ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
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.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 10
N 2
2019

Published since September 2010 DOI 10.17587/issn.2220-3397 ISSN 2220-3397

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.451 DOI: 10.17587/prin.10.51-58

А. Н. Годунов, канд. физ.-мат. наук, зав. отделом, e-mail: nkag@niisi.ras.ru,
В. А. Солдатов, канд. техн. наук, ст. науч. сотр., e-mail: nkvalera@niisi.ras.ru,
Федеральное государственное учреждение "Федеральный научный центр Научно-исследовательский институт системных исследований Российской академии наук" (ФГУ ФНЦ НИИСИ РАН), Москва

Опыт создания компактной операционной системы реального времени

Рассмотрены методы (масштабирование, опции компилятора, размещение программного кода во флэш-памяти), используемые при создании компактной операционной системы жесткого реального времени для отечественных радиационно-стойких микропроцессоров. Вся память, требуемая целевому образу, подразделяется на статическую и динамическую. Описаны инструментальные средства, созданные для оценки требуемого объема статической оперативной памяти, как всего целевого образа, так и его отдельных частей. Приведена методика оценки требуемого объема динамической памяти для отдельных частей образа. Описаны полученные результаты.

Ключевые слова: *ОСРВ "Багет", конфигурирование, масштабирование, флэш-память, формат ELF, оперативная память, ОЗУ, динамическая память*

Введение

В последние годы большое значение приобрела проблема импортозамещения в области вычислительных систем, которая складывается из двух составляющих. Во-первых — это производство современного электронного оборудования на собственной элементной базе. Во-вторых — это собственное программное обеспечение, которое будет работать на отечественном оборудовании. В ФГУ ФНЦ НИИСИ РАН были достигнуты значительные успехи по созданию отечественных микросхем. Так, еще в 1997 г. был разработан первый в России 32-разрядный RISC-процессор (1890ВМ1Т) со встроенным сопроцессором плавающей арифметики. Далее, в 2008 г. разработан 64-разрядный суперскалярный RISC-процессор (1890ВМ5Ф). В 2011 г. закончена разработка комплекта микросхем с коммуникационной средой RapidIO. Каждая микросхема, представляющая собой систему-на-кристалле со встроенными сетевыми и коммуникационными каналами, может использоваться для создания супер-ЭВМ промышленного назначения [1, 2].

Однако несмотря на общий рост быстродействия и других ресурсов, необходимых для проведения вычислительных работ, существует класс задач, выполняемых в жестком реальном времени, для которых оперативная память является лимитированным ресурсом. Таким примером могут служить системы, работающие в условиях радиационного облучения. Связано это с тем, что радиационно-стойкая оперативная память является дорогим ресурсом и требует много места для размещения на модуле [3].

Создание компактной операционной системы (ОС), способной выполняться на ограниченном объеме оперативной памяти, было решено выполнять на базе уже существующей ОС. В качестве базовой системы была выбрана ОС реального времени (ОСРВ) "Багет" семейства 2.х [4, 5], разработанная в ФГУ ФНЦ НИИСИ РАН и предназначенная для создания управляющих систем жесткого реального времени. Операционная система "Багет" семейства 2.х, которую в дальнейшем будем называть ОСРВ "Багет", была выбрана по следующим причинам:

– является отечественной программной разработкой;

– имеет длительную историю развития (первая версия ОСРВ "Багет" была выпущена еще в 2002 г., с тех пор состоялся выпуск еще пяти изданий этой ОС);

– получила достаточно широкое распространение (используется более чем в 100 организациях);

– имеет развитые средства протоколирования, самодиагностики и восстановления;

– полностью удовлетворяет требованиям к информационной безопасности систем и прошла тестирование независимой системой тестов для операционных систем, разработанной в ИСП РАН [6].

В целях повышения мобильности ОСРВ "Багет" из нее выделена отдельная часть, которая называется пакетом поддержки модуля (ППМ) и содержит ту часть ОС, которая зависит от конкретной аппаратуры

(модуля). Пакет поддержки модуля, в частности, содержит драйверы устройств и диспетчер прерываний (за исключением пролога и эпилога).

Для разработки прикладного программного обеспечения под управлением ОСРВ "Багет" использовался специально созданный стенд, состоящий из двух ЭВМ, соединенных между собой по сети. Первая из них называется инструментальной ЭВМ (компьютер с операционной системой LINUX), а вторая – целевой ЭВМ (ЭВМ, для которой разрабатывается программное обеспечение). Радиационно-стойкий процессор, входящий в состав стенда, представляет собой 32-разрядный микропроцессор архитектуры MIPS. Объем оперативной памяти, доступной процессору, зависит от числа банков памяти, подключенных к нему, и может принимать значения 4, 8, 12 или 16 Мбайт. В описываемом стенде используется процессор, оснащенный ОЗУ объемом 4 Мбайта и флэш-памятью объемом 8 Мбайт. У процессора также имеется кэш-память данных первого уровня объемом 8 Кбайт и кэш-память инструкций первого уровня объемом также 8 Кбайт. Кэш-память второго уровня отсутствует.

Способы создания компактной ОСРВ

В работе [7] для сокращения требуемого объема оперативной памяти было предложено использовать:

– опции оптимизации компилятора;

– размещение программного кода во флэш-памяти;

– масштабирование (*scaling*).

У компилятора gcc имеется опция оптимизации *−Os*, которая предписывает создавать объектные файлы с минимальным размером сегмента кода. Для исследования результатов применения этой опции было создано два комплекта библиотек ОСРВ "Багет. Первый комплект библиотек был получен путем трансляции исходных текстов с опцией *−O3*, которая предназначена для максимального уровня оптимизации кода по времени выполнения, а второй комплект — с опцией *−Os*. Для построения целевого образа было создано два каталога. В одном каталоге все исходные тексты, входящие в целевой образ, транслировались с опцией *—O3*, а во втором каталоге все исходные тексты транслировались с опцией *−Os*. Далее с некоторыми идентичными параметрами конфигурации создавали два целевых образа и оценивали размеры сегмента кода. Результаты представлены в табл. 1.

Таблица 1

Влияние опции *−Os* на размер программного кода

Конфигурация образа	Размер кода, Кбайт, при опции *−O3* (100 %)	Размер кода, Кбайт, при опции *−Os*
Небольшой	112,7	98,5 (87,4%)
Средний	678,6	583,6 (86,0%)
Большой	1320,7	1123,9 (85,1%)

Анализируя данные, представленные в табл. 1, можно сделать вывод, что опция компилятора *−Os* позволяет уменьшить программный код в среднем на 13,8 %. При этом следует учитывать, что в некоторых случаях возможно незначительное ухудшение временных характеристик.

При создании целевого образа прикладной системы с помощью редактора связи (линкера) можно настроить секции программного кода исполняемого файла на адреса, соответствующие флэш-памяти. В этом случае программный код будет выполняться из флэш-памяти. Скорость чтения команд программы из флэш-памяти, конечно, ниже, чем скорость чтения команд из оперативной памяти, зато размер требуемой для системы оперативной памяти становится меньше на размер секций программного кода прикладной системы. Кроме того, следует учитывать, что у микропроцессора имеется кэш-память инструкций как для оперативной памяти, так и для флэш-памяти. Таким образом, можно предположить, что время выполнения программного кода не слишком зависит от места расположения кода (в оперативной памяти или во флэш-памяти).

В работе [8] приведены результаты исследования временных характеристик исполнения программного кода, хранящегося во флэш-памяти, по сравнению с выполнением того же кода, хранящегося в оперативной памяти. Проведенное тестирование формально подтвердило гипотезу о том, что использование флэш-памяти для хранения инструкций способно вызвать падение производительности. Так, в наихудшем случае, когда инструкции выполняемой функции отсутствуют в кэш-памяти процессора, используемые данные находятся в кэш-памяти данных, функция состоит из инструкций, выполняющихся за 1 такт при загрузке из кэш-памяти процессора, было получено ухудшение производительности в 1,74 раза. Однако результаты задачи копирования продемонстрировали, что за счет использования кэш-памяти процессора и активной работы с данными влияние флэш-памяти может быть практически устранено. Использование флэш-памяти можно считать оправданным при выполнении больших пользовательских программ, имеющих сопоставимую с размерами кэш-памяти инструкций часть, выполнение которой занимает больше всего процессорного времени. Кроме того, для наихудшего случая можно предложить следующий способ улучшения временных характеристик. Объектные файлы, содержащие программный код, выполнение которого надо ускорить, обрабатываются утилитой objcopy, которая переименовывает все программные секции, например, *".text"* в *".ramtext"*. Далее в скрипте, который управляет работой линкера, делается указание — секции с именами *".text"* настраивать на адреса флэш-памяти, а секции с именами *".ramtext"* — на адреса в оперативной памяти. Этим способом можно распределить весь программный код по двум видам памяти и тем

самым получить образ, которому требуется меньше оперативной памяти практически без проигрыша по быстродействию.

Основным способом сокращения объема ОС является масштабирование (*scaling*), которое включает в себя как исключение какой-то части ОСРВ целиком, так и управление значением максимального количества объектов, которое обслуживается некоторой частью ОСРВ. Некоторые части ОСРВ являются обязательными и не могут быть полностью исключены из образа. К обязательным частям относятся: обработка прерываний и исключений, поддержка многопоточности (включая планирование потоков), служба времени, распределение памяти. Каждая из этих частей масштабируема в той или иной степени, но не может быть полностью исключена.

Все остальные части операционной системы могут быть полностью исключены, если они не используются прикладной программой или ППМ, или существенно уменьшены. К необязательным частям относятся: сигналы; средства синхронизации (семафоры); условные переменные; очереди сообщений; таймеры; региональные настройки; программные каналы; таблицы символов; файловые системы; блочные устройства; потоки ввода/вывода; форматированный ввод/вывод; средства локальной и удаленной отладки; средства взаимодействия с сетью.

Рассмотрим результаты применения масштабирования на отдельных примерах. Начнем с частей ОСРВ "Багет", которые отвечают за взаимодействия с сетью. Сетевое программное обеспечение в ОСРВ базируется на использовании семейства протоколов TCP/IP. Подключение сетевых средств возможно только при условии подключения средств поддержки таймеров и очередей сообщений. Конфигурирование сети позволяет включать (или не включать) сетевые средства в целевой образ системы. Рассмотрим пример, когда в создаваемый образ системы включены взаимодействие с сетью, сетевые утилиты (*ping*, *telnet*-клиент, *telnet*-сервер), получение календарного времени по сети; заданы сетевые адреса; определена таблица компьютеров в локальной сети; включена поддержка сетевых протоколов (RPC, SLIP, PPP), задано использование сетевых файловых систем (*nfs, ftp*), включены удаленный отладчик и динамическая загрузка модулей. Построен образ, и средствами, которые будут описаны ниже, оценен размер памяти, которая отводится под сегменты кода и данных. Далее исключили из конфигурации образа использование сетевых средств, построили образ и опять оценили размер требуемой памяти. В результате получилось, что для поддержки сетевых средств в ОСРВ "Багет" требуется около 600 Кбайт памяти только под сегменты кода и сегменты данных.

Если сетевые средства включаются в образ, то уменьшить их объем можно за счет исключения из образа части поддерживаемых протоколов (RPC, SLIP, PPP), сетевых утилит (*ping*, *telnet*-клиент, *telnet*-сервер, получения времени по сети), таблицы компьютеров, и т. п.

Сетевые средства также используют и память, которая выделяется динамически при выполнении образа. При конфигурировании сетевых средств задается число сетевых буферов (*nb*, значение по умолчанию 8192) и число кластеров (*nk*, значение по умолчанию 1024). При инициализации сетевого интерфейса будет запрошено ($256 \times nb + 2048 \times nk$) байт памяти. Понятно, что значения, заданные по умолчанию, слишком велики и их надо уменьшать, исходя из потребностей и существующих ресурсов.

Конфигурирование средств форматированного вывода является примером более сложного масштабирования системы для уменьшения объема требуемой памяти. В ОСРВ "Багет" для форматированного вывода можно использовать три функции: *kprint(), syslog()* и *printf()*.

Функция *printf()* соответствует стандартам POSIX и Си. Она использует функции форматных преобразований, а также функции обслуживания потоков ввода и вывода (*FILE **). Собственно вывод отформатированных сообщений выполняется функцией *write()*, которая соответствует стандарту POSIX. Сообщения выводятся в файл (в том числе на терминал). Функцию *printf()* нельзя использовать в обработчиках прерывания. Также следует учитывать, что в случае краха системы данные вывода могут быть потеряны.

Функция *syslog()* также соответствует стандарту POSIX. Для форматирования она использует функцию *snprintf()*, и в силу этого в образ ОС попадают функции форматирования, а также функции обслуживания потоков ввода и вывода (*FILE **). Отформатированное сообщение помещается в очередь, и впоследствии выводится специальным потоком (*syslog demon*). Возможен вывод в файл (в том числе на терминал) и/или в память (во флэш-память и в сохраняемую при перезагрузке оперативную память). Последние сообщения (помещенные в очередь, но еще не выведенные) могут быть потеряны в случае краха системы. Возможен вывод, минуя очередь (обычно используется при тяжелых ошибках), в этом случае сообщения не теряются. Функцию *syslog()* можно использовать в обработчиках прерывания.

Функция *kprint()* нестандартна. Для форматирования она использует функцию *kvprintf()*, которая является упрощенной версией стандартной функции *vsprintf()*. Собственно вывод выполняется функцией *kwrite()*. Существующая реализация функции *kwrite()*, содержащаяся в ППМ, выводит сообщения (только) на консоль побайтно в режиме опроса. Функция *kprint()* может быть использована в обработчиках прерываний и обладает наибольшей живучестью (в частности, не использует прерывания). Отметим, что использование режима опроса и запрещение прерываний функцией *kwrite()* ухудшают временные характеристики.

Средства конфигурирования позволяют отключить использование потоков ввода/вывода, что автоматически должно привести к отсутствию необходимости использования стандартных функций вывода. Для того чтобы в этом случае не изменять исходные тексты программ, в которых содержатся обращения

к стандартным функциям форматированного вывода, в сборке целевого образа начинают участвовать одноименные облегченные функции (*outbyte(), printf(), puts(), sprint(), snprintf()*). Также, в случае запрета на использование системного журнала, можно не изменять исходные тексты с вызовом функции *syslog()*, так как в этом случае в сборке будет участвовать одноименная функция, вызывающая *kprint()*. Если при конфигурировании отключить консоль, тогда в сборке будет участвовать одноименная функция *kprint(),* осуществляющая вывод не на консоль, а в оперативную память, сохранющую свое содержимое при перезагрузке образа (адрес и размер сохраняемой памяти указываются при конфигурации). И, наконец, можно полностью отключить вывод, тогда в сборке будет участвовать одноименная функция *kprint()*, которая ничего не делает.

Если в состав стенда добавить еще одну целевую ЭВМ на основе 32-разрядного микропроцессора архитектуры *MIPS*, которая имеет достаточно большой размер оперативной памяти, тогда можно отлаживаться на более мощной машине. Когда система становится уже отлаженной, можно отключить средства отладки и перенести выполнение образа на целевую ЭВМ с малым объемом ОЗУ.

Средства измерения объема используемой памяти

Объем памяти, который требуется для выполнения любой программы, можно условно разделить на следующие три части:

— память, в которой размещается программный код и неизменяемые константы программы;

— память, в которой размещаются глобальные и статические данные, используемые программой;

— память, которая динамически выделяется программе при ее выполнении.

Память первых двух видов будем называть статической памятью, а третий вид памяти – динамической памятью. Сколько статической памяти требуется для выполнения программы можно оценить до ее выполнения путем анализа объектного кода программы. Самый простой способ, позволяющий оценить объем статической памяти, это использование какой-либо из утилит для получения информации о файлах формата ELF, например, readelf или objdump. Но, когда задачей является оценка не только общего объема используемой памяти, но и оценка объема памяти, которая используется отдельными частями, входящими в целевой образ, стандартные утилиты становятся неподходящим инструментом.

Для оценки размера оперативной памяти, которая требуется для каждого объектного модуля, можно использовать опцию редактора связей *gcc — printmap*, которая предписывает выводить подробную информацию о каждом объектном модуле, входящем в образ системы. Другими словами, создавать карту образа (*map*). Информация, выдаваемая редактором связей, состоит из нескольких различных частей, следующих друг за другом. В начале каждой части выводится ее заголовок.

Первая часть выдачи содержит список модулей, включенных в собираемый образ для разрешения внешних ссылок. Каждый элемент списка состоит из двух строк. В первой строке содержится имя модуля, который включается в образ, а во второй — какая именно внешняя ссылка и какого модуля требует включить его в собираемый образ. Фрагмент выдачи представлен на рис. 1.

В случае, когда в образе оказался модуль, который не должен там содержаться, из соответствующей строки списка легко увидеть, какой модуль и какая внешняя ссылка вызвали эту ситуацию.

```
Archive member included because of file (symbol)
      /home/nkvalera/oc277-005h/lib/bspbt205.a(cpuLib.o)
                                                  oc2000.o (cp0PrIDGet)
      /home/nkvalera/oc277-005h/lib/bspbt205.a(cacheR3kLib.o)
                                                  oc2000.o (cacheInit)
            .          .          .
      /home/nkvalera/oc277-005h/lib/oc2000mips.a(queue.o)
                  /home/nkvalera/oc277-005h/lib/oc2000mips.a(q_ddl.o) (queInitDef)
            .          .          .
```

Рис. 1. Фрагмент списка модулей, включенных в образ для разрешения внешних ссылок

```
      .          .          .
.text         0x0000000000000000    0x18698
 *(.text .stub)
 .text        0x0000000000000000     0x3208 oc2000.o
              0x0000000000000ea4             kvprintf
      .          .          .
              0x0000000000000cfc             boardWDReset
 .text        0x0000000000003208     0x210 mem_show.o
      .          .          .
              0x00000000000032b8             start
 .text        0x0000000000003418     0x33c cpuLib.o
      .          .          .
              0x00000000000182cc             queInitDef
.rodata       0x0000000000000000     0x1c54
 *(.rodata)
.rodata   0x0000000000000d40   0x5c /home/nkvalera/oc277-005h/lib/oc2000mips.a(mempool.o)
      .          .          .
 .rodata      0x0000000000001a54     0x200 /home/nkvalera/oc277-005h/lib/oc2000mips.a(fsb.o)
              0x0000000000001a54             fmsbTbl
.          .          .
.data         0x0000000000000000     0x588
 *(.data)
 .data        0x0000000000000000     0x12c oc2000.o
              0x0000000000000014             boardCpuClockRate
      .          .          .
```

Рис. 2. Фрагмент карты памяти (*map*) целевого образа

В последней части выдачи находится карта памяти, состоящая из списка включенных сегментов. Для каждого сегмента указывается его смещение в целевом образе, размер и имя модуля, из которого взят сегмент. Если сегмент модуля содержит внешние имена, тогда для каждого из них указывается смещение в целевом образе (рис. 2).

Формально карта памяти содержит всю информацию, необходимую для анализа использования памяти каждым модулем, входящим в целевой образ. Однако формат, в котором представлена эта информация, не удобен для зрительного восприятия.

Для анализа потребления памяти удобно образ ОС разбить на отдельные части. Наименование части и список модулей, входящих в эту часть, готовятся заранее. Например, вводится понятие части с наименованием *"shell"*, к которой относятся модули *symtbl.c, shell.c, s_y.c, s_lex.c, alias_func.c, help.c, sys_shell.c* и *sym_tbl.S*. По аналогии все модули, входящие в состав образа, должны быть отнесены к какой-либо части. Если модуль не содержится ни в какой части, тогда он будет отнесен к части ППМ.

На основании сказанного была разработана утилита *mini* на языке *java*, которая преобразует карту памяти образа из формата, представленно на рис. 2, в вид удобный для анализа использования статической памяти. Пример такой выдачи представлен в табл. 2.

Таблица 2

Статическая память, требуемая отдельным частям образа

Type	.text	.rodata	.data	.bss	.sdata	.sbss	TOTAL
shell	53 080	51 657	79 252	188	64	52	18 4293
net	133 472	10 764	1040	2228	88	285	147 877
nfs	134 316	2352	600	7392	40	256	144 956
os	85 660	1508	1640	15 368	120	220	104 516
mips	27 796	76 528	0	56	44	36	104 460
PPM	85 604	12 244	104	2004	0	4	99 960
netinet	82 956	4280	860	3004	148	204	91 452
unix	72 220	5900	2580	9476	392	328	90 896
validator	51 456	22 116	0	0	0	8	73 580
debug	44 840	14 956	832	392	36	172	61 228
msdosfs	54 164	2044	464	80	16	68	56 836
rpc	38 632	2804	296	76	52	26	41 886
init_os	26 308	5896	872	4928	0	0	38 004
stdio	28 016	7136	0	0	56	12	35 220
fsck	31 528	2532	0	0	8	4	34 072
cd9660	23 348	864	648	0	8	28	24 896
term	20 016	96	0	0	0	0	20 112
remote-debug	16 652	2876	16	80	20	68	19 712
show	12 768	3444	0	0	4	8	16 224
stdlib	15 060	724	152	128	36	24	16 124
mdload	13 064	2284	136	160	0	56	15 700
utils	12 200	2484	0	0	0	0	14 684
mtx	10 020	148	0	388	4	4	10 564
time	9108	432	324	252	12	4	10 132
sig	8884	76	396	332	4	4	9696
mq	8852	32	16	340	16	16	9272
prsmem	7276	952	0	0	0	4	8232
gcclib	7812	296	0	0	0	0	8108
libm	5256	128	2064	0	624	0	8072
syslog	7012	192	0	116	16	8	7344
tar	5688	628	224	0	8	4	6552
mem	5128	92	0	984	16	0	6220
sem	5872	28	0	196	4	4	6104
dir	5248	28	0	0	0	0	5276
locale	3408	1612	0	0	0	8	5028
kprint	4528	452	0	0	0	0	4980
cond	3852	132	0	368	0	4	4356
tmr	4132	28	0	160	0	4	4324
fileio	4236	4	0	0	0	0	4240
string	3244	512	0	0	0	4	3760
ieee	3180	344	0	56	0	0	3580
encoding	1216	4	0	0	0	16	1236

Окончание табл. 2

Type	.text	.rodata	.data	.bss	.sdata	.sbss	TOTAL
pipe	1176	36	0	0	0	0	1212
posix	760	0	0	0	0	0	760
setjmp	740	0	0	0	0	0	740
user	508	0	0	0	0	0	508
wchar	124	0	0	0	0	0	124
TOTAL	**1 180 416**	**241 645**	**92 516**	**48 752**	**1836**	**1943**	**1 567 108**

Каждая строка табл. 2 содержит информацию об одной части образа ОСРВ "Багет". В заключительной части таблицы выведены итоговая строка.

Столбцы в табл. 2 имеют следующий смысл:

Type — наименование части образа;

TOTAL — суммарный размер всех сегментов.

Столбцы с наименованием ".text", ".rodata", ".data", ".bss", ".sdata", ".sbss" содержат размер в байтах памяти одноименных сегментов в десятичной системе счисления.

Утилита *mini* может также выдавать детализацию по одной части образа или по всем частям образа. В последнем случае в колонке Type выводятся наименования модулей, относящихся к каждой части. Например, детализация по частям образа, которые содержат модули *nfs*-клиента, удаленного отладчика (*remote debugger*) и динамического загрузчика (*mdload*) представлена в табл. 3.

В табл. 3 перед списком модулей, относящихся к части образа, выведен промежуточный заголовок с наименованием части, а в конце списка — строка, содержащая объем требуемой памяти всей части образа.

Для оценки размера динамической памяти, используемой программой, можно применять различные методы, однако все они могут не дать точного результата. Для оценки размера оперативной памяти, которая выделяется динамически, рассмотрим общие принципы распределения памяти. В ОСРВ "Багет" реализованы функции (динамического) распределения памяти в соответствии со стандартом Си, а также реализован механизм распределения памяти на основе так называемых пулов памяти.

При инициализации системы в ОСРВ "Багет" определяется начало свободной памяти, которая начинается сразу за памятью, занятой программным кодом и сегментами с данными. Далее резервируется память для стека обработчика прерываний, и вся оставшаяся (свободная) память отводится под системный пул. Следует также учесть, что в самом начале оперативной памяти в процессорах архитектуры MIPS располагаются векторы обработки исключительных ситуаций. В ОСРВ "Багет" часть памяти, расположенная после векторов прерываний, отводится под "сохраняемую" память, в которую при выполнении системы на целевой ЭВМ можно записывать различную информацию, и она будет там сохранена при перезагрузке без выключения питания. Распределение памяти представлено на рис. 3.

При оценке памяти, используемой в системе динамически, будем учитывать только память системного пула. Для получения информации об использовании памяти можно обратиться к функции *mem_showS()*, разработанной специально для этого. В результате выполнения этой функции можно узнать: размер системного пула, сколько памяти из него уже выделено и сколько памяти осталось еще свободной.

Таблица 3

Пример детализации по отдельным частям образа

Type	.text	.rodata	.data	.bss	.sdata	.sbss	TOTAL
=======nfs:							
nfs_nqlease.o	5256	40	104	0	0	8	5408
nfs_bio.o	12 092	208	0	0	0	0	12 300
nfs_mount.o	992	0	0	0	0	0	992
nfs_socket.o	13 256	852	0	5700	16	20	19 844
nfs_node.o	1860	120	0	0	0	20	2000
nfs_vnops.o	77 492	628	272	160	12	20	78 584
nfs_subs.o	11 444	100	136	0	4	140	11 824
krpc_subr.o	2128	120	0	0	4	0	2252
nfs_syscalls.o	2536	96	0	0	0	4	2636
nfs_stub.o	32	0	0	0	4	4	40
nfs_vfsops.o	7228	188	88	1532	0	40	9076
total nfs	134 316	2352	600	7392	40	256	144 956
======remote-debug:							
xdr_ld.o	304	0	0	0	0	0	304
rdebug.o	13 748	2688	0	80	20	60	16 596
xdr_rdb.o	1064	24	0	0	0	0	1088
xdr_ptrace.o	436	0	16	0	0	0	452
rdebuga.o	1100	164	0	0	0	8	1272
total remote-debug	16 652	2876	16	80	20	68	19 712
=====mdload:							
elfload.o	8716	1544	0	0	0	8	10 268
module.o	4348	740	136	160	0	48	5432
total mdload	13 064	2284	136	160	0	56	15 700

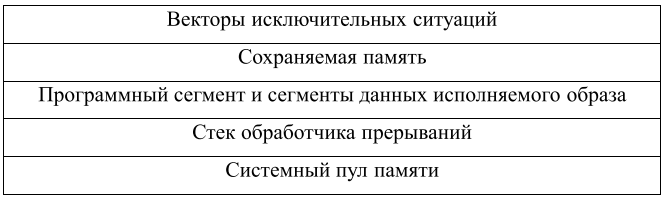

Рис. 3. Распределение оперативной памяти

Выполняя эту функцию перед началом инициализации некоторой части системы и после ее окончания, можно оценить объем динамической памяти, выделенной при инициализации этой части.

Узнать сколько динамической памяти выделено при начальной инициализации всех частей ОСРВ можно, если после окончания инициализации в пользовательской функции вызвать функцию *mem_showS()* и посмотреть значение, указывающее на размер выделенной памяти.

Более подробное описание всех средств измерений объема памяти, требуемой системе, представлено в работе [9].

Результаты

Приведем пример создания образа, который по потребностям в оперативной памяти будет близок к минимальному и будет состоять только из обязательных частей. При конфигурировании ОСРВ было задано:

— размер стека потоков управления, равный 8196 байт;

— исключено все, что можно не включать в образ, и запрещено использование плавающих чисел;

— максимально возможное число мьютексов уменьшено до 10;

— в качестве пользовательской программы включен модуль *"mem_show.c"*, разработанный для тестирования образа, состоящего только из обязательных частей ОСРВ "Багет".

При такой конфигурации получим образ, по размерам близкий к минимальному. Для оценки статической памяти, требуемой образу, воспользуемся утилитой *mini* (табл. 4). Как видно из данных табл. 4, образу, близкому к минимальному, требуется около 105 Кбайтов статической памяти.

Тестовый пример после старта оценивает распределение динамической памяти и порождает два потока, каждый из которых в течение заданного времени в цикле выполняет некоторый итерационный расчет, результат записывает по определенным адресам сохраняемой памяти и вызывает функцию *sched_yield()*. По окончанию выполнения потоков еще раз оценивается распределение динамической памяти и вызывается функция перезагрузки системы.

Для контроля правильности выполнения тестового примера, после его окончания анализируется содержимое сохраняемой памяти, из которого следует, что на инициализацию ОСРВ было выделено 11 Кбайт динамической памяти, по 8 Кбайт памяти было выделено на стек каждого из потоков, оба потока отработали без ошибок (результат итерационных расчетов совпадает с ожидаемым). Таким образом, получается, что для выполнения тестового примера образа, близкого к минимальному, требуется 124 Кбайта памяти.

Таблица 4

Статическая память образа, близкого к минимальному

Type	.text	.rodata	.data	.bss	.sdata	.sbss	TOTAL
os	32 936	916	1096	5400	52	200	40 600
init_os	14 912	3392	300	4864	0	0	23 468
mips	14 508	2004	0	56	16	36	16 620
PPM	7548	0	16	4	0	4	7572
mtx	5460	148	0	388	4	4	6004
mem	4316	92	0	984	16	0	5408
gcclib	3812	256	0	0	0	0	4068
string	752	512	0	0	0	0	1264
setjmp	612	0	0	0	0	0	612
user	560	0	0	0	0	0	560
stdlib	504	0	0	0	0	0	504
shell	0	0	0	0	0	0	0
TOTAL	**85 920**	**7320**	**1412**	**11 696**	**88**	**244**	**106 680**

Если сравнивать полученный результат с другими ОСРВ, можно увидеть, что у ОСРВ, которые поддерживают стандарт POSIX, требования к объему минимальной памяти примерно такого же порядка [10]. Следует заметить, что существуют ОСРВ, предназначенные для встраивания в микроконтроллеры, которым требуется меньше оперативной памяти для их функционирования [11]. Однако эта группа ОСРВ по своему интерфейсу не соответствуют никакому общепринятому стандарту, что существенно ухудшает переносимость прикладного программного обеспечения.

