywords:*** *parametric architecture, parametric, Grasshopper, 3ds max, 3D modeling, Zaha Hadid*

DOI: 10.17587/it.25.679-681


#### References


1. **GrassHopper**, available at: https://archi.place/program/grasshopper/ (date of access: 03/02/2019).
2. **Autodesk 3ds max**, available at: https://www.autodesk.ru/products/3ds-max/overview (date of access: 03/02/2019).
3. **Archi.ru:** Galaxy SOHO complex, available at: https://archi.ru/projects/world/6129/kompleks-galaxy-soho (date of access: 03/02/2019) (in Russian).
4. **Rhinoceros**, available at: https://www.rhino3d.com/ (date of access: 03/02/2019).
5. **Greeble**, available at: https://en.wikipedia.org/wiki/Greeble (date of access: 03/02/2019).
6. **Forest Pack**, available at: https://www.itoosoft.com/ru/ (date of access: 03/02/2019).
7. **Barchugova E. V.** Parametric as a direction of modern project activity, available at: https://www.marhi.ru/AMIT/2013/4kvart13/barchugova/barchugova.pdf (date of access: 03/02/2019) (in Russian).
8. **Official** site of Zaha Hadid Architects, available at: http://www.zaha-hadid.com/ (date of access: 03/02/2019).

УДК 004.2; 004.31 DOI: 10.17587/it.25.682-690

**А. Ю. Попов,** канд. техн. наук, доц., e-mail: alexpopov@bmstu.ru,
Московский государственный технический университет им. Н. Э. Баумана, г. Москва

# Применение гетерогенной вычислительной системы с набором команд дискретной математики для решения задач на графах

*В последнее десятилетие существенно вырос интерес к анализу сложных моделей данных, в том числе графов, являющихся наиболее адекватной формой представления данных социальных сетей, компьютерных программ, топологии интегральных схем, биоинформатики и других важных приложений. По мере того, как размер этих наборов данных увеличивается, становится очевидной необходимость поиска более эффективных методов и средств анализа больших графов, в том числе на основе более совершенных аппаратных технических решений. Современные графические процессоры обладают большим параллелизмом и производительностью, однако не решают основных проблем обработки графов: зависимостей по данным, распределения нерегулярных графов по процессорным устройствам, конфликтов при доступе к памяти. В МГТУ им. Н. Э. Баумана разработан специализированный микропроцессор Leonhard ×64 с набором команд дискретной математики (DISC), предназначенный в том числе для обработки графов большой размерности. В статье приведены сведения об операциях дискретной математики и соответствующих им машинных инструкциях микропроцессора Leonhard x64, обосновывается архитектура гетерогенной вычислительной системы на его основе. На примерах алгоритмов Дейкстры и Белмана—Форда демонстрируется различие между реализацией обработки графов, а также эффективность универсальных вычислительных систем, гетерогенных вычислительных систем на основе графических ускорителей GPGPU и системы на основе микропроцессора Leonhard ×64.*

***Ключевые слова:*** *набор команд дискретной математики, микропроцессор Leonhard, граф, структура данных, графический ускоритель вычислений*

## Обзор современных аппаратных систем

Потоковая обработка сложных моделей данных, таких как графы, становится все более востребованной в аналитических системах. Ускорение их работы может быть достигнуто благодаря более совершенным методам и алгоритмам, а также благодаря повышению эффективности аппаратного обеспечения. Однако традиционные подходы к проектированию параллельных программ и систем, основанные на применении большого числа однотипных универсальных процессоров, уже не обеспечивают требуемого быстродействия, приводят к существенному росту сложности и времени разработки и отладки программного кода. Вместе с этим, мы покажем, что существующие вычислительные системы, использующие массовый параллелизм, еще далеки от совершенства, а заложенные в них аппаратные механизмы не предназначены для обработки зависимых или сильно фрагментированных в оперативной памяти данных. Рассмотрим далее проблемы обработки алгоритмов дискретной оптимизации в универсальных микропроцессорах (подобных семейству ×86 Intel) или же в современных графических ускорителях GPGPU (например, NVidia Tesla), а также в специальных аппаратных системах (на примере отечественного микропроцессора Leonhard ×64).

Современные вычислительные машины используют универсальные микропроцессоры, которые являются не только основными обрабатывающими устройствами с точки зрения архитектуры ЭВМ, но и определяют специфику разработки и функционирования программ. Базовыми принципами построения таких микропроцессоров, позволяющими добиться высокой производительности исполнения программного кода, являются глубокая конвейеризация микроархитектуры,

спекулятивное исполнение команд и многоуровневая подсистема памяти. Как следствие, в микропроцессорах на разных стадиях одновременно находится большое число команд и операндов, продвигаемых по конвейеру в его исполнительную часть. В случае простых итераций, основанных на хорошо предсказуемых итерационных переменных, микропроцессор способен ритмично поставлять на исполнение затребованные операнды, а длинный конвейер оказывается эффективным. Именно на подобных задачах мы можем наблюдать высокую производительность универсальных микропроцессоров. Проблемы зависимых данных, тем не менее, являются характерной чертой подавляющего большинства алгоритмов на графах. Вследствие таких зависимостей нарушается ритмичная работа конвейера, снижается эффективность аппаратной предвыборки, увеличивается число сбросов конвейера из-за неправильно предсказанных переходов [1—3].

Более того, анализ потока команд показывает [4], что команды арифметической и логической обработки составляют лишь порядка 30 % инструкций, и не менее 50 % потока инструкций составляют команды пересылки. Учитывая, что эти данные получены на микропроцессоре с полным набором инструкций (Complex Instruction Set Computing, CISC), при котором доступ к памяти кодируется в одной команде с операциями обработки, можно констатировать большой процент служебных команд в потоке инструкций и, как следствие, малую эффективность универсального подхода в целом.

Одновременно с этим обработка сложных моделей данных, таких как графы, затруднена в связи с особенностями размещения графов в оперативной памяти. Существенной проблемой универсальных микропроцессоров является фрагментация при сегментно-страничной организации виртуальной памяти, которая приводит к потоку двойных обращений к физической оперативной памяти. В этом случае наблюдается замедление обмена данными между процессором и памятью из-за открытия и закрытия большого числа страниц оперативной памяти, а конвейер простаивает в связи с зависимостями по данным. Принятый для современных типов ОЗУ пакетный режим работы также снижает эффективность системы, так как способствует неполному использованию шины и ресурсов процессора. Можно констатировать, что архитектурные решения, заложенные в современные ЭВМ, направлены на ускорение обработки векторных структур и, напротив, замедляют обработку ссылочных структур данных. Также характерно, что в настоящее время отсутствуют средства, ускоряющие обработку зависимых данных.

Одним из перспективных способов повышения производительности универсальных ЭВМ является применение аппаратных ускорителей вычислений (так называемых акселераторов). Ускорение обработки в этом случае достигается благодаря применению более совершенных и параллельных аппаратных механизмов, независимых от работы центрального процессора и системной памяти, а также за счет высвобождения важных ресурсов системы (системных шин, буферов ввода/вывода и пр.). Одновременная обработка многих потоков данных позволяет снять часть вычислительной нагрузки с универсальных микропроцессоров и перенести их на независимое программное и аппаратное обеспечение ускорителей. Таким образом, положительный технический результат достигается благодаря лучшему сочетанию таких параметров, как цена изделия, производительность, простота применения технологии, энергопотребление.

После 2000 г. стало ясно, что мощность графических ускорителей GPU (Graphics Processing Unit) растет быстрее мощности универсальных микропроцессоров. Однако специфика разработки программ для видеоадаптеров ограничивала развитие технологий ускорения вычислений на графических ядрах. Современные процессоры GPU содержат большое число потоковых мультипроцессоров (SM), обладают высокой вычислительной мощностью и относительно низкой стоимостью. Проект NVidia Tesla развивает идею GPGPU (General-purpose computing on graphics processing units) и реализован в целях получения компактных высокопроизводительных систем, обладающих высокой универсальностью и в большей степени использующих универсальные средства разработки программ на языках C/C + + и Fortran. В настоящий момент число параллельных потоков вычислений в чипе GPGPU достигает нескольких десятков тысяч. Однако для эффективной реализации параллельного алгоритма на GPGPU требуется выполнение таких условий как: представление задачи в виде независимых потоков вычислений; отсутствие зависимостей по данным; возможность представления данных в локальной памяти SM.

Очевидно, что при наличии нескольких тысяч потоковых процессоров в GPGPU эффек-

тивность решения задачи зависит от возможности ее представления в виде тысяч параллельных потоков и пропускной способности подсистемы памяти. При этом требуется не только эффективно распределить данные по многим устройствам SM, но и обеспечить получение и анализ результата вычислений. Как следствие, фаза загрузки/подготовки данных и фаза выгрузки результатов занимают достаточно много времени, так как подготовка к запуску вычислений и завершающий этап вычислений сопровождаются перемещением данных между оперативной памятью CPU и глобальной памятью GPGPU по шине PCIe. Этот процесс требует существенных затрат системного времени и ресурсов из-за достаточно высокой латентности шины и необходимости инициализации механизмов копирования (на основе механизмов прямого доступа к памяти, DMA).

Предпочтительным вариантом представления данных для GPGPU являются матрицы и векторы, которые могут быть легко разделены на части, соответствующие объему кэш-памяти SM. Как следствие, при объемах данных, существенно превышающих размеры L1 кэш SM и L2 чипа GPGPU, скорость работы графических ускорителей резко падает. Такой эффект проявляется при обработке разреженных матриц в связи с непредсказуемым шаблоном доступа к глобальной памяти. При существенном увеличении числа вычислительных ядер в GPGPU скорость вычислений возрастает лишь в 1,1...5 раз [5].

В связи с выявленными недостатками обработки зависимых данных на универсальных микропроцессорах и на графических процессорах GPGPU рассмотрим также альтернативный подход к решению задач дискретной оптимизации на основе специализированного микропроцессора Leonhard ×64.

## Микропроцессор с набором команд дискретной математики

В данной работе мы предлагаем альтернативный подход к решению задач на графах, основанный на применении специализированного микропроцессора Leonhard ×64 с набором команд дискретной математики (Discrete Mathematics Instructions Set Computer, DISC), разработанного в МГТУ им. Н. Э. Баумана. Для разработки принципов применения такого устройства потребовалось внести изменения в основы функционирования и взаимодействия микропроцессоров в универсальных вычислительных системах. Поясним сказанное на основе модели вычислительной системы, рассмотренной в работах В. Г. Хорошевского. Согласно [6] вычислительная система представляет собой разновидность вычислителя:

$$c = \langle U, g, a(p(D)) \rangle, \quad (1)$$

где $U$ — множество устройств, обеспечивающих ввод, обработку, хранение и вывод информации; $g$ — структура связей между устройствами; $a$ — алгоритм работы вычислителя или алгоритм управления вычислительными процессами при реализации программы $p$ обработки данных $D$. В работе [6] отмечается, что для любого вычислителя характерны: последовательная обработка информации; фиксированность структуры; неоднородность составляющих устройств и связей между ними.

Дальнейшее совершенствование такой системы невозможно без выделения специальной вычислительной нагрузки и разработки более эффективных средств для ее выполнения. Микропроцессор Leonhard ×64, в частности, предназначен для аппаратного выполнения набора операций над множествами, структурами данных и графами, которые до этого реализовывались на основе универсальных микропроцессоров. Стоит отметить, что обработка структур данных является основой большого числа прикладных алгоритмов. Ускорение их работы всегда являлось приоритетной задачей исследователей, а большинство новых и более эффективных алгоритмов дискретной оптимизации основаны на более совершенных структурах для представления данных в памяти ЭВМ.

Согласно [7] структура данных определяется как совокупность двух множеств:

$$S = (D, R), \quad (2)$$

где $D$ — множество элементов данных; $R$ — множество отношений между элементами данных.

Так как обработка структур данных предполагает взаимодействие устройств из $U$ через каналы связи $g$ и под управлением алгоритма $a$, то свойства всех указанных компонентов модели влияют на время формирования результата вычислителем. Тогда на основании (2) можно констатировать, что при обработке структур данных должна происходить обработка как отношений $R$, так и скалярных данных $D$. В том случае, если такая обработка выполняется раздельно на различных устройствах, можно

определить уточненную модель вычислителя, выполняющего обработку структур данных:

$$c_S = \langle U_D,\ U_R,\ g,\ a_D(p_D(D)),\ a_R(p_R(R))\rangle, \qquad (3)$$

где $U_D$ — множество устройств обработки скалярных данных; $U_R$ — множество устройств обработки отношений данных; $g$ — структура связей между устройствами; $a_D$ и $a_R$ — алгоритмы управления вычислительными процессами в $U_D$ и $U_R$; $p_D$ — программа обработки данных $D$; $p_R$ — программа обработки отношений $R$. Принято следующее допущение: множество устройств вычислителя $U = U_D \cup U_R$; алгоритм работы вычислителя $a = a_D \cup a_R$; программы обработки $p = p_D \cup p_R$. Это означает, что в вычислительной задаче выделяется часть действий по обработке отношений данных, выполнение которых возлагается на специальные вычислительные устройства. В случае полного разделения вычислителя на две независимые подсистемы — части для обработки данных и отношений — будет выполняться более строгое условие: $U_D \cap U_R = \varnothing$; $a_D \cap a_R = \varnothing$, $p_D \cap p_R = \varnothing$.

