ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2016
Т. 5, № 3

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

Учредитель — Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет)

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Математическое и программное обеспечение высокопроизводительных вычислительных систем
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES "COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2016
vol. 5, no. 3

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- High performance computing
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 004.021 DOI: 10.14529/cmse160301

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА ПОИСКА МИНИМАЛЬНЫХ ОСТОВНЫХ ДЕРЕВЬЕВ С ИСПОЛЬЗОВАНИЕМ ЦЕНТРАЛЬНОГО И ГРАФИЧЕСКОГО ПРОЦЕССОРОВ*

© 2016 г. А.С. Колганов[1,2]

[1]*Институт прикладной математики имени М.В. Келдыша РАН (125047 Москва, Миусская пл., д.4),*

[2]*Московский государственный университет имени М.В. Ломоносова (119991 Москва, ул. Ленинские Горы, д. 1)*

E-mail: alexander.k.s@mail.ru

Поступила в редакцию: 06.05.2016

Решение задачи поиска минимальных остовных деревьев является распространенной в различных областях исследований: распознавание различных объектов, компьютерное зрение, анализ и построение сетей (например, телефонных, электрических, компьютерных, дорожных и т.д.), химия и биология и многие другие. Обработка больших графов — достаточно трудоемкая задача для центрального процессора (CPU) и является востребованной в данное время. Все более широкое распространение для решения задач общего назначения получают графические ускорители (GPU), имеющие большую вычислительную мощность, чем CPU. В данной статье рассмотрены методы сжатия и преобразования исходных графов для повышения эффективности их обработки. На примере алгоритма поиска минимальных остовных деревьев исследованы предложенные подходы. Исследована возможность гибридной реализация данного алгоритма. Получены самые высокие результаты по производительности на графах R-MAT и SSCA2.

Ключевые слова: поиск остовных деревье, параллельная обработка графов, алгоритм Борувки, CUDA, большие графы.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Колганов А.С. Параллельная реализация алгоритма поиска минимальных остовных деревьев с использованием центрального и графического процессоров // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 5–19. DOI: 10.14529/cmse160301.

Введение

В данной статье рассматривается задача построения минимального остовного дерева (MST — minimum spanning tree) для неориентированного взвешенного графа. Остовное дерево — такое дерево, которое является максимальным по включению ребер подграфом, не имеющее циклов, и в котором сумма весов ребер — минимальна. Если исходный граф связный, то будет построено остовное дерево, если же в исходном графе несколько несвязных компонент, то результатом будет остовный лес.

Минимальные остовные деревья имеют широкое практическое применение. Они используются при построении различных сетей, например, коммуникационных, компьютерных

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2016»

или транспортных. Также они используются в кластеризации, сегментации при обработке изображений, распознавании рукописного ввода. Вычисление остовных деревьев на графах является нерегулярным алгоритмом обработки данных. Наименьшая вычислительная сложность последовательного алгоритма MST была получена автором данной статьи [1] и равна $O(E\alpha(E,V))$, где E — количество ребер графа, V — количество вершин, α — функция Аккермана [2].

Алгоритм Борувки дает оценку $O(Elog(V))$. Также существуют некоторые параллельные реализации данного алгоритма: со сложностью $O(log(V))$ и временной сложностью $O(Vlog(V))$, на симметричном мультипроцессоре со сложностью $O((V+E)/p)$, где p — количество ядер в процессоре, на графическом процессоре [3–5]. Но все рассмотренные результаты были получены на достаточно старых и разных графических процессорах, что не позволяет корректно сравнивать такие реализации между собой.

В данной работе описывается гибридный параллельный алгоритм Борувки, использующий и графический и центральный процессоры. В разделе 1 приведено описание обрабатываемых графов. В разделе 2 описаны алгоритмы преобразования входных графов. В разделе 3 дано описание параллельной реализации алгоритма Борувки на графическом процессоре. Разделы 4 и 5 описывают принципы построения гибридной реализации алгоритма и результаты полученной производительности.

1. Описание формата представления графов

Рассмотрим структуру хранения неориентированного взвешенного графа, так как в дальнейшем она будет часто упоминаться и преобразовываться. Граф задается в сжатом CSR (Compressed Sparse Row) [6, 7] формате. Данный формат получил широкое распространение для хранения разреженных матриц и графов. Для взвешенного неориентированного графа с N вершинами и M ребрами необходимо три массива: X (массив указателей на смежные вершины), A (массив списка смежных вершин) и W (массив весов ребер, соответствующие списку смежных вершин). Массив X размера $N+1$, остальные два — $2*M$, так как в неориентированном графе для любой пары вершин необходимо хранить прямую и обратную дуги. В массиве X хранятся начало и конец списка соседей, находящиеся в массиве A, то есть весь список соседей вершины J находится в массиве A с индекса $X[J]$ до $X[J+1]$, не включая его. По аналогичным индексам хранятся веса каждого ребра из вершины J. Для иллюстрации на рис. 1 слева показан граф из 4 вершин, записанный с помощью матрицы смежности, а справа — в формате CSR (для упрощения, вес каждого ребра считается одинаковым и не указан на рисунке).

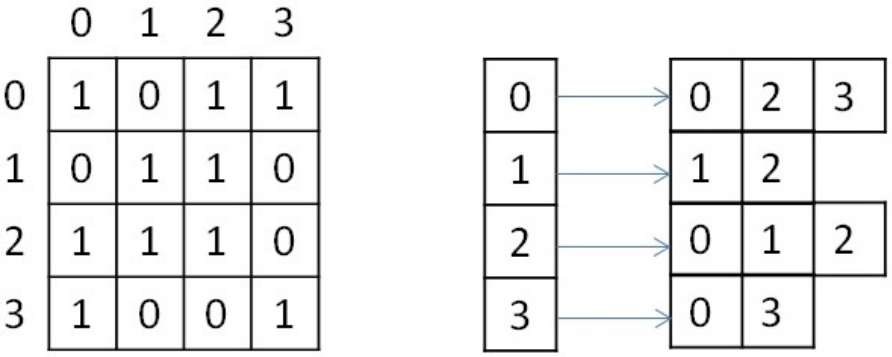

Рис. 1. Представление графа в виде матрицы смежности и в формате CSR

Для описания алгоритмов преобразования графа и параллельного алгоритма MST требуется представление структуры рассматриваемых графов. Для оценки производительности реализации используются два вида синтетических графов: R-MAT-графы (A Recursive Model for Graph Mining) [8] и SSCA2-графы (Scalable Synthetic Compact Applications) [9].

R-MAT-графы хорошо моделируют реальные графы из социальных сетей, Интернета. В данном случае рассматриваются R-MAT-графы со средней степенью связности вершины 32, а количество вершин является степенью двойки. В таком R-MAT графе имеется одна большая связная компонента и некоторое количество небольших связных компонент или висящих вершин.