Заключение

Описаны методы, которые позволяют уменьшить объем оперативной памяти, требуемой для выполнения прикладной системы под управлением ОСРВ "Багет". Для каждого из предложенных методов приведена оценка получаемого выигрыша.

Представлены разработанные средства и методы, позволяющие оценивать потребность целевого образа и его отдельных частей в статической памяти на инструментальной ЭВМ до его выполнения. Определены точки использования специально разработанной функции для оценки потребностей в памяти, выделяемой динамически.

Описан отладочный стенд, созданный для разработки систем, в которых накладываются ограничения на объем используемой оперативной памяти. На инструментальной ЭВМ этого стенда добавлены средства, позволяющие контролировать потребность в оперативной памяти разрабатываемых систем.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП14) по теме (проекту) "Исследование принципов построения компактной операционной системы для отечественных радиационно-стойких процессоров" (№ 0065-2018-0021).

Список литературы

1. **Бобков С. Г.** Импортозамещение элементной базы вычислительных систем // Вестник Российской Академии Наук. 2014. Том 84, № 11. С. 1010—1016.

2. **Бобков С. Г., Задябин С. О.** Перспективные высокопроизводительные вычислительные системы промышленного применения на базе стандарта RapidIO // Электроника, микро и наноэлектроника: сб. научн. тр., М.: МИФИ, 2009. С. 114—121.

3. **Новожилов Е. А.** Микропроцессоры НИИСИ РАН для космического применения // 2-я Международная научная конференция "Интегральные схемы и электронные модули", Республика Крым, г. Алушта, 26—30 сентября 2016 г. Сб. трудов, Республика Крым, г. Алушта, 2016. С. 278—279.
4. **Безруков В. Л., Годунов А. Н., Назаров П. Е., Солдатов В. А., Хоменков И. И.** Введение в ос2000 // Вопросы кибернетики. 1999. С. 76—106.
5. **Годунов А. Н., Солдатов В. А.** Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. С. 68—76.
6. **Герлиц Е. А., Кулямин В. В., Максимов А. В., Петренко А. К., Хорошилов А. В., Цыварев А. В.** Тестирование операционных систем // Труды Института системного программирования РАН. 2014. Том 26. Выпуск 1. С. 73—108.
7. **Асонов А. А., Годунов А. Н., Солдатов В. А.** Методы снижения ресурсоемкости операционной системы жесткого реального времени // Труды НИИСИ РАН. 2018. Том 8, № 1. С. 4—13.
8. **Байков Н. Д., Годунов А. Н.** Исполнение программного кода, хранящегося во флэш-памяти // Труды НИИСИ РАН. 2018. Том 8, № 6. С. 136—141.
9. **Годунов А. Н., Солдатов В. А.** Методы и средства оценки объема памяти, требуемой для выполнения образа ОСРВ Багет // Труды НИИСИ РАН. 2018. Том 8, № 4. С. 42—52.
10. **Золотарев С.** Операционные системы реального времени для 32-разрядных микропроцессоров // Современная электроника. 2006. № 7. С. 52—59.
11. **Курниц А.** FreeRTOS — операционная система для микроконтроллеров // Компоненты и технологии (КиТ). 2011. № 2, 3, 4, 5, 6, 7, 8, 9, 10. С. 96—100, 109—114, 96—102, 97—102, 98—104, 23—32, 132—140, 97—104, 93—100.

Experience Creating a Compact Real-Time Operating System

A. N. Godunov, nkag@niisi.ras.ru, **V. A. Soldatov,** nkvalera@niisi.ras.ru, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA), Moscow, 117218, Russian Federation

Corresponding author:

Godunov Alexander N., Head of Department, Federal State Institution "Scientific Research Institute for System Analysis of the Russian Academy of Sciences" (SRISA), Moscow, 117218, Russian Federation,
E-mail: nkag@niisi.ras.ru

Received on December 03, 2018
Accepted on December 20, 2018

RAM is a limited resource for a certain class of tasks that are performed in a hard real-time. First of all, this refers to the systems operating in the conditions of radiation exposure. Radiation resistant RAM is quite expensive. To reduce the size of memory required for RTOS, it was decided to use already existing RTOS, and to propose the methods allowing significant reduction of the amount of RAM required. The approach described allows reaching significant memory savings without making major changes in the RTOS, enabling the reduction of development efforts and providing greater reliability of the RTOS. The essential methods of reducing the amount of RAM required by the RTOS are the OS configuration, the placement of sections of RTOS used only for reading in flash memory, the use of the compiler optimization option that minimizes the amount of the required RAM. The work also describes the methods and features allowing quantifying the memory savings resulting from the use of methods indicated. Some parts of the RTOS (interrupt and exception handling, multi-threading, time service, memory allocation) are mandatory and cannot be completely excluded from the target image, but can only be reduced to some extents. All other parts of the operating system can be either significantly reduced or excluded unless they are used by the application program or the board support package (BSP). These include networking, file systems, block devices, basic input/output operations, debugging tools, condition variables, message queues, timers, regional settings, program channels, signals and semaphores. Hardware for development of the compact RTOS and the results obtained are also described.

Keywords: *RTOS Baget, configuration, scaling, flash memory, ELF format, Random Access Memory, RAM, dynamic memory*

Acknowledgements: *This work was supported by state research program of the Russian Federation (No. 0065-2018-0021).*

For citation:

Godunov A. N., Soldatov V. A. Experience Creating a Compact Real-Time Operating System, *Programmnaya Ingeneria,* 2019, vol. 10, no. 2, pp. 51—58

DOI: 10.17587/prin.10.51-58

References

1. **Bobkov S. G.** Importozameshhenie jelementnoj bazy vychislitel'nyh sistem (Import Substitution Element Base Computing Systems), *Vestnik Rossijskoj Akademii Nauk*, 2014, vol. 84, no. 11, pp. 1010—1016 (in Russian).
2. **Bobkov S. G., Zadjabin S. O.** Perspektivnye vysokoproizvoditel'nye vychislitel'nye sistemy promyshlennogo primenenija na baze standarta RapidIO (Advanced high-performance industrial computing systems based on the RapidIO standard), *Jelektronika, mikro i nanojelektronika: sb. nauchn. tr.,* Moscow, MIFI, 2009, pp. 114—121 (in Russian).
3. **Novozhilov E. A.** Mikroprocessory NIISI RAN dlja kosmicheskogo primenenija (Microprocessors of NIISI RAS for space applications*), 2-ya Mezhdunarodnaya nauchnaya konferencia "Integral'nye shemy i jelektronnye moduli",* Alushta, 26—30 September, 2016, Sb. trudov, Alushta, 2016, pp. 278—279 (in Russian).
4. **Bezrukov V. L., Godunov A. N., Nazarov P. E., Soldatov V. A., Homenkov I. I.** Vvedenie v oc2000 (Introduction to the oc2000), *Voprosy kibernetiki*, 1999, pp. 76—106 (in Russian).
5. **Godunov A. N., Soldatov V. A.** Operacionnye sistemy semejstva Baget (shodstva, otlichija i perspektivy) (Operating Systems of the Baget Family (Likeness, Differences and Perspectives)), *Programmirovanie,* 2014, no. 5, pp. 68—76 (in Russian).
6. **Gerlic E. A., Kuljamin V. V., Maksimov A. V., Petrenko A. K., Horoshilov A. V., Cyvarev A. V.** Testirovanie operacionnyh sistem (Testing operating systems), *Trudy Instituta sistemnogo programmirovanija RAN*, 2014, vol. 26, no. 1, pp. 73—108 (in Russian).
7. **Asonov A. A., Godunov A. N., Soldatov V. A.** Metody snizhenija resursoemkosti operacionnoj sistemy zhestkogo real'nogo vremeni (Methods for reducing the resource consumption of the hard real-time operating system), *Trudy NIISI RAN,* 2018, vol. 8, no. 1, pp. 4—13 (in Russian).
8. **Bajkov N. D., Godunov A. N**. Ispolnenie programmnogo koda, hranjashhegosja vo fljesh-pamjati (Execution of program code stored in flash memory), *Trudy NIISI RAN,* 2018, vol. 8, no. 6, pp. 136—141 (in Russian).
9. **Godunov A. N., Soldatov V. A.** Metody i sredstva ocenki ob#ema pamjati, trebuemoj dlja vypolnenija obraza OSRV Baget (Methods and tools for estimating the amount of memory required for an RTOS Baget image), *Trudy NIISI RAN,* 2018, vol. 8, no. 4, pp. 42—52 (in Russian).
10. **Zolotarev S.** Operacionnye sistemy real'nogo vremeni dlja 32-razrjadnyh mikroprocessorov (Real-time operating systems for 32-bit microprocessors), *Sovremennaja jelektronika,* 2006, no. 7, pp. 52—59 (in Russian).
11. **Kurnic A.** FreeRTOS — operacionnaja sistema dlja mikrokontrollerov (FreeRTOS — operating system for microcontrollers), *Komponenty i tehnologii (KiT),* 2011, no. 2, 3, 4, 5, 6, 7, 8, 9, 10, pp. 96—100, 109—114, 96—102, 97—102, 98—104, 23—32, 132—140, 97—104, 93—100 (in Russian).

УДК 004.8 DOI: 10.17587/prin.10.59-68

В. П. Тельнов, канд. техн. наук, доц., e-mail: telnov@bk.ru, **Ю. А. Коровин,** д-р физ.-мат. наук, проф., e-mail: korovinyu@mail.ru, Национальный исследовательский ядерный университет "МИФИ", Обнинск

Программирование графов знаний, рассуждения на графах

Рассматривается технология представления знаний и модели рассуждений в системах искусственного интеллекта, оснащенных средствами логического вывода. Визуальная навигация в графах знаний и рассуждения на графах осуществляются с помощью специальных поисковых виджетов и интеллектуального браузера RDF. Представлена архитектура семантического веб-портала, приведены примеры использования графов ядерных знаний.

Ключевые слова: *семантический веб, онтология, граф знаний, графовая база данных, облачные вычисления*

Введение

Экспоненциальный рост объема и скорости обращения информации в масштабах планеты (информационный взрыв) и, как следствие, информационный кризис (противоречие между возрастающими потоками информации и ограниченными возможностями человека по ее восприятию) есть важная предпосылка возникновения семантического веба. В мировом сообществе осознано "направление главного удара" в борьбе с информационным взрывом — переход от хранения и обработки данных к накоплению и обработке знаний.

Справедливо отмечается, что традиционные веб-технологии (иногда называемые web 2.0) не предоставляют адекватных средств поиска и навигации в среде распределенных знаний на семантическом уровне. Как следствие, возникла идея об интеллектуальных агентах (программных средствах), которые могли бы самостоятельно идентифицировать релевантные информационные ресурсы из любого доступного источника данных и содействовать синтезу достоверного знания. Семантические подходы к управлению знаниями по состоянию на 2018 г. находят применение при организации обмена научными данными (EUDAT); при компьютерной обработке текстов, написанных на естественных языках (WordNet); в процессе развития международных баз знаний (DBpedia, Wikidata); при семантическом анализе социальных сетей; в экспертных, прогностических, образовательных системах, в медицине (IBM Watson).

Языки описания онтологий RDF, RDFS [1], OWL [2], графы знаний и дескрипционные логики [3, 4], которые по выразительным качествам сопоставимы с логикой исчисления предикатов первого порядка и близки к модальным логикам, способствуют формированию современной теоретической основы для представления знаний в компьютерных системах, что, в частности, подтверждается действующими стандартами W3C в области семантического веба [4]. Семантическая графовая база данных, также именуемая RDF-хранилищем, выделяется среди других типов графовых баз данных благодаря возможности поддерживать онтологии. Такая база данных интегрирует разнородную информацию из многих источников и хранит взаимосвязи между элементами данных. Семантическая база данных способна выявлять новые знания на основе существующей информации по заданным алгоритмам. Такая база представляет собой эффективный инструментарий генерации когнитивных гипотез и анализа отношений между сущностями.

Актуальность семантических подходов к управлению знаниями отмечена в докладах Третьей международной конференции МАГАТЭ по управлению ядерными знаниями (ноябрь 2016 г.), XIX Международной конференции DAMDID/RCDL "Аналитика и управление данными в областях с интенсивным использованием данных" (октябрь 2017 г.) и др. По состоянию на середину 2018 г. образовательные веб-порталы университетов, центры ядерных данных, базы данных МАГАТЭ и Государственной корпорации по атомной энергии "Росатом" не используют возможности семантической паутины.

Целью работы, представленной в настоящей статье, является создание семантического веб-портала ядерных знаний [6] с опорой на онтологии и с использованием графовых баз данных, развернутых на облачных платформах. Задача исследования заключалась в создании следующих графов ядерных знаний:

- мировые центры ядерных данных;
- ядерные исследовательские центры Российской Федерации;
- события и публикации ЦЕРН;
- базы данных и сетевые сервисы МАГАТЭ;
- учебные материалы МГУ и МИФИ по ядерной физике;
- журналы по ядерной физике;
- объединенный граф ядерных знаний.

Потенциальными бенефициарами информационных решений и технологий, которые предлагаются в статье, являются руководители, эксперты, студенты, преподаватели и специалисты в области ядерной физики, атомной энергетики, компьютерных наук (целевая аудитория).

Смежные работы и новизна

Наиболее полный обзор состояния дел в области представления и вывода знаний применительно к задачам искусственного интеллекта представлен в монографии [7, с. 136—168]. Ряд университетов [8] и компаний-разработчиков программного обеспечения [9] концентрируют свое внимание на вопросах реализации машин вывода (ризонеров) для дескрипционных логик и воплощений языка описания онтологий OWL. В докладах конференции International Workshops on Description Logics 2017 [10] отмечается рост интереса гигантов IT-индустрии (Google, Facebook, Wikimedia) к графовым моделям представления знаний и дескрипционным логикам. Попытки обнаружить в русскоязычном сегменте всемирной паутины действующие сервисы семантического веба редко приводят к успеху. Есть все основания для вывода о том, что в России производится недостаточно связанных открытых данных. Основными источниками данных для русскоязычных пользователей семантической паутины остаются общедоступные международные базы знаний, содержащие многоязычный контент, в первую очередь DBpedia и Wikidata. Представленный в настоящей работе проект [6] призван частично восполнить этот пробел. Интеллектуальный браузер RDF есть главная отличительная особенность проекта, которая выделяет его среди иных информационных решений в областях представления знаний в компьютерных системах и реализации алгоритмов рассуждений на графах знаний.

Авторитетный обзор [11] позволяет ориентироваться среди новейших продуктов и решений, которые используют технологии семантического веба. Лидерами здесь признаются программные продукты AllegroGraph, Apache Jena, ArangoDB, Blazegraph, Cray Graph Engine, DataStax Enterprise Graph, Ontotext GraphDB, IBM Graph, MarkLogic, OrientDB, Neo4j, Stardog, Teradata, Aster, Virtuoso. Перспективным представляется совместный проект компаний Ontotext и Impelsys относительно использования платформ Ontotext GraphDB и Dynamic Semantic Publishing для развития систем персонализированного адаптивного образования. Рассматриваемый в настоящей работе пилотный проект по созданию семантической базы ядерных знаний [6] использует облачные платформы и сетевые сервисы Ontotext Cognitive Cloud [12], Google Cloud Platform [13], Amazon Web Services [14], что вполне соответствует современному уровню разработок в данной области.

Паттерны проектирования онтологий

Онтология в контексте информационных технологий понимается как формальная спецификация с иерархической структурой, которая предназначена для представления знаний. Обычно онтология включает в себя описания классов сущностей (концептов) и их свойств (ролей) применительно к некоторой предметной области, а также отношения между сущностями и ограничения на то, как эти отношения могут использоваться. Онтологии, которые дополнительно включают в себя индивиды (экземпляры классов сущностей) и частные утверждения относительно индивидов, также называют графами знаний. Под формальной моделью онтологии O понимают упорядоченную тройку вида

$$O = <X, R, F>,$$

где X — конечное множество классов сущностей (концептов) для той предметной области, которую представляет онтология O;

R — конечное множество свойств (ролей), которые устанавливают отношения между сущностями для некоторой предметной области;

F — конечное множество функций интерпретации, заданных на сущностях и/или свойствах для онтологии O. Можно сказать, что функции интерпретации отображают формальные онтологии на предметные области.

Основополагающие структуры семантического веба, в частности, онтологии и графы знаний, основаны на совокупности стандартов, установленных международным консорциумом W3C [4]. Ключевыми стандартами, которые используются в проекте [6], являются среда описания ресурсов RDF, язык описания онтологий OWL и язык запросов к семантической паутине SPARQL [15]. Модель RDF (триплеты) — это простой базовый формализм, который применяется для представления данных в графах знаний. Язык описания онтологий OWL основан на дескрипционных логиках. Он существенно расширяет возможности модели RDF и предназначен для логического описания состояния дел в некоторой предметной области, включая иерархию сущностей и отношения между сущностями.

Богатство и разнообразие взаимоотношений между классами и отдельными индивидами отличает полноценные графы знаний от словарей, тезаурусов и таксономий, которые используются в консервативных системах управления знаниями, см. например [16]. Одна из привлекательных особенностей семантического веба заключается в том, что он обеспечивает возможность извлечения (вывода) новых знаний из фактов, которые уже имеются в онтологии (графе знаний). Для этого применяются соответствующие программные агенты, которые называют ризонерами. То, как логический вывод осуществляется алгоритмически, не описывается в самом OWL-документе, поскольку OWL есть декларативный язык. Правильный ответ на любой вопрос в графе знаний предопределяется семантикой той дескрипционной логики, которая лежит в основе стандарта языка, на котором описана онтология (граф знаний). Современный стандарт языка OWL 2 основан на семантике дескрипционной логики с сигнатурой SROIQ(D) [4].

На рис. 1 показан паттерн проектирования графа знаний типа "Центр ядерных исследований", который

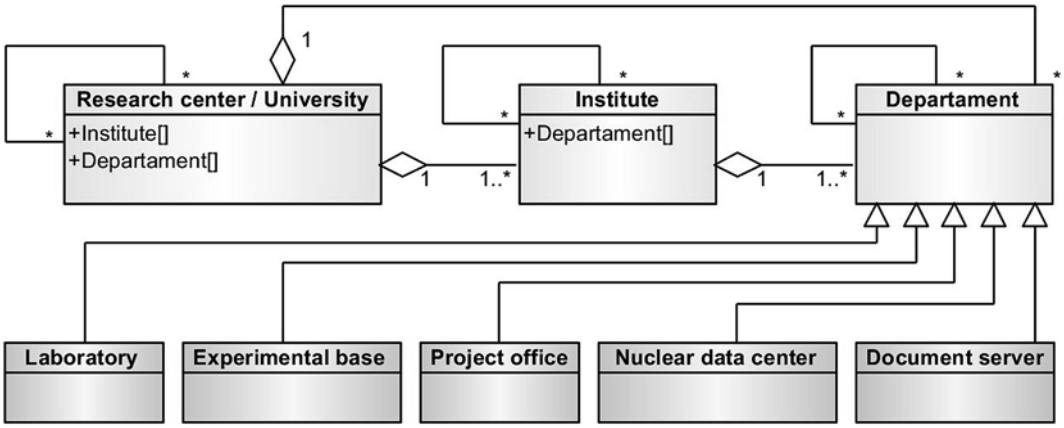

Рис. 1. Паттерн проектирования графа знаний типа "Центр ядерных исследований" в нотации UML

используется в проекте [6]. Эта модель была разработана на основе анализа следующих российских и международных ядерных исследовательских центров: Национальный исследовательский центр "Курчатовский институт", Национальный исследовательский ядерный университет "МИФИ", НИИ ядерной физики МГУ, ЦЕРН, МАГАТЭ.

На рис. 2 показан паттерн проектирования графа знаний типа "Ядерный учебный центр", который ис-

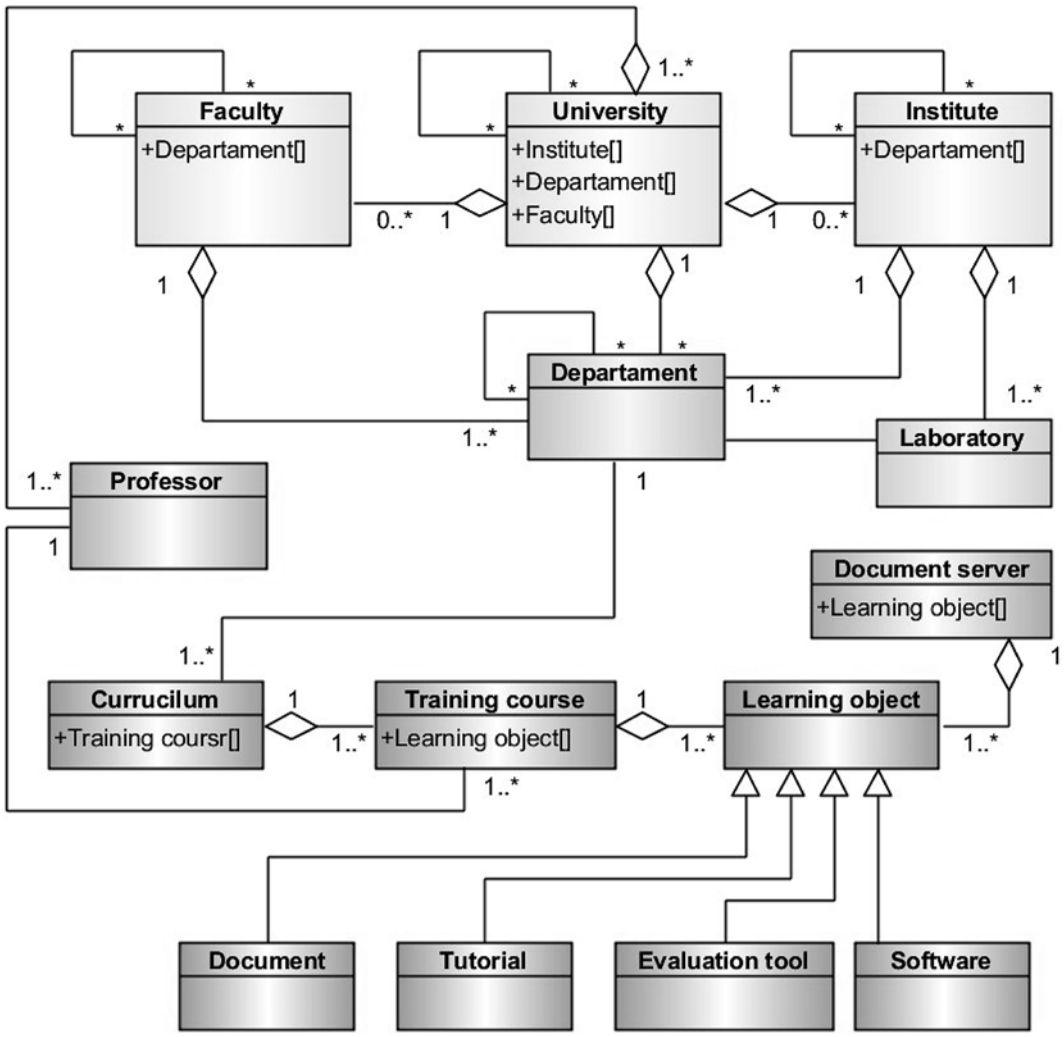

Рис. 2. Паттерн проектирования графа знаний типа "Ядерный учебный центр" в нотации UML

пользуется в проекте [6]. Данная модель была создана на основе анализа образовательных программ следующих российских учебных центров: Национальный исследовательский ядерный университет "МИФИ", Физический факультет МГУ.

Паттерны проектирования онтологий, изображенные на рис. 1 и 2, представлены в нотации языка UML согласно международному стандарту [17]. Пример онтологии в сериализованном виде для графа знаний "Учебные материалы МГУ и МИФИ по ядерной физике" доступен по ссылке [18].

Графовая база данных

Онтологию, обогащенную экстенсиональными знаниями из конкретной предметной области, называют графом знаний или базой знаний. С практической точки зрения графы знаний размещаются в графовых базах данных или в иных репозиториях, которые именуют RDF-хранилищами или хранилищами триплетов. Проект [6] базируется на облачной платформе Ontotext Cognitive Cloud, репозитории семантической базы знаний физически размещаются на серверах Amazon Web Services и обслуживаются виртуальными вычислительными машинами. Удаленная асинхронная работа с облачной платформой Ontotext Cognitive Cloud осуществляется с помощью стандартного языка запросов SPARQL 1.1 через интерфейсы прикладного программирования на языках Java, C#, Python, PHP, Ruby, JavaScript, NodeJS, др. Распространенными операциями являются создание, чтение, обновление и удаление данных в графах знаний. Для практической реализации сетевых запросов к репозиториям используются методы протокола HTTP, такие как GET, POST, PUT, DELETE. Тело каждого HTTP-запроса содержит сгенерированные SPARQL-запросы следующих видов:

- SELECT для извлечения данных из графа знаний;
- CONSTRUCT для создания нового графа знаний в формате RDF;
- INSERT для добавления триплетов к графу знаний;
- DELETE для удаления триплетов из графа знаний.

На рис. 3 показаны примеры работы поисковых виджетов семантической базы ядерных знаний [6], которые предназначены для быстрого погружения в доступные графы знаний. Каждый из графов знаний содержит тысячи триплетов. Поисковые виджеты, показанные на рис. 3, позволяют пользователям попадать в нужное место конкретного графа знаний, где будут обнаружены и визуализированы искомые информационные объекты. Принцип работы поисковых виджетов аналогичен тому, как происходит выборка информации из всемирной паутины с помощью популярных поисковых систем (Google, Яндекс, др.). По мере того как пользователь набирает символы ключевых слов в строке ввода поискового виджета, система "выкатывает" адекватный список сущностей из соответствующего графа знаний. Пользователю предлагается выбрать подходящий концепт или индивид и погрузиться непосредственно в нужную область графа знаний.

Далее представляется возможность более точной визуальной навигации по графу, которая осуществляется интуитивно понятным образом с применением интеллектуального браузера RDF, как будет описано ниже.

Интеллектуальный браузер RDF

Браузер RDF является существенным атрибутом проекта [6], который отличает его от иных известных решений в области семантического веба. Оказавшись в желаемом месте нужного графа знаний с помощью поискового виджета, далее пользователь посредством браузера RDF может выполнять визуальную навигацию по графу, посещая его узлы в нужном порядке и извлекая метаданные, гипертекстовые ссылки, полнотекстовый и медийный контент, ассоциированный с узлом. При этом окрестность (окружение, замыкание) каждого узла графа становится видимой и доступной для навигации. Эта окрестность включает

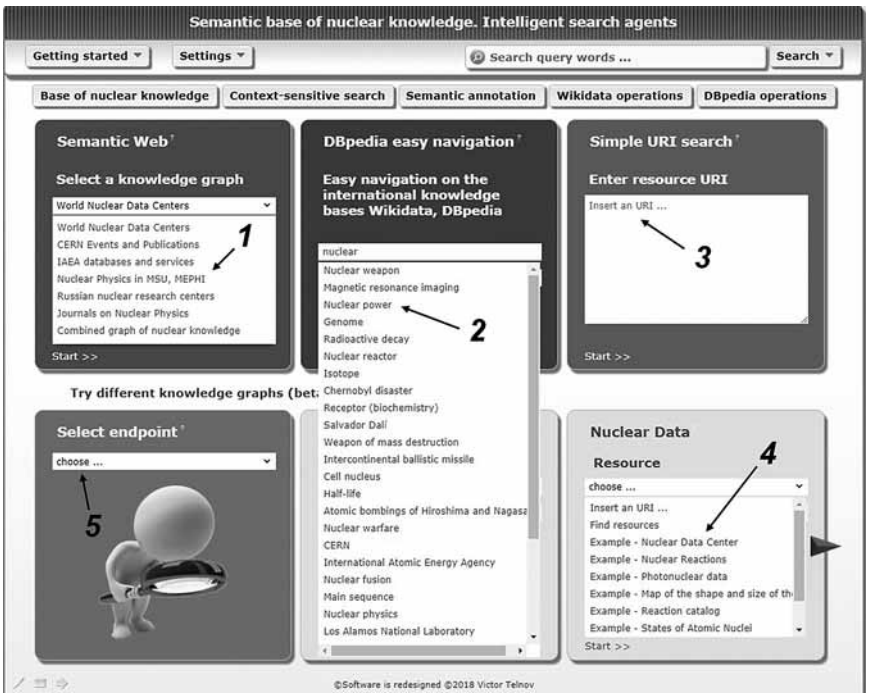

Рис. 3. Поисковые виджеты, предназначенные для быстрого погружения в графы знаний:

1 — выбор графа знаний для работы; *2* — быстрая навигация в DBpedia; *3* — поиск по URI во всемирной семантической паутине; *4* — примеры работы с графами знаний; *5* — выбор графа знаний для демонстрации

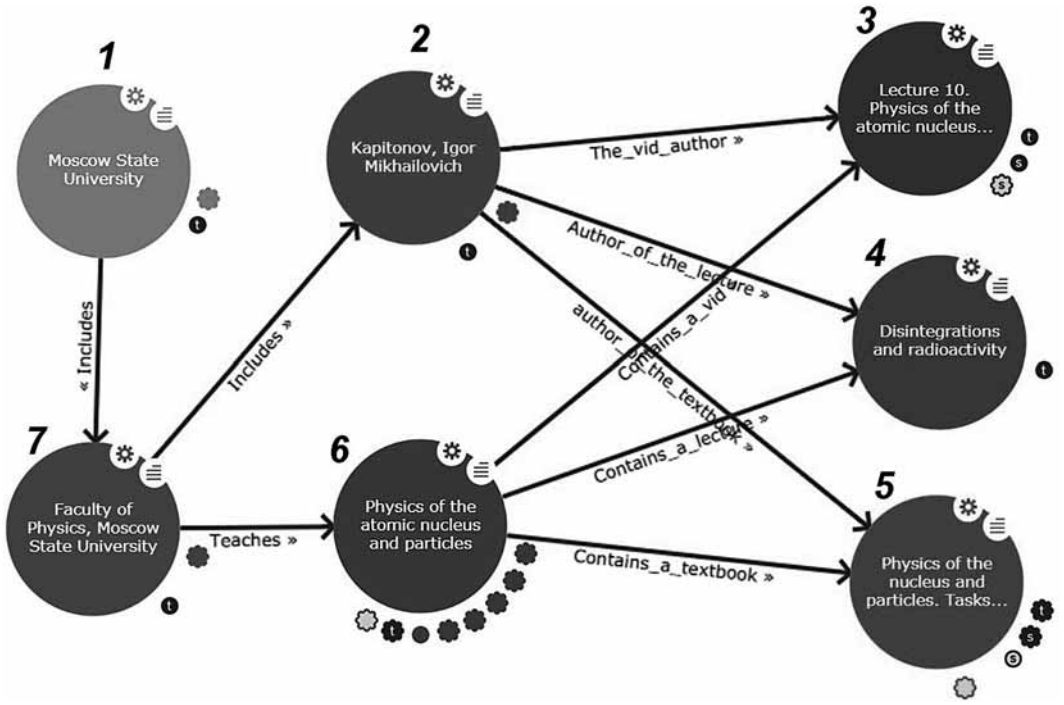

Рис. 4. Фрагмент графа знаний типа "Ядерный учебный центр":

1 — индивид класса "University"; *2* — индивид класса "Professor"; *3* — индивид класса "Training video"; *4* — индивид класса "Lecture"; *5* — индивид класса "Textbook"; *6* — индивид класса "Training course"; *7* — индивид класса "Faculty"

в себя узлы того графа, через который пользователь изначально вошел в семантическую паутину, а также смежные узлы иных графов, которые поддерживаются базой знаний.