Уточненная модель вычислителя позволяет сформулировать принцип декомпозиции архитектуры вычислительной системы, согласно которому устройство обработки отношений имеет доступ к независимой памяти, в которой хранятся множества, структуры данных, графы, а также команды их обработки. Результаты выполнения команд направляются в центральный процессор для дальнейшего использования в ходе вычислительного алгоритма. Таким образом, функциями микропроцессора Leonhard ×64 в вычислительной системе являются хранение и независимая обработка множеств, структур данных и графов.

Для определения набора команд микропроцессора Leonhard ×64 были исследованы операции дискретной математики, а также ряд алгоритмов дискретной оптимизации [8]. В результате были определены основные функции устройства обработки отношений, которые могут быть представлены следующей формальной моделью:

$A = \langle A_1, ..., A_n\rangle$ — функция хранения кортежа $A$ из $n$ множеств;

$R(A_i, x, y)$, $x \in A_i$, $y \in A_i$ — функция определения отношения между элементами $x$ и $y$ множества $A_i$;

$|A_i|$, $i = \overline{1, n}$, — операция определения мощности множества;

$x \in A_i$, $i = \overline{1, n}$, — операция проверки принадлежности элемента $x$ множеству;

$A_i \cup x$, $i = \overline{1, n}$, — операция добавления элемента в множество;

$A_i \setminus x$, $i = \overline{1, n}$, — операция удаления элемента из множества;

$A \setminus A_i$ — операция удаления множества $A_i$ из кортежа $A$;

$A_i \subset A_j$ — операция отношения включения множества $A_i$ в $A_j$;

$A_i \equiv A_j$ — операция отношения эквивалентности;

$A_i \cup A_j$ — операция объединения множеств;

$A_i \cap A_j$ — операция пересечения множеств;

$A_i \setminus A_j$ — операция разности множеств;

$A_i \vartriangle A_j$ — операция симметрической разности;

$\overline{A}$ — операция дополнения $A_i$;

$A_i \times A_j$ — операция декартового произведения;

$2^{Ai}$ — операция определения Булеана.

На основе функциональной модели были разработаны машинные команды процессора обработки структур, которые представляют собой высокоуровневые действия, основанные на операциях дискретной математики [28]. Последняя версия набора команд Leonhard ×64 состоит из 20 высокоуровневых кодов операций, перечисленных ниже:

- **Search (SRCH)** выполняет поиск значения, связанного с ключом.
- **Insert (INS)** вставляет пару ключ—значение в структуру. SPU обновляет значение, если указанный ключ уже находится в структуре.
- Операция **Delete (DEL)** выполняет поиск указанного ключа и удаляет его из структуры данных.
- Последняя версия набора команд Leonhard ×64 была расширена двумя новыми инструкциями (NSM и NGR) для обеспечения требований некоторых алгоритмов. Каждая инструкция набора включает до трех операндов: ключ, значение и номер структуры данных. Команды **NSM/NGR** выполняют поиск соседнего ключа, который меньше (или больше) заданного и возвращает его значение. Операции могут быть использованы для эвристических вычислений, где интерполяция данных используется вместо точных вычислений (например, кластеризация или агрегация).
- **Maximum /minimum (MAX, MIN)** ищут первый или последний ключи в структуре данных.
- Операция **Cardinality (CNT)** определяет число ключей, хранящихся в структуре.
- Команды **AND, OR, NOT** выполняют объединения, пересечения и дополнения в двух структурах данных.

- **Срезы (LS, GR, LSEQ, GREQ)** извлекают подмножество одной структуры данных в другую.
- **Переход к следующему или предыдущему (NEXT, PREV)** находят соседний (следующий или предыдущий) ключ в структуре данных относительно переданного ключа. В связи с тем, что исходный ключ должен обязательно присутствовать в структуре данных, операции NEXT/PREV отличаются от NSM/NGR.
- **Удаление структуры (DELS)** очищает все ресурсы, используемые заданной структурой.
- Команда **Squeeze (SQ)** дефрагментирует блоки памяти DSM, используемые структурой.
- Команда **Jump (JT)** указывает SPU код ветвления, который должен быть синхронизирован с CPU (команда доступна только в режиме MISD).

Далее покажем (табл. 1) соответствие набора команд DISC функциям, операциям и кванторам дискретной математики.

Как следует из табл. 1, в наборе команд DISC остаются не реализованными лишь операция симметрической разности, декартового произведения, а также операция Булеана. Причиной такого ограничения является большая сложность их аппаратной реализации. Также стоит отметить, что перечисленные операции редко требуется применять в алгоритмах дискретной оптимизации, а аппаратные затраты на их реализацию высоки. Кроме того, все указанные операции легко реализуются на основе реализованного подмножества операций:

- симметрическая разность может быть получена с использованием команд NOT и OR ($A \Delta B = (A \setminus B) \cup (B \setminus A)$);
- декартово произведение и Булеан могут быть реализованы на основе последовательного обхода множеств командами NEXT, PREV и INS.

Таблица 1

**Соответствие инструкций DISC функциям, кванторам, операциям дискретной математики**

| Функции, кванторы и операции дискретной математики | Инструкции набора команд DISC |
|---|---|
| Функция хранения кортежа | INS |
| Функция отношения элементов множества | NEXT, PREV, NSM, NGR, MIN, MAX |
| Мощность множества | CNT |
| Функция принадлежности элемента множеству | SRCH |
| Добавление элемента в множество | INS |
| Исключение элемента из множества | DEL, DELS |
| Исключение подмножества из кортежа | DELS |
| Включение подмножества в кортеж | LS, GR, LSEQ, GREQ |
| Отношение эквивалентности множеств | LS, GR, LSEQ, GREQ |
| Объединение множеств | OR |
| Пересечение множеств | AND |
| Разность множеств | NOT |
| Симметрическая разность множеств | Не реализовано |
| Декартово произведение множеств | Не реализовано |
| Булеан множества | Не реализовано |

## Особенности архитектуры гетерогенной вычислительной системы на основе микропроцессора Leonhard x64

Вычислительная система, разработанная в МГТУ им. Н. Э. Баумана, состоит из параллельно работающих микропроцессоров нескольких типов. Функциями центрального процессора (рис. 1) является общее управление вычислительной системой, т. е. загрузка операционной системы, обмен данными с дисковой подсистемой, запуск вычислительных алгоритмов, обмен информацией с внешними устройствами, визуализация результатов вычислений на экран. В качестве центрального процессора (ЦП) в различных модификациях системы были использованы микропроцессоры фирм Intel, AMD, а также отечественный микропроцессор Байкал T1 производства АО Байкал Электроникс.

ЦП соединен с ускорительной частью через высокоскоростную шину PCIe ×8, по которой происходит загрузка команд и операндов в микропроцессор Leonhard ×64, а также обмен данными с локальным арифметическим процессором (ЛАП, Local arithmetic processor). Последний (т. е. ЛАП) размещен на одном кристалле с Leonhard ×64 версии 3.0, исполняет стандартный набор команд арифметической и логической обработки чисел и призван сократить время простоя микропроцессора Leonhard ×64 в ожидании новых команд и операндов.

Стоит отметить, что предыдущая версия микропроцессора Leonhard (версия 2.0) не предусматривала размещение универсального микропроцессора на одном кристалле [8—10].

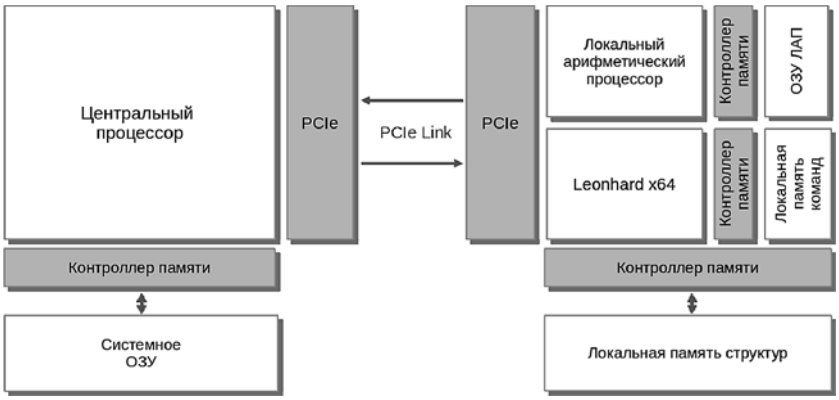



**Рис. 1. Архитектура вычислительной системы на основе микропроцессора Leonhard ×64**

Как следствие, весь обмен данными между вычислительным алгоритмом и ускорительной частью проводился по шине PCIe. Такой обмен характеризуется не только большой пропускной способностью, но и высокой латентностью в связи с необходимостью использования драйверной модели шины и трехуровневой организацией взаимодействия PCI Express. При этом существенное время тратится на формирование пакетов для трех уровней взаимодействия устройств (PCIe поддерживает обмен на уровне транзакций, уровне представления данных и на физическом уровне), а также получение подтверждения от соответствующего уровня "устройства-визави". Проведенные эксперименты показали, что латентность операции записи и последующего чтения, часто используемая при обращении к ускорителям, снижается приблизительно в 8 раз при использовании локальной шины внутри кристалла ПЛИС (табл. 2).

Также нужно учесть, что важной особенностью обмена данными между устройством выполнения команд дискретной математики и устройством выполнения арифметико-логических команд являются малые объемы передаваемых данных и зависимости по данным. Объяснить это можно тем, что большинство действий над множествами в алгоритмах дискретной математики (например, квантор существования, операция добавления элемента в множество и др.) используют простые числовые операнды, а не большие последовательности чисел.

В результате было выявлено, что использование шины PCIe существенно замедляет работу вычислительных алгоритмов дискретной оптимизации, так как не обеспечивает низкой латентности выполнения транзакций. Выходом из этого положения может быть использование более высокоскоростных аппаратно-управляемых интерфейсов (управление транзакциями не использует программных драйверов). Поэтому было принято решение поместить универсальный микропроцессор на один кристалл с микропроцессором Leonhard ×64 и использовать там один из локальных типов шин, предназначенных для подключения ускорителей. В качестве локального арифметического процессора были использованы синтезируемые микропроцессорные ядра Microblaze, OpenRISC и RISCV. В результате скорость выполнения синхронных операций, таких как поиск (SRCH) и некоторых других, увеличилась до 10,5 раз.

Таблица 2

**Сравнение латентности операций обмена данными с ускорителем вычислений (синхронная запись и чтение регистра данных)**

| Конфигурация системы и вариант реализации взаимодействия | Латентность, нс | Число тактов ЦПУ (1,9 ГГц) | Число тактов ускорителя (100 МГц) |
|---|---|---|---|
| ЦПУ Intel ×86 1.9 ГГц, шина PCIe × 8 версия 1.0 2.5 ГГц, частота интерфейсной части ускорителя 100 МГц | 1750 | 3300 | 175 |
| Локальный арифметический процессор Microblaze v5 100 МГц, шина FSL 32 бит 100 МГц, частота интерфейсной части ускорителя 100 МГц | 220 | 418 | 22 |

## Сравнение архитектурной эффективности вычислительных систем при обработке графов

Для экспериментального обоснования принятых архитектурных решений было проведено сравнение числа тактов работы вычислительной системы с использованием и без использования ускорителя операций дискретной математики, а также для различных вариантов взаимодействия ускорителя и центрального процессора. Сравнение аппаратной эффективности приведенных типов систем было рассчитано по результатам тестов на классическом алгоритме Дейкстры и алгоритме Беллмана—Форда поиска кратчайших путей на графах (Single Shortest Path Problem, SSSP). Реализации указанных алгоритмов

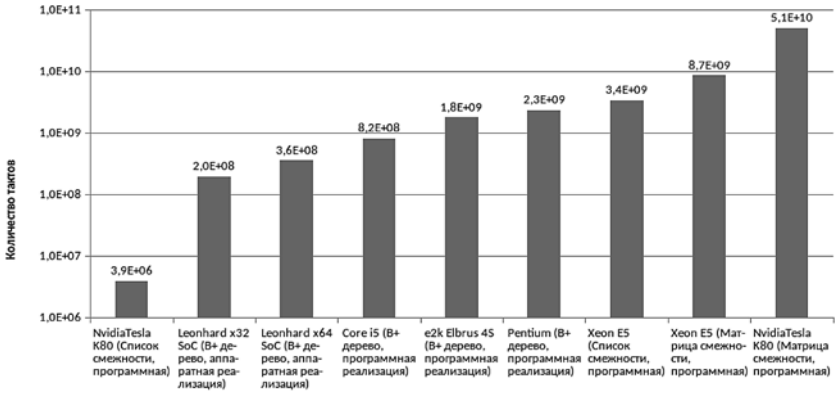



**Рис. 2. Результаты сравнения производительности универсальных микропроцессоров (Intel Xeon E5, Intel Core i5, Elbrus e2k), графического ускорителя Nvidia Tesla K80 и микропроцессора Leonhard ×64 на примере задачи SSSP (алгоритм Дейкстры, граф решетки содержит 16 K вершин)**

для вычислительной системы с аппаратной поддержкой операций дискретной математики были представлены в работах [12, 13].

При этом было выявлено, что эффективность решения задач существенно зависит как от формата представления графа в локальной памяти микропроцессора, так и используемого алгоритма (рис. 2). Традиционный вариант представления графа в виде матрицы смежности [7] оказывается малопригодным для реализации на GPGPU в связи с сильной разреженностью данных. В связи с этим производительность графических ускорителей при таком представлении графа оказывается существенно ниже в сравнении с универсальными микропроцессорами. В других вариантах тестов было использовано представление графа в виде сжатого списка смежности (так называемый формат CSR, Compressed Sparse Row Format), который обеспечивает высокую параллельность работы алгоритма Беллмана—Форда.