SSCA2-граф представляет собой большой набор независимых компонент, соединенных ребрами друг с другом. SSCA2-граф генерируется таким образом, чтобы средняя степень связности вершины была близка к 32, количество вершин в данном графе также является степенью двойки. Таким образом, рассматриваются два совершенно разных по структуре графа с одинаковым количеством вершин.

2. Преобразование входных данных

Так как тестирование алгоритма будет производиться на графах R-MAT и SSCA2, которые получаются с помощью генератора, то для улучшения производительности реализованного алгоритма необходимо проделать некоторые преобразования. Все преобразования не будут учитываться в подсчете производительности.

2.1. Локальная сортировка списка вершин

Для каждой вершины выполним сортировку ее списка соседей по весу в порядке возрастания. Это позволит частично упростить выбор минимального ребра на каждой итерации алгоритма. Данная сортировка является локальной, следовательно, она не дает полное решение задачи.

2.2. Перенумерация всех вершин графа

Занумеруем вершины графа таким образом, чтобы наиболее связные вершины имели наиболее близкие номера. В результате данной операции в каждой связной компоненте разница между максимальным и минимальным номером вершины будет наименьшей, что позволит лучшим образом использовать маленький объем кэша графического процесса. Стоит отметить, что для R-MAT-графов данная перенумерация не дает существенного эффекта, потому что в данном графе присутствует очень большая компонента, которая не помещается в кэш даже после применения данной оптимизации. Для SSCA2-графов эффект от данного преобразования заметен больше, так как в данном графе большое количество компонент, содержащих относительно небольшое количество вершин.

2.3. Отображение весов графа в целые числа

В данной задаче не требуется производить каких-либо операций над весами графа. Необходимо уметь сравнивать веса двух ребер. Для этих целей можно использовать целые числа, вместо чисел двойной точности, так как скорость обработки чисел одинарной точности на GPU намного выше, чем двойной. Данное преобразование можно выполнить для графов, у которых количество уникальных ребер не превосходит 2^{32} (максимальное количество различных чисел, помещающихся в unsigned int языка Си). Если средняя степень

связности каждой вершины равна 32, то самый большой граф, который можно обработать с применением данного преобразования, будет иметь 2^{28} вершин и будет занимать в памяти 64 ГБ. На 2015 год наибольшее количество памяти присутствует в ускорителях NVidia Tesla k40 / NVidia Titan X и AMD FirePro w9100 и составляет 12ГБ и 16ГБ соответственно. Поэтому на одном GPU с применением данного преобразования можно обработать достаточно большие графы.

2.4. Сжатие информации о вершинах

Данное преобразование применимо только к SSCA2-графам из-за их структуры. В данной задаче большую роль играет производительность памятей всех уровней: начиная от глобальной памяти и заканчивая кэшем первого уровня. Для снижения трафика между глобальной памятью и L2-кэшем, можно хранить информацию о вершинах в сжатом виде. Изначально информация о вершинах представлена в виде двух массивов: массива X, в котором хранятся начало и конец списка соседей в массиве A.

Рассмотрим пример на рис. 2. У вершины J есть 10 вершин-соседей, и если номер каждого соседа хранится с использованием типа unsigned int, то для хранения списка соседей вершины J потребуется $10 * sizeof(unsigned\ int)$ байт, а для всего графа — $2 * M * sizeof(unsigned\ int)$ байт. Будем считать, что $sizeof(unsigned\ int) = 4$ байта, $sizeof(unsigned\ short) = 2$ байта, $sizeof(unsigned\ char) = 1$ байт. Тогда для данной вершины необходимо 40 байт для хранения списка соседей.

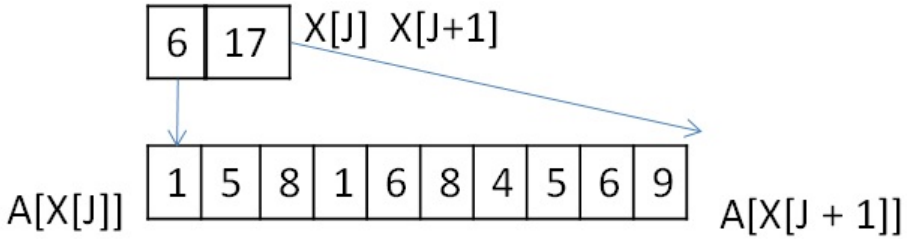

Рис. 2. Смежные 10 вершин для вершины J

Нетрудно заметить, что разница между максимальным и минимальным номером вершины в этом списке равна 8, причем для хранения данного числа необходимо всего 4 бита. Исходя из тех соображений, что разница между максимальным и минимальным номером вершины может быть меньше, чем unsigned int, можно представить номер каждой вершины следующим образом:

$$base\_J + 256 * k + short\_endV,$$

где $<base\_J>$ — например, минимальный номер вершины из всего списка соседей. В данном примере на рис. 2 это будет 1. Данная переменная будет иметь тип unsigned int, и таких переменных будет столько, сколько вершин в графе. Далее посчитаем разницу между номером вершины и выбранной базой. Так как в качестве базы мы выбрали наименьшую вершину, то данная разница будет всегда положительной. Для SSCA2-графа данная разница будет помещаться в unsigned short. $<short\_endV>$ — это остаток от деления на 256. Для хранения данной переменной будем использовать тип unsigned char; а $<k>$ — есть целая часть от деления на 256. Для k выделим 2 бита (то есть k лежит в пределах от 0 до 3). Выбранное представление является достаточным для рассматриваемого графа. В битовом представление данное сжатие проиллюстрировано на рис. 3. Тем самым для хранения списка вершин требуется $(1 + 0,25) * 10 + 4 = 16,5$ байт для данного примера, вместо 40 байт, а

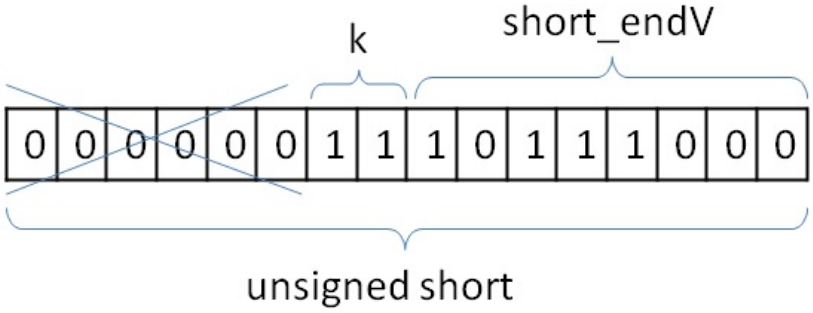

Рис. 3. Сжатия номера одной смежной вершины

для всего графа: $(2 * M + 4 * N + 2 * M/4)$ вместо $2 * M * 4$. Если $N = 2 * M/32$, то общий объем уменьшится в $(8 * M)/(2 * M + 8 * M/32 + 2 * M/4) =$ 2.9 раз.