На рис. 4 в качестве примера показана ближайшая окрестность узла, который соответствует индивиду "Physics of the Atomic Nucleus and Particles" класса "Training course" в графе знаний "Учебные материалы МГУ и МИФИ по ядерной физике", а также некоторые элементы управления, ассоциированные с узлами графа. Если сфокусироваться на любом ином узле, который отображается браузером RDF, он также станет доступен для навигации вместе с его окрестностью и метаданными. Таким образом, пользователь имеет возможность осуществлять произвольные визуальные туры по графам знаний в любом направлении и на любую глубину, выбирая появляющиеся данные. При наведении курсора на различные участки каждого узла графа знаний появляются локальное меню, уточняющая информация и подсказки для вариантов продолжения навигации по графу.

Логические рассуждения на графах знаний

Графы знаний проекта [6] обслуживаются визуальным ризонером, который встроен в интеллектуальный браузер RDF, и с точки зрения функциональных возможностей близок к ризонерам известного программного продукта Apache Jena [9]. В частности, ризонер проекта [6] обеспечивает следующие виды стандартных проверок и рассуждений на графах:

- при проверке выполнимости (*satisfiability*) исследуется вопрос, может ли произвольный класс иметь экземпляры (индивиды);
- при проверке категоризации (*subsumption*) исследуются подклассы классов, строится иерархия вложенности классов и свойств;
- при проверке согласованности (*consistency*) исследуются вопрос, насколько экстенсиональные знания (ABox [4]) согласованы в графе с интенсиональными знаниями (TBox [4]);
- при проверке экземпляров классов (индивидов) исследуется корректность всех утверждений о данном экземпляре с точки зрения ABox;
- поиск всех экземпляров класса (индивидов), а также поиск всех классов, которым конкретный индивид принадлежит прямо или опосредованно;
- поиск всех сущностей, которые связаны между собой определенным свойством, а также группировка (кластеризация) таких сущностей.

Кластеры сущностей, связанные между собой определенным свойством или группой свойств, это примеры выведенных фактов (образцы новых знаний), которые изначально в явном виде не были представлены в графе. Выведенные факты в браузере RDF имеют вид лепестков, сгруппированных вокруг узлов графа, они раскрываются щелчком мыши и удобны для последующей визуальной навигации по графу. Визуальный способ указания правил выво-

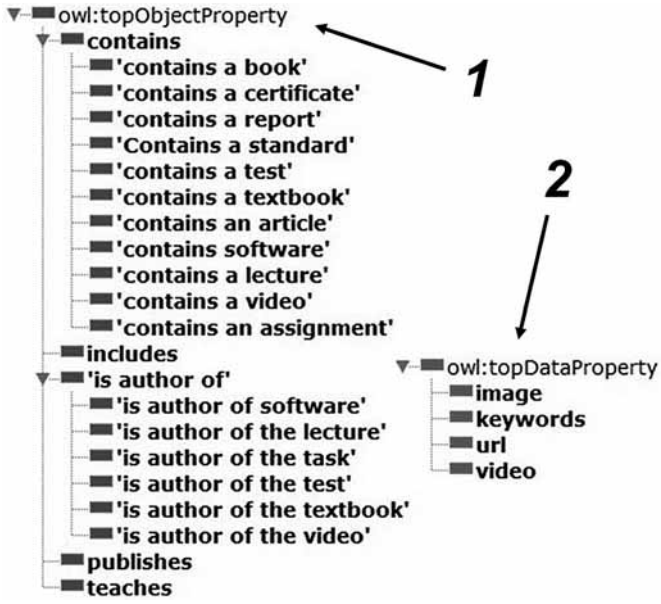

Рис. 5. Основные свойства объектов (*1*) и свойств данных (*2*), которые используются в графах знаний проекта [6]

да на графе знаний является своего рода изюминкой проекта [6], которая отличает его от известных традиционных ризонеров, где правила вывода задаются с использованием логических предикатов и SPARQL-подобного синтаксиса. Представляется, что интуитивно понятный визуальный способ задания правил вывода и навигации является более комфортным для неискушенных пользователей графов знаний.

Все графы знаний, составляющие семантический образовательный веб-портал [6], включают в себя основные свойства объектов и свойства данных (рис. 5). Браузер RDF снабжен встроенными алгоритмами вывода, которые обеспечивают навигацию по графам и средства поиска в графах. Сама возможность логического вывода новых фактов в семантических базах знаний обусловлена тем обстоятельством, что язык описания онтологий OWL базируется на дескрипционной логике. Семантика дескрипционной логики для языка OWL 2 (стандарт см. в работе [2]) расширяет дескрипционную логику с сигнатурой SROIQ(D), которая, в свою очередь, основана на логике с сигнатурой ALC (*Attributive Language with Complement*) [4]. Каждая конкретная дескрипционная логика характеризуется набором конструкторов и индуктивным правилом, с помощью которого составные концепты данной логики строятся из атомарных концептов и атомарных ролей. Для дальнейшего изложения приведем два конструктора, которые связаны с использованием квантора всеобщности $\forall$ и квантора существования $\exists$ в дескрипционной логике ALC.

Выражение $\forall R.C$ интерпретируется как множество тех индивидов, для которых все R-последователи принадлежат интерпретации концепта C, формально:

$$(\forall R.C)^I = \left\{ e \in \Delta^I \middle| \forall d \in \Delta^I : (e,d) \in R^I \Rightarrow d \in C^I \right\}. \quad (1)$$

Выражение $\exists R.C$ интерпретируется как множество тех индивидов, у которых имеется R-последователь, принадлежащий интерпретации концепта C, формально:

$$(\exists R.C)^I = \left\{ e \in \Delta^I \middle| \exists d \in \Delta^I : (e,d) \in R^I \wedge d \in C^I \right\}. \quad (2)$$

Здесь использованы следующие обозначения:

I — интерпретирующая функция, которая отображает дескрипционную логику на конкретную предметную область (домен);

Δ — обозначение для всей предметной области (домена);

R — некоторая роль (свойство);

C — некоторый концепт (класс);

e, d — индивиды, принадлежащие некоторому концепту (классу) и связанные некоторой ролью (свойством).

Все логические рассуждения на графах знаний в проекте [6] реализуются с помощью визуального ризонера, встроенного в RDF-браузер, который автоматически генерирует необходимые SPARQL-запросы к базе знаний, затем обрабатывает и классифицирует полученные результаты. Рассмотрим простой пример автоматизированных логических рассуждений на графе знаний. Предположим, что в ходе навигации по графу знаний "Учебные материалы МГУ и МИФИ по ядерной физике" был обнаружен некий индивид с именем "Objectives of the common nuclear workshop". С помощью браузера RDF легко определить, что данный индивид относится к классу "Textbook" (учебник) и связан с учебным курсом под названием "Physics of the Atomic Nucleus and Particles" (рис. 6). Необходимо найти все остальные учебники, которые присутствуют в данном графе знаний и связаны с данным учебным курсом. Для решения этой задачи применимы как минимум три варианта рассуждений.

Вариант 1. Найти в графе все узлы, которые являются аналогами индивида "Objectives of the common nuclear workshop", т. е. связаны с ним свойством "sameAs". Здесь свойство "sameAs" объединяет всех индивидов класса "Textbox", которые ассоциированы с индивидом класса "Training course".

Вариант 2. Перейти в узел "Textbook" (учебник) и выявить всех индивидов этого класса, которые ассоциированы с индивидами класса "Training course".

Вариант 3. Перейти в узел "Physics of the Atomic Nucleus and Particles", который связан с данным индивидом свойством "contains a textbook", и там раскрыть всю группу учебников, принадлежащих данному учебному курсу.

Рассуждения по всем трем вариантам решения задачи основаны на использовании конструкторов (1) и (2), которые задают правила интерпретации квантора всеобщности и квантора существования при прямом и обратном выводе на графе знаний. Рис. 7 иллюстрирует результат рассуждений на графе, выполненных по варианту 3. Использование любого другого из предложенных вариантов рассуждений дало бы идентичный результат.

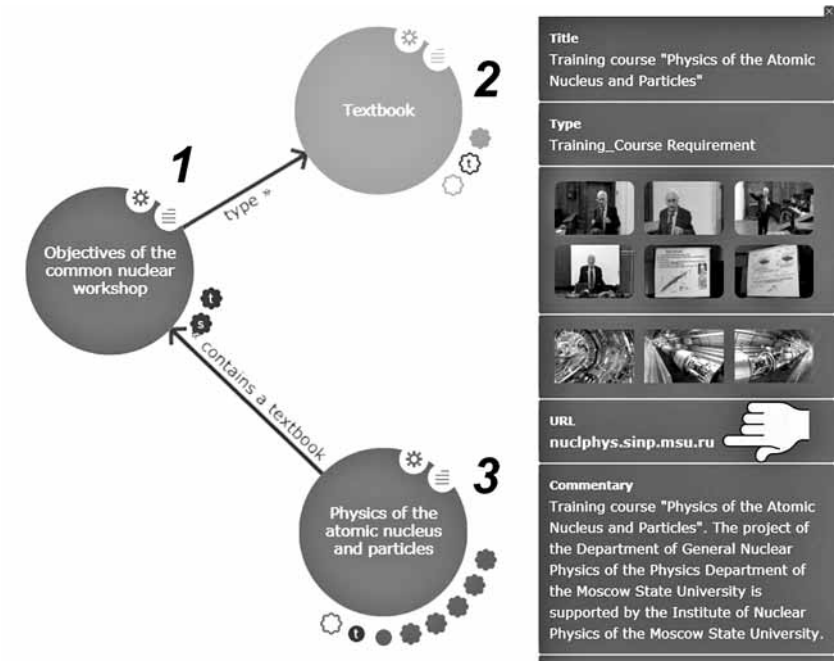

Рис. 6. К логическим рассуждениям на графах знаний:
1 — индивид класса "Textbook"; *2* — класс "Textbook"; *3* — индивид класса "Training course"

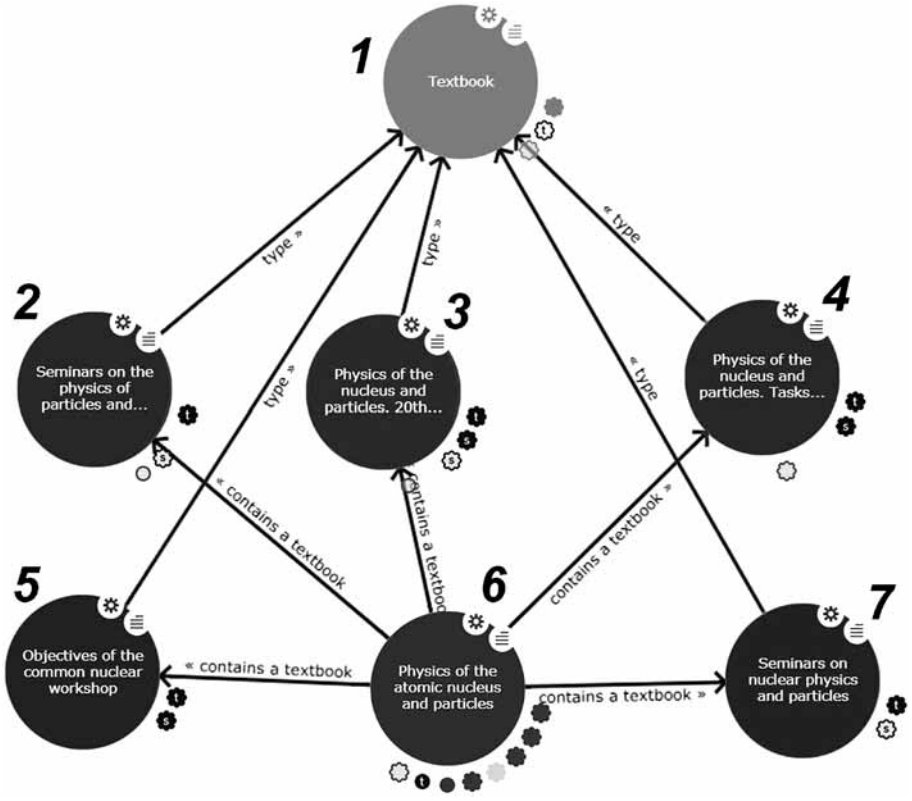

Рис. 7. Пример рассуждений на графе знаний с использованием свойств "type" и "contains a textbook":
1 — класс "Textbook"; *2*, *3*, *4*, *5*, *7* — индивиды класса "Textbook"; *6* — индивид класса "Training course"

Для полноценного использования графов знаний, которые входят в состав семантической базы ядерных знаний [6], достаточно элементарных рассуждений, подобных приведенному выше. Вместе с тем дескрипционные логики с сигнатурами SHIQ, SHOIN, SROIQ и их расширения позволяют строить менее тривиальные цепочки рассуждений, используя многоместные отношения и их реификацию, специального вида домены и диапазоны свойств. Возможна реализация индуктивных правил вывода на основе транзитивных свойств и составных аксиом вложенности свойств, вывод на основе прецедентов, логическое обоснование когнитивных гипотез и аргументация путем рассмотрения возможных альтернатив при ограничениях на кардинальность свойств. Исследование и апробация этих технологий рассуждений на графах знаний планируется в ближайшее время.

Архитектура семантического веб-портала

Программное обеспечение проекта [6] имеет унифицированный пользовательский интерфейс и включает в себя следующие компоненты: поисковые виджеты для быстрого погружения в графы ядерных знаний, интеллектуальный браузер RDF с встроенным визуальным ризонером, удаленные хранилища триплетов (репозитории) на облачной платформе Ontotext Cognitive Cloud, универсальное удаленное хранилище полнотекстового контента Google Drive, сетевой сервис Ontotext Text Analytics, вспомогательные поисковые агенты для быстрой навигации в международных базах знаний DBpedia и Wikidata, автономный поисковый агент "Contextual search" (подробнее о нем см. в работе [19]) для наполнения и актуализации имеющихся графов знаний. Все перечисленные компоненты программного обеспечения функционируют параллельно и независимо друг от друга.

На рис. 8 показана укрупненная диаграмма компонентов программного обеспечения проекта [6], выполненная в соответствии с международным стандартом UML 2 [17]. Находящийся слева на диаграмме компонент "Semantic Educational Web Portal"

обеспечивает пользовательский интерфейс, настройку и координацию работы иных компонентов, которые привязаны к одноименным вкладкам на центральной панели веб-портала. В верхней части диаграммы показаны компоненты, которые непосредственно обслуживают графы знаний. Компоненты "RDF browser" и "Search widgets" работают в связке друг с другом, обеспечивая пользователям комфортный поиск, визуальную навигацию и логические рассуждения на графах знаний. Эти компоненты взаимодействуют с семантической графовой базой данных Ontotext GraphDB, формируя и отправляя сетевые SPARQL-запросы в асинхронном режиме. Операции с семантическими репозиториями выполняются на удаленных облачных платформах с использованием моделей обслуживания DaaS, IaaS и PaaS, что обеспечивает масштабируемость задействованных сетевых сервисов.

Полнотекстовые учебные объекты и медийный контент размещаются в произвольных удаленных хранилищах данных (на рис. 8 показан файловый хостинг Google Drive), для видеоконтента более предпочтителен хостинг YouTube. Выбор конкретного удаленного хранилища не принципиален, могут использоваться произвольные облачные репозитории, оснащенные средствами отображения контента (Microsoft OneDrive, Яндекс.Диск и др.).

В нижней части диаграммы на рис. 8 показаны вспомогательные компоненты программного обеспечения проекта [6]. Облачный сервис Ontotext Text Analytics предоставляет инструментарий для семантического аннотирования сетевых ресурсов в ходе селекции и "авторинга" новых образовательных объектов для графов знаний. Агент "Contextual search" предназначен для осуществления разведочного поиска,

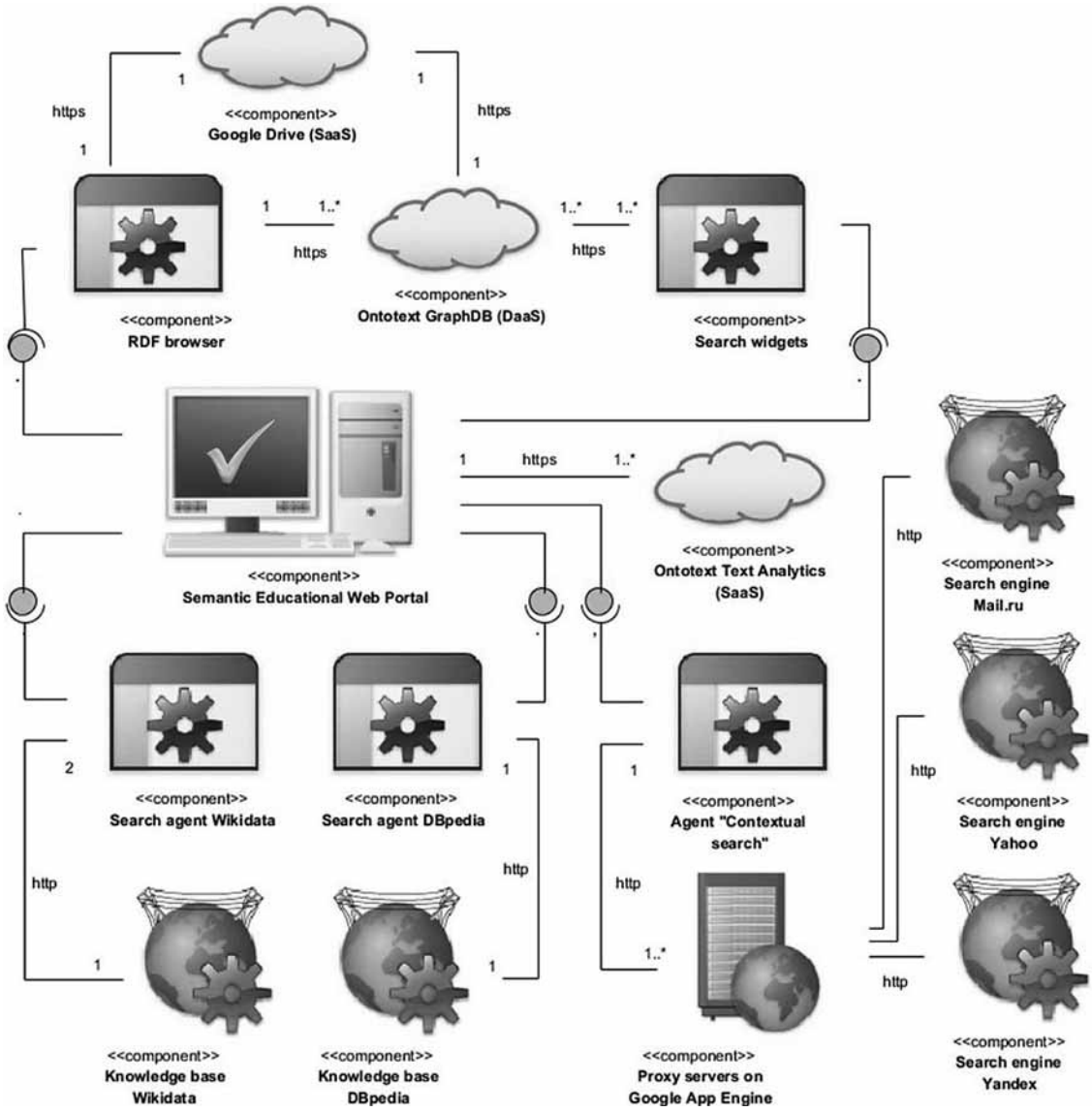

Рис. 8. Укрупненная диаграмма компонентов программного обеспечения проекта [6]

автоматизации процесса обнаружения во всемирной паутине и кластеризации релевантных информационных ресурсов — кандидатов для "авторинга" и пополнения (актуализации) семантической базы ядерных знаний. Автономные поисковые агенты DBpedia и Wikidata представляют собой публичные точки доступа к одноименным международным базам знаний, они снабжены библиотеками шаблонов употребительных поисковых запросов и способны доставлять найденный контент в различных форматах, включая графику. Серверная составляющая агента "Contextual search" функционирует на Google Cloud Platform (язык Python). Прочие компоненты представляют собой автономные JavaScript-объекты, которые функционируют на стороне клиента.

Обсуждаемый проект [6] открыт для свободного использования неограниченным кругом заинтересованных лиц, однако следует отметить два обстоятельства:

- на дату подготовки статьи работает пилотная версия семантической базы ядерных знаний, наполнение графов знаний продолжается;
- серверная часть программного обеспечения функционирует на облачных платформах по бесплатной квоте, поэтому возможны задержки с обслуживанием запросов.

Заключение

К 2018 г. созданы достаточный математический аппарат и программный инструментарий для работы с онтологиями, графами знаний и семантическими хранилищами данных, в том числе на облачных платформах. Существует значительное число публичных точек доступа SPARQL к международным базам знаний. Англоязычный сегмент всемирной паутины наполнен связанными открытыми данными. Это в основном справочные, библиографические, медийные и другие данные энциклопедического характера.

Инновационный потенциал проекта [6] применительно к образовательной деятельности достаточно высок. Студенты и преподаватели университетов расходуют немало времени и усилий на поиск полезной информации во Всемирной паутине, вместо того чтобы воспринимать и интерпретировать достоверный учебный материал. Современные реалии высшего образования таковы, что значительное число студентов и преподавателей не подозревают о существовании семантического веба и связанных открытых данных. В учебном процессе они продолжают использовать традиционные системы управления контентом (CMS), системы управления обучением (LMS) и виртуальные учебные среды (VLE), которые построены на таксономиях и тезаурусах. Студенты широко практикуют поиск информации во всемирной паутине по ключевым словам, используя для этого общедоступные поисковые системы, при этом они усваивают значительное количество недостоверных сведений, рекламы и информационного мусора. Существенную роль здесь играют сформировавшиеся традиции, а также легкость и большая скорость формирования поисковых запросов к публичным поисковым системам по сравнению с запросами к семантической паутине. При этих обстоятельствах выглядит оправданным известный скептицизм относительно того, что семантические интернет-технологии в их нынешнем виде получат признание и широкое распространение в университетской и в профессиональной средах. Есть все основания предполагать, что коммерциализированные поисковые системы, наряду с различными вариантами википедий, еще долго будут оставаться наиболее востребованными "универсальными учебниками" для многочисленных студентов и специалистов, если на смену им не станут приходить полноценные графы знаний и интеллектуальные поисковые агенты, снабженные адекватными средствами визуализации.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 18-07-00583А.

Список литературы

1. **W3C** RDF Schema 1.1. URL: http://www.w3.org/TR/rdf-schema (дата обращения: 21.08.2018).
2. **W3C** OWL 2 Web Ontology Language. URL: http://www.w3.org/TR/owl2-overview (дата обращения: 21.08.2018).
3. **Description** Logics. URL: http://dl.kr.org (дата обращения: 21.08.2018).
4. **Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P.** The Description Logic Handbook: Theory, Implementation and Applications. 2nd Ed. New York: Cambridge University Press, 2010.
5. **W3C** Semantic Web. URL: http://www.w3.org/standards/semanticweb (дата обращения: 21.08.2018).
6. **Семантическая** база ядерных знаний. URL: http://vt.obninsk.ru/x (дата обращения: 21.08.2018).
7. **Harmelen F., Lifschitz V., Porter B.** Handbook of Knowledge Representation. England, Oxford: Elsevier Science Oxford, 2008, 1035 p.
8. **The University** of Manchester. List of Reasoners. URL: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners (дата обращения: 21.08.2018).
9. **Reasoners** and rule engines: Jena inference support. URL: http://jena.apache.org/documentation/inference (дата обращения: 21.08.2018).
10. **Description** Logics. Proceedings of the 30th International Workshop on Description Logics. Montpellier, France, 18—21 July 2017. URL http://ceur-ws.org/Vol-1879 (дата обращения: 21.08.2018).
11. **Howard P.** Graph and RDF databases. Market Report Paper by Bloor 2018. URL: http://www.bloorresearch.com/technology/graph-databases (дата обращения: 21.08.2018).
12. **Ontotext** Cognitive Cloud. URL: http://cloud.ontotext.com (дата обращения: 21.08.2018).
13. **Google** Cloud Platform. URL: http://cloud.google.com (дата обращения: 21.08.2018).
14. **Amazon** Web Services. URL: http://aws.amazon.com/ru (дата обращения: 21.08.2018).
15. **SPARQL** 1.1 Query Language. URL: http://www.w3.org/TR/sparql11-query (дата обращения: 21.08.2018).
16. **Knowledge** Management for Nuclear Research and Development Organizations. IAEA TECDOC No. 1675. URL: http://www-pub.iaea.org/books/IAEABooks/8644/Knowledge-Management-for-Nuclear-Research-and-Development-Organizations (дата обращения: 21.08.2018).
17. **ISO** 19505 UML Part 2 Superstructure. URL: http://drive.google.com/file/d/0B0jk0QU2E5q9NVIwMFNieGxOZVU (дата обращения: 21.08.2018).
18. **Пример** онтологии "Учебные материалы МГУ и МИФИ по ядерной физике". URL: http://drive.google.com/file/d/1AIXMsm3cfAxR6NX220R4ZeFeoSFp0mj5 (дата обращения: 21.08.2018).
19. **Тельнов В. П.** Контекстный поиск как технология извлечения знаний в сети Интернет // Программная инженерия. 2017. Т. 8, № 1. С. 26—37.

Programming of Knowledge Graphs, Reasoning on Graphs

V. P. Telnov, telnov@bk.ru, **Yu. A. Korovin,** korovinyu@mail.ru, National Research Nuclear University "MEPhI", Obninsk, 249040, Russian Federation

Corresponding author:

Telnov Victor P., Associate Professor, National Research Nuclear University "MEPhI", Obninsk, 249040, Russian Federation,
E-mail: telnov@bk.ru

Received on September 09, 2018
Accepted on September 20, 2018

The paper is devoted to knowledge representation technologies, reasoning models and algorithms for generating cognitive hypotheses in intelligent systems. The emphasis is placed on applying problem—oriented graphs of knowledge in nuclear physics and nuclear power. The proposed technologies and methods cover the tasks of computer—aided detection and classification of nuclear knowledge and competences based on ontologies and representation of information objects in semantic graph data bases equipped with automated reasoning means. Templates for knowledge graphs designing for nuclear research and training centers are presented. Visual navigation and reasoning on the knowledge graphs are performed by means of special retrieval widgets and the smart RDF browser. The visual way of specifying the inference rules on the knowledge graphs is the highlight of the reasoner in the proposed project, which makes it stand out from the more traditional known reasoners, where inference rules are specified using logical predicates and SPARQL—like syntax. Operations with the semantic repository are implemented on cloud platforms through SPARQL queries and RESTful services. The proposed software solutions are based on the cloud computing using DaaS, IaaS and PaaS service models to provide scalability of data warehouses and network services. The architecture of the software in UML notation is presented, examples of real use of the created technologies and software are given.