Большинство команд Leonhard ×64 требуют $O(\log_8 n)$ операций доступа к памяти (исключая операции AND/OR/NOT и операции срезов). Однако в связи с аппаратной реализацией механизмов обработки B + деревьев команды микропроцессора Leonhard ×64 выполняются за меньшее число тактов по сравнению с универсальными процессорами. В представленных экспериментах был использован микропроцессор Leonhard ×64, реализованный на основе ПЛИС Virtex 6 со встроенным локальным микропроцессором Microblaze. Мы использовали третью версию Leonhard ×64 со следующими параметрами: 64-битные ключи и значения; максимальное число ключей в памяти структур — 100 млн (100 663 296 записей); максимальное число структур данных — 7; 4 ГБайт объем локальной памяти структур; рабочая частота Leonhard ×64 100 МГц.

Казалось бы, результаты экспериментов однозначно демонстрируют преимущества многоядерных систем на основе графических ускорителей: число тактов для решения задачи оказывается в ~ $10^2$ раз меньше в сравнении с микропроцессором Leonhard и в $10^3$ меньше Intel Xeon E5. Однако стоит также принимать во внимание аппаратную сложность микропроцессоров, выраженную в количестве ресурсов кристалла (транзисторов или вентилей), необходимых для их реализации. Например, аппаратная сложность GPGPU NVidia Tesla K80 составляет порядка 7,1 млрд транзисторов, в то время как аппаратная сложность микропроцессора Leonhard ×64 составляет порядка 1 млн вентилей вместе с микропроцессорными ядрами PowerPC или Microblaze. Для расчета архитектурной эффективности различных вычислительных платформ будем использовать следующую формулу:

$$E_{HW} = \frac{10^6}{TICK \cdot GATE},$$

где $TICK$ — число тактов работы алгоритма; $GATE$ — число млн вентилей, необходимых для реализации микропроцессора.

В результате сравнения оказалось, что аппаратная эффективность графических ускорителей при решении задач на графах в $10^2$ раз ниже специализированного микропроцессора Leonhard ×64 (рис. 3) и Leonhard ×32. Также показано, что более простые по структуре микропроцессоры (отечественный VLIW микропроцессор Elbrus e2k, Intel Pentium IV) обладают большей эффективностью в расчете на один вентиль в сравнении с современными многоядерными микропроцессорами Intel.

Полученные результаты показывают, что высокий уровень производительности современных многоядерных микропроцессоров и ускорителей GPGPU достигается благодаря их высокой параллельности и аппаратной сложности. При этом условный вклад в прирост производительности одного транзистора кристалла оказывается существенно ниже, чем у предшествующих моделей или менее параллельных микропроцессоров. При этом следует учитывать объективные ограничения крем-

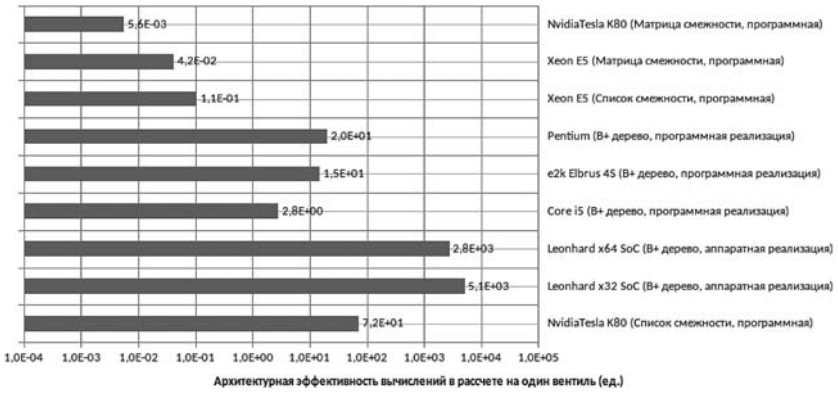



**Рис. 3. Сравнение архитектурной эффективности вычислительных систем в расчете на один вентиль**

ниевой технологии, влияющие на дальнейшее сохранение указанной тенденции. Размеры кристалла ограничены, а топология критически важных участков заказных СБИС определяется ручной трассировкой, что увеличивает трудозатраты на создание схемной топологии. В результате стоимость изделия зависит от площади кристалла и процента дефектных микросхем [14]. Можно утверждать, что дальнейшее наращивание числа ядер в микропроцессорах приведет к снижению удельной эффективности транзисторов на кристалле.

## Выводы и варианты внедрения системы

Проведенные эксперименты и сравнение аппаратной эффективности различных микропроцессоров и ускорителей показывают необходимость разработки и внедрения специальных средств ускорения алгоритмов на графах, несмотря на существенные достижения разработчиков графических ускорителей GPGPU. Вследствие специфики задач обработки графов даже при использовании оптимизированных алгоритмов и структур данных аппаратная эффективность GPGPU и CPU, выраженная в тактах или рассчитанная на один транзистор, оказывается существенно ниже эффективности специальных ускорителей и микропроцессора Leonhard ×64 в частности.

Мы рассматриваем несколько важных областей, в которых применение процессора Leonhard должно улучшить производительность и энергопотребление вычислительных систем.

В функции контроллера программно-определяемых сетей (SDN controller) входит сбор данных по протоколу OpenFlow от многих виртуальных или физических контроллеров сети. Большинство алгоритмов управления сетями строятся на основе графовых моделей и дискретной оптимизации. При этом ключевой характеристикой эффективности контроллера SDN является время его отклика и пропускная способность. Вместе с тем растут требования по аналитической обработке сетевых транзакций. Уже сейчас применяются средства машинного обучения для прогнозирования изменения траффика и предиктивного управления сетевыми ресурсами. Также требуется выполнять анализ траффика на предмет подозрительной активности приложений, сетевых атак и других угроз информационной безопасности. В связи с этим аппаратная поддержка SDN-контроллеров на основе микропроцессора Leonhard является актуальной.

Другим направлением внедрения микропроцессора Leonhard являются робототехнические системы. По мере продвижения по маршруту (или заранее) робот строит граф видимости, в котором ребрами соединены видимые точки пространства. Вес ребра может отражать энергетические затраты, которые необходимы для перехода робота из одного положения в другое. В связи с этим требуется решать задачу поиска кратчайшего пути на графе видимости для эффективного перемещения робота.

Микропроцессор Leonhard был использован для реализации алгоритмов машинного обучения и показал хорошие результаты производительности и точности на задаче классификации телеметрических данных космического аппарата (достигнута точность распознавания 99,7 %). Это позволяет использовать Leonhard в системах управления сложными системами. В настоящее время развивается проект управления беспилотными летательными аппаратами и аппаратный автопилот. Также развивается проект применения Leonhard в системах компьютерного зрения, в которых необходимо не только распознавать видимые предметы, но и управлять роботом на основе анализа сцены.

### Список литературы

1. **Patel R., Kumar S.** Visualizing effect of dependency in superscalar pipelining // 2018 4th International Conference on Recent Advances in Information Technology (RAIT). Dhanbad. 2018. P. 1—5.

2. **Patel G. R., Kumar S.** The Effect of Dependency on Scalar Pipeline Architecture // IUP Journal of Computer Sciences. Jan 2017. Vol. 11, N. 1. P. 38—50.

3. **Harish P., Narayanan P. J.** Accelerating Large Graph Algorithms on the GPU Using CUDA // High Performance Com-

puting — HiPC 2007. HiPC 2007. Lecture Notes in Computer Science. Vol. 4873. Springer, Berlin, Heidelberg

4. **Huang I. J., Peng T. C.** Analysis of ×86 instruction set usage for DOS / Windows applications and its implication on superscalar design // IEICE Transactions on Information and Systems. 2002. Vol. E85—D, N. 6. P. 929—939.

5. **Kasmi N., Mahmoudi S. A., Zbakh M., Manneback P.** Performance evaluation of sparse matrix-vector product (SpMV) computation on GPU architecture // 2014 Second World Conference on Complex Systems (WCCS), Agadir. 2014. P. 23—27.

6. **Хорошевский В. Г.** Архитектура вычислительных систем: Учеб. пособ. Москва: Изд-во МГТУ им. Н. Э. Баумана, 2008. С. 520.

7. **Кормен Т., Лейзерсон Ч., Ривест Р.** Алгоритмы: построение и анализ. М.: МЦНМО, 2000. 960 с.

8. **Попов А. Ю.** О реализации алгоритма Форда-Фалкерсона в вычислительной системе с многими потоками команд и одним потоком данных // Наука и образование. МГТУ им. Н. Э. Баумана. Электрон. журн. 2014. № 9. С. 162—180.

9. **Popov A.** An Introduction to the MISD Technology // HICSS50. Proceedings of the 50th Hawaii International Conference on System Sciences. 2017. P. 1003—1012.

10. **Abdymanapov C., Popov A.** Motion Planning Algorithms Using DISC // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 2019. P. 1844—1847.

11. **Rasheed B., Popov A. Y.** Network Graph Datastore Using DISC Processor // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 2019. N. 1. P. 1582—1587.

12. **Попов А. Ю.** Применение вычислительных систем с многими потоками команд и одним потоком данных для решения задач оптимизации // Инженерный журнал: наука и инновации (электронное издание). 2012. № 1. URL: http://engjournal.ru/catalog/it/hidden/80.html (дата обр. 27.05.2019).

13. **Подольский В. Э.** Об организации параллельной работы некоторых алгоритмов поиска кратчайшего пути на графе в вычислительной системе с многими потоками команд и одним потоком данных // Наука и Образование. Электронный журнал. 2015. № 4.

14. **Sharma P., Rehal M., Bangotra P. K.** Defect Data Classification and Analysis in VLSI Fabrication // 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore. 2018. P. 1447—1453.

**A. Yu. Popov,** Ph. D., Associate Professor of the Department "Computer Systems and Networks",
e-mail: alexpopov@bmstu.ru,
Bauman Moscow State Technical University, Moscow

# Application of a Heterogeneous Computing System with a Discrete Mathematics Instruction Set to Solve Large-Scale Graphs Problems

*The interest to the complex data models analysis has grown significantly in the last decade. Graphs known as the most adequate form of the real data representing in many areas from social networks, computer programs structure, topology of integrated circuits, to the bioinformatics and many others. As the size of datasets increases the need to find more efficient methods and tools for analyzing large graphs, including on the basis of more advanced hardware technical solutions, becomes obvious. Modern GPUs have great parallelism and performance, but cannot solve the fundamental problems of the graph processing: the data dependencies problem, the distribution of irregular graphs computing workload on multiple processor devices, as well as conflicts with memory access for many cores. We introduce a specialized microprocessor Leonhard x64 and the Discrete mathematics Instruction Set Computer (DISC) for large dimension graph processing, which was developed in the Bauman University. This article provides information about the discrete mathematics operators and the corresponding Leonhard x64 instructions, and justifies the architecture principles of a heterogeneous computing system based on it. With examples of Dijkstra and Belman-Ford algorithms we show the difference between the implementation of graph processing and the efficiency of universal computing systems, heterogeneous computing systems based on GPGPU, as well as Leonhard x64 microprocessor-based system.*

***Keywords:*** *discrete mathematics instruction set computer, Leonhard microprocessor, graph, data structure, GPGPU*

DOI: 10.17587/it.25.682-690

**References**

1. **Patel R., Kumar S.** *2018 4th International Conference on Information Technology (RAIT)*, Dhanbad, 2018, pp. 1—5.

2. **Patel G. R., Kumar S.** *IUP Journal of Computer Sciences*, Jan 2017, vol. 11, no. 1, pp. 38—50.

3. **Harish P., Narayanan P. J.** (2007) Accelerating Large Graph Algorithms on the GPU Using CUDA, *High Performance Computing — HiPC 2007. Lecture Notes in Computer Science,* vol. 4873, Springer, Berlin, Heidelberg.

4. **Huang I. J., Peng T. C.** *I. J. Trans.*, 2002, vol. E85—D, no. 6, p. 929—939.

5. **Kasmi N., Mahmoudi S. A., Zbakh M., Manneback P.** *2014 Second World Conference on Complex Systems (WCCS),* Agadir, 2014, pp. 23—27.

6. **Khoroshevsky B. G.** Architecture of computing systems: Proc. Allowance, Moscow, Publishing House of Moscow State Technical University H. E. Bauman, 2008, 520 p. (in Russian).

7. **Kormen T., Leyzerson Ch., Rivest R.** Algorithms: construction and analysis, Moscow, MTSNMO, 2000, 960 p. (in Russian).

8. **Popov A. Yu.** *Science and Education. MGTU them. N. E. Bauman. Electron. Journals*, 2014, no. 9, pp. 162-180 (in Russian).

9. **Popov A.** *HICSS50. Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017, pp. 1003—1012.

10. **Abdymanapov C., Popov A.** *2019 IEEE Conference of Young Researchers in Electrical and Electronic Engineering (EIConRus),* 2019, no. 1, pp. 1844—1847.

11. **Rasheed B., Popov A. Y.** *2019 IEEE Conference of Young Researchers in Electrical and Electronic Engineering (EIConRus),* 2019, no. 1, pp. 1582—1587.

12. **Popov A. Yu.** *Engineering Journal: Science and Innovations (electronic edition),* 2012, no. 1, available at: http://engjournal.ru/catalog/it/hidden/80.html (date sample 05/27/2019) (in Russian).

13. **Podolsky V. E.** *Science and Education. Electronic journal*, 2015, no. 4 (in Russian).

14. **Sharma P., Rehal M., Bangotra P. K.** 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, 2018, pp. 1447—1453.