3. Общее описание алгоритма и этапы его реализации

Для реализации алгоритма MST был выбран алгоритма Борувки. Данный алгоритм имеет минимальную временную сложность и большой потенциал для распараллеливания по сравнению, например, с алгоритмом Прима или Крускала. Базовое описание алгоритма Борувки и иллюстрация его итераций хорошо представлена [10]. Согласно алгоритму, все вершины изначально включены в минимальное дерево — каждая вершина в своем дереве. Далее необходимо выполнить следующие шаги:

1. Найти минимальные ребра между всеми деревьями для их последующего объединения. Если на данном шаге не выбрано ни одно ребро, то ответ задачи получен;
2. Выполнить объединение соответствующих деревьев. Данный шаг разбивается на два этапа: удаление циклов, так как два дерева могут в качестве кандидата на объединение указать друг друга, и этап объединения, когда выбирается номер дерева, в которое входят объединяемые поддеревья. Для определенности будем выбирать минимальный номер такого дерева. Если в ходе объединения осталось лишь одно дерево, то ответ задачи получен;
3. Выполнить перенумерацию полученных деревьев для перехода на первый шаг (чтобы все деревья имели номера от 0 до k).

Схематично реализованный алгоритм показан на рис. 4. Выход из всего алгоритма происходит в двух случаях:

- Если все вершины после N итераций объединены в одно дерево;
- Если невозможно найти минимальное ребро из каждого дерева (в таком случае минимальные остовные деревья найдены).

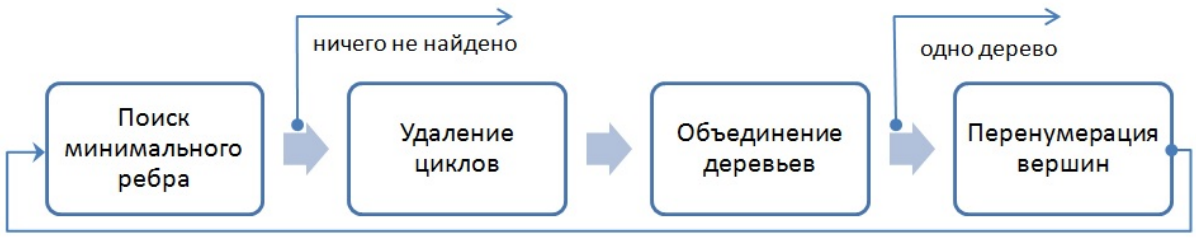

Рис. 4. Схема работы алгоритма Борувки

3.1. Поиск минимального ребра

Сначала каждая вершина графа помещается в отдельное дерево. Далее происходит итеративный процесс объединения деревьев, состоящий из четырех рассмотренных выше процедур. Процедура поиска минимального ребра позволяет выбрать именно те ребра, которые будут входить в минимальное остовное дерево. Как было описано выше, на входе у данной процедуры преобразованный граф, хранящийся в формате CSR. Так как для списка соседей была выполнена частичная сортировка ребер по весу, то выбор минимальной вершины сводится к просмотру списка соседей и выбора первой вершины, которая принадлежит другому дереву. Если предположить, что в графе нет петель, то на первом шаге алгоритма выбор минимальной вершины сводится к выбору первой вершины из списка соседей для каждой рассматриваемой вершины, потому что список соседних вершин (которые составляют вместе с рассматриваемой вершиной ребра графа), отсортированы по возрастанию веса ребра и каждая вершина входит в отдельное дерево. На любом другом шаге необходимо просмотреть список всех соседних вершин по порядку и выбрать ту вершину, которая принадлежит другому дереву.

Почему же нельзя выбрать вторую вершину из списка соседних вершин и положить данное ребро минимальным? После процедуры объединения деревьев (которая будет рассмотрена далее) может возникнуть ситуация, что некоторые вершины из списка соседних могут оказаться в том же дереве, что и рассматриваемая, тем самым данное ребро будет являться петлей для данного дерева, а по условию алгоритма необходимо выбирать минимальное ребро до других деревьев.

Для реализации обработки вершин и выполнения процедуры поиска, объединения и слияния списков хорошо подходит структура Union Find [11]. К сожалению, не все структуры оптимально обрабатываются на GPU. Наиболее выгодно в данной задаче (как и в большинстве других) использовать непрерывные массивы в памяти GPU, вместо связных списков. Ниже будут рассмотрены похожие алгоритмы Union-Find для поиска минимального ребра, объединения сегментов, удаления циклов в графе. Рассмотрим алгоритм поиска минимального ребра. Его можно представить в виде двух шагов:

- Выбор исходящего из каждой вершины минимального ребра (которое входит в какой-то сегмент) рассматриваемого графа;
- Выбор ребра минимального веса для каждого дерева.

Для того, чтобы не перемещать информацию о вершинах, записанную в формате CSR, будем использовать два вспомогательных массива, которые будут хранить индекс начала и конца массива A списка соседей. Два данных массива будут обозначать сегменты списков вершин, принадлежащих одному дереву. Например, на первом шаге массив начал или нижних значений будет иметь значения $X[0]...X[N]$, а массив концов или верхних значений будет иметь значения $X[1]...X[N+1]$. После процедуры объединения деревьев, данные сегменты перемешаются, но массив соседей A не будет перемещен в памяти.

Оба шага могут быть выполнены параллельно. Для выполнения первого шага необходимо просмотреть список соседей каждой вершины (или каждого сегмента) и выбрать первое ребро, принадлежащее другому дереву. Можно выделить один warp — набор из 32х нитей GPU, работающие физически параллельно и синхронно — для просмотра списка соседей каждой вершины. На рис. 5 красным выделены сегменты, принадлежащие дереву 0, а зеленым — дереву 1.

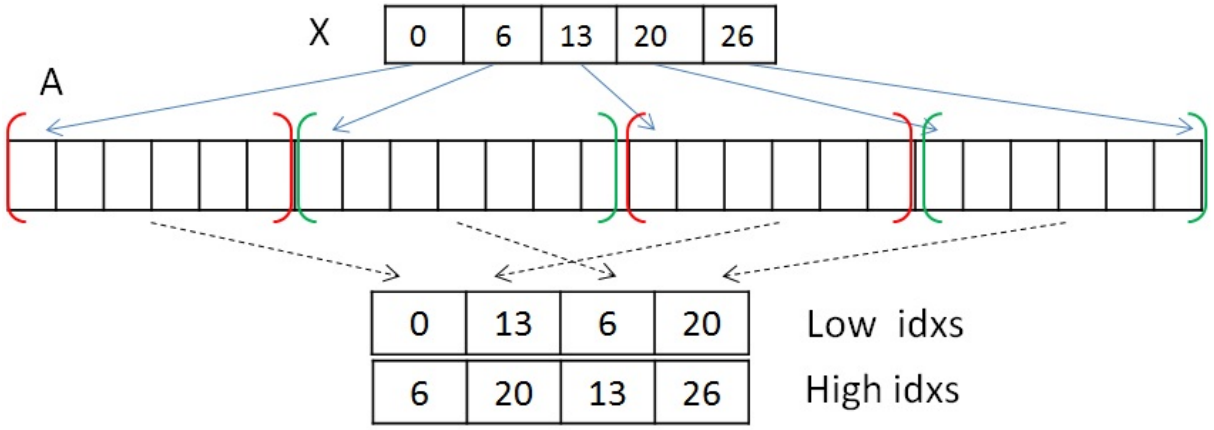

Рис. 5. Расположение сегментов в памяти

В силу того, что каждый сегмент списка соседей отсортирован, то не обязательно просматривать все вершины. Так как один warp состоит из 32 нитей, то просмотр будет осуществляться порциями по 32 вершины. После того, как просмотрены 32 вершины, необходимо объединить результат и если ничего не найдено, то просмотреть следующие 32 вершины. Для объединения результата можно воспользоваться префиксной суммой. В результате данного шага для каждого сегмента будет записана следующая информация: номер вершины в массиве A, входящее в ребро минимального веса и вес самого ребра. Если ничего не найдено, то в номер вершины можно записать, например, число $N + 2$.