Keywords: *semantic web, ontology, knowledge graph, graph database, cloud computing*

Acknowledgements: *The reported study was funded by Russian Foundation for Basic Research according to the research project № 18-07-00583 A.*

For citation:

Telnov V. P., Korovin Yu. A. Programming of Knowledge Graphs, Reasoning on Graphs, *Programmnaya Ingeneria,* 2019, vol. 10, no. 2, pp. 59—68

DOI: 10.17587/prin.10.59-68

References

1. **W3C** RDF Schema 1.1, available at: http://www.w3.org/TR/rdf-schema
2. **W3C** OWL 2 Web Ontology Language, available at: http://www.w3.org/TR/owl2-overview
3. **Description** Logics, available at: http://dl.kr.org
4. **Baader F., Calvanese D., McGuinness D., Nardi D., Patel-Schneider P.** *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd Ed. New York, Cambridge University Press, 2010.
5. **W3C** Semantic, available at: http://www.w3.org/standards/semanticweb
6. **The semantic** base of nuclear knowledge, available at: http://vt.obninsk.ru/x
7. **Harmelen F., Lifschitz V., Porter B.** *Handbook of Knowledge Representation*. England, Oxford: Elsevier Science Oxford, 2008, 1035 p.
8. **The University** of Manchester. List of Reasoners, available at: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners
9. **Reasoners** and rule engines: Jena inference support, available at: http://jena.apache.org/documentation/inference
10. **Description** Logics. Proceedings of the 30th International Workshop on Description Logics (Montpellier, France, 18—21 July 2017), available at: http://ceur-ws.org/Vol-1879
11. **Howard P.** Graph and RDF databases. Market Report Paper by Bloor 2018, available at: http://www.bloorresearch.com/technology/graph-databases
12. **Ontotext** Cognitive Cloud, available at: http://cloud.ontotext.com
13. **Google** Cloud Platform, available at: http://cloud.google.com
14. **Amazon** Web Services, available at: http://aws.amazon.com/ru
15. **SPARQL** 1.1 Query Language, available at: http://www.w3.org/TR/sparql11-query
16. **Knowledge** Management for Nuclear Research and Development Organizations. IAEA TECDOC No. 1675, available at: http://www-pub.iaea.org/books/IAEABooks/8644/Knowledge-Management-for-Nuclear-Research-and-Development-Organizations
17. **ISO** 19505 UML Part 2 Superstructure, available at: http://drive.google.com/file/d/0B0jk0QU2E5q9NVIwMFNieGxOZVU
18. **Example** of ontology "MSU and MEPhI educational materials on nuclear physics", available at: http://drive.google.com/file/d/1AIXMsm3cfAxR6NX220R4ZeFeoSFp0mj5
19. **Telnov V. P.** Kontekstny`j poisk kak texnologiya izvlecheniya znanij v seti Internet (Contextual search as a technology of knowledge extraction in the Internet), *Programmnaya Ingeneria,* 2017, vol. 8, no. 1, pp. 26—37 (in Russian).

DOI: 10.17587/prin.10.69-76

A. Yu. Morozov[1], Postgraduate Student, alex-icez@yandex.ru,
D. L. Reviznikov[1, 2], Professor, reviznikov@gmail.com,
[1] The State Federal-Funded Educational Institution of Higher Professional Training "Moscow Aviation Institute (National Research University)", Moscow,
[2] Federal Research Centre "Information and Control" of the Russian Academy of Sciences, Moscow

Corresponding author:

Morozov Aleksandr Yu., Postgraduate Student, The State Federal-Funded Educational Institution of Higher Professional Training "Moscow Aviation Institute (National Research University)", Moscow, 125993, Russian Federation, E-mail: alex-icez@yandex.ru

Modelling of Dynamic Systems with Interval Parameters on Graphic Processors

Received on June 07, 2018
Accepted on October 23, 2018

This paper presents the algorithm of adaptive interpolation on the basis of kd-tree for modeling of dynamic systems with interval parameters. The complexity of the algorithm is exponential against the number of interval parameters; therefore acceleration is a vital question. The main aspects of its parallelization and implementation using CUDA technology are presented here. The data structures used in the algorithm and their features in terms of parallel operation are described. The computation experiments conducted in some cases demonstrate hundred-fold acceleration in comparison with the central processor computing, which reveals the efficiency of suggested approaches to parallelization.

Keywords: *CUDA, parallelization, kd-tree, algorithm of adaptive interpolation, structured grids, interval dynamic systems*

А. Ю. Морозов[1], аспирант, e-mail: alex-icez@yandex.ru,
Д. Л. Ревизников[1, 2], д-р физ.-мат. наук, проф., e-mail: reviznikov@gmail.com,
[1] Федеральное государственное бюджетное образовательное учреждение высшего образования "Московский авиационный институт (национальный исследовательский университет)",
[2] Федеральный исследовательский центр "Информатика и управление" РАН, г. Москва

Моделирование динамических систем с интервальными параметрами на графических процессорах

Рассмотрен алгоритм адаптивной интерполяции на основе kd-дерева для моделирования динамических систем с интервальными параметрами. Сложность алгоритма — экспоненциальная относительно числа интервальных параметров, поэтому вопрос ускорения программной реализации для него является актуальным. Представлены основные аспекты распараллеливания алгоритма и его программной реализации с применением технологии CUDA. Описаны используемые в алгоритме структуры данных и их особенности с точки зрения параллельной обработки. Проведенные вычислительные эксперименты демонстрируют в ряде случаев ускорение на два порядка по сравнению с вычислениями на центральном процессоре, что свидетельствует об эффективности предлагаемых подходов к распараллеливанию.

Ключевые слова: *CUDA, распараллеливание, kd-дерево, алгоритм адаптивной интерполяции, структурированные сетки, интервальные динамические системы*

Introduction

This paper considers the problems of parallelization of the adaptive interpolation algorithm on the basis of a kd-tree [1] for modeling of dynamic systems with interval parameters. This algorithm allows finding the dependence of a problem solution on specific values of interval parameters in the form of a piecewise polynomial function with controlled precision. It deals with construction of a dynamic structural grid on the basis of a kd-tree [2] over the space formed by interval parameters and its adaptive reconstruction in the course of the problem solution. The complexity of the algorithm is exponential against the spatial dimension over which the grid is constructed, and parallelization for computational speedup is a natural need for it.

Over the last years, CUDA technology is developing extensively [3]. The technology allows using NVIDIA graphic processors for general computing [4—6]. The key difference between GPU and CPU lies in thousands of cores capable of performing computations simultaneously. Often, even when using even not very expensive graphic cards, a ten-fold or even a hundred-fold performance improvement can be achieved as compared to CPU computing. Please note that with all its attractiveness, GPU development has a set of features; for example, instead of presence of RAM alone, it has six different memory types and the possibility to directly interact with cache memory.

1. The algorithm of adaptive interpolation on the basis of a kd-tree

The process of modeling of a dynamic system containing m interval parameters can be presented in the form of a mapping sequence:

$$\begin{cases} y_{k+1} = F_k(y_k); \\ y_0 \in \mathbf{y_0} = \left(\mathbf{y_0^1},\ \mathbf{y_0^2},\ ...,\ \mathbf{y_0^m}, y_0^{m+1},\ ...,\ y_0^n\right)^{\mathrm{T}}; \\ k = 0,\ ...,\ N-1, \end{cases}$$

where $F_k : \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{y_0} \in \mathbb{IR}^n$ are interval initial conditions. Vector-function F_k meets the conditions which guarantee that there exists a unique solution for all $y_0 \in \mathbf{y}_0$. Hereinafter, intervals are highlighted in bold type and their set is designated as $\mathbb{IR}$:

$$\mathbb{IR} = \left\{\mathbf{x} = \left[\underline{\mathbf{x}}, \overline{\mathbf{x}}\right], \underline{\mathbf{x}}, \overline{\mathbf{x}} \in \mathbb{R}\right\}.$$

If a dynamic system is defined by an ODE system, then $F_k(y_k) = y(t_{k+1})$ and is represented by the solution of the corresponding ODE system:

$$\begin{cases} y' = f(y); \\ y(t_k) = y_k; \\ t \in \left[t_k,\ t_{k+1}\right]. \end{cases}$$

In case of a discrete dynamic system F_k describes the law of transition from one state to another.

For further convenient description of the algorithm, we will assume that the dynamic system is autonomous and that intervals are contained only in the initial conditions. If it is not the case, then additional fictitious equations are added to the system, and intervals with an independent variable are transferred to initial conditions.

The goal of the present algorithm lies in the construction of a piecewise polynomial function for each k; the function interpolates the dependence y_k on the specific values of interval parameters. The availability of such function makes it possible to quickly compute the solution for any initial point and, as a consequence, to quickly determine the interval estimation $\mathbf{y}_k$.

Suppose that at the step k the solution $y_k(y_0)$ is known. Let us construct an interpolation grid over the space formed by the interval initial conditions. Let us suppose that it is regular and uniform:

$$G_k^0 = \left\{\left(y_0^{i_1 i_2 \ldots i_m},\ y_k^{i_1 i_2 \ldots i_m}\right) \middle| 0 \leqslant i_l \leqslant p,\ 1 \leqslant l \leqslant m\right\},$$

$$y_0^{i_1 i_2 \ldots i_m} = \left(\begin{array}{c} \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{1}} + \dfrac{\overline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{1}} - \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{1}}}{p} i_1,\ \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{2}} + \dfrac{\overline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{2}} - \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{2}}}{p} i_2,\ ..., \\ \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{m}} + \dfrac{\overline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{m}} - \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{m}}}{p} i_m,\ y_0^{m+1},\ ...,\ y_0^n \end{array}\right)^{\mathrm{T}},$$

$$y_k^{i_1 i_2 \ldots i_m} = y_k\left(y_0^{i_1 i_2 \ldots i_m}\right),$$

where p is the degree of the interpolation polynomial.

For each $y_k^{i_1 i_2 \ldots i_m}$ let us compute $y_{k+1}^{i_1 i_2 \ldots i_m} = F_k\left(y_k^{i_1 i_2 \ldots i_m}\right) = F_k\left(F_{k-1}\left(...F_0\left(y_0^{i_1 i_2 \ldots i_m}\right)...\right)\right)$. In accordance with the obtained table function

$$G_{k+1}^0 = \left\{\left(y_0^{i_1 i_2 \ldots i_m},\ y_{k+1}^{i_1 i_2 \ldots i_m}\right) \middle| 0 \leqslant i_l \leqslant p,\ 1 \leqslant l \leqslant m\right\}$$

let us construct the interpolation polynomial $P_{k+1}^0(y)$, for example, in the Lagrangian form [7]:

$$P_{k+1}^0(y) = \sum_{i_1=0}^{p} ... \sum_{i_m=0}^{p} y_{k+1}^{i_1 i_2 \ldots i_m} l^{i_1 i_2 \ldots i_m}(y),$$

where

$$l^{i_1 i_2 \ldots i_m}(y) = \prod_{j=0}^{p} \prod_{\substack{l=0 \\ i_l \neq j}}^{m} \frac{p\dfrac{y^l - \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{l}}}{\overline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{l}} - \underline{\mathbf{y}}_{\mathbf{0}}^{\mathbf{l}}} - j}{i_l - j}.$$

If the interpolation error

$$e = \max_{y_0 \in \mathbf{y_0}} \left\| y_{k+1}(y_0) - P_{k+1}^0(y_0) \right\|$$

is greater than a certain predetermined number ε, then G_k^0 is divided into two grids G_k^1 and G_k^2 in such a way that their interpolation error estimation is less than e. The same operations are performed for them as for the grid G_k^0, and, when needed, they also are partitioned. If $y_{k+1}(y_0)$ is continuously differential $p + 1$ times with respect to y_0, we can demonstrate that the fragmentation process is finite. In general case, from the theoretical point of view, the interpolation error is estimated by majorization of the highest derivative with proportionality coefficients that define the interpolation grid, as well as its size. During each fragmentation, these coefficients decrease; as a result, the interpolation error decreases as well. In practice, the minimal mesh size, usually

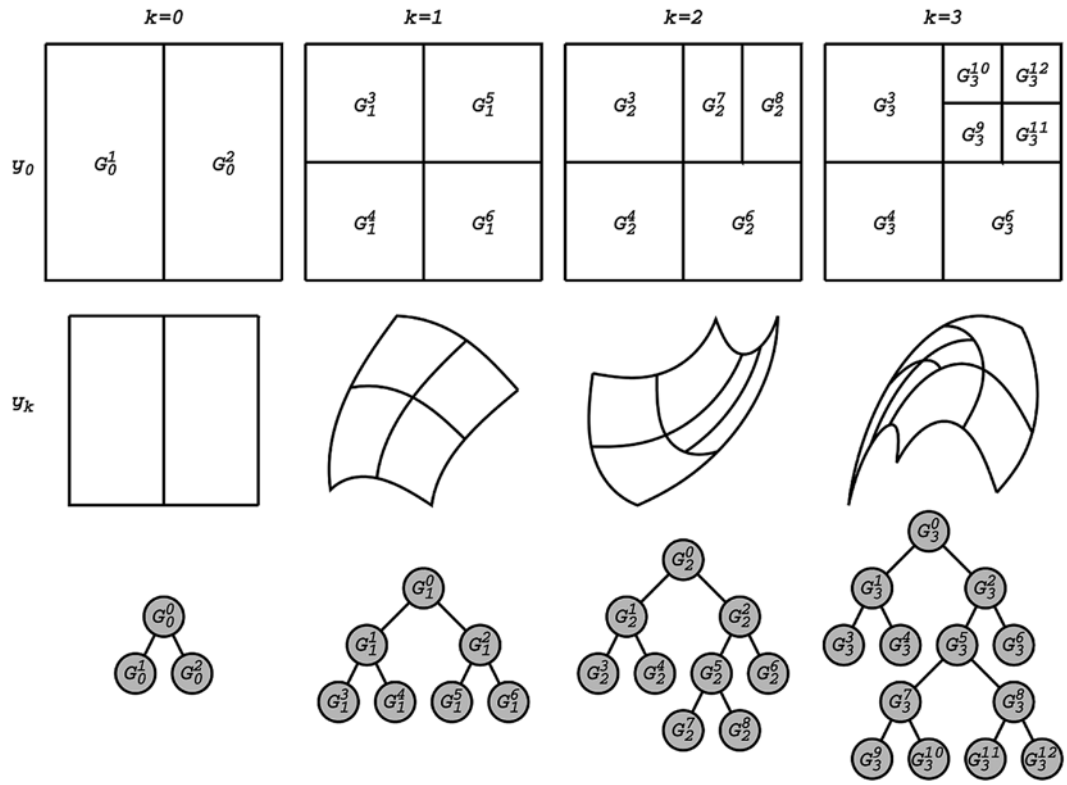

Fig. 1. Algorithm work demonstration

commensurable with machine epsilon, is defined; when this minimal size is reached, the fragmentation process stops and the corresponding area is marked as a "discontinuity point".

As a result, at the step $k + 1$ a kd-tree and the corresponding piecewise polynomial function will be obtained; the function interpolates the solution with the specified accuracy. There is no need to construct a kd-tree from scratch at each step, instead the tree obtained at the previous step is used, and it is reconstructed depending on the interpolation error estimation. In fig. 1, *a* kd-tree reconstruction process is shown.

In practice, interpolation error estimation is performed only for some points from the domain corresponding to each specific vertex of the tree. Two approaches to their selection can be singled out. The first involves supplementing a test set of points to each vertex in a random manner (fig. 2, *a*). The second approach is based on the reduction of interpolation order: some nodes are taken off from the grid, and their values are interpolated with respect to remaining nodes of reduced order (fig. 2, *b*). Although error estimation in the second case is inflated, it requires fewer computing resources, therefore let us choose this option.

In fig. 3, the geometric interpretation of interpolation error in the Euclidean norm for one vertex is shown. All lines were obtained by interpolation with respect to painted points and correspond to the lines in fig. 2, *b*.

Fig. 4 illustrates algorithm's pseudocode.

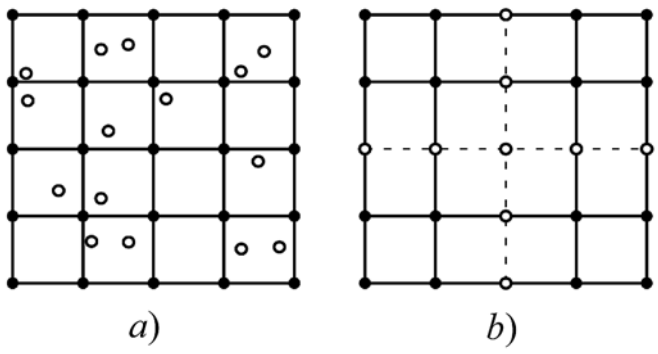

Fig. 2. Selection of points in which interpolation error estimation is performed

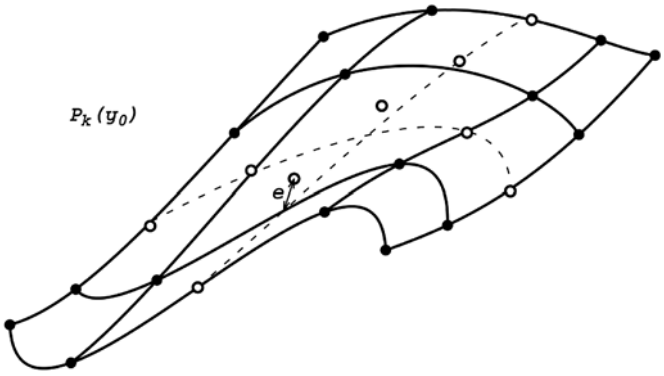

Fig. 3. Geometric interpretation of interpolation error for one vertex

```
G_0 = {G_0^0}
for k = 1,...,N:
    G_k = { }
    for G_{k-1}^i in G_{k-1}:
        G_k^i = { }
        for (y_0, y_{k-1}) in G_{k-1}^i:
            y_k = F_k(y_{k-1})
            G_k^i += (y_0, y_k)
        G_k += G_k^i
    for G_k^i in G_k:
        if eval_error(G_k^i) < ε and not is_leaf(G_k^i):
            if all(eval_error(G_k^{i'}) < ε for G_k^{i'} in childs(G_k^i)):
                G_k -= childs(G_k^i)
    for G_k^i in G_k:
        if eval_error(G_k^i) > ε and is_leaf(G_k^i):
            ({G_{k-1}^{i_1}, G_k^{i_1}}, {G_{k-1}^{i_2}, G_k^{i_2}}) = build_split(G_{k-1}^i)
            G_{k-1} += {G_{k-1}^{i_1}, G_{k-1}^{i_2}}
            G_k += {G_k^{i_1}, G_k^{i_2}}
```

Fig. 4. Pseudocode algorithm

2. Parallelization and implementation

Let us consider the algorithm presented above in terms of parallelization. One step of the algorithm can be divided into several phases:

1) application of F_k to all values connected with nodes;
2) computation of interpolation error in all tree vertices;
3) accomplishment of removal and partition of vertices.

Prior to the parallelization phase, it is necessary to analyze what performance improvement can be achieved. The first phase of the algorithm is collection of fully independent tasks that can be performed at the same time. If F_k is quite a complex function in terms of computation (for example, it represents an ODE solver of several thousand equations), then operations are immaterial compared to it in terms of man-hours, and the problem of their parallelization is neglected. Often in that case straightforward application of CUDA technology, when each thread computes one F_k, is inefficient. First, the process of $F_k(y_k)$ computation depending on y_k can differ greatly, which will lead to so-called thread divergence, when a part of threads of one warp goes along one computation branch and the other part goes along the other computation branch. Secondly, a large number of variables may become necessary, and, as a consequence, register spilling will happen (a part of variables will move from the register memory to the local memory), and the access speed will fall several hundred times. In that case, CUDA technology is used at the level of computing of F_k only. And if a computational cluster is available, MPI technology [8] can be used for task distribution among computing machines.

However if F_k does not require such substantial computational resources, for example, as is the case with a discrete dynamic system, when F_k is an explicitly defined function, then we can consider parallelization of operations over a tree and predominant use of CUDA technology.

For efficient use of CUDA technology it is preferable to store all data in arrays. Partly this is related to the fact that for obtaining maximum performance, each of 32 consecutive threads forming a warp should follow the same instruction and promptly access a continuous memory section [9]. If this is not accomplished, a significant decrease in operating speed occurs. That is why dynamic data structures, such as trees, are stored in arrays.

Usually, while storing a tree in an array, the positions of daughter vertices are clearly defined for each vertex, but this approach is efficient when the tree is balanced. However, in the present case, in the process of the algorithm operation, the obtained kd-tree can be very much unbalanced, which leads to substantial data fragmentation and excessive memory consumption. That is why two indices are additionally stored in the node structure. The indices designate where daughter vertices are stored. Adding new vertices is always carried out at the array end (fig. 5). With this storage method, data fragmentation is inevitable, since the tree is reconstructed frequently. At a certain point, when the number of blank locations becomes comparable to the number of occupied locations, the defragmentation process is started, which performs data compaction and recomputes all indices. One of the fundamental GPU algorithms (compact, [10]) is the cornerstone of the defragmentation process. Storing the list of free vertices is an alternative solution, but in general it does not help in avoiding data defragmentation and does not store the sequence of vertices added.

For availability of simultaneous access to all interpolation grid nodes and for elimination of node duplication in different vertices (fig. 6), they are stored separately

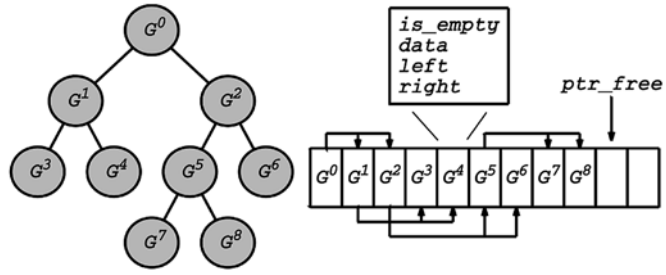

Fig. 5. Implementing a tree in an array

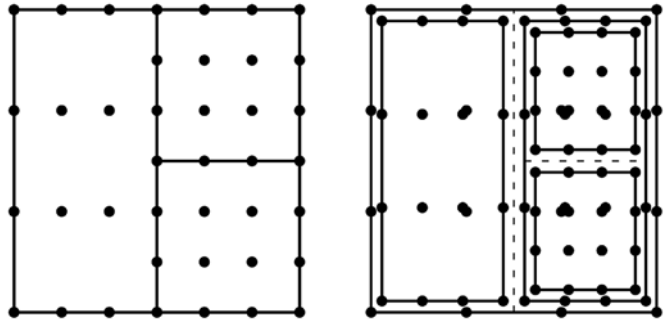

Fig. 6. Illustration of interpolation grid node duplication

from the kd-tree. For their storage, both a hash table [11] allocated in an array and a sorted array can be suited. In both structures, for decrease of queries to the global memory, the key is stored in an array separated from the data, which makes it possible not to refer to the associated data during a search [12]. If we limit ourselves to seamless access to elements with GPU requirement, then at $m = 1$ the sorted array outperforms the hash-table at the average by two-fold, at $m = 2$ they are roughly the same, and at $m > 2$ the hash-table becomes far more efficient. Partly it is due to the fact that the nodes located close in space in a sorted array are propelled into different sections of memory.

Let us take a closer look at the implementation of the hash-table (fig. 7). As in the implementation of a tree, adding elements at the array end is always performed here with periodic compaction. First, it allows maintaining insertion sequence; second, in the course of the algorithm operation, the insertion is performed with stacks of nodes belonging to one vertex, and it is desirable that they are located in a continuous section of the memory. The following formula is used as the hash function of a real valued vector:

$$hash(x) = \left(\sum_{i=1}^{m} \left\lfloor \left\lfloor x_i / \varepsilon_0 + \frac{1}{2} \right\rfloor \bmod p \right\rfloor \times p^{m-i} \right) \bmod q,$$

where ε_0 is a discretization parameter; p, q are prime numbers. For obtaining good distribution, the additional

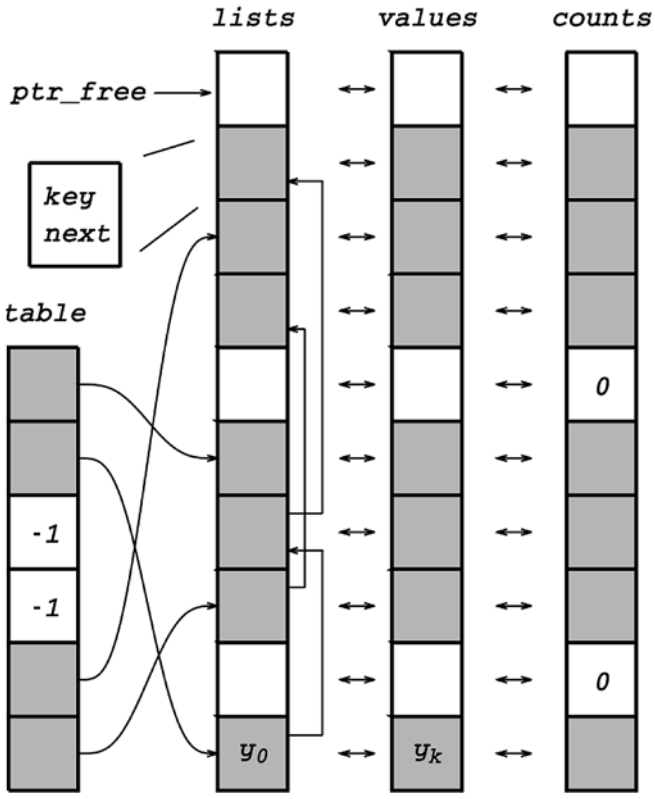

Fig. 7. Implementation of a hash table in arrays

transformation adopted from OpenJDK6 (structure HashMap) is used:

$$h := hash(x);$$
$$h := h \,\hat{}\, (h \gg 20) \,\hat{}\, (h \gg 12);$$
$$h := h \,\hat{}\, (h \gg 7) \,\hat{}\, (h \gg 4).$$

The computation of interpolation error in a certain vertex amounts to multiplication of the values associated with nodes by interpolation coefficients and their summation with subsequent comparison and search of the maximum deviation. For implementation of summation and search of the maximum value, another fundamental GPU algorithm (reduction [10]) is used. When using the approach based on the interpolation order reduction (see fig. 2, *b*), each interpolation coefficient can be represented as multiplication of one-dimensional interpolation coefficients. In this case, constant memory is used for their storage. It has high access speed due to caching.

When one vertex is partitioned into two vertices, it is necessary to compute all values associated with nodes by means of interpolation. If we assume that a vertex is always partitioned into halves by a hyperplane perpendicular to one of coordinate axis, then, exactly as in error computing, interpolation coefficients can be computed in advance and placed in the constant memory.

The partitioning process is performed in several stages, each of which is parallelized.

1. Identifying all vertices to be partitioned. If there are any, go to the second step.
2. Building all possible partition options.
3. Applying F_k map to all nodes of obtained vertices.
4. Computing interpolation error.
5. Selecting the best option. Moving to step 1.

When parallelizing the work with an interpolation grid, the number of nodes contained within it should be taken into account. If the number is not large, then parallelization at the level of one vertex can be inefficient. If all nodes are fitted into the shared memory, then the parallelization is performed in one way, if they are not fitted, then in another way. Generally, a specific implementation depends heavily on the parameters of solved problem: m, n, p and F_k.

Instead of kd-trees, other space partition trees can be used, for example, a quadtree in a two-dimensional case and an octree in a three-dimensional case [13]. On the one hand, they simplify the grid compaction process, since here it is possible to go without partition option enumeration. But on the other hand, 2^m vertices are created at once during each partition instead of two new vertices, and frequently it is unjustified. If we assume that a relevant space area should be partitioned in 2^m parts so that the error becomes acceptable, then in that case $2(2^m - 1)$ new vertices will be created in the kd-tree. When using uniform grids, the interpolation nodes in daughter vertices will fully duplicate the interpolation nodes in parent vertices, so even a two-fold increase of the number of vertices will have insignificant impact on the performance. In whole, the use of a kd-tree appears to be the optimal option in terms of computational cost.

3. Results

Let us perform some tasks using a central processing unit and a graphic processing unit and compare the operation time of separate phases of the algorithm. CPU characteristics: Intel Core i7-3770 3.40GHz, RAM — 16GiB DDR3 1333MHz. GPU characteristics: NVidia GeForce GTX 760, 2GiB, 1152 CUDA cores.

For integration of ODE systems, the fourth-order Runge—Kutta method was used. The integration step was 10^{-3}. The sample spacing $\tau = t_{k+1} - t_k$ was defined as constant and was equal to 5×10^{-2}. It determined the interval after which error estimation and reconstruction of a kd-tree were performed. The order of the interpolation polynomial was $p = 4$. Relative accuracy — 10^{-5} (while computing the error, normalization by "exact value" was performed).

Let us consider the Lotka-Volterra model [14] with interval initial conditions and one interval coefficient. The Cauchy problem is given by:

$$\begin{cases} x' = 4x - \frac{5}{4}xy - \alpha x^2; \\ y' = -2y + \frac{1}{2}xy - \frac{1}{20}y^2; \\ x(0) = x_0 \in [4,\ 5] \\ y(0) = y_0 \in [2.8,\ 3.2]; \\ \alpha \in [-0.05,\ 0.05]; \\ t \in [0,\ 50]. \end{cases} \quad (1)$$

This model describes interactions of predator-prey type. The system has the following features: at $\alpha < 0$ there is an unstable focus and the swing of the number of species increases, and at $\alpha > 0$ there is a stable focus and in time the system state converges to a steady state.

In fig. 8, the solution set in different moments is presented. The following picture can be clearly observed: a part of the set is directed to the point which corresponds to a stable focus, and a part of the set grows in size, which corresponds to an unstable focus. Such behavior of the system leads to constant increase in the grid density along α (fig. 9). As far as uncertainty is present in parameters, the solution set on the phase plane is just a projection of a three-dimensional set on two-dimensional space. The interval parameter α corresponds to an additional dimension.

In table 1 work time measurements for each phase of the algorithm are presented. Most of the time is spent on computing F_k. Since this phase of the algorithm is a set of fully independent tasks, the maximum speed gain from parallelization is observed here: about 30x. Computing of the error and reconstruction of a kd-tree require some interaction between threads and are performed not so trivially as computing of F_k, so the acceleration from parallelization amounts only to 8-fold.