УДК 004.89 DOI: 10.17587/it.25.691-697

**А. Б. Менисов,** канд. техн. наук, e-mail: men.arty@yandex.ru,
**И. А. Шастун,** канд. техн. наук, e-mail: shastunivan1982@gmail.com,
Военно-космическая академия имени А. Ф. Можайского,
**С. Ю. Капицын,** канд. техн. наук, доц., e-mail: wolf76@inbox.ru,
Военная академия Генерального Штаба ВС РФ

# Подход к выявлению вредоносных сайтов сети Интернет на основе обработки лексических признаков адресов (URL) и усредненного ансамбля моделей

*В настоящее время выявление и блокирование доступа к вредоносным сайтам сети Интернет выполняется, в основном, путем включения URL-адресов в черные списки. Однако черные списки не могут быть исчерпывающими, и с их помощью нет возможности выявлять вновь созданные вредоносные сайты. Целью статьи является разработка нового подхода для выявления вредоносных сайтов на основе усредненного ансамбля моделей, позволяющего улучшить точность выявления. В разработанном подходе по сравнению с современными техническими решениями в сфере информационной безопасности учтены лексические признаки адресной строки (URL) вредоносных сайтов, которую злоумышленники пытаются видоизменить под адрес известного или безопасного сайта.*

*Также авторы приводят сравнение результатов выявления вредоносных URL-адресов, полученных современными и разработанными подходами. Статья будет полезна не только исследователям в области машинного обучения, но и специалистам в отрасли кибербезопасности.*

***Ключевые слова:*** *информационная безопасность, вредоносные веб-сайты, машинное обучение*

## Введение

Технологические достижения современного общества сочетаются с новыми угрозами информационной безопасности, такими как вредоносные сайты, распространяющие негативный и экстремистский контент, сайты, через которые реализуются такие атаки, как взлом, социальная инженерия, фишинг, человек посередине, SQL-инъекции, DOS- и DDOS-атаки и многие другие [1], которые в конечном итоге приводят к краже личных данных или даже к установке вредоносных программ в корпоративные системы.

С учетом разнообразия типов атак, проводимых с помощью вредоносных сайтов, становится сложно спроектировать надежные системы для обнаружения нарушений информационной безопасности. Например, большинство фишинговых атак реализуются путем распространения URL-адресов вредоносных сайтов (или распространение таких URL-адресов является одним из этапов атаки) [2].

URL-адрес (Uniform Resource Locator) является глобальным адресом документов и других ресурсов в сети Интернет [3]. URL имеет два основных компонента: 1) идентификатор протокола, который указывает, какой протокол использовать; 2) имя ресурса, указывающего IP-адрес или имя домена, где расположен ресурс.

URL-адреса, используемые для атак, в рамках данной статьи будем называть вредоносными URL-адресами. Отметим, что около трети всех веб-сайтов являются потенциально вредоносными [4].

Наиболее распространенным методом обнаружения вредоносных URL-адресов является метод включения в черные списки. Черные списки — это база URL-адресов, которые в прошлом были определены как вредоносные. Эта база данных формируется с течением времени (часто с помощью краудсорсинговых решений, например, PhishTank [5]). Такой подход очень прост в реализации в связи с простым формированием запросов к базе данных и имеет очень низкий уровень ложноположи-

тельных результатов [6]. Однако невозможно оперативно поддерживать исчерпывающий список вредоносных URL-адресов, тем более что новые URL-адреса создаются каждый день с высокой интенсивностью. Учитывая невозможность пополнения черного списка URL-адресами, сформированными сервисами сокращения ссылок, и с целью замаскировать вредоносный URL-адрес злоумышленники часто изменяют URL-адреса следующими путями [7, 8]: обфускацией хоста IP-адресом, обфускацией хоста другим доменом, обфускацией хоста большим именем и использованием неправильного написания.

Таким образом, метод занесения в черный список имеет серьезные ограничения и не эффективен для выявления вновь созданных URL-адресов. Чтобы преодолеть эти проблемы, в последнее десятилетие исследователи применяли методы машинного обучения для выявления вредоносных URL-адресов [7, 9—16].

## 1. Постановка задачи

Задано множество URL-адресов $U$, множество их состояний $Y = \{0, 1\}$, и существует целевая функция $y^* : U \to Y$, значения которой $y_i = y^*(u_i)$ известны только на конечном подмножестве объектов $\{u_1, ..., u_n\} \subset U$, причем при $y_i = 1$ сайт является вредоносным, $y_i = 0$ — безопасным. Пары "объект—состояние" $(u_i, y_i)$ являются прецедентами. Совокупность пар прецедентов $U^l = (u_i, y_i)_{i=1}^l$ является обучающей выборкой для восстановления зависимости $y^*$.

Задача выявления вредоносных URL-адресов заключается в том, чтобы построить решающую функцию $a : U \to Y$, которая приближала бы целевую функцию $y^*(u)$, причем не только на объектах обучающей выборки, но и на всем множестве $U$, т. е. была бы способной классифицировать с точки зрения вредоносности произвольный URL-адрес $u \in U$. Вероятность правильной классификации и вероятности ошибок задают средний риск ошибочного выявления вредоносных URL:

$$П = M[C] = p_0 \cdot 0 + p_1 C_1 + ... + p_i C_i, \qquad (1)$$

где $C$ — множество ошибок выявления, $\langle C_1, ..., C_i \rangle$ — ошибки выявления, $i$ — число классов состояний URL-адресов, $p_0$ — вероятность правильного решения, $\langle p_1, ..., p_i \rangle$ — вероятность ошибок.

Таким образом, задача выявления вредоносных URL-адресов по двум классам (безопасные и вредоносные URL-адреса) заключается в следующем:

$$П = M[C] = p_0 \cdot 0 + p_1 C_1 + ... + p_2 C_2 \to \min, \qquad (2)$$

где $p_1$ — вероятность ошибки первого рода (когда ошибочно классифицирован безопасный URL-адрес как вредоносный), а $p_2$ — вероятность ошибки второго рода (когда происходит пропуск вредоносного URL-адреса).

## 2. Лексические признаки вредоносных URL-адресов

Практически в любой задаче классификации возникают вопросы: какие признаки использовать, а какие нет; нужно ли как-то преобразовывать исходные признаки. Для задачи выявления вредоносных URL-адресов вопрос извлечения лексических признаков из адресной строки (URL) является наиболее актуальным.

Признак $f$ URL-адреса $u$ — это результат измерения некоторой характеристики URL-адреса. Формально признаком называется отображение $f: U \to D_f$, где $D_f$ — множество допустимых значений признака. Лексические признаки — это результат анализа строки URL, которая состоит из совокупности знаков, слов, взаимосвязи слов, характеризующих принадлежность к вредоносному или безопасному URL-адресу.

Наиболее часто используемые лексические признаки включают статистические параметры строки URL, такие как длина URL, длина каждого из компонентов URL (имени хоста, домена верхнего уровня, основного домена и т. д.) [17].

В зависимости от того, что содержит строка URL, существует возможность идентифицировать вредоносную природу URL-адреса. Например, существует метод скрытия, когда вредоносные URL-адреса пытаются выдать за безопасные, имитируя их имена и добавляя к ним незначительные дополнения в строке URL, или злоумышленники добавляют схожие слова в разные части домена.

Для расширения признаков в работе использовались значения TF-IDF имени домена и поддоменов.

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса [30]. Вес некоторого слова $z_{i,j} \in Z$ пропорционален частоте употребления $i$-го слова в $j$-м документе и обратно пропорционален частоте употребления $i$-го

слова во всех документах коллекции и вычисляется как

$$z_{i,j} = TF_{i,j} \cdot \log\left(\frac{D}{d_i}\right), \qquad (3)$$

где $TF_{i,j}$ (term frequency — частота $i$-го слова $0 \leqslant i \leqslant |Z|$) в $j$-м документе) — отношение числа вхождений некоторого слова к общему числу слов документа, и $\log\left(\frac{D}{d_i}\right)$ — частотная характеристика, с которой некоторое $i$-е слово встречается в $d_i$ документах коллекции мощностью $D$.

Такой подход уменьшает вес широкоупотребительных слов. Это актуально, так как все домены можно разделить по географической и специальной принадлежности, и для задачи выявления этот признак является малоинформативным.

Таким образом, извлечение лексических признаков из адресной строки (URL) состоит из нижеописанной последовательности действий.

*Шаг 1.* Подсчет числа $f_1$ разделителей (точек) в доменных и поддоменных частях URL-адреса. Является количественным признаком: $f_1 \in \mathbb{R}$.

*Шаг 2.* Подсчет числа $f_2$ других разделителей (';' ,'_', '?', ' = ', '&'). Является количественным признаком: $f_2 \in \mathbb{R}$.

*Шаг 3.* Проверка кодирования URL-адреса. Если в качестве альтернативы имени домена в URL-адресе используется IP-адрес, например, "http://125.98.3.123/fake.html", то это информативный признак вредоносного сайта. Иногда IP-адрес даже преобразуется в шестнадцатеричный код, как показано в следующей ссылке: "http: //0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html". Является бинарным признаком: $f_3 = \{0, 1\}$.

*Шаг 4.* Проверка на использование символа "@" в URL заставляет браузер игнорировать все, что предшествует символу "@", а реальный адрес часто следует за символом "@". Является бинарным признаком: $f_4 = \{0, 1\}$.

*Шаг 5.* Проверка на наличие символа "//" в пути URL означает, что пользователь будет перенаправлен на другой веб-сайт. Пример таких URL-адресов: "http://www.legitimate.com//http://www.phishing.com". Является бинарным признаком: $f_5 = \{0, 1\}$.

*Шаг 6.* Подсчет длины $f_6$ составляющих URL-адресов. Является количественным признаком: $f_6 \in \mathbb{R}$.

*Шаг 7.* Вычисление TF-IDF ключевых слов доменов и поддоменов. Являются количественными признаками: $\langle f_7, ..., f_m \rangle \in \mathbb{R}$, где $m$ — число частей адресной строки (URL).

***Пример.*** Если URL-адрес содержит 10 слов, и слово "fake" встречается в нем два раза, то частота слова для слова "fake" в документе будет 0,2 (2/10). Вычислим логарифм отношения числа всех URL-адресов к числу URL-адресов, содержащих слово "fake". Таким образом, если "fake" содержится в 1000 документах из 10 000 000 документов, то IDF будет равной: log(10 000 000/1000) = 4. Для расчета окончательного значения веса слова необходимо TF умножить на IDF. В данном примере TF-IDF-вес для слова "fake" в выбранном документе будет равен: $0{,}2 \cdot 4 = 0{,}8$. Большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

Таким образом, полученный вектор признаков $\langle f_1, ..., f_7, ..., f_m \rangle$ является признаковым описанием URL-адреса $u \in U$. Совокупность признаковых описаний всех объектов выборки $U^l$, записанная в виде таблицы размера $l \times m$, является матрицей объектов-признаков:

$$F = \|f_j(u_i)\|_{l \times m} = \begin{pmatrix} f_1(u_1) & \ldots & f_m(u_1) \\ \ldots & \ldots & \ldots \\ f_1(u_l) & \ldots & f_m(u_l) \end{pmatrix}. \qquad (4)$$

Для множества сайтов значения $m$ различны, и это накладывает определенные ограничения на выбор моделей для обучения.

## 3. Описание подхода выявления вредоносных URL-адресов на основе усредненного ансамбля моделей

Процесс выявления вредоносных URL-адресов следует разделить на следующие этапы:

*Этап 1.* Сбор данных: эта фаза направлена на сбор всей доступной информации о URL-адресе, такой как наличие URL-адресов в черном списке; явные признаки URL-адреса, такие как строка URL и информация о хосте; контент веб-сайта, такой как HTML и JavaScript; информация о доступности и т. д.

*Этап 2.* Предварительная обработка данных: на этом этапе неструктурированная информация о URL-адресе (например, текстовое описание) должна быть преобразована в матрицу объектов-признаков.

*Этап 3.* Выбор и обучение моделей.

В современной литературе упоминается большое число отдельных алгоритмов машин-

| № п/п | Название модели | Организация | Страна разработки | Год разработки | Значение СКО |
|---|---|---|---|---|---|
| 1 | Gradient Boosting | INRIA | Франция | 2010 | 0,2730239 |
| 2 | XGBoost | Вашингтонский университет | США | 2014 | 0,2693472 |
| 3 | LightGBM | Microsoft | США | 2016 | 0,2756339 |
| 4 | Catboost | Яндекс | РФ | 2017 | 0,2675411 |
| 5 | **Усредненный ансамбль** | **ВКА имени А. Ф. Можайского** | **РФ** | **2019** | **0,2674244** |

ного обучения, которые могут быть применены для выявления вредоносных URL-адресов [21—26].

Однако проблема применения матрицы объектов-признаков, отображающей состояние URL-адреса, заключается в том, что число признаков может быть не фиксировано или не известно заранее. Например, можно рассматривать каждое слово как отдельный признак для текстового обозначения домена и всех поддоменов URL-адреса, который мог встречаться в данных для обучения. Модель выявления вредоносных URL-адресов может быть обучена на этих данных, но при работе с новыми URL-адресами, которые могут содержать слова, которых не было в обучающих данных, точность выявления будет снижаться.

Моделью выявления вредоносных URL-адресов называется параметрическое семейство отображений $A = \{g(u, \theta)|\theta \in \Theta\}$, где $g$: $U \times \Theta \to Y$ — некоторая фиксированная функция, $\Theta$ — множество допустимых значений параметра $\theta$. Для задачи с $m$ признаками $f_j$: $U \to \mathbb{R}$, $j = 1, ..., m$, используются модели с вектором параметров $\theta = (\theta_1, ..., \theta_m) \in \Theta = \mathbb{R}^m$:

$$g(u, \theta) = \text{sign} \sum_{j=1}^{m} \theta_j f_j(u). \quad (5)$$

Так, для повышения точности выявления вредоносных URL-адресов целесообразно объединить несколько алгоритмов в ансамбль моделей, чтобы повысить точность классификатора и уменьшить дисперсию его работы [27].