Второй шаг необходим для редуцирования найденной информации для выбора ребра с минимальным весом для каждого из деревьев. Данный шаг делается из-за того, что сегменты, принадлежащие одному и тому же дереву, просматриваются параллельно и независимо, и для каждого из сегментов выбирается ребро минимального веса. В данном шаге один warp может редуцировать информацию по каждому дереву (по нескольким сегментам). После выполнения данного шага будет известно с каким деревом каждое из деревьев соединено минимальным ребром (если оно существует). Для записи данной информации введем еще два вспомогательных массива, в одном из которых будем хранить номера деревьев, до которых есть минимальное ребро, во втором — номер вершины в исходном графе, которая является корнем входящих в дерево вершин. Результат работы данного шага показан на рис. 6.

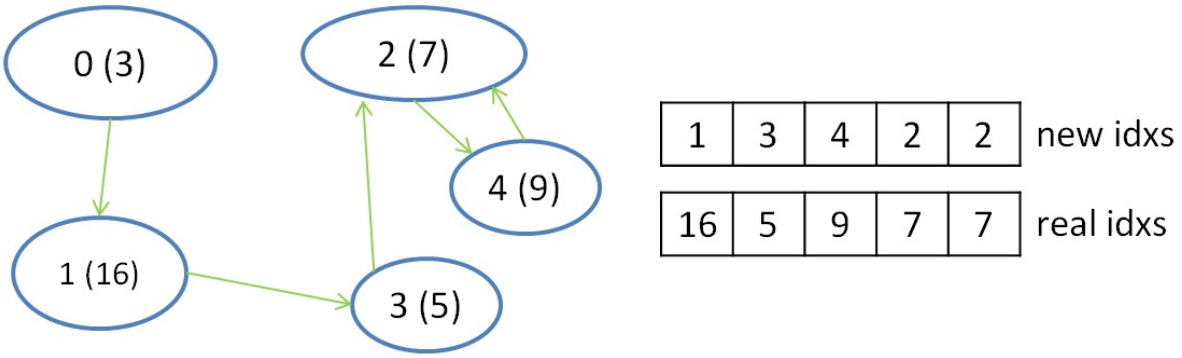

Рис. 6. Результат работы процедуры поиска минимального ребра

Для работы с индексами необходимо еще два массива, которые помогают конвертировать первоначальные индексы в новые индексы и получать по новому индексу первоначаль-

ный. Рассмотренные таблицы переконвертации индексов обновляются с каждой итерацией алгоритма. Таблица получения нового индекса по первоначальному индексу имеет размер N — количества вершин в графе, а таблица получения первоначального индекса по новому сокращается с каждой итерацией и имеет размер, равный количеству деревьев на какой-либо выбранной итерации алгоритма (на первой итерации алгоритма эта таблица имеет также размер N элементов).

3.2. Удаление циклов

Данная процедура необходима для удаления циклов между двумя деревьями. Такая ситуация возникает тогда, когда у дерева N1 минимальное ребро до дерева N2, а у дерева N2 минимальное ребро до дерева N1. На рис. 6 присутствует цикл только между двумя деревьями с номерами 2 и 4. Так как деревьев на каждой итерации становится меньше, то будем выбирать минимальный номер из двух деревьев, составляющих цикл. В данном случае, дерево с номером 2 станет указывать на само себя, а дерево с номером 4 продолжит указывать на дерево с номером 2. С помощью проверок, исходный код на языке Си которых показан на рис. 7, можно определить такой цикл и устранить его в пользу минимального номера. Такая процедура может быть выполнена параллельно, так как каждая вершина может быть обработана независимо и записи в новый массив вершин без циклов не пересекаются.

```
// cF - массив старых значений,
// F - массив новых значений,
// N - количество элементов в массиве.
__global__ void remove_cycles(const unsigned *cF, unsigned *F, unsigned N) {
    // глобальный номер нити
    unsigned i = blockIdx.x * blockDim.x + threadIdx.x;
    if(i < N) {
        unsigned local_f = cF[i];
        if (cF[local_f] == i) {
            if (i < local_f)
                F[i] = i;
        }
    }
```

Рис. 7. Реализация алгоритма удаления циклов на GPU

3.3. Объединение деревьев

Данная процедура производит объединение деревьев в более крупные деревья. Процедура удаления циклов между двумя деревьями, по сути, является предобработкой перед данной процедурой. Она позволяет избежать зацикливания при объединении деревьев. Объединение деревьев представляет собой процесс выбора нового корня путем изменения ссылок. Если, допустим, дерево с номером 0 указывало на дерево с номером 1, а в свою очередь дерево с номером 1 указывало на дерево с номером 3, то можно сменить ссылку дерева 0 с дерева 1 на дерево 3. Описанное изменение ссылки стоит производить, если оно не приводит к появлению цикла между двумя деревьями. Применим описанный алгоритм к рис. 6. После процедур удаления циклов и объединения деревьев останется только одно дерево с номером 2. Процесс объединения представлен на рис. 8. Структура графа и принцип

его обработки таков, что не возникнет ситуации, когда будет происходить зацикливание процедуры и она также может быть выполнена параллельно.

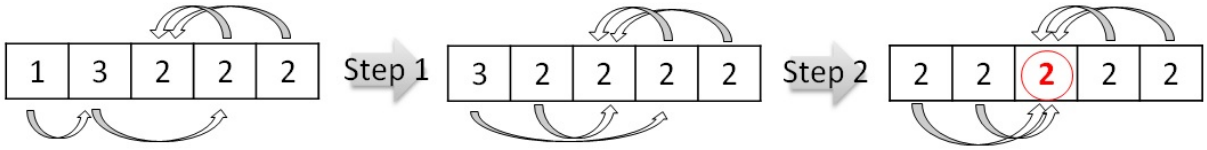

Рис. 8. Итерационный процесс объединения деревьев

3.4. Перенумерация вершин (деревьев)

После выполнения процедуры объединения деревьев необходимо перенумеровать полученные деревья так, чтобы их номера шли подряд от 0 до P. По построению новые номера должны получить элементы массива, удовлетворяющие условию $F[i] == i$ (для рассмотренного примера на рис. 6, данному условию удовлетворяет только элемент с индексом 2). Тем самым, с помощью атомарных операций можно разметить весь массив новыми значениями от 1 ... $(P + 1)$. Далее выполнить заполнение таблиц получения нового индекса по первоначальному и первоначального индекса по новому. Пример такой перенумерации представлен на рис. 9.