Let us consider the following nonlinear ODE system:

$$\begin{cases} x' = -\frac{y}{\sqrt{\sin^2(x) + y^2} + 0.5}; \\ y' = \frac{x}{\sqrt{x^2 + \cos^2(y)} + 0.5}; \\ x(0) = x_0 \in [-5, 5]; \\ y(0) = y_0 \in [-5, 5]. \end{cases} \quad (2)$$

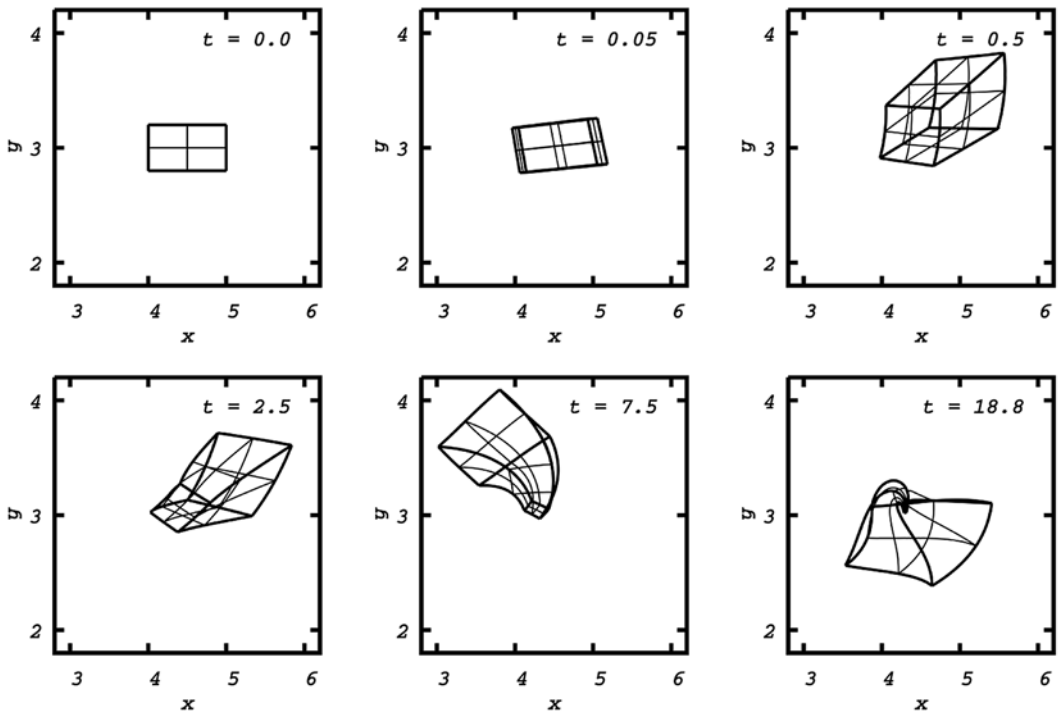

Fig. 8. Solution set of the system (1) in different moments *t*

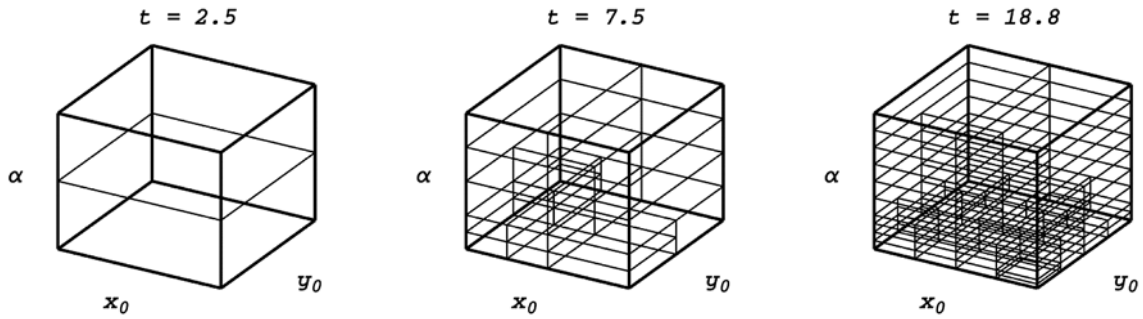

Fig. 9. Space partitions obtained by the integration of the system (1)

Table 1

Comparison of operation time of the algorithm phases in integration of the system (1)

Processor	Phase		
	Computing F_k, s	Computing of error, s	Reconstruction of kd-tree, s
CPU	68.438	8.058	10.881
GPU	2.338	1.074	1.319

Table 2

Comparison of operation time of the algorithm phases during integration of system (2)

Processor	Phase		
	Computing F_k, s	Computing of error, s	Reconstruction of kd-tree, s
CPU	651.702	4.196	65.004
GPU	5.506	0.4688	0.977

In fig. 10, the solution set of the system (2) in different moments is shown. It is deformed strongly and gradually begins to evolve in a spiral structure.

This ODE system is characterized by frequent reconstruction of the kd-tree during its integration, since the areas of condensation and coarsening of the grid move in space, which can be seen in fig. 11. This results in increased operation time of the corresponding algorithm phase (table 2). Another characteristic feature of tasks is that there are heavy functions, such as $\sqrt{}$, sin, cos in

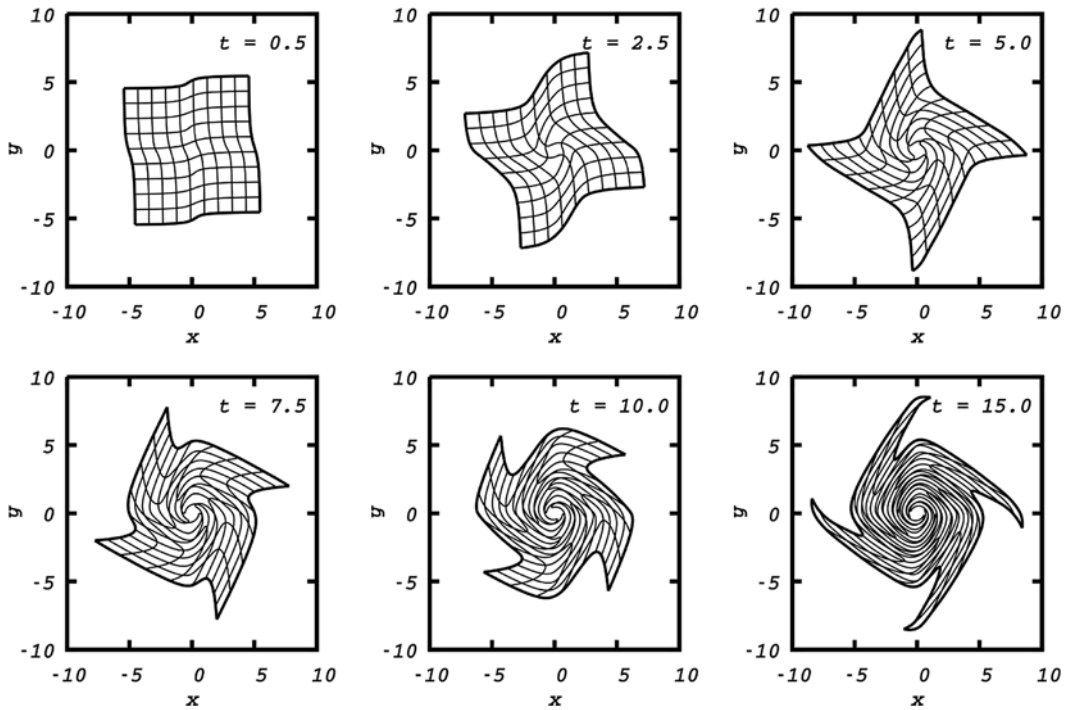

Fig. 10. The solution set of the system (2) in different moments *t*

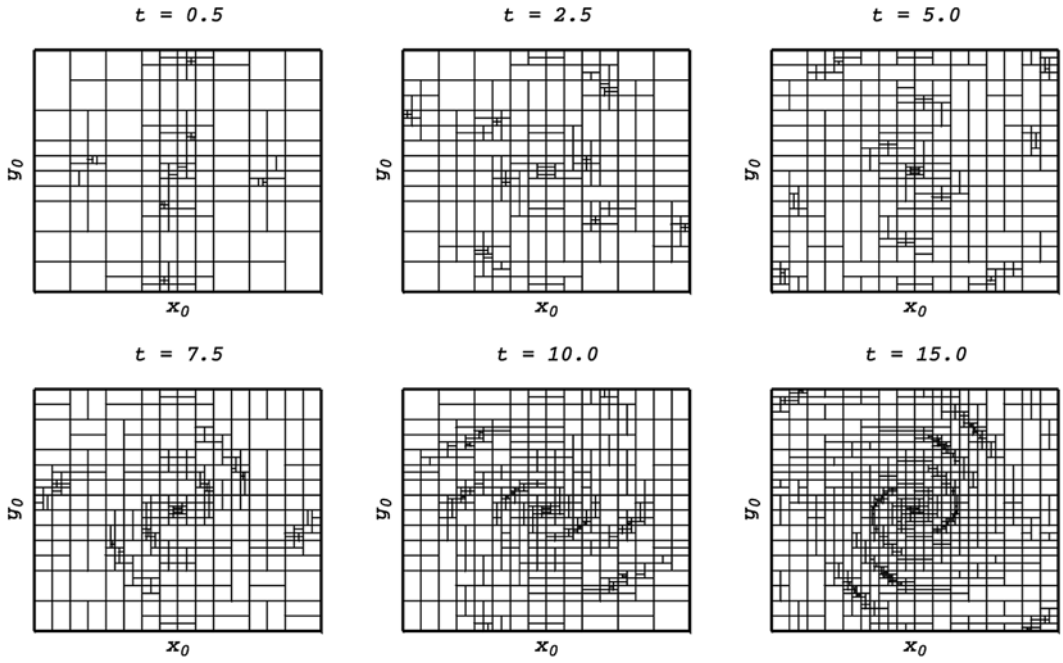

Fig. 11. Space partitions obtained by the integration of the system (2)

the right part of ODE, and this has a significant impact on the total operation time. Almost 100x speed gain when using parallelization results from the fact that the task allows the graphic processor to be fully loaded, unlike the above example where the obtained grids were not so dense.

Conclusion

This paper describes various aspects of parallelization and implementation of the adaptive interpolation algorithm on the basis of a kd-tree for modeling of dynamic systems with interval parameters. The algorithm consists of several phases, each of which is parallelized using the CUDA technology. According to computational experiments, most of the time during integration of ODE systems is spent on the first algorithm phase, which involves multiple computations of the same transformation. However, parallelization of the second and third phase involves some interaction between the threads, using of classical parallel algorithms and different memory types. Overall, in computations on not the most advanced card, almost 100x acceleration was achieved as compared to the central processing unit.

For citation:

Morozov A. Yu., Reviznikov D. L. Modelling of Dynamic Systems with Interval Parameters on Graphic Processors, *Programmnaya Ingeneria,* 2019, vol. 10, no. 2, pp. 69—76.

DOI: 10.17587/prin.10.69-76

References

1. **Morozov A. Y., Gidaspov V. Y., Reviznikov D. L.** Primenenie adaptivnoj interpolyacii v zadachah modelirovaniya dinamicheskih sistem s interval'nymi parametrami (Using adaptive interpolation in the tasks of modeling of dynamic systems with interval parameters), *Materials of XX Jubilee International Conference on Computational Mechanics and Modern Application Systems (VMSSPPS'2017): Theses. Intern. Conf.,* Alushta, 24—31 May 2017, Crimea, 2017, pp. 90—92 (in Russian).

2. **Brown R. A.** Building a Balanced k-d Tree in O(kn log n) Time, *Journal of Computer Graphics Techniques (JCGT),* 2015, vol. 4, no. 1, pp. 50—68.

3. **Shun-Liang J. K.** Gpu application in cuda memory, *Advanced Computing: An International Journal (ACIJ),* 2015, vol. 6, no. 2, pp. 1—10.

4. **Mikhaylyuk M. V., Trushin A. M.** Algoritmy opredeleniya kollizij sfer na GPU (Algorithms for determination of sphere collision on gpu), *Programmnaya Ingeneria,* 2017, vol. 8, no. 8, pp. 354—358 (in Russian).

5. **Semenov S. A., Reviznikov D. L.** Ehffektivnoe ispol'zovanie programmiruemyh graficheskih processorov v zadachah molekulyarno-dinamicheskogo modelirovaniya (Efficient application of programmable graphic processing unit in molecular-dynamic modeling tasks), *Sistemy i sredstva informatiki,* 2017, no. 4, pp. 109—121 (in Russian).

6. **Reviznikov D. L., Semenov S. A.** Osobennosti molekulyarno-dinamicheskogo modelirovaniya nanosistem na graficheskih processorah (Characteristics of molecular-dynamic modeling of nanosystems on graphic processors), *Programmnaya Ingeneria,* 2013, no. 2, pp. 31—35 (in Russian).

7. **Sauer T., Xu Y.** On multivariate lagrange interpolation, *Mathematics of Computation,* 1995, vol. 64, no. 211, pp. 1147—1170.

8. **Burns G., Daoud R., Vaigl J.** LAM: An Open Cluster Environment for MPI, *Proceedings of Supercomputing Symposium*, June 1994, Toronto, Canada, 1994, pp. 379—386.

9. **Dae-Hwan K.** Evaluation of the performance of GPU global memory coalescing, *Journal of Multidisciplinary Engineering Science and Technology (JMEST),* 2017, vol. 4, no. 4, pp. 7009—7013.

10. **Sengupta S., Harris M., Garland M.** Efficient Parallel Scan Algorithms for GPUs: NVIDIA Technical Report, dec. 2008.

11. **Alcantara D. A., Sharf A., Abbasinejad F., Sengupta S.** and other. Real-time Parallel Hashing on the GPU, *ACM Transactions on Graphics* (*Proceedings of ACM SIGGRAPH Asia 2009),* 2009, vol. 28, no. 5, pp. 1—9.

12. **Mei G., Tian H.** Performance Impact of Data Layout on the GPU-accelerated IDW Interpolation, *SpringerPlus*, 2016, vol. 5, pp. 1—18.

13. **Dinesh P. M., Sartaj S.** *Handbook of Data Structures And Applications,* Chapman & Hall / CRC, 2004, 1321 p.

14. **Arnold V. I.** *Obyknovennye differencial'nye uravneniya* (Ordinary differential equations), Izhevsk, Izhevskaya respublikanskaya tipografiya, 2000, 368 p. (in Russian).

УДК 514.18 DOI: 10.17587/prin.10.77-86

Е. В. Конопацкий, канд. техн. наук, доц., e-mail: e.v.konopatskiy@mail.ru, ГОУВПО "Донбасская национальная академия строительства и архитектуры", г. Макеевка

Подход к построению геометрических моделей многофакторных процессов и явлений многомерной интерполяции

Изложен подход к геометрическому моделированию многофакторных процессов и явлений методом многомерной интерполяции. Основу этого подхода составляет использование дуг алгебраических кривых, проходящих через наперед заданные точки. Приведены примеры построенных с использованием предложенного подхода геометрических моделей физико-механических свойств и состава композиционных строительных материалов.

Ключевые слова: *геометрическая модель, многофакторный процесс, многомерная интерполяция, функция отклика, факторы влияния, точечное уравнение, дерево геометрической модели, дуга кривой, отсек поверхности, отсек гиперповерхности*

Введение

Моделирование многофакторных процессов и явлений является важной частью любых научных исследований, поскольку обеспечивает решение не только задач оптимизации, но и прогнозирования поведения процесса или явления при изменении некоторого числа факторов. Среди различных методов моделирования таких объектов выделяется геометрическое моделирование. Его использование позволяет найти аналитическую зависимость между любыми, даже совершенно несоизмеримыми и несовместимыми факторами и функцией отклика, представив ее в виде единого геометрического объекта. Причем размерность пространства полученного геометрического объекта напрямую зависит от числа факторов, влияющих на функцию отклика.

Многомерная интерполяция не является чем-то новым. Ее важность подчеркивается в работе [1] применительно к использованию для моделирования многомерных таблиц в физике и технике. Примером такой двухмерной таблицы могут служить таблицы термодинамических функций газов, где независимыми переменными обычно являются температура и плотность. Трехмерные таблицы составляют и используют значительно реже, но не по причине, что таких зависимостей нет, а потому, что таблицы слишком громоздки. Четырехмерных таблиц практически нет, хотя в физике немало задач с большим числом параметров. Например, проводимость плазмы $\sigma(T, \rho, E, H)$ зависит от ее температуры T, плотности ρ, а также от напряженностей электрического (если сказываются нелинейные эффекты) и магнитного полей (E, H). Следует отметить, что с появлением современных вычислительных машин, способных оперировать большими объемами информации, использование таких таблиц, а вместе с ними и многомерной интерполяции, в значительной мере возросло.

Научных подходов к решению задач многомерной интерполяции встречается достаточно много. Например, работе [2] приведено сравнение различных способов многомерной интерполяции, которые включают: интерполяцию многочленом Лагранжа, полиномиальную рекурсивную интерполяцию и рациональную интерполяцию. В работе [3] рассмотрена интерполяция периодической функции многих переменных, заданной в узлах обобщенной параллелепипедной сетки целочисленной решетки. Задача интерполяции функции, заданной на регулярной сетке, для случая большого числа переменных приведена в работе [4]. Решению задачи многомерной сплайн-интерполяции на хаотических сетках с огромным числом интерполяционных точек посвящена работа [5]. Кроме того задачи многомерной интерполяции и аппроксимации могут быть решены на основе теории случайных функций [6].

Анализируя разнообразие научных подходов к решению задач многомерной интерполяции, можно сделать вывод о наличии в них как универсальных классических, так и инновационных модифицированных методов решения. Тем не менее каждый метод в большей степени ориентирован на решение конкретной практической задачи. Применение его к другим задачам может иметь как позитивные, так и негативные последствия.

Теоретические основы геометрического моделирования многофакторных процессов и явлений методом многомерной интерполяции

Основное отличие геометрических моделей от их математических аналогов, полученных с помощью многомерной интерполяции, заключается в том, что результат моделирования представляет

собой не абстрактный математический, а конкретный геометрический параметризированный объект, обладающий наперед заданными геометрическими свойствами. В такой постановке наиболее близкими по методологии являются работы [7, 8]. В них задача геометрического моделирования ставится следующим образом: в пространстве размерности n задан набор фиксированных точек (геометрическое отображение экспериментальных или статистических данных в некоторой системе координат), через которые необходимо построить геометрический объект и найти его уравнение. В работе [8] приведены как аппарат геометрического моделирования, так и частные его случаи в виде широкого спектра решенных с помощью многомерной интерполяции практических задач. Однако используемый в работах [7, 8] аппарат моделирования ограничивается использованием геометрических объектов второго порядка, что в значительной мере сужает сферу его практического использования в инженерной практике.

В основу геометрического моделирования многофакторных процессов и явлений методом многомерной интерполяции [9] (в контексте настоящей работы) положен простейший принцип принадлежности одного геометрического объекта к другому из начертательной геометрии [10]. Например, прямая линия принадлежит плоскости, если две точки этой прямой принадлежат плоскости. В свою очередь, точка принадлежит плоскости, если она принадлежит прямой, лежащей в этой плоскости. Рассматривая прямую линию как частный случай кривой, а плоскость — как частный случай поверхности, получим, что для того чтобы провести через исходные точки поверхность, нужно организовать их в виде опорных (в начертательной геометрии используется термин "направляющих") линий, объединив их впоследствии с помощью образующей. Тогда все точки, принадлежащие опорным линиям, будут принадлежать искомой поверхности, которая, в свою очередь, принадлежит трехмерному пространству. Причем образующая также является кривой линией, только проходящей через текущие точки опорных линий искомой поверхности. Обобщая этот подход на многомерное пространство, получим следующее утверждение. Для того чтобы гиперповерхность четырехмерного пространства была носителем наперед заданных точек, координаты которых соответствуют исходной экспериментально-статистической информации, необходимо чтобы исходные точки принадлежали семейству поверхностей, объединенных в гиперповерхность образующей линией, проходящей через текущие точки семейства поверхностей. Тогда задача опять сводится к определению кривых линий, проходящих через наперед заданные точки, только таких линий становится больше. Таким образом, формируется дерево описывающего модель процесса, которое можно представить в виде некоторой геометрической схемы. По этой модели впоследствии может быть построен и аналитически описан геометрический объект. Аналогичным образом можно построить геометрическую модель процесса, принадлежащую пространству любой размерности. Собственно говоря, размерность пространства напрямую

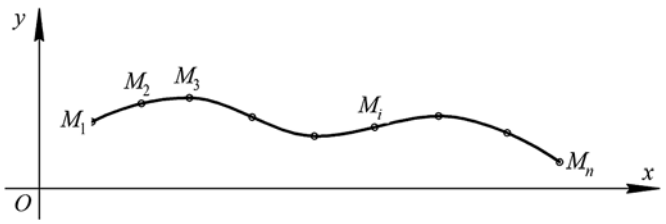

Рис. 1. Геометрическая модель однофакторного процесса

зависит от числа факторов, влияющих на процесс, и потому получивших название — факторы влияния. Каждому фактору влияния соответствует параметр геометрической модели процесса. При этом чем больше параметров определяют геометрический объект, тем в более высокоразмерном пространстве он находится.

С учетом изложенного выше однофакторный процесс или явление геометрически можно представить однопараметрическим множеством точек — линией двухмерного пространства, проходящей через наперед заданные точки (рис. 1), которая определяется следующим точечным уравнением [11, 12]:

$$M = \sum_{i=1}^{n} M_i p_i(u), \qquad (1)$$

где M — текущая точка дуги кривой линии (здесь прямые линии рассматриваются как кривые с нулевой кривизной), проходящей через наперед заданные точки; M_i — исходные точки, через которые должна проходить искомая дуга кривой, координаты этих точек соответствуют исходным экспериментально-статистическим данным; $p_i(u)$ — функции от параметра u (методика их определения на основе полиномов Бернштейна изложена ниже); u — текущий параметр, который изменяется от 0 до 1, при этом значение $u = 0$ определяет начальную точку дуги кривой M_1, а $u = 1$ — конечную точку дуги кривой M_n; n — число исходных точек дуги кривой, соответствующее числу экспериментов; i — порядковый номер исходной точки, через которую должна проходить искомая дуга кривой, который изменяется от 1 до n и соответствует порядковому номеру эксперимента.

При этом обязательным является выполнение условия принадлежности геометрического объекта к определенному пространству. В данном случае принадлежность дуги кривой к плоскости xOy декартовой системы координат: $\sum_{i=1}^{n} p_i(u) = 1$. Следует отметить, что точечные уравнения, полученные на инвариантах параллельного проецирования, являются справедливыми как для декартовой, так и для аффинной систем координат.

Точечное уравнение (1) представляет собой символьную запись. Выполнив покоординатный расчет, суть которого заключается в замене в точечном уравнении символьного обозначения точек на их конкретные координаты, получим систему параметрических уравнений

$$\begin{cases} x_M = \sum_{i=1}^{n} x_{M_i} p_i(u) \\ y_M = \sum_{i=1}^{n} y_{M_i} p_i(u) \end{cases},$$

где x_M и y_M — координаты текущей точки дуги кривой, проходящей через наперед заданные точки; x_{M_i} и y_{M_i} — координаты исходных точек, соответствующих исходным экспериментальным данным.

Аналогичным образом любое точечное уравнение можно представить в виде системы параметрических уравнений. Эта система представляет собой аналитическое описание проекций дуги плоской кривой на оси глобальной системы координат. При этом на рис. 1 оси *Ox* соответствует фактор, влияющий на процесс или явление, а оси *Oy* — искомая функция отклика. Связь между функцией отклика и фактором влияния аналитически осуществляется с помощью параметра *u*, который изменяется от 0 до 1. Графически она представляет собой линию проекционной связи, если представить проекции кривой на комплексном чертеже Монжа. Далее все уравнения будут представлены в символьной точечной форме без перехода к системе параметрических уравнений для более краткого представления.

Следует отметить, что каждой точке, через которую проходит геометрический объект, соответствует конкретное значение экспериментально-статистической информации исходных данных искомой модели процесса или явления. А исходные данные, представленные геометрически, расположены в некоторой области и для их описания геометрический объект должен располагаться в пределах этой области. Таким образом, для описания однофакторного процесса используется не вся кривая линия, а только ее фрагмент — дуга кривой, что соответствует изменению параметра от 0 до 1. Соответственно двухфакторный процесс описывает отсек поверхности, трехфакторный — отсек гиперповерхности и т. д. При этом значения параметров, вне зависимости от их числа, принадлежат интервалу от 0 до 1. Для того чтобы сделать прогноз на основе полученной геометрической модели, необходимо использовать значения соответствующих параметров либо ниже нуля, либо больше единицы.

В свою очередь, двухфакторный процесс определяется двухпараметрическим множеством точек — поверхностью трехмерного пространства, проходящей через наперед заданные точки (рис. 2):

$$\begin{cases} M_1 = \sum_{j=1}^{n} M_{1j} p_{1j}(u) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ M_i = \sum_{j=1}^{n} M_{ij} p_{ij}(u) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ M_m = \sum_{j=1}^{n} M_{mj} p_{mj}(u) \\ M = \sum_{i=1}^{m} M_i q_i(v) \end{cases}, \qquad (2)$$

где M — текущая точка отсека поверхности, проходящей через наперед заданные точки в количестве $m \times n$, она же текущая точка образующей дуги кривой; M_i — текущая точка i-й опорной (направляющей) дуги кривой, проходящей через наперед заданные точки, при этом движение всех опорных линий согласовывается с помощью единого параметра u; M_{ij} — исходные точки, через которые должен проходить искомый отсек поверхности, координаты этих точек соответствуют исходным экспериментально-статистическим данным; $p_{ij}(u)$ — функции от параметра u (в данном случае полиномиальные, но в общем случае могут быть и другие, в том числе трансцендентные), определяющие вид опорных линий; $q_i(v)$ — функции от параметра v, определяющие вид образующей линии; m — число опорных линий; n — число исходных точек в каждой опорной линии (на рис. 2 n одинаково для всех опорных линий, но это не обязательно); i — порядковый номер опорной линии; j — порядковый номер исходной точки в каждой опорной линии; u — текущий параметр опорных линий отсека поверхности, который изменяется от 0 до 1; v — текущий параметр образующей линии отсека поверхности, который также изменяется от 0 до 1.

Для описания геометрической модели двухфакторного процесса (рис. 2) используется трехмерная система координат. Причем осям *Ox* и *Oy* соответствуют факторы, влияющие на процесс, а оси *Oz* — искомая функция отклика.

Трехфакторный процесс определяется трехпараметрическим множеством точек — гиперповерхностью четырехмерного пространства, проходящей через наперед заданные точки (рис. 3):

$$\begin{cases} M_{ij} = \sum_{k=1}^{l} M_{ijk} p_{ijk}(u) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ M_i = \sum_{j=1}^{n} M_{ij} q_{ij}(v) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ M = \sum_{i=1}^{m} M_i r_i(w) \end{cases}, \qquad (3)$$

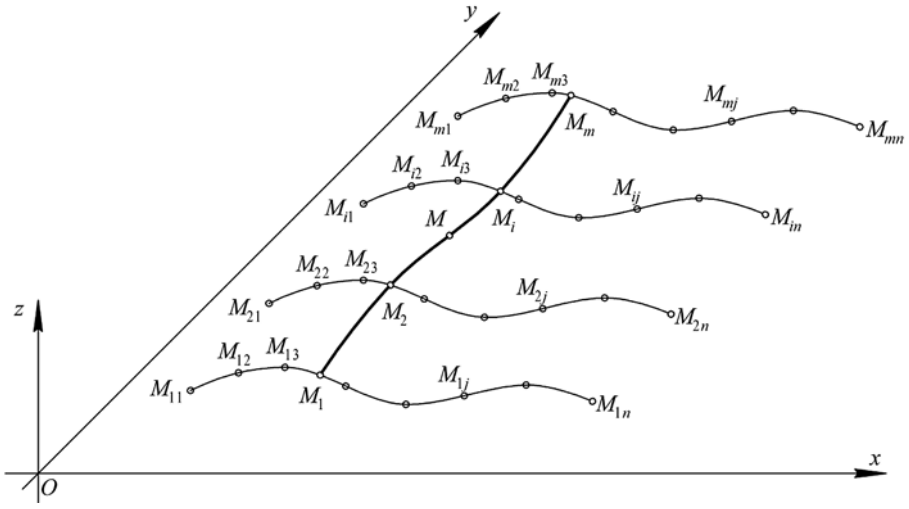

Рис. 2. Геометрическая модель двухфакторного процесса

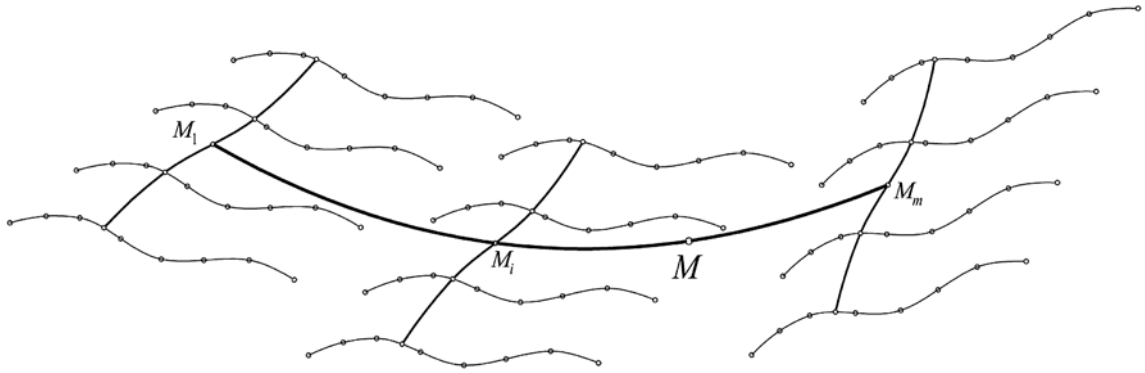

Рис. 3. Дерево геометрической модели трехфакторного процесса

где M — текущая точка отсека гиперповерхности, проходящей через наперед заданные точки в количестве $m \times n \times l$; M_i — текущая точка образующей i-го отсека поверхности, проходящая через наперед заданные точки, которая является опорной для построения гиперповерхности; M_{ij} — текущая точка j-й опорной дуги кривой, проходящей через наперед заданные точки; M_{ijk} — исходные точки, через которые должен проходить искомый отсек гиперповерхности, координаты этих точек соответствуют исходным экспериментально-статистическим данным; $p_{ijk}(u)$ — функции от параметра u, определяющие вид опорных линий; $q_{ij}(v)$ — функции от параметра v, определяющие вид образующих линий отсеков поверхностей; $r_i(w)$ — функции от параметра w, определяющие вид образующей линии отсека гиперповерхности; l — число исходных точек в каждой опорной линии; n — число опорных линий для построения опорных поверхностей; m — число опорных поверхностей для построения гиперповерхности; i — порядковый номер опорной поверхности; j — порядковый номер опорной линии; k — порядковый номер исходной точки в каждой опорной линии; u, v и w — текущие параметры, которые изменяются от 0 до 1.