Усреднение ансамбля моделей — это отображение $\mu$: $(U \times Y)^l \to A$, которое произвольной конечной выборке $U^l = (u_i, y_i)_{i=1}^l$ ставит в соответствие алгоритм:

$$g_{\text{ср}} = \frac{1}{N} \sum_{k=1}^{N} g_k, \quad g_k \in A. \quad (6)$$

Создание усредненного ансамбля моделей заключается в следующем:

1. Разработка $N$ моделей, каждая со своими значениями точности выявления.

2. Обучение каждой модели отдельно.

3. Объединение моделей и усреднение значений в конечном классификаторе.

Преимущество такого подхода заключается в том, что ансамбль моделей может быть легче обучен на небольших входных наборах данных [28] и часто повышает точность по любому отдельному алгоритму [29].

Чтобы подход имел эффективную программную реализацию, выбор моделей основывался на успешном применении моделей на основе градиентного бустинга (LightGBM [32], Gradient Boosting [33], XGBoost [34] и CatBoost [35]) в соревнованиях по машинному обучению [31].

Для измерения эффективности отдельных моделей и разработанного метода будем использовать метрику среднеквадратичной ошибки, которая рассчитывается как

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - y_i^*)^2}, \quad (7)$$

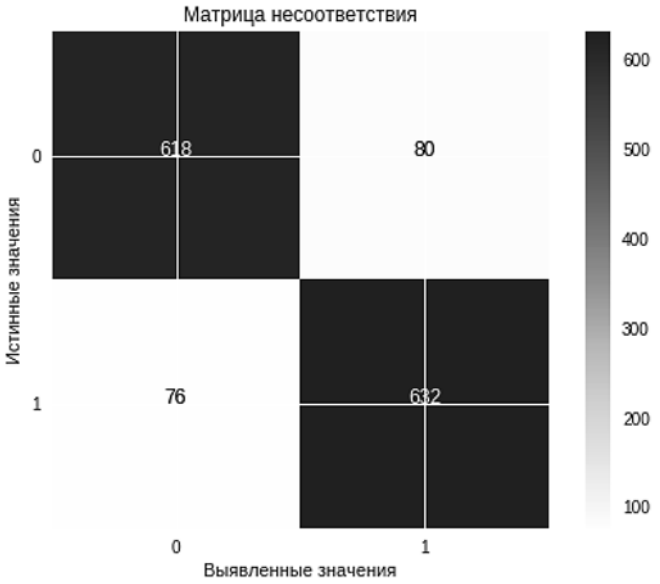
где $y_i$ — результат выявления состояния URL-адреса, а $y_i^*$ — истинное значение. Результаты измерений представлены в таблице.

*Этап 4.* Определение состояния URL-адреса.

## 4. Оценивание эффективности выявления вредоносных URL-адресов

Для определения качества разработанной методики будем использовать набор данных [5], который содержит 7030 записей URL-адресов и их значений, включающих 3494 вредоносных и 3536 безопасных URL-адресов. Валидационную часть определим равной 0,2 от общего набора данных, что включает 696 вредоносных и 712 безопасных URL-адресов.

Для оценивания качества выходных данных применим матрицу несоответствия выявления вредоносных URL-адресов, представленную на рисунке.




**Матрица несоответствия выявления вредоносных URL-адресов**

Матрица несоответствия отображает число верных и ошибочных выявлений по сравнению с фактическими данными. Рассмотрим полученные результаты:

- (0, 0) — правильно выявлены состояния не вредоносных URL-адресов;
- (1, 1) — правильно выявлены состояния вредоносных URL-адресов;
- (0, 1) — URL-адреса безопасны, а принято решение о том, что они вредоносны;
- (1, 0) — URL-адреса вредоносны, а принято решение о том, что они безопасны.

Эти вероятности первого и второго рода можно вычислить как вероятность попадания случайной величины $y_i$ в область допустимых значений классов состояний URL-адресов, т. е. $p_1 = P(0, 1)$ и $p_2 = P(1, 0)$. Подставив эти значения из матрицы несоответствия в формулу (2), получим

$$П = M[C] = \frac{80}{696} \cdot 80 + \frac{76}{712} \cdot 76 \approx 17,3.$$

Сравним средний риск рандомного определения состояний (т. е. с вероятностью 0,5) URL-адресов:

$$П_{\text{ранд}} = M_{\text{ранд}}[C_{\text{ранд}}] = \\ = \frac{348}{696} \cdot 348 + \frac{356}{712} \cdot 356 \approx 350.$$

Таким образом, разработанный ансамбль моделей показал повышение качества выявления вредоносных URL-адресов.

## Заключение

Разработка новых подходов к повышению защищенности компаний и пользователей информационных web-систем является постоянной и актуальной задачей.

Элементом новизны разработанного подхода является предложение по применению признаков, определяющих лексическую природу вредоносного URL-адреса. Особенностью данного подхода является учет возрастающей схожести составных частей URL-адресов у злоумышленников, что дополняет существующие подходы по явному выявлению вредоносных сайтов и способствует более точному выявлению вновь созданных в целях их дальнейшего блокирования.

Практическая значимость заключается в возможности применения подхода при обосновании и разработке технических решений информационной безопасности.

Разработанный подход к выявлению вредоносных URL-адресов на основе лексической обработки URL-адресов и усредненного ансамбля моделей имеет преимущество перед современными алгоритмами машинного обучения при выявлении информационных признаков и позволяет повысить качество выявления вредоносных URL-адресов.

Но несмотря на многообещающую способность подходов машинного обучения одним из основных недостатков разработанного подхода выявления вредоносных URL-адресов может быть его ресурсоемкий характер (особенно при извлечении признаков), что снижает их практическую ценность по сравнению с методами внесения в черный список.

Таким образом, дальнейшим развитием исследований может быть:

- исследование вопросов сбора дополнительных данных об URL-адресах;
- исследование вопросов влияния дисбаланса данных для обучения моделей;
- исследование вопросов повышения производительности выявления вредоносных URL-адресов;
- разработка технических решений безопасности почтовых сервисов разных типов.

### Список литературы

1. **Sahoo D., Liu C., Steven C.** Malicious URL Detection using Machine Learning // A Survey. 2017. URL: https://arxiv.org/pdf/1701.07179.pdf

2. **Hong J.** The state of phishing attacks // Communications of the ACM. 2012. Vol. 55, N. 1. P. 74—81.

3. **Berners-Lee T.** Uniform Resource Locators (URL): A Syntax for the Expression of Access Information of Objects on the Network // World Wide Web Consortium. 2015.

4. **Liang B., Huang J., Liu F., Wang D., Dong D., Liang Z.** Malicious web pages detection based on abnormal visibility recognition // E-Business and Information System Security: International Conference on. IEEE. 2009. P. 1—5.

5. **Данные** вредоносных сайтов компании Phishtank. URL: https://www.phishtank.com/ (дата обращения 26.01.2019).

6. **Sinha S., Bailey M., Jahanian F.** Shades of grey: On the effectiveness of reputation-based "blacklists" in Malicious and Unwanted Software // MALWARE 2008. 3rd International Conference on. IEEE. 2008. P. 57—64.

7. **Garera S., Provos N., Chew M., Rubin A.** A framework for detection and measurement of phishing attacks // ACM workshop on Recurring malcode. ACM, 2007. P. 1—8.

8. **Alshboul Y., Nepali R., Wang Y.** Detecting malicious short urls on twitter // Twenty-first Americas Conference on Information Systems, Puerto Rico, 2015.

9. **Patil D. R., Patil J.** Survey on malicious web pages detection techniques // Science and Technology. 2015. Vol. 8, N. 5. P. 195—206.

10. **McGrath D. K., Gupta M.** Behind phishing: An examination of phisher modi operandi // LEET. 2008. Vol. 8. P. 4.

11. **Ma J., Saul L. K., Savage S., Voelker G. M.** Beyond blacklists: learning to detect malicious web sites from suspicious urls // The 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009. P. 1245—1254.

12. **Learning** to detect malicious urls // ACM Transactions on Intelligent Systems and Technology (TIST). 2011. Vol. 2, N. 3. P. 30.

13. **Purkait S.** Phishing counter measures and their effectiveness—literature review // Information Management & Computer Security. 2012. Vol. 20, N. 5. P. 382—420.

14. **Khonji M., Iraqi Y., Jones A.** Phishing detection: a literature survey // IEEE Communications Surveys & Tutorials. 2013. Vol. 15, N. 4. P. 2091—2121.

15. **Nepali R. K., Wang Y.** You look suspicious: Leveraging visible attributes to classify malicious short urls on twitter // The 49th Hawaii International Conference on System Sciences (HICSS). IEEE, 2016. P. 2648—2655.

16. **Kuyama M., Kakizaki Y., Sasaki R.** Method for detecting a malicious domain by using whois and dns features // The Third International Conference on Digital Security and Forensics (DigitalSec2016). 2016. P. 74.

17. **Kolari P., Finin T., Joshi A.** Svms for the blogosphere: Blog identification and splog detection, // AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs. 2006. P. 92—99.

18. **Ma J., Saul L. K., Savage S., Voelker G. M.** Identifying suspicious urls: an application of large-scale online learning // The 26th Annual International Conference on Machine Learning. ACM, 2009. P. 681—688.

19. **Yadav S., Reddy A. K. K., Reddy A., Ranjan S.** Detecting algorithmically generated malicious domain names // The 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010. P. 48—61.

20. **Le A., Markopoulou A., Faloutsos M.** Phishdef: Url names say it all // INFOCOM, 2011. IEEE, 2011. P. 191—195.

21. **Ramanathan V., Wechsler H.** Phishing website detection using latent dirichlet allocation and adaboost // Intelligence and Security Informatics (ISI), 2012. IEEE International Conference on. IEEE, 2012. P. 102—107.

22. **Astorino A., Chiarello A., Gaudioso M.** Piccolo Malicious url detection via spherical classification // Neural Computing and Applications. 2016. P. 1—7.

23. **Hu Z., Chiong R., Pranata I., Susilo W., Bao Y.** Identifying malicious web domains using machine learning techniques with online credibility and performance data // Evolutionary Computation (CEC). IEEE, 2016. P. 5186—5194.

24. **Dekel O., Shalev-Shwartz S., Singer Y.** The forgetron: A kernelbased perceptron on a budget // SIAM Journal on Computing. 2008. Vol. 37, N. 5. P. 1342—1372.

25. **Lu J., Hoi S. C., Wang J., Zhao P., Liu Z.-Y.** Large scale online kernel learning // JMLR. 2016.

26. **Hoi S. C., Jin R., Zhao P., Yang T.** Online multiple kernel classification // Machine Learning. 2013. Vol. 90, N. 2. P. 289—316.

27. **Документация** разработчика модуля SciKit Python. URL: https://scikit-learn.org/stable/modules/ensemble.html (дата обращения: 5.02.2019).

28. **Haykin S.** Neural networks: a comprehensive foundation. N. J.: Prentice Hall, 1999.

29. **Hashem S.** Optimal linear combinations of neural networks // Neural Networks. 1997. Vol. 10, N. 4. P. 599—614.

30. **Описание** TF-IDF. URL: https://ru.wikipedia.org/wiki/TF-IDF (дата обращения: 5.02.2019).

31. **Свалин А.** CatBoost против Light GBM против XGBoost. URL: https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db (дата обращения: 6.02.2019).

32. **Документация** разработчика LGBM. URL: https://lightgbm.readthedocs.io (дата обращения: 7.02.2019).

33. **Документация** разработчика Gradient Boosting. URL: (дата обращения: 7.02.2019).

34. **Документация** разработчика XGBoost. URL: https://xgboost.readthedocs.io (дата обращения: 7.02.2019).

35. **Документация** разработчика CatBoost. URL: https://tech.yandex.ru/catboost/ (дата обращения: 7.02.2019).

36. **Набор** данных URL-адресов. URL: http://www.squidguard.org/blacklists.html (дата обращения: 12.02.2019).

37. **Набор** данных URL-адресов. URL: https://www.kaggle.com/teseract/datasets (дата обращения: 12.02.2019).

**A. B. Menisov,** Ph. D., Scientist of Military Researching Department, e-mail: men.arty@yandex.ru,
**I. A. Shastun,** Ph. D., Lecturer, e-mail: shastunivan1982@gmail.com,
Space military academy named by A. F. Mozhaysky, Saint-Petersburg,
**S. U. Kapitsin**, Ph. D., e-mail: wolf76@inbox.ru, Military Academy of the General Staff, Moscow

# The Approach of Detecting Malicious Internet Sites Based on the Processing of Lexical Attributes of Addresses (URLs) and Averaged Ensemble of Models

*Currently, the detection and blocking of access to malicious Internet sites is performed mainly by including URLs in blacklists. However, blacklists cannot be exhaustive, and they cannot be used to identify newly created malicious sites. The purpose of the article is to develop a new approach to detect malicious sites based on the average ensemble of models, allowing to improve the accuracy of detection. The developed approach, compared to modern technical solutions in the field of information security, takes into account the lexical features of the address bar (URL) of malicious sites that attackers try to modify to the address of a well-known or secure site. Also, the authors compare the results of identifying malicious URLs obtained by modern and developed approaches. The article will be useful not only to researchers in the field of machine learning, but also to experts in the cybersecurity industry.*

***Keywords:*** *information security, malicious websites, machine learning*

DOI: 10.17587/it.25.691-697

**References**

1. **Sahoo D., Liu C., Steven C.** Malicious URL Detection using Machine Learning, *A Survey*, 2017, available at: https://arxiv.org/pdf/1701.07179.pdf

2. **Hong J.** *Communications of the ACM*, 2012, vol. 55, no. 1, pp. 74—81.