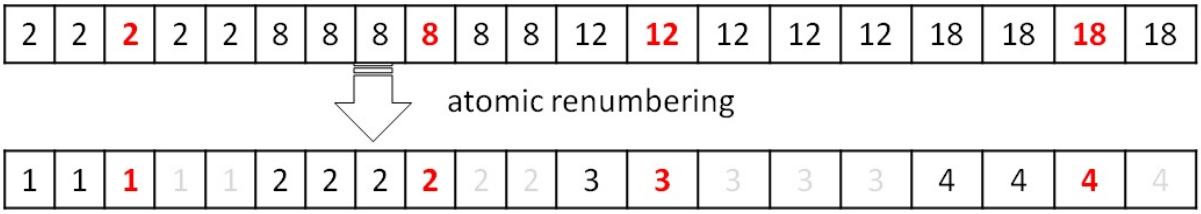

Рис. 9. Процесс перенумерации деревьев

Работа с данными таблицами описана в процедуре поиска минимального ребра. Следующая итерация не может корректно выполняться без обновления данных таблиц. Все описанные операции выполняются параллельно и на GPU.

4. Гибридная реализация процедуры поиска минимального ребра

Описанный алгоритм в разделе 3, показывает хорошие ускорения на одном GPU. Решение задачи MST организовано таким образом, что можно распределить нагрузку процедуры поиска минимального ребра между центральными процессором (CPU) и GPU. Данное распределение можно сделать только на общей памяти. Для этого был использован стандарт OpenMP [12] для параллельного счета на CPU и передача данных между CPU и GPU по шине PCIe с помощью технологии CUDA [13]. Выполнение всех процедур на одной итерации на линии времени при использовании одного GPU представлено на рис. 10.

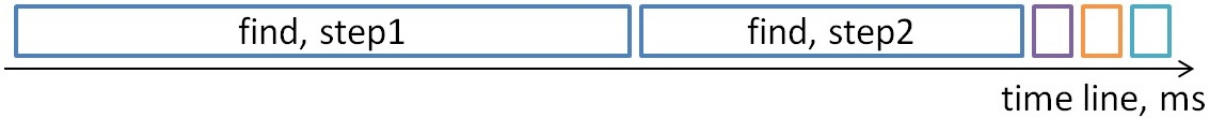

Рис. 10. Время выполнения процедур на одной итерации алгоритма на GPU

Изначально все данные о графе хранятся как на CPU, так и на GPU. Для того, чтобы CPU мог считать, необходимо передать информацию о перемещенных во время объединения деревьев сегментах. Также для того, чтобы GPU продолжил итерацию алгоритма, необходимо вернуть подсчитанные на CPU данные. Логичным было бы использование асинхронного копирования между центральным процессором и ускорителем (см рис. 11).

Алгоритм на CPU повторяет алгоритм, используемый на GPU, только для распараллеливания цикла используется OpenMP. Как и стоило ожидать, CPU вычисляет медленнее, чем GPU. Чтобы CPU успевал посчитать свою часть, данные между CPU и GPU надо делить в отношении $1:5$. Остальные процедуры невыгодно считать на обоих устройствах, так как они занимают очень мало времени, а накладные расходы и медленная скорость CPU только увеличат время алгоритма. Также очень важна скорость копирования между CPU и GPU. На тестируемой платформе поддерживался протокол PCIe 3.0, который позволял достигать 15 ГБ/с. В итоге, использование CPU позволило получить около 15% прироста производительности.

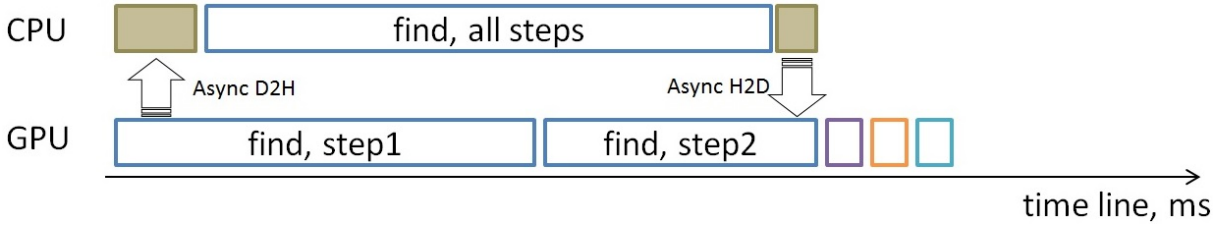

Рис. 11. Время выполнения процедур на одной итерации алгоритма на GPU и CPU

На сегодняшний день количество оперативной памяти на GPU и CPU существенно отличается в пользу последнего. На тестовой платформе на GPU доступно 6 ГБ GDDR5, в то время как на CPU доступно 40 ГБ. Ограничения по памяти на GPU не позволяют обсчитывать большие графы.

С версии CUDA Toolkit 6 доступна для использования технология Unified Memory [14], которая позволяет обращаться с GPU в память CPU. Так как информация о графе необходима только в процедуре поиска минимального ребра, то для больших графов можно сделать следующее: сначала поместить все вспомогательные массивы в памяти GPU, а далее расположить часть массивов графа (массив соседей A, массив X и массив весов W) в памяти GPU, а то что не посместилось — в памяти CPU. Далее, во время счета, можно делить данные так, чтобы на CPU, по возможности, обрабатывалась та часть, которая не поместилась на GPU, а GPU минимально использовал доступ в память CPU (так как доступ в память CPU с графического ускорителя осуществляется через шину PCIe на скорости не более 15 ГБ/с).

Используя такой подход, можно обработать графы, которые изначально не помещаются на GPU даже при использовании описанных алгоритмов сжатия, но с меньшей скоростью, так как пропускная способность протокола PCIe значительно меньше, чем память ускорителя.

5. Результаты тестирования и обзор существующих решений

Тестирование производилось на GPU NVidia GTX Titan и на 6 ядерном процессоре Intel Xeon E5 v1660 с частотой 3,7 ГГц. Использовались графы в масштабе от $2^{16}(0,023$ ГБ) до $2^{26}(25,2$ ГБ). Граф масштаба 2^{16} достаточно мал (порядка 25 МБ) и даже без преобразова-

ний легко помещается в кэш одного современного процессора Intel Xeon. Однако, большие графы требуют достаточного количества памяти и граф 2^{24} масштаба уже не помещается в память тестируемого GPU без сжатия. Производительность измеряется в количестве обработанных ребер в секунду (traversed edges per second — TEPS).