Выполнив покоординатный расчет, получим проекции гиперповерхности на оси глобальной системы координат, для которой оси Ox, Oy и Oz на рис. 2 будут соответствовать факторам влияния, а Ot — искомой функции отклика.

Обобщая предложенный подход, получим геометрическую модель n-факторного процесса, как n-параметрическое множество точек или гиперповерхность $(n + 1)$-го пространства. Прохождение всего геометрического объекта через наперед заданные точки обеспечивается прохождением всех точек через направляющие линии на каждом этапе формирования дерева модели. При этом любая линия представляется организованным определенным образом однопараметрическим множеством точек. Таким образом, для геометрического моделирования многофакторных процессов и явлений необходимо иметь специальную библиотеку дуг кривых, проходящих через наперед заданные точки.

Моделирование дуг кривых, проходящих через наперед заданные точки

При моделировании дуг кривых, проходящих через наперед заданные точки, возможны два подхода — использование заранее заданных кривых линий и использование составных кривых. Чтобы сделать выбор, необходимо знать конкретные условия моделирования, поскольку составные кривые более эффективно используются для большого количества исходной экспериментально-статистической информации. Однако обязательным условием является возможность представления этих данных на регулярной сети точек. Это могут быть различного рода сплайны [6, 13] и обводы [14—16]. Однако сложность визуального восприятия многомерного пространства приводит к необходимости использования не зрительной, а логической наглядности, основанной на методах обобщения и аналогии. Поэтому использование единой кривой является более предпочтительным и дает возможность не только определить оптимальные значения факторов, влияющих на функцию отклика, но и в перспективе спрогнозировать ее поведение. Это приводит к необходимости создания методики определения универсальных дуг кривых, которые должны быть носителями исходных точек вне зависимости от их взаимного положения. Тогда их можно систематизировать в виде регулярной или нерегулярной сети точек.

Рассмотрим методику модулирования дуг алгебраических кривых, проходящих через наперед заданные точки, на основе полиномов Бернштейна [17]. Пусть задана ломаная линия $A_1A_2...A_n, A_{n+1}$. Дуга алгебраической кривой n-го порядка имеет следующее точечное уравнение:

$$M = A_1\bar{t}^n + A_2C_n^1\bar{t}^{n-1}t + A_3C_n^2\bar{t}^{n-2}t^2 + ... + \\ + A_nC_n^{n-1}\bar{t}t^{n-1} + A_{n+1}t^n = \sum_{i=0}^{n} A_{i+1}C_n^i\bar{t}^{n-i}t^i, \quad (4)$$

где M — текущая точка дуги кривой; $A_i,..., A_{n+1}$ — исходные точки, определяющие форму дуги кривой; $C_n^1, ..., C_n^i$ — коэффициенты Бернштейна; t — те-

кущий параметр, который изменяется от 0 до 1; $\overline{t} = 1 - t$ — дополнение параметра до 1.

Тогда исходное точечное уравнение кривых, полученных на основе полиномов Бернштейна, которые в общем случае находятся в пространстве размерности n, являются кривыми $(n + 1)$-й кривизны (например, пространственная дуга кривой третьего порядка в общем случае является дугой двоякой кривизны), может быть записано в следующем виде:

$$M = \sum_{i=0}^{n} A_{i+1} \frac{n!}{i!(n-i)!} \overline{t}^{n-i} t^{i}. \quad (5)$$

Использовав равномерное распределение текущего параметра $t = \frac{j}{n}$, получим

$$M_{j+1} = \sum_{i=0}^{n} A_{i+1} \frac{n!}{i!(n-i)!} \left(\frac{n-j}{n}\right)^{n-i} \left(\frac{j}{n}\right)^{i}. \quad (6)$$

Переопределим точки ломаной линии $A_1A_2...A_n$, A_{n+1} через точки $M_1M_2...M_n$, M_{n+1}, которые принадлежат дуге кривой, определенной с помощью уравнения (6), пропорционально изменяя значение параметра t от 0 до 1. В результате получим систему линейных $n + 1$ алгебраических уравнений, каждая строка которой определяется следующим уравнением:

$$\sum_{i=0}^{n} \frac{n!}{i!(n-i)!} (n-j)^{n-i} (j)^{i} A_{i+1} = n^{n} M_{j+1}. \quad (7)$$

Решив эту систему уравнений методом Крамера и подставив в исходное уравнение (5), получим уравнение дуги кривой, проходящей через наперед заданные точки $M_1M_2...M_n$, M_{n+1}..

Полученная дуга кривой представлена в точечной форме (т. е. в символьной форме, для которой вместо координат используются непосредственно точки). Для практического использования полученных уравнений необходимо выполнить покоординатный расчет [11, 12]. При этом одно точечное уравнение заменяется системой параметрических уравнений, которые представляют собой аналитическое описание проекций текущей точки на оси глобальной системы координат. Здесь следует отметить очень важную отличительную особенность полученного уравнения. Для точечных уравнений принадлежность геометрического объекта к пространству конкретной размерности определяется суммой функций от параметра t, которая обязательно должна быть равна 1. Поскольку функции от параметра t определяются биномом Ньютона, который раскладывается для параметра t и его дополнения до 1, то условие принадлежности дуги кривой конкретному пространству будет выполняться вне зависимости от размерности пространства. Иными словами, полученные параметрические уравнения дуги кривой могут быть использованы для пространства любой размерности.

Другой важной особенностью полученной дуги кривой является равномерное распределение параметра, изначально заложенное в методику определения дуги кривой, проходящей через наперед заданные точки. При этом для каждой конкретной координатной оси, имеющей равномерное распределение координат исходных точек, справедлива линейная зависимость между натуральным значением фактора, принадлежащего i-й оси проекций, и текущим параметром:

$$x_i = nl_i t + b_i, \quad (8)$$

где x_i — i-я ось проекций глобальной системы координат; n — порядок дуги кривой; b_i — начальное значение фактора влияния, соответствующее i-й оси проекций; l_i — шаг равномерного распределения проекции исходных точек на i-ю ось.

Эта особенность в значительной мере сокращает объем необходимых вычислений при моделировании геометрических объектов на регулярной сети точек. Сама методика при этом носит универсальный характер и без внесения каких-либо модификаций может в полном объеме использоваться как для регулярной, так и для нерегулярной сети точек.

Алгоритм геометрического моделирования многофакторных процессов и явлений методом многомерной интерполяции

Исходя из изложенного выше, для решения задачи геометрического моделирования и оптимизации многофакторных процессов и явлений с помощью многомерной интерполяции можно выделить отмеченную далее концептуальную последовательность действий.

1. Анализ, сортировка и предварительная обработка исходных данных.

2. Составление геометрической схемы конструирования геометрического объекта, так называемого дерева геометрической модели, которая соответствует исходной экспериментальной или статистической информации.

3. Выбор дуг интерполирующих кривых, проходящих через наперед заданные точки, которые на основе дерева геометрической модели будут формировать геометрический объект.

4. Моделирование геометрического объекта в соответствии с деревом геометрической модели на основе дуг кривых, проходящих через наперед заданные точки, которые бывают двух видов: фиксированные, являющиеся графическим отображением исходной экспериментальной или статистической информации, и текущие, которые своим движением заполняют пространство и тем самым формируют геометрический объект.

5. Проверка достоверности полученной геометрической модели. Наиболее удобно это делать с помощью коэффициента детерминации. Особенность заключается в том, что традиционно для оценки моделирования используются исходные точки, в данном случае полученный геометрический объект уже является носителем исходных точек. Это является непременным условием моделирования и обеспечи-

вает коэффициент детерминации, равный единице. То есть с точки зрения математического моделирования такая модель уже имеет наивысшую степень достоверности. Однако если есть такая возможность, для еще более качественной проверки достоверности геометрической модели можно использовать дополнительные точки, определив их координаты между исходными точками (узлами интерполяции).

6. Аппроксимация полученной модели более простым геометрическим объектом. Этот этап является не обязательным, но в некоторых случаях позволяет значительно сократить итоговые вычисления, оказывая при этом незначительное влияние на достоверность полученной модели, которая должна быть подтверждена соответствующим значением коэффициента детерминации. Для аппроксимации геометрической модели процесса могут быть эффективно использованы дуги алгебраических кривых, проходящие через наперед заданные точки.

7. Определение экстремумов на поверхности геометрического объекта функции отклика, которые могут иметь особое значение для исследуемого процесса или явления и в большинстве случаев позволяют оптимизировать результат моделирования.

Рассмотрим далее предложенный подход, реализованный на нескольких примерах в виде геометрических моделей двухфакторного [18], трехфакторного [19] и четырехфакторного [20] процессов.

Оптимизация состава комбинированного заполнителя мелкозернистого бетона с помощью многомерной интерполяции

Физико-механические свойства любого композиционного строительного материала, в том числе и мелкозернистого бетона, напрямую зависят от его состава. Поэтому оптимизация состава композиционных строительных материалов является актуальной научной прикладной задачей.

Исследуем с помощью геометрического моделирования методом многомерной интерполяции влияние трехкомпонентного заполнителя: мартеновский шлак (МШ), горелая порода (ГП) и доменный граншлак (ГрШ) на физико-механические свойства мелкозернистого бетона [18]. Особенностью такой геометрической модели является то обстоятельство, что содержание заполнителей в смеси определяется долевым участием и вместе составляет единое целое. Поскольку суммарное участие всех трех компонентов всегда составляет 100 %, один из компонентов можно исключить. На геометрической схеме (рис. 4) в качестве компонентов используются МШ и ГрШ, что позволяет абсолютно точно определить смесь заполнителя. Далее исключаем лишние комбинации компонентов заполнителя, принимая во внимание, что сумма долевого участия всех трех компонентов равна 100 %. Таким образом, число выполненных экспериментов равно 10. Полученные в результате эксперимента 10 точек распределим следующим образом: первый опорный контур состоит из четырех

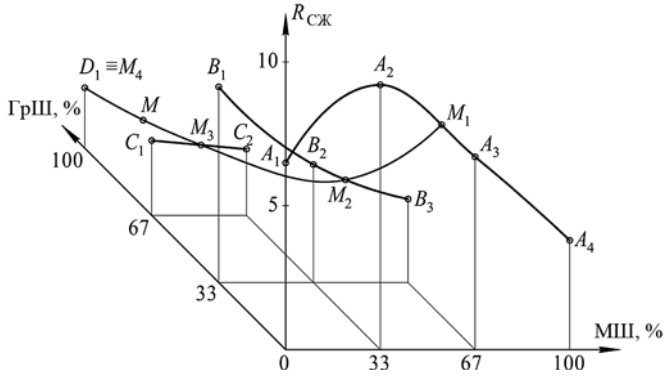

Рис. 4. Геометрическая схема построения модели зависимости мелкозернистого бетона от состава комбинированного заполнителя

точек, второй — из трех, третий — из двух и четвертый — из одной точки (рис. 4).

Первую опорную линию $A_1A_2A_3A_4$ (рис. 4) определим с помощью точечного уравнения дуги кривой третьего порядка, проходящей через четыре наперед заданные точки. Вторую ($B_1B_2B_3$) и третью (C_1C_2) опорные линии определим соответственно с помощью точечных уравнений параболы и прямой:

$$\begin{gathered} M_1 = A_1\left[\overline{u}^3 - 2,5\overline{u}^2u + \overline{u}u^2\right] + A_2\left[9\overline{u}^2u - 4,5\overline{u}u^2\right] + \\ + A_3\left[-4,5\overline{u}^2u + 9\overline{u}u^2\right] + A_4\left[\overline{u}^2u - 2,5\overline{u}u^2 + u^3\right], \\ M_2 = B_1\overline{u}\,(1-2u) + 4B_2\overline{u}u + B_3u\,(2u-1), \\ M_3 = C_1\overline{u} + C_2u, \end{gathered} \tag{9}$$

где A_i, B_i и C_i — исходные точки для построения геометрической модели, координаты которых соответствуют исходным экспериментальным данным, представленным графически на рис. 4; M_i — текущие точки опорных линий отсека поверхности; u — текущий параметр, который изменяется от 0 до 1 и согласовывает движение текущих точек по опорным линиям; $\overline{u} = 1 - u$ — дополнение параметра до 1.

Образующую искомого отсека поверхности определим с помощью точечного уравнения дуги кривой третьего порядка, проходящей через четыре текущие точки:

$$\begin{gathered} M = M_1\left[\overline{v}^3 - 2,5\overline{v}^2v + \overline{v}v^2\right] + M_2\left[9\overline{v}^2v - 4,5\overline{v}v^2\right] + \\ + M_3\left[-4,5\overline{v}^2v + 9\overline{v}v^2\right] + M_4\left[\overline{v}^2v - 2,5\overline{v}v^2 + v^3\right], \end{gathered} \tag{10}$$

где M — текущая точка искомого отсека поверхности; v — текущий параметр, который изменяется от 0 до 1 и определяет положение текущей точки M образующей линии отсека поверхности; $\overline{v} = 1 - v$ — дополнение параметра до 1.

В результате получен вычислительный алгоритм построения поверхности отклика для определения зависимости физико-механических свойств мелкозернистого бетона от состава комбинированного заполнителя. С учетом исходных данных для опреде-

ления предела прочности при сжатии $R_{СЖ}$, получим следующую систему параметрических уравнений:

$$\begin{cases} МШ = -100uv + 100u; \\ ГрШ = 100v; \\ R_{СЖ} = -99,225u^3v^3 + 198,45u^3v^2 + 247,05u^2v^3 - \\ - 121,275u^3v - 184,725uv^3 - 480,6u^2v^2 + 22,05u^3 + \\ + 37,35v^3 + 279,45u^2v + 341,1uv^2 - 45,9u^2 - \\ - 58,05v^2 - 177,625uv + 21,25u + 16,4v + 6,4. \end{cases} \quad (11)$$

Исходя из полученной модели, определим такой состав комбинированного заполнителя, при котором обеспечивается наибольшее значение предела прочности при сжатии. Для этого необходимо решить систему дифференциальных уравнений:

$$\begin{cases} \dfrac{\partial z}{\partial u} = 0; \\ \dfrac{\partial z}{\partial v} = 0. \end{cases}$$

Кроме того, были исследованы на наличие экстремальных точек границы исследуемой области. В результате максимальный предел прочности при сжатии 9,24 МПа был достигнут при следующем составе заполнителей: мартеновский шлак (29,4 %) и горелая порода (70,6 %). Таким образом, с помощью геометрического моделирования и математического анализа функции двух переменных выполнена оптимизация состава комбинированного заполнителя для изготовления наиболее качественного мелкозернистого бетона из отходов промышленности.

Использование метода многомерной интерполяции и аппроксимации для моделирования процесса распределения прочностных характеристик в бетонной колонне

Рассмотрим в качестве второго примера моделирование процесса распределения прочностных характеристик по всему объему бетонной колонны, предложенного в работе [19]. С точки зрения геометрического моделирования этот процесс можно представить в виде гиперповерхности отклика, принадлежащей четырехмерному пространству, которая определяется тремя факторами, влияющими на процесс. В качестве факторов влияния используются координаты x, y и z, определяющие положение искомой точки по всему объему бетонной колонны, а в качестве функции отклика — показатели соотношения характеристик бетона к показателям характеристик стандартных образцов в процентах.

В результате после использования метода многомерной интерполяции и аппроксимации получено следующее уравнение геометрический модели гиперповерхности отклика, которая определяется параметрами u, v и w:

$$\frac{E_b^э}{E_b} = 123,5 - 92wu^2 + 92wu + 92wv - 92wv^2 + 51,2w^2u^2 - \\ - 51,2w^2u - 51,2w^2v + 51,2w^2v^2 + 518,4w^2u^2v + \\ + 518,4w^2uv^2 - 745,6wu^2v + 745,6wu^2v^2 - \\ -518,4w^2u^2v^2 - 745,6wuv^2 + 745,6wuv - \\ - 518,4w^2uv - 90,6w + 59,2w^2,$$

где $\dfrac{E_b^э}{E_b}$ — показатель соотношения прочностных характеристик бетона в процентах к показателям характеристик стандартных образцов; u, v и w — текущие параметры, которые изменяются от 0 до 1 и определяют показатель соотношения прочностных характеристик бетона по всему объему бетонной колонны.

Для перехода от значений параметров к натуральным значениям факторов, в соответствии с уравнением (8) для равномерного распределения значений исходных точек были получены следующие линейные зависимости:

$$\begin{cases} x = 320u; \\ y = 320v; \\ z = 1280w. \end{cases}$$

Здесь хотелось бы остановиться на проверке достоверности полученной модели. Традиционно достоверность моделирования подтверждается сравнением полученной модели с исходными данными. В нашем случае условие принадлежности исходных точек к искомому геометрическому объекту была заложена непосредственно в процессе моделирования и потому уже удовлетворяет требованиям достоверности полученной модели по сравнению с исходными данными. Для более точной проверки необходимо использование дополнительных промежуточных точек. В нашем случае при построении модели была использована многомерная параболическая аппроксимация гиперповерхности отклика. Поэтому вместо 125 точек, соответствующих исходным данным, для аппроксимации было использовано с учетом симметрии бетонной колонны только 15 точек. А для проверки достоверности полученной модели были использованы все 125 исходных точек.

Следует отметить, что автору не удалось найти информацию о специализированных методах проверки достоверности геометрических моделей многомерного пространства, поэтому для оценки достоверности был использован коэффициент детерминации из регрессионного анализа, значение которого составило $R^2 = 0,94$.

Графическая визуализация полученной геометрической модели представлена на рис. 5 (см. третью сторону обложки) в виде семейства поверхностей отклика. Визуализация реализована в программном пакете Maple. Для большей наглядности по оси z используется вертикальный масштаб.

Геометрическая модель зависимости предела прочности при сжатии мелкозернистого дегтеполимербетона от четырех факторов

Третий пример — геометрическая модель зависимости предела прочности при сжатии мелкозернистого дегтеполимербетона от четырех факторов: вязкости дегтя $\left(C_{30}^{10}\right)$, концентрации отсева поливинилхлорида в каменноугольном дорожном дегте $\left(C_m^{\text{ПВХ}}\right)$, концентрации активатора на поверхности минерального порошка $\left(C_m^{\text{КМ-МТ}}\right)$ и температуры была описана в работе [20]. Эта модель представляет собой четырехпараметрическое множество точек или отсек гиперповерхности отклика, принадлежащий пятимерному пространству:

при моделировании таких процессов и явлений, которые имеют большое число взаимосвязанных факторов. Представленные практические примеры подтверждают эффективность использования метода многомерной интерполяции и аппроксимации применительно к задачам геометрического моделирования многофакторных процессов и явлений. Представлена методика аналитического определения дуг алгебраических кривых, проходящих через наперед заданные точки, которая является основой предложенного в работе подхода. Перспективой дальнейших исследований является применение метода многомерной интерполяции для геометрического моделирования и оптимизации социально-экономических, термодинамических и светотехнических процессов и явлений.

$$
\begin{aligned}
R = {} & (((44w - 52{,}8w^2 + 22{,}4)u^2 + (-49{,}2w + 58{,}4w^2 - 24)u - 32{,}4w + 31{,}2w^2 + 9{,}2)v^2 + \\
& +((-83{,}6w + 72{,}8w^2 - 8)u^2 + (81{,}4w - 74w^2 + 10{,}8)u + 31{,}8w - 29{,}2w^2 - 9)v + \\
& +(7{,}6w - 2{,}4w^2 - 3{,}2)u^2 + (-11w + 7{,}6w^2 + 3{,}2)u - 4w^2 + 4{,}4w + 3{,}6)t^2 + \\
& +(((-119{,}6w + 119{,}2w^2 - 23{,}2)u^2 + (148{,}6w - 149{,}2w^2 + 22)u + 15{,}4w - 12{,}4w^2 - 9{,}8)v^2 + \\
& +((179{,}4w - 158w^2 + 6)u^2 + (-196{,}7w + 179{,}4w^2 - 6{,}8)u - 21{,}9w + 16{,}2w^2 + 9{,}1)v + \\
& +(-8{,}6w + 1{,}2w^2 + 4{,}4)u^2 + (13{,}9w - 9{,}4w^2 - 5)u + 7{,}6w^2 - 9{,}8w - 8)t + \\
& ((90{,}8w - 80{,}8w^2 - 0{,}8)u^2 + (-119{,}4w + 110w^2 + 4)u - 1 + 25{,}2w - 26{,}4w^2)v^2 + \\
& +((-115{,}8w + 104{,}4w^2 + 3{,}6)u^2 + (141{,}7w - 131w^2 - 6)u + 1{,}7 - 19{,}4w + 22w^2)v + \\
& +(1{,}6w - 1)u^2 + (1{,}9 - 5{,}2w + 4{,}8w^2)u + 4{,}9 - 5{,}2w^2 + 7{,}4w,
\end{aligned}
\tag{12}
$$

где R — предел прочности при сжатии мелкозернистого дегтеполимербетона; u, v, w и t — текущие параметры, которые изменяются от 0 до 1 и определяют значение предела прочности при сжатии мелкозернистого дегтеполимербетона R.

Воспользуемся методами математического анализа для оптимизации итоговой геометрической модели процесса. Для этого нужно найти частные производные по каждому из параметров и решить полученную систему уравнений. В результате получим, что в пределах исследуемой области максимальное значение предела прочности при сжатии 12 МПа достигается при вязкости дегтя $C_{30}^{10} = 208$с, концентрации отсева поливинилхлорида в каменноугольном дорожном дегте $C_m^{\text{ПВХ}} = 2$ %, концентрации активатора на поверхности минерального порошка $C_m^{\text{КМ-МТ}} = 1$ % и температуре $T = 0$ °С.

Заключение

В работе изложен подход к геометрическому моделированию многофакторных процессов и явлений на основе многомерной интерполяции, который позволяет строить и оптимизировать геометрические модели с использованием дискретного массива точек в виде исходной экспериментально-статистической информации. Такой подход особенно эффективен

Список литературы

1. **Калиткин Н. Н.** Численные методы. М.: Наука, 1978. 512 с.
2. **Пахнутов И. А.** Многомерная интерполяция // Интерактивная наука. 2017. № 15. URL: https://cyberleninka.ru/article/n/mnogomernaya-interpolyatsiya (дата обращения: 26.08.2018).
3. **Добровольский Н. М., Есаян А. Р., Андреева О. В., Зайцева Н. В.** Многомерная теоретико-числовая Фурье интерполяция // Чебышевский сборник. 2004. Т. 5. Вып. 1. С. 122—143.
4. **Шустов В. В.** Многомерная интерполяция сеточной вектор-функции // Молодой ученый. 2010. № 8—1. С. 17—20.
5. **Бежаев А. Ю.** Сплайн-интерполяция многомерных данных большого объема // Сибирский журнал вычислительной математики. 2003. Т. 6, № 3. С. 249—261.
6. **Бахвалов Ю. Н.** Метод многомерной интерполяции и аппроксимации и его приложения. М.: Спутник+, 2007. 108 с.
7. **Вертинская Н. Д.** Теория нелинейных многомерных моноидальных поверхностей и ее приложения: автореф. дис.... доктора техн. наук: 05.01.01. Иркутск, 2006. 31 с.
8. **Вертинская Н. Д.** Математическое моделирование многофакторных и многопараметрических процессов в многокомпонентных системах на базе конструктивной геометрии. Иркутск: ИрГТУ, 2009. Ч. 1 (Лекции). 230 с.
9. **Конопацкий Е. В.** Геометрическое моделирование и оптимизация многофакторных процессов и явлений методом многомерной интерполяции // Труды Международной научной конференции по физико-технической информатике CPT2018, 28—31 мая 2018 г. Москва—Протвино, 2018. С. 299—306.
10. **Короев Ю. И.** Начертательная геометрия: учебник. 3-е изд., стер. М.: КНОРУС, 2013. 424 с.

11. **Балюба И. Г.** Конструктивная геометрия многообразий в точечном исчислении: дис.... доктора техн. наук: 05.01.01. Макеевка, 1995. 227 с.
12. **Балюба И. Г., Найдыш В. М.** Точечное исчисление: учебное пособие. Мелитополь: МГПУ им. Б. Хмельницкого, 2015. 236 с.
13. **Квасов Б. И.** Методы изогеометрической аппроксимации сплайнами. М.: ФИЗМАТЛИТ, 2006. 360 с.
14. **Иванов Г. С.** Конструирование одномерных обводов, принадлежащих поверхностям, путем их отображения на плоскость // Геометрия и графика. 2018. Т. 6. № 1. С. 3—9. DOI: 10.12737/article_5ad07ed61bc114.52669586.
15. **Конопацкий Е. В., Крысько А. А., Рубцов Н. А.** Особенности конструирования замкнутого обвода первого порядка гладкости в БН-исчислении // Сучасні проблеми моделювання: зб. наук. праць. 2016. Вып. 6. С. 65—70.
16. **Крысько А. А., Конопацкий Е. В., Чураков А. Я.** Геометрические основы конструирования одномерного обвода через *k* наперед заданных точек в БН-исчислении // Сучасні проблеми моделювання: зб. наук. праць. 2015. Вып. 4. С. 76—81.
17. **Бумага А. И.** Геометрическое моделирование физико-механических свойств композиционных строительных материалов в БН-исчислении: дис.... канд. техн. наук: 05.23.05, 05.01.01. Макеевка, 2016. 164 с.
18. **Бумага А. И., Братчун В. И., Конопацкий Е. В.** Оптимизация состава комбинированного заполнителя мелкозернистого бетона методами БН-исчисления // Современное промышленное и гражданское строительство. 2016. Т. 12, № 2. С. 92—98.
19. **Конопацкий Е. В., Воронова О. С.** Геометрическая модель процесса распределения прочностных характеристик в бетонной колонне // Прикладная математика и вопросы управления. 2017. № 1. С. 37—44.
20. **Конопацкий Е. В., Бумага А. И., Бочоришвили В. А.** Геометрическая модель зависимости предела прочности при сжатии модифицированного мелкозернистого дегтебетона от четырех параметров // Вестник Донбасской национальной академии строительства и архитектуры. Современные строительные материалы: сб. науч. тр. 2016. Вып. 2016-1 (117). С. 55—61.

Approach to the Construction of Geometric Models of Multifactor Processes and Phenomena by the Method of Multidimensional Interpolation

E. V. Konopatskiy, e.v.konopatskiy@mail.ru, Donbas National Academy of Civil Engineering and Architecture, Makeyevka, 286123, Donetsk People's Republic

Corresponding author:

Konopatskiy Evgeniy V., Associate Professor, Donbas National Academy of Civil Engineering and Architecture, Makeyevka, 286123, Donetsk People's Republic,
E-mail: e.v.konopatskiy@mail.ru

Received on August 28, 2018
Accepted on September 17, 2018

The paper presents an approach to the theoretical foundations the geometric modeling of multifactor processes and phenomena by the method of multidimensional interpolation, which include the method of modeling the arcs of algebraic curves passing through the pre-set points and the fundamental algorithm of the multifactor processes modeling. The proposed method of multi-dimensional interpolation allows one to analytically determine and optimize geometric models based on a discrete array of points in the form of initial experimental and statistical information. This approach is particularly effective in modeling those processes and phenomena that have a large number of interrelated factors. These examples confirm the effectiveness of the method of multidimensional interpolation and approximation in relation to the geometric modeling problems of multifactor processes and phenomena. They include a two-factor process and optimization of the composition the combined aggregate of fine-grained concrete by means of multidimensional interpolation, a three-factor process — modeling of the distribution of strength characteristics throughout the volume of the concrete column, as well as an analytical description and optimization of the geometric model of the dependence of the pre-case of the compressive strength fine-grained degtepolymerbeton from 4 factors. The prospect of further research is the application of the method of multidimensional interpolation to the geometric modeling and optimization of socio-economic, thermodynamic and lighting processes and phenomena.

Keywords: *geometric model, multifactor process, multidimensional interpolation, response function, influence factors, point equation, geometric model tree, curve arc, surface compartment, hypersurface compartment*

For citation:

Konopatskiy E. V. Approach to the Construction of Geometric Models of Multifactor Processes and Phenomena by the Method of Multidimensional Interpolation*, Programmnaya Ingeneria,* 2019, vol. 10, no. 2, pp. 77—86.