3. **Berners-Lee T.** Uniform Resource Locators (URL): A Syntax for the Expression of Access Information of Objects on the Network, *World Wide Web Consortium*, 2015.

4. **Liang B., Huang J., Liu F., Wang D., Dong D., Liang Z.** *E-Business and Information System Security: International Conference on. IEEE*, 2009, pp. 1—5 (in Rssian).

5. **Malicious** URL dataset of Phishtank, available at: https://www.phishtank.com/ (date: 26.01.2019).

6. **Sinha S., Bailey M., Jahanian F.** *MALWARE 2008. 3rd International Conference on. IEEE*, 2008, pp. 57—64.

7. **Garera S., Provos N., Chew M., Rubin A.** *ACM workshop on Recurring malcode. ACM*, 2007, pp. 1—8.

8. **Alshboul Y., Nepali R., Wang Y.** Detecting malicious short urls on twitter, *Twenty-first Americas Conference on Information Systems, Puerto Rico,* 2015.

9. **Patil D. R., Patil J.** *Science and Technology*, 2015, vol. 8, no. 5, pp. 195—206.

10. **McGrath D. K., Gupta M.** *LEET*, 2008, vol. 8, pp. 4.

11. **Ma J., Saul L. K., Savage S., Voelker G. M.** *The 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM*, 2009, pp. 1245—1254.

12. **Learning** to detect malicious urls, *ACM Transactions on Intelligent Systems and Technology (TIST),* 2011, vol. 2, no. 3, pp. 30.

13. **Purkait S.** *Information Management & Computer Security*, 2012, vol. 20, no. 5, pp. 382—420.

14. **Khonji M., Iraqi Y., Jones A.** *IEEE Communications Surveys & Tutorials*, 2013, vol. 15, no. 4, pp. 2091—2121.

15. **Nepali R. K., Wang Y.** The 49th Hawaii International Conference on System Sciences (HICSS), IEEE, 2016, pp. 2648—2655.

16. **Kuyama M., Kakizaki Y., Sasaki R.** The Third International Conference on Digital Security and Forensics (DigitalSec 2016), 2016, pp. 74.

17. **Kolari P., Finin T., Joshi A.** AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs, 2006, pp. 92—99.

18. **Ma J., Saul L. K., Savage S., Voelker G. M.** *The 26th Annual International Conference on Machine Learning. ACM,* 2009, pp. 681—688.

19. **Yadav S., Reddy A. K. K., Reddy A., Ranjan S.** *The 10th ACM SIGCOMM conference on Internet measurement. ACM*, 2010, pp. 48—61.

20. **Le A., Markopoulou A., Faloutsos M.** *INFOCOM*, 2011, *IEEE*, 2011, pp. 191—195.

21. **Ramanathan V., Wechsler H.** *Intelligence and Security Informatics (ISI), 2012. IEEE International Conference on. IEEE*, 2012, pp. 102—107.

22. **Astorino A., Chiarello A., Gaudioso M.** *Neural Computing and Applications*, 2016, pp. 1—7.

23. **Hu Z., Chiong R., Pranata I., Susilo W., Bao Y.** *Evolutionary Computation (CEC), IEEE*, 2016, pp. 5186—5194.

24. **Dekel O., Shalev-Shwartz S., Singer Y.** *SIAM Journal on Computing*, 2008, vol. 37, no. 5, pp. 1342—1372.

25. **Lu J., Hoi S. C., Wang J., Zhao P., Liu Z.-Y.** Large scale online kernel learning, *JMLR*, 2016.

26. **Hoi S. C., Jin R., Zhao P., Yang T.** *Machine Learning*, 2013, vol. 90, no. 2, pp. 289—316.

27. **SciKit** Development documentation, available at: https://scikit-learn.org/stable/modules/ensemble.html (date: 5.02.2019).

28. **Haykin S.** Neural networks: a comprehensive foundation, N. J., Prentice Hall, 1999.

29. **Hashem S.** *Neural Networks*, 1997, vol. 10, no. 4, pp. 599—614.

30. **Definition** of TF-IDF, available at: https://wikipedia.org/wiki/TF-IDF (date: 5.02.2019) (in Rssian).

31. **Svalin A.** CatBoost vs Light GBM vs XGBoost, available at: https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db (date: 6.02.2019) (in Russian).

32. **LGBM** Development documentation, available at: https://lightgbm.readthedocs.io (date: 7.02.2019) (in Rssian).

33. **Gradient** Boosting Development documentation, available at: (date: 7.02.2019) (in Rssian).

34. **XGBoost** Development documentation, available at: https://xgboost.readthedocs.io (date: 7.02.2019) (in Rssian).

35. **CatBoost** Development documentation, available at: https://tech.yandex.ru/catboost/ (date: 7.02.2019) (in Rssian).

36. **URL dataset**, available at: http://www.squidguard.org/blacklists.html (date: 12.02.2019) (in Rssian).

37. **URL dataset**, available at: https://www.kaggle.com/teseract/datasets (date: 12.02.2019) (in Rssian).

УДК 378.1, 004.7 DOI: 10.17587/it.25.698-704

**С. Л. Лобачев,** д-р техн. наук, проф., зав. каф., e-mail: lsl7777@mail.ru,
Российский университет транспорта (МИИТ), Москва

# Дистанционные образовательные технологии в очном обучении: опыт, анализ и некоторые выводы

*Рассматриваются вопросы внедрения дистанционных образовательных технологий в учебный процесс вуза в рамках смешанного обучения студентов очной формы обучения. Представлены результаты анализа мнений студентов и преподавателей, влияние этих результатов на планирование и организацию учебного процесса с использованием этих технологий.*

***Ключевые слова:*** *дистанционные образовательные технологии, система дистанционного обучения, электронные образовательные ресурсы, информационные технологии, высшее образование, смешанное обучение*

## Введение

Дистанционные образовательные технологии (ДОТ) в последние несколько лет перешли из состояния технологий будущего в состояние повседневной реальности практически во всех учебных заведениях от средней школы до уважаемых и авторитетных университетов. Информационные и дистанционные технологии стали важным инструментом в образовательном процессе, без которого сегодня трудно представить качественное обучение на всех уровнях образования [1]. Накопленный опыт использования дистанционных и электронных технологий, поддержанный министерством образования и науки в виде официально утвержденного на законодательном уровне "Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ" в 2014 и 2017 гг. [2, 3], привел к тому, что сейчас сложно найти вуз, в котором в той или иной форме не использовались бы эти технологии.

При всех, несомненно, положительных факторах имеющей сегодня место картины в части использования ДОТ, приходится констатировать, что данная ситуация порой напоминает кампанию по широкому внедрению, без оглядки на готовность самих образовательных организаций, преподавателей, обучающихся и, порой, даже специалистов, занимающихся внедрением этих технологий. Практически полностью забыт положительный опыт создания и использования федеральных образовательных порталов [4], на смену которым с 2015 г. упорно продвигается проект "Национальная платформа открытого образования" [5], предлагающая онлайн-курсы по базовым дисциплинам, изучаемым в российских университетах. При этом порядок интеграции этих более чем 300 курсов в учебные планы вузов, их связь с другими дисциплинами учебных планов и еще многие вопросы до сих пор так и остаются открытыми.

Многочисленные публикации о перспективности и удобстве ДОТ носят скорее информационный и качественный характер. Почти нет публикаций с анализом результатов внедрения ДОТ, включая статистические данные об опросах преподавателей и студентов об их отношении к различным аспектам применения ДОТ в учебном процессе. Немногочисленные публикации по анализу результатов внедрения ДОТ, к сожалению, опираются на весьма ограниченный опыт — единицы учебных курсов, изученных несколькими десятками студентов. Примеров анализа внедрения ДОТ в крупном государственном университете в рамках полных образовательных программ бакалавриата, специалитета или магистратуры по нескольким специальностям и нескольким сотням студентов от первого до последнего курса обнаружить практически невозможно.

## Постановка задачи и характеристики опыта реализации ДОТ

Целью настоящей публикации является анализ результатов внедрения и использова-

ния на протяжении нескольких лет смешанного обучения по ряду специальностей высшего образования, основанный на опросах преподавателей и студентов о различных аспектах использования ДОТ, а также выработка рекомендаций по оптимальному сочетанию очного и дистанционного обучения в образовательных программах и отдельных дисциплинах высшего образования. Особое внимание уделено очной форме обучения и интеграции в нее ДОТ.

Представленные ниже данные опираются на опыт внедрения ДОТ в юридическом институте Российского университета транспорта (ЮИ РУТ) на протяжении последних шести лет [6]. Внедрение ДОТ началось в 2012 г. с реализации нескольких образовательных программ бакалавриата и специалитета с полным дистанционным обучением без посещения вуза в заочной форме обучения. С 2015 г. к этим программам добавилась одна образовательная программа среднего профессионального образования. Первый выпуск специалистов по программам высшего образования состоялся в 2015 г. с небольшой группы, прошедшей обучение по ускоренной программе. Динамика реализации ДОТ по программам с полным дистанционным обучением в период с 2012 по 2018 г. иллюстрирует диаграмма на рис. 1.

В силу изменения нормативной базы на федеральном уровне и локальной нормативной базы университета в период 2015—2017 гг. реализация трех образовательных программ в заочной форме обучения была закрыта. Однако уже с 2014 г. ДОТ стали активно использоваться в программах очного обучения, где оказались очень востребованными. Таким образом, к настоящему моменту опыт смешанного обучения по программам высшего образования в Юридическом институте составляет более пяти лет.


**Рис. 1. Динамика развития полного дистанционного обучения в заочной форме обучения**


**Рис. 2. Рост объемов реализации ДОТ в программах очного обучения**

График на рис. 2 показывает рост объемов реализации ДОТ в программах очного обучения в период с 2014 г. по весенний семестр 2019 г. На данном графике замедление роста объема часов, начиная с осени 2018 г., связано с принятием организационных решений, направленных на сдерживание процесса увеличения доли ДОТ в учебном процессе очной формы обучения. Такие решения были приняты в связи с тем, что по ряду дисциплин объем часов, реализуемых с использованием ДОТ, стал приближаться к объему, реализуемому очно. Характерно, что перевод изучения дисциплин на использование ДОТ проводился по инициативе преподавателей, которые очень активно включились в этот процесс [7].

К настоящему моменту преподавательский состав, реализующий обучение с использованием ДОТ, составляет около 120 человек, и по своей структуре выглядит следующим образом:

- доктора наук — около 20 %;
- кандидаты наук — около 75 %;
- преподаватели без ученой степени — около 5 %.

Если обратиться к анализу видов занятий, переводимых преподавателями на изучение с использованием ДОТ, то тут картина следующая:

- изучение теоретического материала (лекции и т. д.) — 38 %;
- практические работы (семинарские занятия) — 57 %;
- контрольные мероприятия (тесты) — 5 %.

В содержательном плане объем реализации образовательных программ высшего образования с использованием ДОТ характеризует табл. 1, где представлены данные за 2017/18 и через косую дробь — за 2018/19 учебный год. Следует подчеркнуть, что использованием ДОТ уже в 2017/18 учебном году были охвачены

Таблица 1

**Объем реализации образовательных программ с использованием ДОТ**

| Виды программ высшего образования | Формы обучения | Число образовательных программ, реализуемых с использованием ДОТ | Численность обучающихся по программам с ДОТ |
|---|---|---|---|
| Программы бакалавриата | Очная<br>Очно-заочная<br>Заочная | 3/3<br>1/1<br>3/3 | 468/461<br>61/109<br>146/363 |
| Программы специалитета | Очная<br>Очно-заочная<br>Заочная | 4/4<br>2/2<br>2/3 | 1284/1284<br>61/130<br>120/510 |
| Программы магистратуры | Очная<br>Очно-заочная<br>Заочная | 0/1<br>0/0<br>0/1 | 0/7<br>0/0<br>0/8 |
| ВСЕГО | | 15/18 | 2140/2872 |

все образовательные программы, реализуемые в ЮИ РУТ по очной форме обучения. Содержание таблицы констатирует устойчивый рост доли ДОТ как в числе охваченных студентов, так и в качественном плане — распространение ДОТ на программы магистратуры.

За границами данной работы остались вопросы реализации ДОТ в программах среднего и дополнительного профессионального образования, по которым в ЮИ РУТ проходит обучение с использованием ДОТ еще около 2000 человек ежегодно. Ограниченный объем статьи не позволяет рассмотреть все аспекты реализации ДОТ в учебном процессе. В частности, организационно-технологические решения, способствующие внедрению смешанного обучения в ЮИ РУТ, достаточно подробно рассмотрены в работе [6], а организация учебно-методического обеспечения опирается на результаты работы [8].

## Методика и результаты анализа мнений студентов и преподавателей об использовании ДОТ

С превращением ДОТ в регулярно используемый инструмент учебного процесса появляется возможность накопления данных для проведения анализа потребительских свойств системы дистанционного обучения (СДО) и методик, применяемых в учебном процессе с использованием ДОТ. Кроме того, такие данные могут явиться основой для дальнейшего совершенствования технологии и методики дистанционного обучения, а также локальной нормативной базы, регламентирующей использование ДОТ в учебном процессе как по отдельным учебным дисциплинам, так и по образовательным программам в целом.