На рис. 12 показаны производительности следующих алгоритмов: параллельного алгоритма на GPU, параллельного алгоритма на CPU, параллельного алгоритма на CPU и GPU. Из построенных графиков производительности видно, что с использованием всех оптимизаций на SSCA2-графах гибридный алгоритм показывают хорошую масштабируемость: чем больше граф, тем лучше производительность. Данный рост сохраняется до тех пор, пока все данные помещаются в память GPU. На 2^{25} и 2^{26} масштабах был использован механизм Unified Memory, что приводит к ухудшению производительности из-за более медленного доступа к памяти CPU.

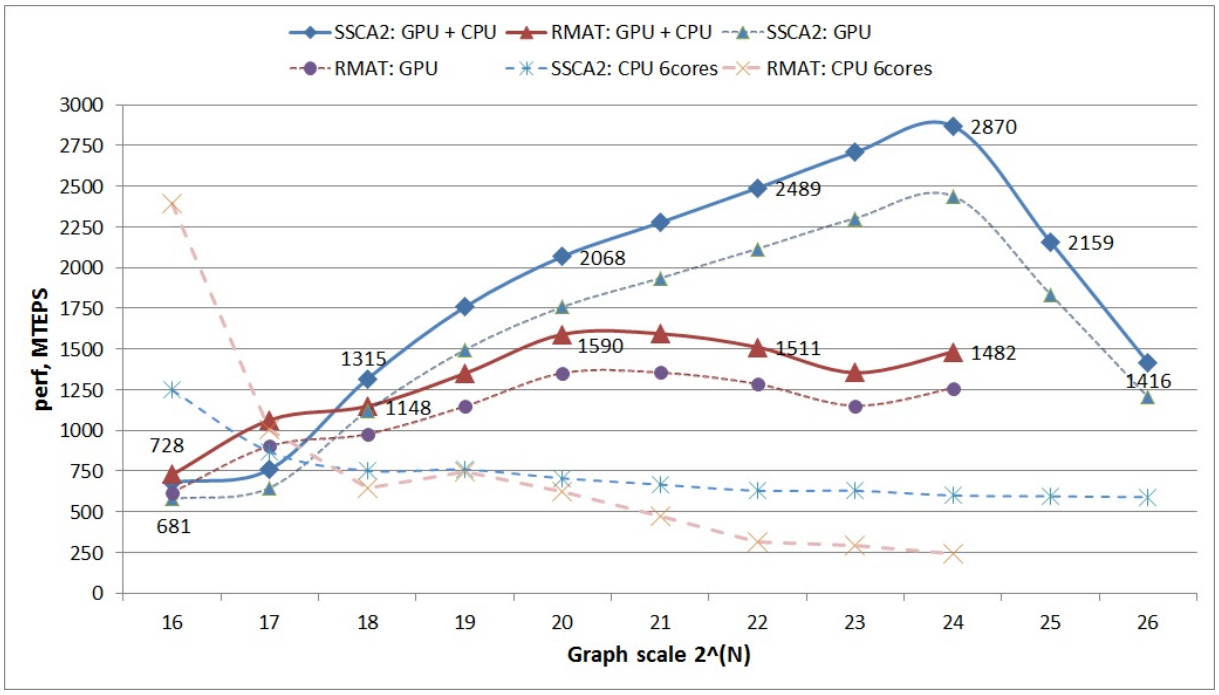

Рис. 12. Производительность гибридного параллельного и параллельного на CPU алгоритмов Борувки

Во многих зарубежных статьях, описывающих реализацию алгоритма MST с использованием GPU (например, [3–5]), не использовались какие-либо предобработки входного графа. В статье [5] приводятся результаты производительности на графах дорог различных штатов Америки. Данные графы имеют среднюю степень связности вершин 2,5, количество вершин — от 264 тыс. до 6 млн., а полученная производительность варьируется от 16 MTEPS до 36 MTEPS в зависимости от размера графа. Авторами данной статьи получены ускорения по сравнению с последовательной версией более, чем в 30 раз. В статье [4] алгоритм MST используется для кластеризации.

В статье [15] был реализован алгоритм Прима для поиска минимального остовного дерева. Авторы данной статьи использовали некоторые описанные выше преобразования графа, например, сортировка ребер по возрастанию веса. В статье приводятся результаты производительности не только на графах различных штатов Америки, но и на R-MAT-графах и SSAC2-графах. Для генерации графов авторы статьи использовали инструмент

«Georgia Tech graph generator suite». Полученная производительность варьируется от 20 MTEPS до 40 MTEPS на всех протестированных графах. RMAT-граф генерировался со средней степенью связности вершины 3, 6 и 9.

В статье [16] и других статьях, описываются похожие техники реализации алгоритма MST. Все описанные реализации демонстрируют невысокую производительность на GPU, по сравнению с реализованным гибридным алгоритмом.

Заключение

В результате проделанной работы был реализован гибридный параллельный алгоритм Борувки, использующий ядра центрального процесса и графического процессора. В данной работе был предложен адаптированный под архитектуру графического процессора фортмат хранения графа. Данный формат представляет собой формат CSR, расширенный алгоритмами сжатия, что позволило обрабатывать большие графы в ограниченном объеме памяти на графическом процессоре. Специально для алгоритма поиска остовных деревьев выполнялась предобработка предложенного формата — локальная сортировка ребер по весу и глобальная сортировка по степени связности вершин графа. Данные сортировки выполнялись параллельно на центральном процессоре, время таких сортировок на самом большом рассматриваемом графе оценивается в несколько десятков секунд.

С помощью данного алгоритма была достигнута на SSCA2-графах максимальная производительность около 3000 MTEPS и для RMAT-графов максимальная производительность около 1500 MTEPS. Были получены достаточно высокие результаты на двух графах различной структуры по сравнению с последовательной версией алгоритма Борувки, а также сравнительно высокие результаты относительно существующих параллельных реализаций алгоритма Борувки на GPU. В будущем планируется исследовать возможность выполнения данного алгоритма на архитектуре Intel Xeon Phi.

Литература

1. Chazelle B.A. Minimum spanning tree algorithm with inverse-ackermann type complexity // Journal of the ACM. 2000. Vol. 47, No. 6. P. 1028–1047.
2. Pettie S. An inverse-Ackermann style lower bound for the online minimum spanning tree verification problem // Foundations of Computer Science. 2002. P. 155–163.
3. Wei W., Shaozhong G., Fan Y., Jianxun C. GPU-Based Fast Minimum Spanning Tree Using Data Parallel Primitives // Information Engineering and Computer Science (ICIECS), 2nd International Conference (TBD Wuhan, China). 2010. P. 1–4.
4. Arefin A.S., Riveros C., Berretta R., Moscato P. kNN-MST-Agglomerative: A fast and scalable graph-based data clustering approach on GPU // Computer Science & Education (ICCSE), 7th International Conference (The University of Melbourne, Australia). 2012. P. 585–590.
5. Vineet V., Harish P., Patidar S., Narayanan P.J. Fast Minimum Spanning Tree for Large Graphs on the GPU // HPG '09 Proceedings of the Conference on High Performance Graphics (New Orleans, LA, USA). 2009. P. 167–171.
6. Gibbs N.E., Poole W.G., Stockmeyer P.K. A comparison of several bandwidth and profile reduction algorithms // ACM Transactions on Mathematical Software. 1976. Vol. 2, No. 4. P. 322–330.