DOI: 10.17587/prin.10.77-86

References

1. **Kalitkin N. N.** *Chislennye metody* (Numerical method), Moscow, Nauka, 1978, 512 p. (in Russian).
2. **Pahnutov I. A.** Mnogomernaja interpoljacija (Multidimensional interpolation), *Interaktivnaja nauka*, 2017, no. 15, available at: https://cyberleninka.ru/article/n/mnogomernaya-interpolyatsiya. (in Russian).
3. **Dobrovol'skij N. M., Esajan A. R., Andreeva O. V., Zajceva N. V.** Mnogomernaja teoretiko-chislovaja Fur'e interpoljacija (Multidimensional numerical-theoretic Fourier interpolation), *Chebyshevskij sbornik*, 2004, vol. 5. no. 1, pp. 122—143 (in Russian).
4. **Shustov V. V.** Mnogomernaja interpoljacija setochnoj vektor-funkcii (Multidimensional interpolation grid vector-functions), *Molodoj uchjonyj*, 2010, no. 8—1, pp. 17—20 (in Russian).
5. **Bezhaev A. Ju.** Splajn-interpoljacija mnogomernyh dannyh bol'shogo ob#ema (Spline interpolation of high-volume multidimensional data), *Sibirskij zhurnal vychislitel'noj matematiki*, 2003, vol. 6, no. 3, pp. 249—261 (in Russian).
6. **Bahvalov Ju. N.** *Metod mnogomernoj interpoljacii i approksimacii i ego prilozhenija* (Multivariate interpolation and approximation method and its applications), Moscow, Sputnik +, 2007, 108 p. (in Russian).
7. **Vertinskaja N. D.** Teorija nelinejnyh mnogomernyh monoidal'nyh poverhnostej i ejo prilozhenija (Theory of nonlinear multidimensional monoidal surfaces and its applications) Dokt. Diss., Irkutsk, 2006, 31 p. (in Russian).
8. **Vertinskaja N. D.** *Matematicheskoe modelirovanie mnogofaktornyh i mnogoparametricheskih processov v mnogokomponentnyh sistemah na baze konstruktivnoj geometrii* (Mathematical modeling of multifactorial and multiparameter processes in multicomponent systems on the basis of the constructive geometry), Irkutsk, IrGTU, 2009, vol. 1 (Lekcii), 230 p. (in Russian).
9. **Konopackij E. V.** Geometricheskoe modelirovanie i optimizaciya mnogofaktornyx processov i yavlenij metodom mnogomernoj interpolyacii (Geometric modeling and optimization of multifactorial processes and phenomena by the method of multidimensional interpolation), *Trudy Mezhdunarodnoj nauchnoj konferencii po fiziko-texnicheskoj informatike CPT2018*, Moskow—Protvino, 2018, pp. 299—306 (in Russian).
10. **Koroev Ju. I.** *Nachertatel'naja geometrija* (Descriptive geometry), Moscow, KNORUS, 2013, 424 p. (in Russian).
11. **Balyuba I. G.** Konstruktivnaya geometriya mnogoobrazij v tochechnom ischislenii (Constructive geometry of manifolds in a point calculation), Dokt. Diss., Makeevka, 1995, 227 p. (in Russian).
12. **Balyuba I. G., Najdysh V. M.** *Tochechnoe ischislenie* (Point calculation). Melitopol, 2015, 236 p. (in Russian).
13. **Kvasov B. I.** *Metody izogeometricheskoj approksimacii splajnami* (Isogeometric spline approximation methods), Moscow, FIZMATLIT, 2006, 360 p. (in Russian).
14. **Ivanov G. S.** Konstruirovanie odnomernyx obvodov, prinadlezhashhix poverxnostyam, putem ix otobrazheniya na ploskost' (Construction of Belonging to Surfaces One-Dimensional Contours by Mapping Them to a Plane), *Geometriya i grafika*, 2018, vol. 6, no. 1, pp. 3—9. DOI: 10.12737/article_5ad07ed61bc114. 52669586. (in Russian).
15. **Konopackij E. V., Krys'ko A. A., Rubcov N. A.** Osobennosti konstruirovaniya zamknutogo obvoda pervogo poryadka gladkosti v BN-ischislenii (The features of construction of the closed perimeter of the first order smoothness in BN-calculation), *Suchasni problemi modelyuvannya*, 2016, vol. 6, pp. 65—70 (in Russian).
16. **Krys'ko A. A., Konopackij E. V., Churakov A. Ya.** Geometricheskie osnovy konstruirovaniya odnomernogo obvoda cherez k napered zadannyx tochek v BN-ischislenii (Geometrical bases of construction of one-dimensional circle through k in advance of the given points in BN-calculation), *Suchasni problemi modelyuvannya*, 2015, vol. 4, pp. 76—81 (in Russian).
17. **Bumaga A. I.** Geometricheskoe modelirovanie fiziko-mexanicheskix svojstv kompozicionnyx stroitel'nyx materialov v BN-ischislenii (Geometric modeling the physical and mechanical properties of composite building materials in BN-calculation), Kand. Diss., Makeevka, 2016. 164 p. (in Russian).
18. **Bumaga A. I., Bratchun V. I., Konopackij E. V.** Optimizacija sostava kombinirovannogo zapolnitelja melkozernistogo betona metodami BN-ischislenija (Optimization of the composition of the combined aggregate of fine-grained concrete by bn-calculation methods), *Sovremennoe promyshlennoe i grazhdanskoe stroitel'stvo*, 2016, vol. 12, no. 2, pp. 92—98 (in Russian).
19. **Konopackij E. V., Voronova O. S.** Geometricheskaja model' processa raspredelenija prochnostnyh harakteristik v betonnoj kolonne (Geometric model of the distribution of strength characteristics in a concrete column), *Prikladnaja matematika i voprosy upravlenija*, 2017, no. 1, pp. 37—44 (in Russian).
20. **Konopackij E. V., Bumaga A. I., Bochorishvili V. A.** Geometricheskaja model' zavisimosti predela prochnosti pri szhatii modificirovannogo melkozernistogo degtebetona ot chetyrjoh parametrov (Geometric model the relationship between the compressive strength of the modified fine-grained degaetano of the four parameters), *Vestnik Donbasskoj nacional'noj akademii stroitel'stva i arhitektury. Sovremennye stroitel'nye materialy*: sb. nauch. tr., 2016, vol. 2016-1 (117), pp. 55—61 (in Russian).

ИНФОРМАЦИЯ

УДК 519.876.5 DOI: 10.17587/prin.10.87-96

А. С. Акопов, д-р техн. наук, проф., гл. науч. сотр., e-mail: aakopov@hse.ru,
А. Л. Бекларян, канд. техн. наук, доц., ст. науч. сотр., e-mail: abeklaryan@hse.ru,
Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет "Высшая школа экономики", г. Москва, Федеральное государственное бюджетное учреждение науки Центральный экономико-математический институт РАН, г. Москва, **А. К. Сагателян**, д-р геол.-мин. наук., проф., директор, e-mail: ecocentr@sci.am, **Л. В. Саакян**, канд. геогр. наук, зам. директора по научной части, e-mail: lilit.sahakyan@cens.am, **О. А. Беляева**, канд. биол. наук, руководитель отдела, e-mail: olga.belyaeva@cens.am, **Г. О. Тепаносян**, канд. биол. наук, руководитель отдела, e-mail: gevorg.tepanosyan@cens.am, Центр эколого-ноосферных исследований Национальной академии наук Республики Армения, г. Ереван

Система поддержки принятия решений для рационального озеленения города на примере г. Ереван, Республика Армения*

Представлена разработанная авторами система поддержки принятия решений для рационального управления озеленением на примере г. Ереван, Республика Армения. С использованием методов агентного моделирования разработана имитационная модель распространения выбросов вредных веществ в атмосферу, учитывающая их взаимодействие с зелеными насаждениями (деревьями). Целью моделирования является минимизация концентрации вредных выбросов в защищаемых (социально значимых) районах, в частности, в зонах расположения детских садов. Важным элементом разработанной системы является предложенный генетический алгоритм вещественного кодирования, агрегированный с имитационной моделью распространения выбросов, реализованной на платформе AnyLogic. В результате проведенных численных экспериментов получена наилучшая конфигурация посадки деревьев вокруг детских садов г. Ереван, обеспечивающая минимальный уровень концентрации вредных веществ в соответствующих защищаемых районах с учетом ограничений на стоимость программы озеленения.

Ключевые слова: *генетический алгоритм, озеленение, вредные выбросы, экологическое моделирование, имитационное моделирование*

Введение

В настоящее время проблема защиты окружающей среды от вредных выбросов становится особенно актуальной в условиях городской агломерации. В связи с ростом числа автомобилей, открытием новых промышленных предприятий и постоянным увеличением существующих производственных мощностей наблюдается устойчивый рост концентрации вредных выбросов в атмосферу. Существуют различные способы решения данной проблемы, среди которых следует отметить такие направления, как:

- экологическая модернизация предприятий [1, 2], нацеленная на внедрение современных технологий малоотходного производства;
- закрытие вредных предприятий, систематически нарушающих установленные предельно допустимые нормы выбросов, либо их перенос за черту города [2, 3];
- использование решений типа *Natural-based solution* (NBS), предполагающих использование средств живой природы для защиты окружающей среды [4].

В условиях дефицита инвестиционного капитала, необходимого на экологическую модернизацию предприятий либо их географический перенос, единственным рациональным решением является подход, основанный на NBS, в частности, применение растительности для защиты социально значимых объектов от вредных выбросов. Среди исследований по оценке влияния деревьев на концентрацию выбросов следует отметь работы [5—7]. Данные исследования

* Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (проект № 18-51-05004) и Комитета по науке Министерства образования и науки РА (грант 18RF-092). Номер гранта 2015—2017 гг.: 15RF-061.

в основном посвящены оценке абсорбционных возможностей растительности по отношению к различным типам вредных выбросов (например, CO_2, SO_2, NO_x, VOC, пыль и т.д.) на микроуровне. Такой подход позволяет получить некоторую оценку возможностей абсорбции одного элемента растительности (например, лист, дерево и т. п.) определенного объема вредных выбросов в течение ограниченного периода времени. Далее для определения необходимого числа высаживаемых деревьев применяют простые методы линейной экстраполяции. При этом для моделирования динамики вредных выбросов применяют Гауссовы модели рассеивания примеси — математические модели распространения примесей в атмосфере [8, 9]. Однако подобный подход оказывается неэффективным на глобальном уровне целого города. Причина в том, что численная реализация подобных моделей требует решения сложных систем дифференциальных уравнений в частных производных, учитывающих взаимодействие каждого дерева с каждым типом вредных выбросов, а также физико-химическое взаимодействие различных видов выбросов друг с другом с учетом влияния внешних факторов (например, температуры, направления ветра и т. д.).

С учетом представленных выше соображений создание системы поддержки принятия решений для рационального озеленением города требует разработки специального класса агентно-ориентированных моделей. Их назначение — осуществлять компьютерное моделирование сложных процессов взаимодействия вредных выбросов и зеленых насаждений с учетом возникающих абсорбционно-диффузионных эффектов. При этом, что немаловажно, исключается использование сложных физико-химических моделей. Объектом исследования является г. Ереван, Республика Армения, исходные данные по которому были предоставлены Центром эколого-ноосферных исследований Национальной академии наук РА (http://cens.am/, ЦЭНИ НАН РА).

Целью исследования, результаты которого представлены в настоящей работе, является разработка системы поддержки принятия решений для рационального озеленения города (на примере г. Ереван, РА) для минимизации среднесуточной концентрации выбросов в защищаемых районах, в частности, в зонах расположения детских садов как наиболее важных объектов социальной инфраструктуры.

1. Агентная модель распространения выбросов

Далее будет представлена разработанная авторами агентная модель распространения выбросов в городе. В этой модели рассматриваются следующие типы взаимодействующих агентов:

- агенты-выбросы, представляющие собой массы вредных веществ определенного вида (например, CO, CO_2, SO_2, VOC, NO_x и т. д.) и характеризуемые радиусом личного пространства, который уменьшается в результате взаимодействия с растительностью;
- агенты-деревья, представляющие собой кластеры однотипных деревьев (например, тополь, клен, дуб и т. д.), распределенных в пространстве определенным образом, которые используются для защиты определенных районов города от выбросов;
- агенты — транспортные средства, представляющие собой кластеры автомобилей, которые распределяются на дорогах города с учетом известной плотности и являются основными нестационарными источниками вредных выбросов в городе;
- агенты-предприятия, расположенные в городе и являющиеся стационарными источниками вредных выбросов.

Предложенная модель основана на известных абсорбционно-диффузионных эффектах [5, 6], возникающих в результате взаимодействия агентов-деревьев с агентами-выбросами (рис. 1).

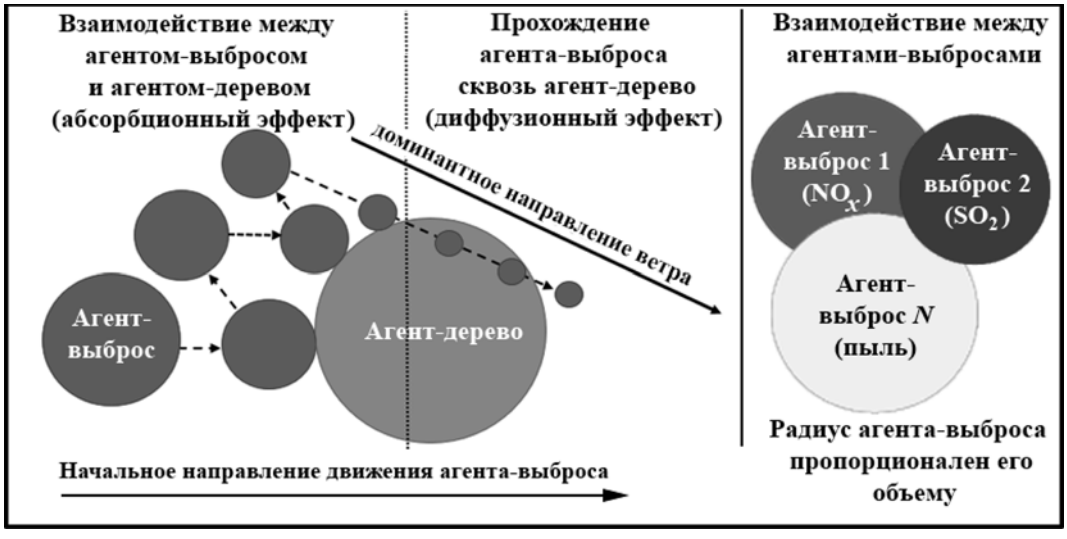

Рис. 1. Взаимодействие агентов-деревьев и агентов-выбросов

Отметим, что в процессе движения произвольного агента-выброса с учетом влияния доминантного направления ветра возможны множественные столкновения с агентами-деревьями (рис. 1). Такие столкновения приводят к уменьшению радиуса личного пространства агента-выброса при каждом подобном столкновении в результате абсорбционного эффекта, т. е. эффекта поглощения вредных выбросов листьями деревьев. Радиус личного пространства агента-выброса пропорционален физическому объему данного типа выброса (например, NO_x, SO_2 и т. д.), представляющего собой кластер, внутри которого концентрация соответствующего вещества имеет постоянное значение. В процессе взаимодействия одного типа выбросов с другими возможны эффекты поглощения (например, пыль частично поглощает SO_2), однако в данной модели такие поглощения не учитываются в силу высокой вычислительной сложности. Вместо этого взаимодействие агентов-выбросов друг с другом считается независимым, т. е. траектории их движения могут пересекаться без влияния на их радиусы и объемы.

Если радиус "личного" пространства агента-выброса станет кратно меньше радиуса агента-дерева, с которым происходит соответствующее взаимодействие, то агент-выброс беспрепятственно проходит сквозь агент-дерево. Таким образом реализуется диффузионный эффект (рис. 1). В результате действия подобных эффектов концентрация вредных веществ в атмосфере может существенно измениться. С одной стороны, деревья являются естественным барьером для выбросов и обеспечивают их эффективную абсорбцию. С другой стороны, сильный диффузионный эффект, вызванный низкой плотностью их посадки, может привести к существенному повышению концентрации вредных веществ в защищаемых зонах. Геометрия посадки деревьев при этом оказывает существенное влияние на направление движения агентов-выбросов, которые меняют свою траекторию в результате столкновения с агентами-деревьями с учетом доминантного направления ветра. Соответственно, рациональная геометрия посадки и конфигурация древесных кластеров может обеспечить эффективное рассеивание агентов-выбросов и уменьшение их радиусов. При этом концентрация вредных выбросов в атмосфере может оцениваться как сумма радиусов (площадей) личного пространства всех агентов-выбросов (дифференцируемых по видам выбросов), находящихся в защищаемой зоне, например, над территорией детского сада. Далее по тексту в качестве защищаемого городского района будем рассматривать детский сад, что не исключает возможности использовать какой-либо другой объект (район) защиты.

Таким образом, агенты-выбросы двигаются в направлениях с учетом взаимодействия с различными ландшафтными объектами, в частности, с агентами-деревьями. Такое перемещение вредных веществ может быть описано с использованием дифференциальных уравнений с переменной структурой.

Введем следующие обозначения:

- $\tilde{T} = \{t_0,\ t_1,\ ...,\ t_T\}$ — набор временных интервалов по дням (t_0 — начальное время; T — один год);
- $\tilde{I} = \{i_1,\ i_2,\ ...,\ i_I\}$ — набор индексов источников выбросов (агентов-предприятий и агентов-транспортных средств); I — число источников вредных выбросов в городе;
- $\tilde{V} = \{v_1,\ v_2,\ ...,\ v_V\}$ — набор индексов вредных веществ (например, CO_2, органическая и неорганическая пыль, тяжелые металлы, SO_2, NO_x); V — число видов вредных веществ;
- $\tilde{J}_{i,v} = \{j_{1,i,v},\ j_{2,i,v},\ ...,\ j_{J_{i,v}(t)}\}$ — набор индексов агентов-выбросов, производимых i-ми источниками выбросов, содержащих v-е вредные вещества, $J_{i,v}(t)$ — число агентов-выбросов в атмосфере города в момент времени t $(i \in \tilde{I},\ v \in \tilde{V},\ t \in \tilde{T})$;
- $\{\tilde{x}_{j_{i,v}}(t),\ \tilde{y}_{j_{i,v}}(t)\}$ — координаты $j_{i,v}$-го агента-выброса в момент времени t в системе WGS 84 (World Geodetic System 1984 — Всемирная система геодезических параметров Земли 1984 г., в число которых входит система геоцентрических координат); $j \in \tilde{J},\ i \in \tilde{I},\ v \in \tilde{V},\ t \in \tilde{T}$;
- $\tilde{Z} = \{\zeta_1,\ \zeta_2,\ ...,\ \zeta_Z\}$ — набор индексов зон высадки агентов-деревьев (кластеры деревьев); Z — число кластеров деревьев, равное числу защищаемых городских зон (детских садов);
- $\tilde{K}_\zeta = \{k_{1,\zeta},\ k_{2,\zeta},\ ...,\ k_{K_\zeta}\}$ — набор индексов агентов-деревьев, принадлежащих своим ζ-м кластерам; K_ζ — число агентов-деревьев в ζ-м кластере $(\zeta \in \tilde{Z})$;
- $s_{j_{i,v}}(t)$ — скорость распространения выбросов, определяемая средней скоростью ветра;
- $\alpha_{j_{i,v}}(t)$ — угол, определяющий направление движения j_i-го агента-выброса с учетом доминантного направления ветра при условии отсутствия какого-либо препятствия (агента-дерева) на его пути, которое вычисляется как угол между известным вектором направления движения j_i-го агента-выброса в текущий и предыдущий моменты времени соответственно, скорректированный на значение угла сноса, вызванного влиянием ветра;
- $\pm\beta_{j_{i,v}k_\zeta}(t)$ — угол обхода j_i-го агента-выброса вокруг k_ζ-го агента-дерева, который вычисляется как угол между известным вектором движения j_i-го агента-выброса в направлении k_ζ-го агента-дерева и вектором движения j_i-агента-выброса в направлении известных координат точки обхода агента-дерева (слева или справа в зависимости от знака угла $\beta_{j_i k_\zeta}(t)$, задаваемого случайным образом);
- $\gamma_{j_i}(t)$ — угол отскока j_i-го агента-выброса от k_ζ-го агента-дерева, который вычисляется как угол, определяющий направление, противоположное текущему направлению движения агента-выброса, фиксируемому в момент столкновения с k_ζ-м агентом-деревом;
- $c_{j_i}(t)$ — известный коэффициент отскока j_i-го агента-выброса от каждого агента-дерева;
- $dist_{j_i k_\zeta}(t)$ — евклидово расстояние между j_i-м агентом-выбросом и ближайшим k_ζ-м агентом-деревом;
- $r_{j_i}(t)$ — радиус j_i-го агента-выброса, уменьшающегося в результате каждого столкновения с каким-либо агентом-деревом; при этом начальное значение

радиуса j_i-го агента-выброса является известным и зависит от характеристик источника выброса (агента-предприятия или агента-транспортного средства);

- $R_{k_\zeta}(t)$ — радиус k_ζ-го агента-дерева, значение которого зависит от вида дерева (например, тополь, дуб, ель и т. п.) и может быть связано с сезонным фактором (меньше поздней осенью и зимой, чем поздней весной и летом);
- ϖ — коэффициент, отражающий диффузионный эффект ($\varpi = 10$);
- $st_{j_i}(t) \in \{1,\ 0\}$ — состояние j_i-го агента-выброса, при $st_{j_i}(t) = 1$ j_i-й агент-выброс активен, при $st_{j_i}(t) = 0$ j_i-й агент-выброс полностью утратил свой радиус.

Динамика каждого агента-выброса в каждый момент времени t описывается следующей системой дифференциальных уравнений:

$$\frac{d\tilde{x}_{j_{i,v}}(t)}{dt} = \begin{cases} s_{j_{i,v}} \cos\alpha_{j_{i,v}}(t), \text{ если выполняется I,} \\ s_{j_{i,v}} \cos\left(\pm\beta_{j_{i,v}k_\zeta}(t)\right) + \dfrac{c_{j_{i,v}}}{dist_{j_{i,v}k_\zeta}} \cos\gamma_{j_{i,v}}(t), \\ \text{если выполняется II,} \\ 0, \text{ если выполняется III,} \end{cases} \quad (1)$$

$$\frac{d\tilde{y}_{j_{i,v}}(t)}{dt} = \begin{cases} s_{j_{i,v}} \sin\alpha_{j_i}(t), \text{ если выполняется I,} \\ s_{j_{i,v}} \sin\left(\pm\beta_{j_{i,v}k_\zeta}(t)\right) + \dfrac{c_{j_{i,v}}}{dist_{j_{i,v}k_\zeta}} \sin\gamma_{j_{i,v}}(t), \\ \text{если выполняется II,} \\ 0, \text{ если выполняется III,} \end{cases} \quad (2)$$

где

$$i \in \tilde{I},\ v \in \tilde{V},\ j_{i,v} \in \tilde{J}_{i,v},\ \zeta \in \tilde{Z},\ k_\zeta \in \tilde{K}_\zeta.$$

В представленных уравнениях:

I. $st_{j_{i,v}}(t) = 1$ и $\left(dist_{j_{i,v}k_\zeta}(t) > r_{j_{i,v}}(t) + R_{k_\zeta}(t)\right.$ для всех $k_\zeta \in \{1, 2, ..., K_\zeta(t)\}$ или $r_{j_{i,v}}(t) < R_{k_\zeta}(t)/\varpi$ для ближайшего $k_\zeta \in \left.\{1, 2,\ ...,\ K_\zeta(t)\}\right)$;

II. $st_{j_{i,v}}(t) = 1$ и $\left(dist_{j_{i,v}k_\zeta}(t) \leqslant r_{j_{i,v}}(t) + R_{k_\zeta}(t)\right.$ для ближайшего $k_\zeta \in \{1, 2,\ ...,\ K_\zeta(t)\}$ и $r_{j_{i,v}}(t) \geqslant R_{k_\zeta}(t)/\varpi$ для ближайшего $k_\zeta \in \left.\{1, 2,\ ...,\ K_\zeta(t)\}\right)$.

III. $st_{j_{i,v}}(t) = 0$ или $\left(dist_{j_{i,v}k_\zeta}(t) \leqslant r_{j_{i,v}}(t) + R_{k_\zeta}(t)\right.$ и $\left.r_{j_{i,v}}(t) \geqslant R_{k_\zeta}(t)/\varpi\right.$ для всех $\left.k_\zeta \in \{1, 2, ..., K_\zeta(t)\}\right)$.

Дневная концентрация выбросов $j_{i,\ v}$-х агентов-выбросов, состоящих из v-го типа выброса, оцениваемая в p-м детском саду, имеющем радиус личного пространства $\hat{R}_{\tilde{p}}$, в момент времени t $(t \in \tilde{T})$:

$$g^*_{j_{i,v,p}}(t) = \begin{cases} \pi r^2_{j_{i,v}}(t), \text{ если } \sqrt{\left(\tilde{x}_{j_{i,v}}(t) - \tilde{x}_p\right)^2 + \left(\tilde{y}_{j_{i,v}}(t) - \tilde{y}_p\right)^2} \leqslant \\ \leqslant \hat{R}_p \text{ и } st_{j_i}(t) = 1, \\ 0, \text{ если } \sqrt{\left(\tilde{x}_{j_{i,v}}(t) - \tilde{x}_p\right)^2 + \left(\tilde{y}_{j_{i,v}}(t) - \tilde{y}_p\right)^2} > \hat{R}_p, \end{cases}$$

$$i \in \tilde{I},\ v \in \tilde{V},\ j_{i,v} \in \tilde{J}_{i,v},\ p \in \tilde{P}.$$

Дневная концентрация выбросов, суммируемая по всем агентам-выбросов и усредненная по защищаемым городским районам (детским садам):

$$DC^*(t) = \frac{1}{PV} \sum_{p=1}^{P} \sum_{i=1}^{I} \sum_{v=1}^{V} \left(\kappa_v \sum_{j_{i,v}=1}^{J_{i,v}(t)} g^*_{j_{i,v,p}}(t) \right).$$

Здесь:

- $\tilde{P} = \{p_1,\ p_2,\ ...,\ p_P\}$ — набор индексов детских садов, расположенных в городе, которые должны быть защищены от вредных выбросов; P — общее число детских садов;
- $\hat{R}_p$ — радиус личного пространства p-го детского сада, в котором должна быть минимизирована концентрация вредных выбросов;
- κ_v — весовой коэффициент, отражающий значимость (приоритетность) v-го вредного вещества для лица, принимающего решения: $\sum_{v=1}^{V} \kappa_v = 1,\ 0 \leqslant \kappa_v$.

Среднесуточная концентрация вредных выбросов:

$$ADC^* = \frac{1}{T} \sum_{t=t_0}^{T} DC^*(t).$$

Итак, задача лица, принимающего решения, состоит в определении наилучшей конфигурации озеленения вокруг защищаемых детских садов.

При этом управляющими параметрами являются:

- $cnf_\zeta(t_0) \in \tilde{C}_\zeta$ — конфигурация древесных кластеров; $\tilde{C}_\zeta = \{0,\ 1,\ ...,\ 4\}$ — набор индексов конфигураций древесных кластеров, например, без агентов-деревьев, простая окружность, арифметическая спираль, двойная окружность, двойная окружность с переменным расстоянием между ближайшими агентами-деревьями;
- $tp_\zeta(t_0) \in \tilde{S}_\zeta$ — тип древесных кластеров; $\tilde{S}_\zeta = \{1,\ 2,\ ...,\ 5\}$ — набор индексов типов древесных кластеров, например, тополь, дуб, клен, ель, вяз;
- $\delta_\zeta(t_0)$ — базовое расстояние между ближайшими агентами-деревьями, принадлежащими ζ-му древесному кластеру, которое может быть сформировано как $\underline{\delta} \leqslant \delta_\zeta(t) \leqslant \overline{\delta},\ \zeta \in \tilde{Z}$, где $\underline{\delta}$, $\overline{\delta}$ — нижняя и верхняя границы диапазона значений расстояния;
- $\tilde{R}_\zeta(t_0)$ — радиус ζ-го древесного кластера, определяющего зону озеленения, который определяется как $\underline{\tilde{R}} \leqslant \tilde{R}_\zeta(t_0) \leqslant \overline{\tilde{R}},\ \zeta \in \tilde{Z}$, где $\underline{\tilde{R}}$, $\overline{\tilde{R}}$ — нижняя и верхняя границы диапазона значений радиуса;
- n_ζ — число деревьев, принадлежащих ζ-му кластеру (зоне озеленения); $\zeta \in \tilde{Z}$;
- c_{k_ζ} — стоимость посадки и годового обслуживания одного дерева, принадлежащего k_ζ-му агенту-дереву (состоящему из нескольких близко расположенных однотипных деревьев); $k_\zeta \in \tilde{K}_\zeta,\ \zeta \in \tilde{Z}$;
- GB^* — затраты на озеленение города, вычисляемые с учетом требуемого числа высаживаемых деревьев;
- $\overline{GB}$ — максимально допустимые затраты на озеленение города (бюджет на озеленение).

Теперь можно сформулировать задачу рационального управления озеленением города, которая нацелена на минимизацию концентрации вредных выбросов в зонах расположения детских садов.

Задача А. Требуется минимизировать среднесуточную дневную концентрацию выбросов, оцениваемую в защищаемых городских районах *(детских садах)* с помощью набора управляющих параметров $\{cnf_\zeta(t_0),\ tp_\zeta(t_0),\ \delta_\zeta(t_0),\ \tilde{R}_\zeta(t_0)\}$:

$$\min_{\{cnf_\zeta(t_0),\ tp_\zeta(t_0),\ \delta_\zeta(t_0),\ \tilde{R}_\zeta(t_0)\}} [ADC^*], \qquad (3)$$

где

$$ADC^* = \frac{1}{T}\sum_{t=t_0}^{T} DC^*(t),\ GB^* = \sum_{\zeta=1}^{Z}\sum_{k_\zeta}^{K_\zeta} c_{k_\zeta} n_\zeta,$$

$$DC^*(t) = \frac{1}{PV}\sum_{p=1}^{P}\sum_{i=1}^{I}\sum_{v=1}^{V}\left(\kappa_v \sum_{j_{i,v}=1}^{J_{i,v}(t)} g^*_{j_{i,v},p}(t)\right),\ DC^*(t) \leqslant \overline{DC},$$

$$\sum_{v=1}^{V}\kappa_v = 1,\ 0 \leqslant \kappa_v,$$

$$\underline{\delta} \leqslant \delta_\zeta(t_0) \leqslant \overline{\delta},\ \underline{\tilde{R}} \leqslant \tilde{R}_\zeta(t_0) \leqslant \overline{\tilde{R}},$$

$$GB^* \leqslant \overline{GB},$$

$$cnf_\zeta(t_0) \in \tilde{C}_\zeta,\ tp_\zeta(t_0) \in \tilde{S}_\zeta,$$

$$\zeta \in \tilde{Z},\ k_\zeta \in \tilde{K}_\zeta,\ i \in \tilde{I},\ v \in \tilde{V},\ j_{i,v} \in \tilde{J}_{i,v},\ p \in \tilde{P},\ t_0 \in \tilde{T}.$$

Для решения Задачи А был разработан многоагентный генетический алгоритм вещественного кодирования, агрегированный с реализованной имитационной моделью распространения выбросов (AnyLogic) по целевому функционалу (3).