В соответствии с поставленными целями в 2018 г. проводилось анкетирование преподавательского состава и студентов очной формы обучения, использующих ДОТ. В основу анкетирования были положены принципы анонимности и добровольности. По этой причине невозможно было провести анкетирование всех участников этого процесса, однако масштаб проведенного анкетирования позволил сделать выводы, повлиявшие на организацию смешанного обучения в дальнейшем. Полученные результаты представлены ниже.

В анкетировании приняли участие:

- около 100 преподавателей из 120 участвующих в реализации ДОТ;
- около 460 студентов очной формы обучения из 1750 обучающихся с использованием ДОТ.

Распределение студентов, принявших участие в анкетировании, по старшим и младшим курсам отражено в табл. 2.

Существенно меньшее число студентов старших курсов, принявших участие в анкетировании, по сравнению со студентами младших курсов, объясняется тем, что значительная часть студентов старших курсов совмещает учебу с работой, что существенно затруднило привлечение их к анкетированию.

На начальном этапе проведенного исследования был поставлен вопрос об анализе имеющегося у преподавателей опыта реализации обучения с использованием ДОТ, результаты которого отражает диаграмма на рис. 3.

Существенный процент преподавателей, имеющих небольшой опыт работы в СДО, объясня-

Таблица 2

**Распределение студентов, принявших участие в анкетировании**

| Курсы | Число опрошенных студентов | Процент от общего числа |
|---|---|---|
| Младшие (1—2) | 323 | ~70 |
| Старшие (3—5) | 138 | ~30 |
| Всего | 461 | 100 |


**Рис. 3. Опыт преподавателей, реализующих обучение с использованием ДОТ**

ется текучкой кадров, а также постоянным расширением состава образовательных программ с использованием ДОТ. Однако эти процессы не мешают преподавателям активно включаться в освоение и использование ДОТ. В значительной степени это объясняется пользовательскими свойствами используемой СДО, которые подтверждаются результатами, представленными на рис. 4 и 5. Показательно, что на обеих представленных диаграммах нет ни одной отрицательной оценки как интерфейса, так и инструкции преподавателя по работе в СДО. Данный результат является следствием организации постоянной работы по совершенствованию программной, технологической и документационной компонентам СДО, которая ведется на протяжении десяти лет, с момента апробации первой версии программного обеспечения используемой СДО. Доработка программного обеспечения СДО является непрерывным процессом, обусловленным как возрастанием потребностей пользователей, так и изменениями нормативной базы отечественной системы образования и конкретного университета. Естественным следствием этого является ежегодная корректировка всего комплекта эксплуатационной документации, включая инструкции администратора учебного процесса СДО, преподавателя и студента. В процессе такой корректировки учитываются не только объективные причины, но и замечания и пожелания пользователей, что позволило обеспечить высокие оценки, отраженные на рис. 4 и 5.

Поскольку преподаватели непосредственно реализуют учебный процесс в СДО, то было естественно выяснить их оценку инструкции и интерфейса студента в СДО, а затем сопоставить полученные оценки с мнением студентов. Результаты этой работы представлены на рис. 6.

Анализ полученных данных позволяет констатировать:

- при положительной оценке инструкции студента всеми категориями пользователей наиболее критично к ней отнеслись студенты старших курсов;
- именно в категории старших курсов наибольший процент опрошенных не смог сформулировать конкретное мнение по поставленному вопросу;
- среди тех, кто изучал инструкцию, у студентов старших курсов существенно меньше доля тех, кто что-то не понял и считает инструкцию не достаточно подробной;
- во всех категориях отрицательная оценка или вообще отсутствует или составляет крайне не значительную часть.

В качестве вывода можно предположить, что полученные результаты отражают скорее психологические особенности той или иной категории пользователей. Преподаватели подошли наиболее ответственно к анализу инструкции, студенты младших курсов — менее ответственно, но достаточно добросовестно, а студенты старших курсов в значительной ча-


**Рис. 4. Оценка удобства интерфейса преподавателя**


**Рис. 5. Оценка качества инструкции преподавателя**

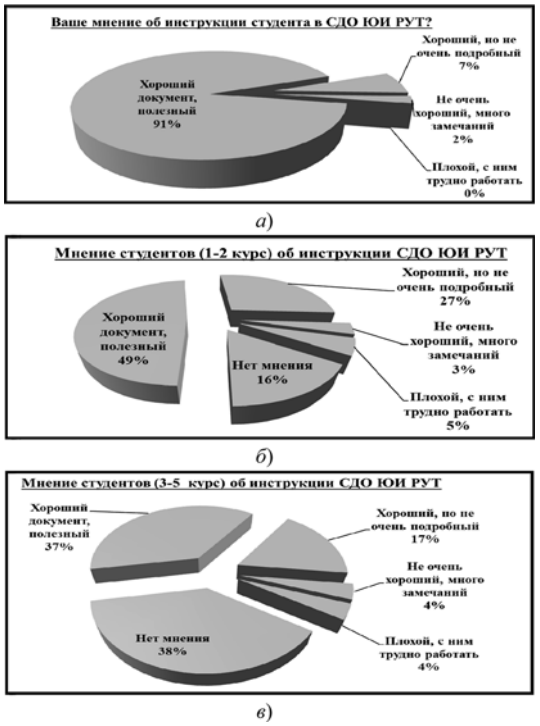



**Рис. 6. Оценки инструкции студента:**

*а* — оценка преподавателями инструкции студента; *б* — оценка студентами младших курсов; *в* — оценка студентами старших курсов

сти вообще не изучали инструкцию. Об этом говорит высокий процент студентов старших курсов, не имеющих никакого мнения.

Полученные результаты полностью соответствуют данным исследований, проведенных специалистами кафедры психологии развития Пермского государственного национального исследовательского университета в 2014 г. [9]. И хотя данные исследования не имели отношения к дистанционному обучению и использованию информационных технологий в учебном процессе, они получили очень интересные результаты, где отмечается, что студенты младших курсов:

- способны более успешно выделять значимые условия достижения поставленных целей;
- в условиях изменения обстоятельств могут более гибко подстраиваться и менять программу своих действий;
- более автономны в организации своей активности по достижению поставленных целей.

Нетрудно видеть, что эти выводы могут объяснить результаты, представленные на рис. 6. И, как результат данного совпадения, можно предположить, что при внедрении смешанного обучения следует учитывать результаты исследования психологов хотя бы на начальном этапе его внедрения, а именно, начинать с младших курсов.

Следующим шагом стал анализ интерфейса личного кабинета студента в СДО ЮИ РУТ.

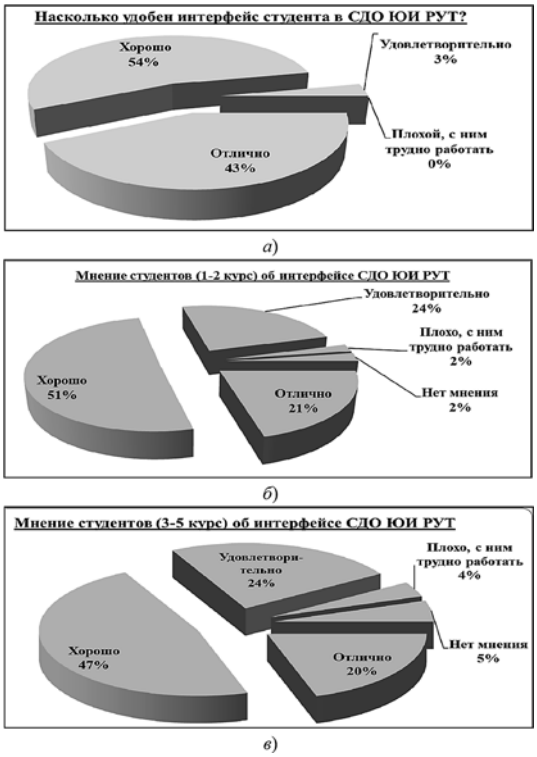



**Рис. 7. Оценки интерфейса студента**

Полученные результаты представлены на рис. 7 в той же последовательности, что и на рис. 6. Характерно, что ответ "нет мнения" здесь дали гораздо меньше студентов, чем на вопрос об инструкции. Это вполне объяснимо, поскольку инструкцию можно не читать, а не увидеть интерфейс, в котором выполняются различные действия, практически невозможно. Надо отметить, что плохую оценку интерфейсу дал практически тот же процент студентов, что и при оценке инструкции студента. Можно предположить, что именно эти 2...5 % относятся к той части студентов, которые в принципе не принимают ДОТ как средство получения знаний и являются активными их противниками.

Полученные оценки инструкции и интерфейса студента являются весьма важными, поскольку далее проводился анализ еще ряда аспектов учебного процесса с использованием ДОТ. Если бы представленные выше результаты показали преимущественно низкие оценки, то последующий анализ трудно было бы считать показательным. Технологические проблемы, в этом случае, могли бы существенно повлиять на его результаты и вызвать сомнения, не являются ли получаемые оценки отражением несовершенства СДО, а не содержательной части смешанного обучения. Поскольку проведенный выше анализ показал высокие оценки инструкций и интерфейса, можно перейти к анализу показателей организации собственно учебного процесса с использованием ДОТ.

Одним из основных вопросов, возникающих при реализации смешанного обучения с использованием ДОТ, является вопрос о соотношении объема занятий в очной форме и занятий с использованием ДОТ. Результаты опроса студентов по данному вопросу отражает табл. 3.

Характерно, что процент противников использования ДОТ для младших и старших курсов примерно одинаков. И для обеих категорий наиболее предпочтительным является соотношение 25 % занятий с использованием ДОТ и

Таблица 3

**Желательное соотношение занятий в очной форме и с использованием ДОТ**

| Объем часов с ДОТ, % | Объем очных часов, % | Мнение младших курсов (1—3 курсы), % | Мнение старших курсов (4—5 курсы), % |
|---|---|---|---|
| 0 | 100 | 13 | 15 |
| 25 | 75 | 56 | 45 |
| 50 | 50 | 23 | 27 |
| 75 | 25 | 8 | 10 |
| 100 | 0 | 0 | 3 |

75 % очные занятия. Именно эти данные легли в основу ограничений по переводу преподавателями занятий на использование ДОТ, о которых упоминалось при анализе диаграммы на рис. 2. В процессе проведения опроса стало понятно, что желательно проводить дифференциацию на профильные и образовательные дисциплины. Однако для младших курсов результаты данного опроса представляются весьма приемлемыми, а для старших требуется более углубленный анализ, хотя практический опыт говорит о том, что если студенты приступили к использованию ДОТ с первого курса, то в дальнейшем большинство становится их активными сторонниками.

Другим важным вопросом при внедрении ДОТ является вопрос о видах занятий, переводимых на использование ДОТ. Мнение студентов по этому вопросу отражают диаграммы на рис. 8. При сопоставлении данных этих диаграмм с данными табл. 3 можно увидеть различия в части позиции студентов о не использовании ДОТ в принципе. Это объясняется тем, что анкетирование проводилось анонимно и часть студентов не отвечали на все вопросы, а при обработке ответов в учет принимались только вопросы, имеющие однозначный ответ.

Отмеченные различия не являются принципиальными и не меняют общей картины. Число противников использования ДОТ составляет существенное меньшинство, а следовательно, смешанное обучение пользуется поддержкой студентов.

Вместе с тем полученные результаты могут являться ориентиром при планировании учебного процесса в части соотношения очной части и части с использованием ДОТ, а также в содержательной части видов занятий, переводимых на изучение с использованием ДОТ.

При планировании смешанного обучения рассматривался вопрос о необходимости проведения очных консультаций по использованию ДОТ, и ориентиром стали данные, представленные на рис. 9.

Результатом данного опроса явилось включение в график учебного процесса как минимум одного занятия в семестр, посвященного особенностям использования ДОТ при изучении той или иной дисциплины. Если студенты обучались с использованием ДОТ два-три семестра, то необходимость таких консультаций отпадает. Востребованность их студентами старших курсов на диаграмме рис. 9 объясняется тем, что часть студентов приступила к обучению с использованием ДОТ, уже находясь на третьем или четвертом курсах.

## Заключение

Представленные в работе результаты стали основой для планирования учебного процесса с использованием ДОТ. Несомненно, что полученные оценки могут зависеть от получаемой студентами специальности или даже особенностей конкретной дисциплины, но они могут служить ориентиром на начальном этапе внедрения смешанного обучения в любом учебном заведении. Если внедрение смешанного обучения начинать с младших курсов, то отмеченная выше специфика образовательной программы будет сказываться несущественно, поскольку состав учебных дисциплин на младших курсах различается в меньшей степени.

Хотелось бы обратить внимание на тот факт, что успешная реализация смешанного обучения в ЮИ РУТ опиралась на двухлетний опыт реализации образовательных программ с полным использованием ДОТ, без посещения вуза студентами. Именно за эти два года преподаватели накопили опыт практической реализации ДОТ, подготовки учебно-методического обеспечения для СДО. Кроме того, в течение этого периода были отработаны организационные и нормативные компоненты организации учебного процесса с использованием ДОТ, без которых реализация смешанного обучения на должном уровне была бы очень затруднительна.