7. Gilbert J.R., Moler C., Schreiber R. Sparse matrices in MATLAB: Design and Implementation // SIAM Journal on Matrix Analysis and Applications. 1992. Vol. 13, No. 1. P. 333—356.

8. Chakrabarti D., Zhan Y., Faloutsos C. R-MAT: A Recursive Model for Graph Mining // In Proceedings of 4th International Conference on Data Mining (Brighton, UK). 2004. P. 442–446.

9. Bader D.A., Madduri K. Design and implementation of the HPCS graph analysis benchmark on symmetric multiprocessors. Technical report. 2005.

10. Borůvka O. O Jistém Problému Minimálním. Práce Moravské Přírodovědecké Společnosti III. 1926. No. 3. P. 37–58.

11. Описание алгоритмов для структуры Union-Find. URL: `https://www.topcoder.com/community/data-science/data-science-tutorials/disjoint-set-data-structures/` (дата обращения: 30.11.2015).

12. Описание стандарта OpenMP. URL: `http://www.openmp.org/mp-documents/openmp-4.5.pdf` (дата обращения: 30.11.2015).

13. Compute Unified Device Architecture. URL: `http://www.nvidia.ru/object/cuda-parallel-computing-ru.html` (дата обращения: 30.11.2015).

14. Unified Memory in CUDA 6. URL: `http://devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6/` (дата обращения: 30.11.2015).

15. Nobari S., Cao T., Karras P., Bressan S. Scalable Parallel Minimum Spanning Forest Computation // PPoPP'12 Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (New York, NY, USA). 2012. P. 205–214.

16. Wei W., Shaozhong G., Fan Y., Jianxun C. Design and Implementation of GPU-Based Prim's Algorithm // Information Engineering and Computer Science (ICIECS) 2nd International Conference (TBD Wuhan, China). 2010. P. 1–4.

DOI: 10.14529/cmse160301

PARALLEL IMPLEMENTATION OF MINIMUM SPANNING TREE ALGORITHM ON CPU AND GPU

© 2016 A.S. Kolganov[1,2]

[1]Keldysh Institute of Applied Mathematics (Miusskaya sq. 4, Moscow, 125047 Russia),
[2]Lomonosov Moscow State University (GSP-1, Leninskie Gory 1, Moscow, 119991 Russia)
E-mail: alexander.k.s@mail.ru

Received: 06.05.2016

Solution of the finding a minimum spanning tree problem is common in various areas of research: recognition of different objects, computer vision, analysis and construction of networks (eg, telephone, electrical, computer, travel, etc.), chemistry and biology, and many others. Processing large graphs is a quite time-consuming task for the central processor (CPU), and in high demand at the present moment. The usage of Graphics processing units

(GPUs) as a mean to solve general-purpose problems grows every day, because GPUs have more computing power than CPUs. This article describes the methods of compression and conversion of graphs in standard formats to increase the efficiency of their processing. The search algorithm of minimum spanning trees has been used for researching the proposed approaches. The possibility of a hybrid implementation of this algorithm has been investigated. The highest results were obtained on the large R-MAT and SSCA2 graphs.

Keywords: MST, parallel graph processing, Boruvka algorithm, CUDA, large graphs.

FOR CITATION

Kolganov A.S. Parallel Implementation of Minimum Spanning Tree Algorithm on CPU and GPU. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 5–19. (in Russian) DOI: 10.14529/cmse160301.

References

1. Chazelle B.A. Minimum Spanning Tree Algorithm with Inverse-Ackermann Type Complexity. *Journal of the ACM*. 2000. vol. 47, no. 6. pp. 1028–1047. DOI: 10.1145/355541.355562.

2. Pettie S. An Inverse-Ackermann Style Lower Bound for the Online Minimum Spanning Tree Verification Problem. *Foundations of Computer Science*. 2002. pp. 155–163. DOI: 10.1109/sfcs.2002.1181892.

3. Wei W., Shaozhong G., Fan Y., Jianxun C. GPU-Based Fast Minimum Spanning Tree Using Data Parallel Primitives. *Information Engineering and Computer Science (ICIECS), 2nd International Conference (TBD Wuhan, China)*. 2010. pp. 1–4. DOI: 10.1109/iciecs.2010.5678261.

4. Arefin A.S., Riveros C., Berretta R., Moscato P. kNN-MST-Agglomerative: A Fast and Scalable Graph-Based Data Clustering Approach on GPU. *Computer Science & Education (ICCSE), 7th International Conference (The University of Melbourne, Australia)*. 2012. pp. 585–590. DOI: 10.1109/iccse.2012.6295143.

5. Vineet V., Harish P., Patidar S., Narayanan P.J. Fast Minimum Spanning Tree for Large Graphs on the GPU. *HPG '09 Proceedings of the Conference on High Performance Graphics (New Orleans, LA, USA)*. 2009. pp. 167–171. DOI: 10.1145/1572769.1572796.

6. Gibbs N.E., Poole W.G., Stockmeyer P.K. A Comparison of Several Bandwidth and Profile Reduction Algorithms. *ACM Transactions on Mathematical Software*. 1976. vol. 2, no. 4. pp. 322—330. DOI: 10.1145/355705.355707.

7. Gilbert J.R., Moler C., Schreiber R. Sparse Matrices in MATLAB: Design and Implementation. *SIAM Journal on Matrix Analysis and Applications*. 1992. vol. 13, no. 1. pp. 333—356. DOI: 10.1137/0613024.

8. Chakrabarti D., Zhan Y., Faloutsos C. R-MAT: A Recursive Model for Graph Mining. *In Proceedings of 4th International Conference on Data Mining (Brighton, UK)*. 2004. pp. 442–446. DOI: 10.1137/1.9781611972740.43.

9. Bader D.A., Madduri K. *Design and Implementation of the HPCS Graph Analysis Benchmark on Symmetric Multiprocessors*. Technical Report. 2005.

10. Borůvka O. O Jistém Problému Minimálním. *Práce Moravské Přírodovědecké Společnosti III*. 1926. no. 3. pp. 37–58.

11. *Opisanie algoritmov dlya struktury Union-Find* [Union-Find Algorithms]. Available at: `https://www.topcoder.com/community/data-science/data-science-tutorials/disjoint-set-data-structures/` (accessed: 30.11.2015).

12. *Opisanie strandarta OpenMP* [OpenMP Standard]. Available at: `http://www.openmp.org/mp-documents/openmp-4.5.pdf` (accessed: 30.11.2015).

13. *Compute Unified Device Architecture*. Available at: `http://www.nvidia.ru/object/cuda-parallel-computing-ru.html` (accessed: 30.11.2015).