2. Генетический алгоритм вещественного кодирования

Задача А относится к классу задач большой размерности, характеризуется сложным рельефом целевого функционала (среднесуточной концентрации выбросов), зависящего как от дискретных $\{cnf_\zeta(t_0),\ tp_\zeta(t_0)\}$, так и от непрерывных $\{\delta_\zeta(t_0),\ \tilde{R}_\zeta(t_0)\}$ значений управляющих параметров. При этом решение подобной задачи с использованием точных аналитических (ньютоновских и квазиньютоновских) методов невозможно ввиду трудностей, связанных с вычислением градиента целевой функции, которая является результатом сложного взаимодействия агентов-выбросов с агентами-деревьями. Поэтому для решения подобных задач эффективнее оказываются эвристические методы оптимизации, в том числе генетические оптимизационные алгоритмы [10—12]. Подобные алгоритмы основаны на использовании операторов кроссовера и мутации. Кроссовер обеспечивает рекомбинацию ранее отобранных из популяции родительских решений, а оператор мутации позволяет преодолеть сложности застревания в области локальных экстремумов. При этом происходит генерация новых потенциальных решений, которые существенно отличаются от соответствующих родительских решений.

Ранее авторами был предложен и реализован программно параллельный многоагентный генетический алгоритм (ГА), адаптированный для решения многокритериальных оптимизационных задач большой размерности [12], использующий бинарное кодирование. Такой алгоритм наиболее эффективен для решения задач, в которых искомые переменные имеют дискретное пространство поиска. Например, когда требуется бинарное управление переходами между состояниями агентов [2]. Однако в случае искомых переменных с непрерывным пространством поиска большой размерности (таких как расстояние между агентами-деревьями, радиус зоны озеленения и т. п.) временная эффективность ГА с бинарным кодированием существенно снижается. В основном это происходит вследствие необходимости существенного увеличения разрядности двоичного кодирования действительных чисел (для порядков и мантисс) для достижения требуемой точности вычислений. Кроме того, при использовании программной реализации этого алгоритма (далее, для краткости изложения — ГА) дополнительные ресурсы затрачиваются на операции кодирования и декодирования значений искомых переменных, что также приводит к снижению эффективности ГА. По этой причине появляется необходимость в разработке ГА с вещественным кодированием.

Основное отличие предлагаемого ГА от других подобных эвристических алгоритмов с вещественным кодированием (*real-coded genetic algorithms*) состоит в комбинировании различных операторов кроссовера. Они обеспечивают эффективную одновременную процедуру поиска, как в дискретном, так и в непрерывном пространствах в условиях большой размерности.

Одним из основных типов операторов кроссовера в ГА вещественного кодирования является SBX-кроссовер (*Simulated Binary Crossover*), предложенный в работе [13]. Основная цель этого оператора состоит в формировании особей-потомков с использованием предварительно отобранных родительских особей. Отметим, что в ГА с вещественным кодированием особями-потомками являются потенциально наилучшие значения искомых переменных. Родительскими особями в этом случае являются наиболее приспособленные решения, оцениваемые по соответствующим значениям фитнесс-функций (функций приспособленности) на каждой итерации ГА.

SBX-оператор описывается следующей системой уравнений:

$$\hat{\beta}(u) = \begin{cases} \left(2u(0,\ 1)\right)^{1/(\tilde{n}+1)}, & \text{если } u < 0{,}5, \\ \left(2\left(1-u(0,\ 1)\right)\right)^{-1/(\tilde{n}+1)}, & \text{если } u \geqslant 0{,}5, \end{cases}$$

$$o_{\tilde{i},1} = 0.5\left(\left(1+\hat{\beta}(u)\right)p_{1,\tilde{i}} + \left(1-\hat{\beta}(u)\right)p_{2,\tilde{i}}\right),$$

$$o_{\tilde{i},2} = 0.5\left(\left(1-\hat{\beta}(u)\right)p_{1,\tilde{i}} + \left(1+\hat{\beta}(u)\right)p_{2,\tilde{i}}\right),$$

$$\hat{x}_{\tilde{i},1} = \begin{cases} o_{\tilde{i},1}, \text{ если } o_{\tilde{i},1} \in \left[a_{\tilde{i}},\ b_{\tilde{i}}\right], \\ a_{\tilde{i}} + h(0,\ 1)\ + \left(b_{\tilde{i}} - a_{\tilde{i}}\right), \text{ если } o_{\tilde{i},1} \notin \left[a_{\tilde{i}},\ b_{\tilde{i}}\right], \end{cases}$$

$$\hat{x}_{\tilde{i},2} = \begin{cases} o_{\tilde{i},2}, \text{ если } o_{\tilde{i},2} \in \left[a_{\tilde{i}},\ b_{\tilde{i}}\right], \\ a_{\tilde{i}} + h(0,\ 1) + \left(b_{\tilde{i}} - a_{\tilde{i}}\right), \text{ если } o_{i,2} \notin \left[a_{\tilde{i}},\ b_{\tilde{i}}\right], \end{cases}$$

$$p_{\tilde{i},1},\ p_{\tilde{i},2} \in \left[a_{\tilde{i}},\ b_{\tilde{i}}\right],\ p_{i,1} \neq p_{i,2},$$

$$\tilde{i} = 1,\ 2,\ \ldots,\ n,$$

где

- $\tilde{i} = 1,\ 2,\ \ldots,\ n$ — индекс (перечня) искомых переменных;
- $\hat{\beta}(u)$ — случайная величина, имеющая нормальное распределение и определяющая мощность оператора кроссовера;
- $\tilde{n} \in [2,\ 5]$ — параметр SBX-кроссовера, влияющий на "удаленность" особей-потомков от соответствующих родительских особей;
- $u(a,\ b)$, $h(a,\ b)$ — случайные величины, равномерно распределенные на отрезке $[a,\ b]$;
- $\left\{p_{\tilde{i},1},\ p_{\tilde{i},2}\right\}$ — пара выбранных (с помощью процедуры селекции) родительских особей (для $\tilde{i}$-х искомых переменных);
- $\left\{\hat{x}_{\tilde{i},1},\ \hat{x}_{\tilde{i},2}\right\}$ — пара особей-потомков, сформированных родительскими особями;
- $\{\eta,\ \omega\}$ — параметры SBX-кроссовера (коэффициенты), влияющие на значения особей-потомков;
- $\tilde{N} \in [0,\ 32]$ — параметр, который имитирует число единичных битов, используемых в операторе кроссовера (например, одноточечном) в ГА с бинарным кодированием.

Для обеспечения эффективного поиска в дискретном пространстве предлагается представленный далее модифицированный оператор кроссовера (*D-SBX*). Он основан на взятии целой части от значений соответствующих особей-потомков, полученных с помощью стандартного SBX-кроссовера:

$$\eta = 2^{-1}\left(2^{-u(0,\ 1)}\right)^{\tilde{N}-1},\ \omega = 1 - \eta,$$

$$\hat{x}_{\tilde{i},1} = \left\lfloor \eta p_{\tilde{i},1} + \omega p_{\tilde{i},2} + 0,5 \right\rfloor,$$

$$\hat{x}_{i,2} = \left\lfloor \omega p_{\tilde{i},1} + \eta p_{\tilde{i},2} + 0,5 \right\rfloor,$$

$$p_{\tilde{i},1}, p_{\tilde{i},2} \in \left[a_{\tilde{i}},\ b_{\tilde{i}}\right],\ p_{i,1} \neq p_{i,2}$$

$$\tilde{i} = 1,\ 2,\ \ldots,\ n.$$

Особенностями предлагаемого ГА с вещественным кодированием являются: комбинированное использование операторов кроссовера {SBX, D-SBX} в зависимости от типов искомых переменных; применение внутренних итераций, обеспечивающих генерацию множественных особей-потомков со своими характеристиками приспособленности. Обозначим через $q = 0,\ 1,\ \ldots,\ Q$ индексы внешних итераций ГА; $\tilde{g}_q = 0,\ 1,\ \ldots,\ \tilde{G}_q$ — индексы внутренних итераций ГА. Укрупненная схема предложенного ГА с вещественным кодированием представлена на рис. 2.

В ГА с вещественным кодированием (рис. 2) также применятся известный оператор адаптивной мутации, предложенный в работе [14]. Его особенностью является изменение значения вероятности мутации в зависимости от итерации ГА и удаленности от экстремума.

3. Описание программного комплекса

Предложенный ГА с вещественным кодированием (рис. 2) был агрегирован по целевому функционалу с разработанной агентной моделью распространения вредных выбросов, реализованной в системе AnyLogic.

Разработка ГА с вещественным кодированием была осуществлена с использованием языка программирования C++, а также с использованием

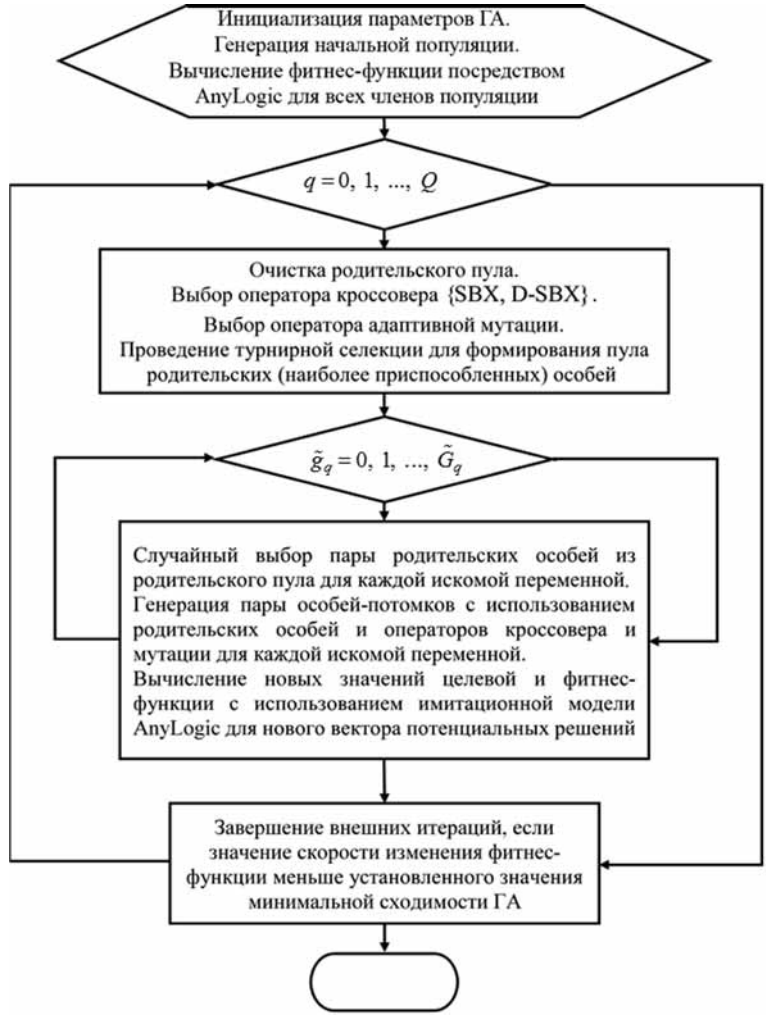

Рис. 2. Схема предложенного ГА с вещественным кодированием

технологии MPI (*Message Passing Interface*), обеспечивающей эффективную процедуру распараллеливания ГА на основе хорошо известной "островной модели". Такая модель позволяет разбить популяцию потенциальных решений на наборы относительно небольших популяций и выполнять эволюционный поиск в параллельных процессах (в рамках мультипроцессорной архитектуры). При этом поддерживаются процедуры периодического обмена наилучшими потенциальными решениями между всеми выполняемыми процессами. Для интеграции оптимизационного модуля с имитационной моделью AnyLogic (представляющей собой набор Java-классов в jar-архиве) была использована технология JNI (*Java Native Interface*), позволяющая обеспечить передачу вектора искомых переменных (потенциальных решений) в модель для расчета целевой функции в процессе выполнения ГА.

В целях сохранения результатов имитационного моделирования, а также для обеспечения механизма загрузки исходных статистических данных (например, таких как координаты агентов-предприятий; координаты возможного расположения автомобильных кластеров; структура и объемы вредных выбросов по предприятиям и др.) была выполнена интеграция имитационной модели (AnyLogic) с предметно-ориентированной базой данных (Oracle) с использованием протокола JDBC.

Для визуализации динамики распространения выбросов были использованы ГИС-технологии, поддерживаемые в системе AnyLogic и позволяющие отображать точные координаты агентов-выбросов и агентов-деревьев на карте г. Ереван с использованием системы WSG-84.

Основной фрагмент (панель управления) разработанной имитационной модели распространения выбросов в г. Ереван (Республика Армения), реализованной в системе AnyLogic, представлен на рис. 3 (см. третью сторону обложки).

Разработанная система обеспечивает численную реализацию модели распространения выбросов, описываемой системой дифференциальных уравнений с переменной структурой (1)—(2). При этом в связи с большим числом агентов-выбросов (10 000 и более) и сложной системой их взаимодействия с агентами-деревьями для решения данной системы уравнений используется хорошо известный метод Эйлера. Этот метод обеспечивает наивысшую скорость вычисления координат соответствующих агентов-выбросов в каждый момент модельного времени (день).

4. Результаты численных экспериментов

Основным результатом, численных экспериментов являются результаты сравнительного анализа эффективности различных сценариев озеленения города. Результаты расчетов, полученные в результате решения оптимизационной задачи (3), представлены в таблице.

Как видно из данных таблицы, наиболее рациональной является стратегия озеленения, основанная на использовании различных типов деревьев в различных древесных кластерах, посаженных вокруг детских садов в городе. Отметим, что в подобном сценарии преобладает использование простой окружности, состоящей из древесных кластеров тополя, обеспечивающих минимальный уровень среднесуточной концентрации вредных веществ в атмосфере в защищаемых зона детских садов г. Ереван (0,032 мг/м^3) при приемлемых затратах на озеленение (12,55 млн долл. США).

Соответствующая данному сценарию динамика дневной концентрации вредных выбросов представлена на рис. 4.

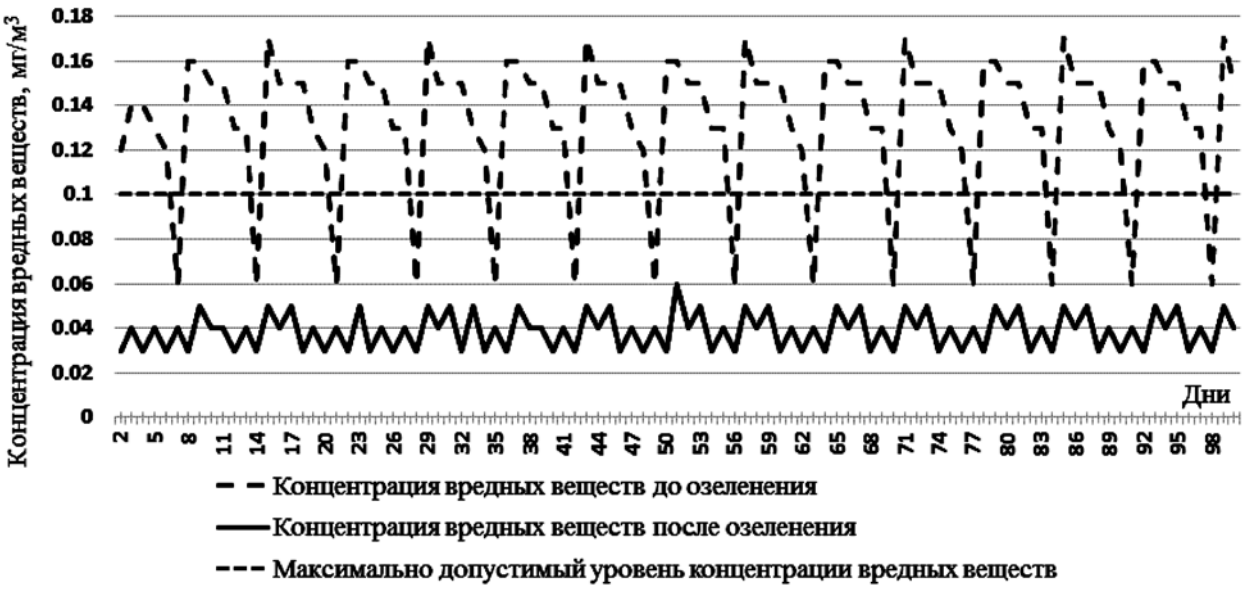

Рис. 4. Дневная концентрация вредных выбросов при индивидуальных конфигурациях древесных кластеров

Сравнительный анализ эффективности различных сценариев озеленения г. Ереван, Республика Армения

Конфигурация древесных кластеров	Тип древесных кластеров	Среднесуточная концентрация выбросов, мг/м^3	Бюджет на озеленение, млн долл. США
Без древесных кластеров	—	0,127	0,00
Простая окружность (для всех древесных кластеров)	Тополь	0,053	9,71
	Дуб	0,052	7,55
	Клен	0,054	6,47
	Ель	0,053	10,79
	Вяз	0,059	10,79
Арифметическая спираль (для всех древесных кластеров)	Тополь	0,081	2,24
	Дуб	0,083	1,74
	Клен	0,093	1,49
	Ель	0,087	2,49
	Вяз	0,082	2,49
Двойная окружность (для всех древесных кластеров)	Тополь	0,046	19,42
	Дуб	0,047	15,11
	Клен	0,052	12,95
	Ель	0,051	21,58
	Вяз	0,052	21,58
Двойная окружность (для всех древесных кластеров)	Тополь	0,049	29,13
	Дуб	0,051	22,66
	Клен	0,053	19,42
	Ель	0,051	32,37
	Вяз	0,050	32,37
Индивидуальная (смешанная) конфигурация древесных кластеров	Различные типы деревьев в различных древесных кластерах, посаженных вокруг детских садов в городе	0,032	12,55

Отметим, что точные координаты древесных кластеров (в формате WSG 84), индивидуальные конфигурации древесных кластеров, расстояния между ними и др. были вычислены с использованием разработанного генетического оптимизационного алгоритма с вещественным кодированием.

Предложенная имитационная модель распространения выбросов с учетом взаимодействия с зелеными насаждениями была верифицирована на реальных данных, собранных Центром эколого-ноосферных исследований Национальной академии наук РА (аккредитованного по ISO IEC 17025), в отдельных защищаемых городских районах (зонах расположения детских садов г. Ереван) с использованием современного оборудования, такого как газоанализаторы, пыледетекторы, сенсоры, спектроскопы и др.

Заключение

Представлена разработанная авторами программная система поддержки принятия решений для рационального управления озеленением на примере г. Ереван, Республика Армения. Предложенная в качестве базовой составляющей этой системы имитационная модель распространения выбросов вредных веществ в атмосферу реализована с использованием инструментальных средств AnyLogic. Она учитывает взаимодействие этих выбросов с зелеными насаждения (деревьями). Моделирование проводили в целях минимизации концентрации вредных выбросов в защищаемых (социально значимых) районах, к числу которых, в частности, относятся зоны расположения детских садов. Важным элементом разработанной

системы является предложенный генетический алгоритм вещественного кодирования, агрегированный с имитационной моделью распространения выбросов. В результате проведенных численных экспериментов получена наилучшая конфигурация посадки деревьев вокруг детских садов в Ереване. Эта конфигурация обеспечивает минимальный уровень концентрации вредных веществ в соответствующих защищаемых районах с учетом ограничений на стоимость программы озеленения.

Наиболее важными направлениями дальнейших исследований являются:

- детализация имитационной модели распространения выбросов посредством включения других агентов, в частности, агентов-зданий, являющихся естественным барьером для выбросов;
- моделирование стратегии вертикального озеленения (т.е. озеленение стен и кровли) и его влияния на абсорбцию вредных выбросов с использованием высотных зданий, расположенных в непосредственной близости от защищаемых районов города (детских садов, школ, и т. д.);
- уточнение методологии прогнозирования динамики распространения вредных выбросов с учетом влияния климатических факторов (температуры и влажности воздуха, скорости и направления ветра, турбулентности вихревых потоков и т. д.);
- развитие программного комплекса как системы поддержки принятия решений экологического планирования, учитывающей плотность распределения жителей и удаленность городских застроек от промышленных предприятий и транспортных потоков (дорог) при формировании стратегии оптимального озеленения.

Список литературы

1. **Akopov A. S., Beklaryan A. L., Saghatelyan A. K., Sahakyan L. V.** Control system for ecological modernization of enterprises (on the example of the Republic of Armenia) // Business Informatics. 2016. N 2 (36). P. 71—78.
2. **Akopov A. S., Beklaryan L. A., Saghatelyan A. K.** Agent-based modelling for ecological economics: A case study of the Republic of Armenia // Ecological Modelling. 2017. Vol. 346. P. 99—118.
3. **Акопов А. С., Бекларян А. Л., Бекларян Л. А., Сагателян А. К.** Моделирование региональной эколого-экономической системы с механизмом государственного регулирования на примере Республики Армения // Экономическая наука современной России. 2016. Т. 72, № 1. С. 109—119.
4. **Nesshöver C., Assmuthe T., Irvine K. M., Rusch G. M., Waylenf K. A., Delbaere B., Haase D., Jones-Walters L., Keune H., Kovacs E., Krauze K., Külvik M., Rey F., Dijk J., Vistad O. I., Wilkinson M. E., Wittmer H.** The science, policy and practice of nature-based solutions: An interdisciplinary perspective // Science of The Total Environment. 2017. Vol. 579. P. 1215—1227.
5. **Fuiii S., Chaa H., Kagib K., Miyamurac H., Kim Y.** Effects on air pollutant removal by plant absorption and adsorption // Building and Environment. 2005. Vol. 40, N 1. P. 105—112.
6. **Omasa K., Saji H., Youssefin S., Kondo N.** Air pollution and plant biotechnology. Tokyo: Springer-Verlag, 2002. 455 p.
7. **Bell J. N. B., Treshow M.** Air pollution and plant life. Chichester: John Wiley & Sons, 2002. 480 p.
8. **Turner D. B.** A diffusion model for an urban area // Journal of Applied Meteorology and Climatology. 1964. Vol. 3, N 1. P. 83—91.
9. **Beychok M. R.** Fundamentals of Stack Gas Dispersion. 4th ed. author-published, 2005. 201 p.
10. **Акопов А. С., Бекларян А. Л., Хачатрян Н. К., Фомин А. В.** Разработка адаптивного генетического оптимизационного алгоритма с использованием методов агентного моделирования // Информационные технологии. 2018. Т. 24, № 5. С. 321—329.
11. **Хивинцев М. А., Акопов А. С.** Применение многоагентного генетического алгоритма для поиска оптимальных стратегических и оперативных решений // Бизнес-информатика. 2014. № 1 (27). С. 23—33.
12. **Akopov A. S., Hevencev M. A.** A Multi-agent genetic algorithm for multi-objective optimization // Proceedings of IEEE International Conference on Systems, Man and Cybernatics, 2013. Manchester: IEEE, 2013. P. 1391—1395.
13. **Kumar A., Deb K.** Real-coded genetic algorithms with simulated binary crossover: studies on multimodal and multiobjective problems // Complex Systems. 1995. Vol. 9, N 6. P. 431—454.
14. **Bandaru S., Tulshyan R., Deb K.** Modified sbx and adaptive mutation for real world single objective optimization // Evolutionary Computation (CEC), 2011 IEEE Congress on. IEEE, 2011. P. 1335—1342.

Decision Support System for the Rational Greening of the City on the Example of Yerevan, Republic of Armenia

A. S. Akopov, aakopov@hse.ru, **A. L. Beklaryan,** abeklaryan@hse.ru, National Research University Higher School of Economics, Moscow, 105187, Russian Federation, **A. Saghatelyan,** ecocentr@sci.am, **L. Sahakyan,** lilit.sahakyan@cens.am, **O. Belyaeva,** olga.belyaeva@cens.am, **G. Tepanosyan,** gevorg.tepanosyan@cens.am, Center for Ecological-Noosphere Studies National Academy of Sciences, Yerevan, 0025, Armenia

Corresponding author:

Beklaryan Armen L., Associate Professor, National Research University Higher School of Economics, Moscow, 105187, Russian Federation,
E-mail: abeklaryan@hse.ru

Received on October 11, 2018
Accepted on October 31, 2018

This article presents the developed decision support system for the rational management of landscaping on the example of Yerevan, Republic of Armenia. Using agent-based modeling methods, a simulation model has been developed for the distribution of emissions of harmful substances into the atmosphere, taking into account their interaction with green plantings (trees) in order to minimize the concentration of harmful emissions in protected (socially significant) areas, in particular, in kindergarten areas. An important element of the developed system is the proposed genetic algorithm for real coding, aggregated with a simulation model of emission propagation, implemented on the AnyLogic platform. As a result of numerical experiments, the best configuration of tree planting around kindergartens in Yerevan has been obtained, ensuring the minimum level of concentration of harmful substances in the respective protected areas, taking into account the restrictions on the cost of the greening program. The developed software complex allows for: further detailing of the emission distribution simulation model; modeling strategies for vertical gardening (i.e., landscaping walls and roofs); clarification of the methodology for predicting the dynamics of the spread of harmful emissions; development of the software complex as a decision support system for environmental planning, taking into account the density of residents distribution, and remoteness of urban buildings from industrial enterprises and traffic flows (roads) in the formation of an optimal greening strategy.

Keywords: *genetic algorithm, greening, harmful emissions, environmental modeling, simulation modeling*

Acknowledgements:

This work was partially supported by the Russian Foundation for Basic Research, project no. 18-51-05004, and by the Committee on Science of the Ministry of Education and Science of the Republic of Armenia, project no. 18RF-092. Project number in the period 2015—2017 was 15RF-061.

For citation:

Akopov A. S., Beklaryan A. L., Saghatelyan A., Sahakyan L., Belyaeva O., Tepanosyan G. Decision Support System for the Rational Greening of the City on the Example of Yerevan, Republic of Armenia, *Programmnaya Ingeneria*, 2019, vol. 10, no. 2, pp. 87—96.

DOI: 10.17587/prin.10.87-96

References

1. **Akopov A. S., Beklaryan A. L, Saghatelyan A. K., Sahakyan L. V.** Control system for ecological modernization of enterprises (on the example of the Republic of Armenia), *Business Informatics*, 2016, no. 2 (36), pp. 71—78.

2. **Akopov A. S., Beklaryan L. A., Saghatelyan A. K.** Agent-based modelling for ecological economics: A case study of the Republic of Armenia, *Ecological Modelling*, 2017, vol. 346, pp. 99—118.

3. **Akopov A. S., Beklaryan L. A., Beklaryan A. L., Saghatelyan A. K.** Modelirovanie regional'noj ehkologo-ehkonomicheskoj sistemy s mekhanizmom gosudarstvennogo regulirovaniya na primere Respubliki Armeniya (Modelling the Regional Ecological-Economic System with the Mechanism of the Government Regulation for the Case-Study of the Republic of Armenia), *Ehkonomicheskaya nauka sovremennoj Rossii*, 2016, no. 1, pp. 109—119 (in Russian).

4. **Nesshöver C., Assmuthe T., Irvine K. M., Rusch G. M., Waylenf K. A., Delbaere B., Haase D., Jones-Walters L., Keune H., Kovacs E., Krauze K., Külvik M., Rey F., Dijk J., Vistad O. I., Wilkinson M. E., Wittmer H.** The science, policy and practice of nature-based solutions: An interdisciplinary perspective, *Science of The Total Environment*, 2017, vol. 579, pp. 1215—1227.

5. **Fuiii S., Chaa H., Kagib K., Miyamurac H., Kim Y.** Effects on air pollutant removal by plant absorption and adsorption, *Building and Environment*, 2005, vol. 40, no. 1, pp. 105—112.

6. **Omasa K., Saji H., Youssefian S., Kondo N.** *Air pollution and plant biotechnology*, Tokyo, Springer Japan, 2002, 455 p.

7. **Bell J. N. B., Treshow M.** *Air pollution and plant life*, Chichester, John Wiley & Sons, 2002, 480 p.

8. **Turner D. B.** A diffusion model for an urban area, *Journal of Applied Meteorology and Climatology*, 1964, vol. 3, no. 1, pp. 83—91.

9. **Beychok M. R.** *Fundamentals of Stack Gas Dispersion* (4th ed.), author-published, 2005, 201 p.

10. **Akopov A. S., Beklaryan A. L., Khachatryan N. K., Fomin A. V.** Razrabotka adaptivnogo geneticheskogo optimizatsionnogo algoritma s ispol'zovaniem metodov agentnogo modelirovaniya (Development of an Adaptive Genetic Optimization Algorithm using Agent Modeling Methods), *Informatsionnye tekhnologii*, 2018, vol. 24, no. 5, pp. 321—329 (in Russian).

11. **Hevencev M. A., Akopov A. S.** Primenenie mnogoagentnogo geneticheskogo algoritma dlya poiska optimal'nykh strategicheskikh i operativnykh reshenij (Application of multi-agent genetic algorithm for search of optimum strategic and operational decisions), *Biznes-informatika*, 2014, no. 1 (27), pp. 23—33 (in Russian).

12. **Akopov A. S., Hevencev M. A.** A Multi-agent genetic algorithm for multi-objective optimization, *Proceedings of IEEE International Conference on Systems, Man and Cybernatics*, 2013, Manchester, IEEE, pp. 1391—1395.

13. **Kumar A., Deb K.** Real-coded genetic algorithms with simulated binary crossover: studies on multimodal and multiobjective problems, *Complex Systems*, 1995, vol. 9, no. 6, pp. 431—454.

14. **Bandaru S., Tulshyan R., Deb K.** Modified sbx and adaptive mutation for real world single objective optimization, *Evolutionary Computation (CEC), 2011 IEEE Congress on*, 2011, IEEE, pp. 1335—1342.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *Н. В. Яшина*

Сдано в набор 06.12.2018 г. Подписано в печать 00.01.2019 г. Формат 60×88 1/8. Заказ PI219
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**