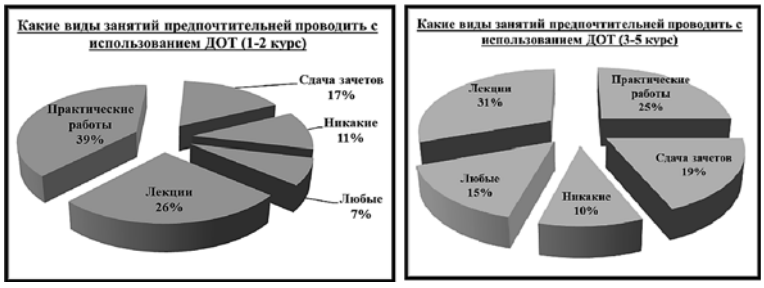



**Рис. 8. Опрос о видах занятий, где желательно использование ДОТ**

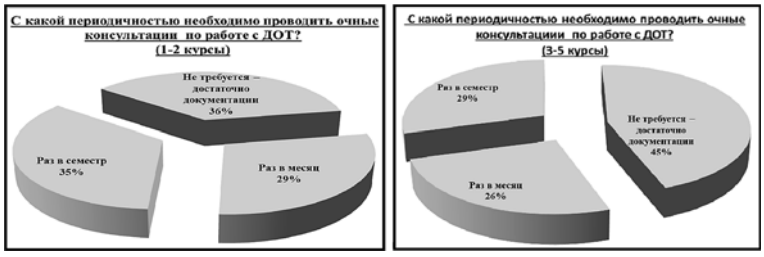



**Рис. 9. Оценка периодичности очных консультаций по использованию ДОТ**

**Список литературы**

1. **Бершадский А. М., Глотова Т. В., Кревский И. Г.** Современный университет: образовательные и информационные технологии в многоуровневой системе высшего образования // Труды XIX Международной объединенной научной конференции "Интернет и современное общество", С.-Петербург, 2016. С. 143—160.
2. **Приказ** Министерства образования и науки Российской Федерации (Минобрнауки России) от 9 января 2014 г. № 2 г. Москва "Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ".
3. **Приказ** Министерства образования и науки РФ от 23 августа 2017 г. № 816 "Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ".
4. **Иванников А. Д.** Тематические Интернет-порталы как средство агрегации электронного контента в заданной предметной области // Информационные технологии. 2014. № 3. С. 43—48.
5. **Проект** "НАЦИОНАЛЬНАЯ ПЛАТФОРМА ОТКРЫТОГО ОБРАЗОВАНИЯ". URL: http://npoed.ru/about
6. **Груздева Л. М., Лобачев С. Л., Малыгин О. А., Петровская Е. Ю.** Комплексная система дистанционного обучения Юридического института Российского университета транспорта (МИИТ) // Информатика и образование. 2018. № 1 (290). С. 27—33.
7. **Лобачев С. Л.** Востребованность дистанционных образовательных технологий студентами и преподавателями как естественный этап внедрения этих технологий в учебный процесс // Сб. докл. Междунар. науч. конф., посвященной 90-летию С. П. Капицы "Человеческий капитал в формате цифровой экономики". Москва, 2018. С. 273—282.
8. **Лобачев С. Л.** Теоретические основы и принципы построения информационно-образовательной среды открытого образования и ее практическая реализация. Дисс. на соискание ученой степени доктора технических наук, Московский государственный университет электроники и математики. Москва, 2005.
9. **Никитин А. А., Петрованова Н. А.** Особенности саморегуляции студентов младших и старших курсов // Вестник Пермского университета. 2014. Вып. 3. С. 75—82.

**S. L. Lobachev**, Professor, Head of the Department, e-mail: lsl7777@mail.ru,
Russian University of Transport (MIIT), Moscow, Russian Federation

# Distance Learning Technologies in Full-Time Education: Experience, Analysis and Some Conclusions

*The article deals with the implementation of distance learning technologies in the educational process of the University within the mixed training of full-time students. The results of the analysis of opinions of students and teachers, the impact of these results on the planning and organization of the educational process using these technologies are presented.*

***Keywords:*** *distance learning technologies, learning management system, electronic educational resources, information technologies, higher education, mixed learning*

DOI: 10.17587/it.25.698-704

**References**

1. **Bershadsky A. M., Glotova T. V., Krevsky I. G.** *Works of the XIX International United Scientific Conference "The Internet and Modern Society",* St. Petersburg, 2016, pp. 143—160 (in Russian).
2. **Order** of the Ministry of Education and Science of the Russian Federation (Ministry of Education and Science of Russia) of January 9, 2014, N 2 Moscow "On approval of the procedure for the use of e-learning and distance learning technologies by educational organizations in the implementation of educational programs" (in Russian).
3. **Order** of the Ministry of Education and Science of the Russian Federation of August 23, 2017, N 816 "On approval of the procedure for the use by organizations engaged in educational activities, e-learning, distance learning technologies in the implementation of educational programs" (in Russian).
4. **Ivannikov A. D.** *Information Technologies*, 2014, no. 3, pp. 43—48 (in Russian).
5. **Project** "NATIONAL PLATFORM OF OPEN EDUCATION", available at: http://npoed.ru/about_(in Russian).
6. **Gruzdeva L. M., Lobachev S. L., Malygin O. A., Petrovskaya E. Yu.** *Informatics and Education*, 2018, no. 1 (290), pp. 27—33 (in Russian).
7. **Lobachev S. L.** *Proc. of the International Scientific Conference dedicated to the 90th anniversary of S. P. Kapitsa, "Human Capital in the Digital Economy Format"*, Moscow, 2018, pp. 273—282 (in Russian).
8. **Lobachev S. L.** Theoretical foundations and principles of building an information and educational environment of open education and its practical implementation, Diss. for the degree of Doctor of Technical Sciences, Moscow State University of Electronics and Mathematics. Moscow 2005, 350 p. (in Russian).
9. **Nikitin A. A., Petrovanov N. A.** *Bulletin of Perm University*, 2014, no. 3, pp. 75—82 (in Russian).

**Адрес редакции:**

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**
E-mail: it@novtex.ru

Технический редактор *Е. В. Конова*.
Корректор *Е. В. Комиссарова*.
Сдано в набор 05.09.2019. Подписано в печать 24.10.2019. Формат 60×88 1/8. Бумага офсетная.
Усл. печ. л. 8,86. Заказ IT1119. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**

Рисунки к статье О. А. Сосниной, А. Д. Филинских, Н. А. Ложкиной

# «АНАЛИЗ МЕТОДОВ СОЗДАНИЯ ВИРТУАЛЬНЫХ МОДЕЛЕЙ НЕТРИВИАЛЬНЫХ ФОРМ»

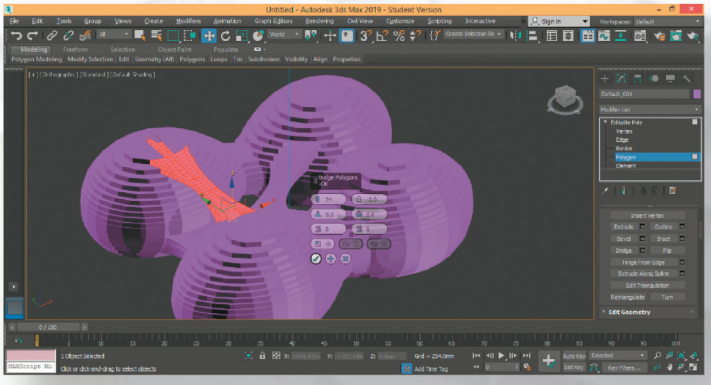


**Рис. 7. Соединение башен на уровне полигонального редактирования**

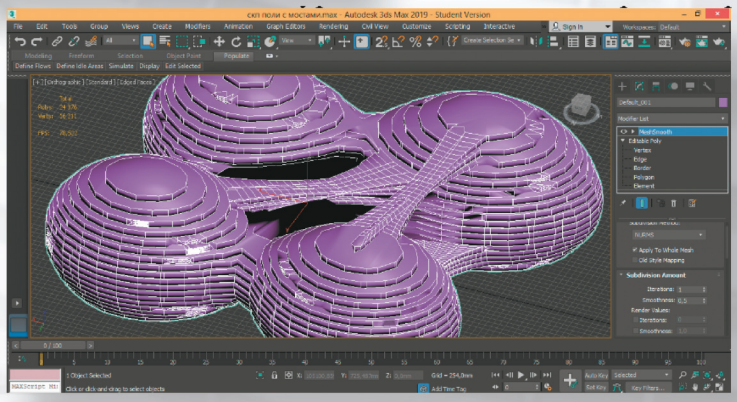


**Рис. 8. Объект после применения модификатора сглаживания**

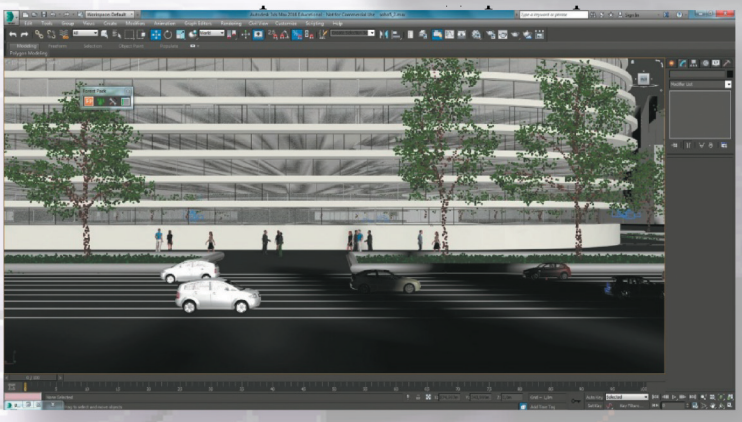
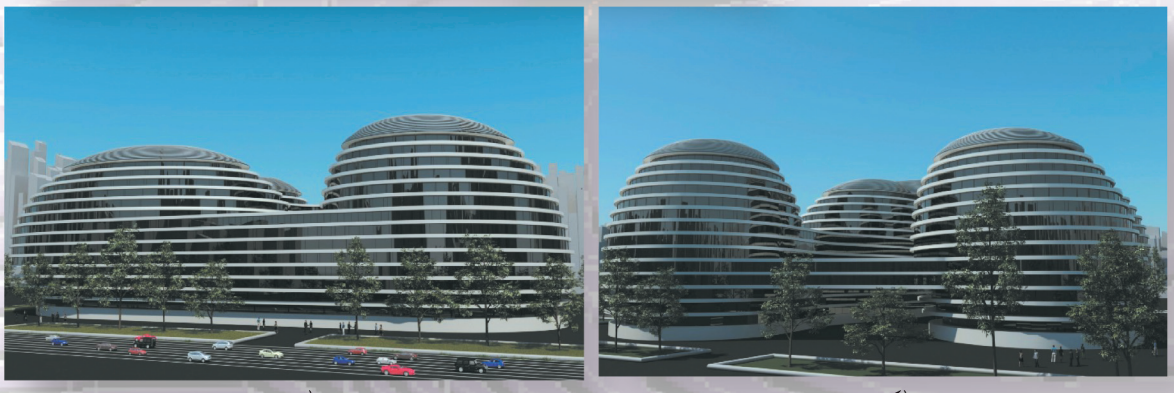


**Рис. 9. Создание окружения**



*а)* *б)*

**Рис. 10. Результат визуализации «Galaxy Soho»**

# Издательство «НОВЫЕ ТЕХНОЛОГИИ»

## выпускает *научно-технические журналы*


Ежемесячный теоретический
и прикладной научно-технический журнал

### ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В журнале освещаются современное состояние, тенденции и перспективы развития основных направлений в области разработки, производства и применения информационных технологий.

**Подписной индекс по Объединенному каталогу**
**«Пресса России» – 72656**


**Подписной индекс по Объединенному каталогу**
**«Пресса России» – 79963**

Научно-практический
и учебно-методический журнал

### БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

В журнале освещаются достижения и перспективы в области исследований, обеспечения и совершенствования защиты человека от всех видов опасностей производственной и природной среды, их контроля, мониторинга, предотвращения, ликвидации последствий аварий и катастроф, образования в сфере безопасности жизнедеятельности.

Ежемесячный
междисциплинарный
теоретический и прикладной
научно-технический журнал

### НАНО- и МИКРОСИСТЕМНАЯ ТЕХНИКА

В журнале освещаются современное состояние, тенденции и перспективы развития нано- и микросистемной техники, рассматриваются вопрос разработки и внедрения нанои микросистем в различные области науки, технологии и производства.


**Подписной индекс по Объединенному каталогу**
**«Пресса России» – 79493**


**Подписной индекс по Объединенному каталогу**
**«Пресса России» – 79492**

Ежемесячный теоретический
и прикладной
научно-технический журнал

### МЕХАТРОНИКА, АВТОМАТИЗАЦИЯ, УПРАВЛЕНИЕ

В журнале освещаются достижения в области мехатроники, интегрирующей механику, электронику, автоматику и информатику в целях совершенствования технологий производства и создания техники новых поколений. Рассматриваются актуальные проблемы теории и практики автомати ческого и автоматизированного управления техническими объектами и технологическими процессами в промышленности, энергетике и на транспорте.

Теоретический
и прикладной
научно-технический журнал

### ПРОГРАММНАЯ ИНЖЕНЕРИЯ

В журнале освещаются состояние и тенденции развития основных направлений индустрии программного обеспечения, связанных с проектированием, конструированием, архитектурой, обеспечением качества и сопровождением жизненного цикла программного обеспечения, а также рассматриваются достижения в области создания и эксплуатации прикладных программно-информационных систем во всех областях челове ческой деятельности.


**Подписной индекс по Объединенному каталогу**
**«Пресса России» – 22765**

**Адрес редакции журналов для авторов и подписчиков:**
107076, Москва, Стромынский пер., 4. Издательство "НОВЫЕ ТЕХНОЛОГИИ".
Тел.: (499) 269-55-10, 269-53-97. Факс: (499) 269-55-10. E-mail: antonov@novtex.ru

ISSN 1684-6400. Информационные технологии. 2019. Том 25. № 11. 641–704. Индекс: 72656 (Объединенный каталог «Пресса России»)