14. *Unified Memory in CUDA 6*. Available at: `http://devblogs.nvidia.com/parallelforall/unified-memory-in-cuda-6/` (accessed: 30.11.2015).

15. Nobari S., Cao T., Karras P., Bressan S. Scalable Parallel Minimum Spanning Forest Computation. *PPoPP'12 Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (New York, NY, USA)*. 2012. pp. 205–214. DOI: 10.1145/2145816.2145842.

16. Wei W., Shaozhong G., Fan Y., Jianxun C. Design and Implementation of GPU-Based Prim's Algorithm. *Information Engineering and Computer Science (ICIECS), 2010 2nd International Conference (TBD Wuhan, China)*. 2010. pp. 1–4.

УДК 517.96, 517.956, 519.63 DOI: 10.14529/cmse160302

МЕТОД РЕШЕНИЯ ОБРАТНОЙ ЗАДАЧИ ИДЕНТИФИКАЦИИ ФУНКЦИИ ИСТОЧНИКА С ИСПОЛЬЗОВАНИЕМ ПРЕОБРАЗОВАНИЯ ЛАПЛАСА

© 2016 г. Н.М. Япарова

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: ddjy@math.susu.ac.ru

Поступила в редакцию: 09.08.2016

В статье предложен метод решения задачи идентификации неизвестной функции источника в параболическом уравнении с постоянными коэффициентами с граничными условиями Дирихле и Неймана. Представленный метод основан на использовании прямого и обратного преобразований Лапласа, что позволило свести исходную задачу к решению интегрального уравнения Вольтерра первого рода, характеризующую прямую зависимость неизвестной функции источника от известных граничных условий. Для численного решения полученного уравнения предлагается использовать регуляризующие алгоритмы. В качестве одного из параметров регуляризации в предложенном численном методе выступает количество слагаемых в конечномерном аналоге ядра. С целью оценки эффективности предложенного подхода и получения экспериментальных оценок погрешности численных решений задачи идентификации функции источника был проведен вычислительный эксперимент. Результаты эксперимента и свидетельствуют о достаточной устойчивости численных решений, полученных на основе предложенного метода.

Ключевые слова: идентификации функции источника, преобразования Лапласа, уравнения Вольтерра, численные методы, вычислительный эксперимент.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Япарова Н.М. Метод решения обратной задачи идентификации функции источника с использованием преобразования Лапласа // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 20–35. DOI: 10.14529/cmse160302.

Введение

Обратные задачи идентификации источника возникают при исследовании различных процессов, связанных с переносом и распределением вещества или энергии внутри тела [1–3]. Математически в линейном приближении эти процессы описываются уравнениями параболического типа с неизвестным неоднородным слагаемым, называемым функцией источника и дополнительно известными начальными и граничными условиями. Для получения устойчивого численного решения рассматриваемых задач используют различные методы регуляризации [4–8].

В статье предложен метод решения обратной задачи идентификации функции источника, основанный на применении прямого и обратного преобразований Лапласа. В подходах, использующих преобразования Лапласа, как правило, после выполнения прямого преобразования получают устойчивые решения с помощью методов регуляризации, а затем уже для этих решений применяют обратное преобразование (см., например, [9, 10]). Основная трудность при таком подходе состоит в том, что численная реализация обратного преобразования Лапласа сильно неустойчива. В данном исследовании предложен другой под-

ход, обобщающий результаты, представленные в работе [11]. В результате его применения сначала получают интегральное уравнение, устанавливающее явную зависимость функции источника от известных граничных и начальных данных, а затем уже для решения этого уравнения используют методы регуляризации. Аналогичная идея была использована в работах [12, 13] для решения обратной граничной задачи.

Практическая реализация предложенного подхода предусматривает разработку численного метода решения полученного интегрального уравнения. С разработкой и исследованием устойчивых численных алгоритмов связаны работы многих исследователей, например, [14–17]. В данном исследовании в качестве одного из возможных методов решения используется численный метод, основанный на схеме Лаврентьева с апостериорным выбором параметра регуляризации.

Статья имеет следующую структуру. В первом разделе приведена постановка задачи а также дополнительные свойства и ограничения на рассматриваемый класс функций. Второй раздел посвящен построению основного интегрального уравнения. В трерьем разделе предложен численный метод решения полученного уравнения В четвертой части статьи приведены результаты вычислительного эксперимента, основными целями которого являлись проверка эффективности предложенного подхода к решения задачи идентификации, основанному на преобразованиях Лапласа, а также получение экспериментальных оценок погрешностей регуляризованных решений. В заключении сформулированы итоговые выводы, полученные в результате проведенных исследований, а также указаны основные направления дальнейших исследований.

1. Постановка задачи

Рассмотрим следующую задачу:

$$u_t = au_{xx} + f(t), \quad x \in (0, \ell), \quad t > 0, \tag{1}$$

$$u(0,t) = u(\ell,t) = u(x,0) = 0, \quad x \in [0, \ell], \quad t \geq 0, \tag{2}$$

$$u_x(0,t) = g(t), \quad t \geq 0, \tag{3}$$

где функция $f(t)$ подлежит определению. Полагаем, что $g \in C^{2+\eta}[0,T]$ при всех $T > 0$, а $\eta \in (0,1)$ и существуют константы M, m такие, что $|g(t)| \leq Me^{mt}$ для любого $t \in [0,T]$ при всех $T > 0$. Известно, что при некотором $g(t) = g_0(t)$ существуют точное значение $f_0(t)$ и функция $u_0(x,t)$, удовлетворяющая (1)-(3) при $f(t) = f_0(t)$, но вместо g_0 известны некоторые приближения g_δ и уровень погрешности $\delta > 0$ такие, что $\|g_\delta - g_0\|_{C(0,T)} \leq \delta$. Требуется сначала определить функцию $f_\delta^\alpha(t) \in C^{1+\eta}(0,T)$ при всех $T > 0$, а затем найти $u_\delta^\alpha(x,t) \in H^{2,1}((0,\ell)\times(0,T))$ при $T > 0$, удовлетворяющую (1)–(3). Единственность решения рассматриваемой задачи (1)–(3) была доказана в работе [18].

В данном исследовании предлагается разделить решение обратной задачи идентификации источника на два этапа. Первый этап заключается в сведении исходной задачи идентификации функции $f(t)$ к решению уравнения Вольтерра первого рода, характеризующего прямую зависимость функции $f(t)$ от граничных условий. На втором этапе осуществляется численное решение полученного интегрального уравнения и по результатам решения вычисляются значения функции $u_\delta^\alpha(x,t)$.

2. Построение интегрального уравнения

Сведем задачу (1)–(3) к интегральному уравнению. С этой целью найдем решение прямой задачи, предположив, что искомая функция $f(t)$ нам известна, то есть рассмотрим следующую задачу задачу:

$$u_t = au_{xx} + f(t), \quad x \in (0, \ell), \quad t \geq 0, \tag{4}$$

$$u(0,t) = u(\ell,t) = u(x,0) = 0, \quad x \in [0,\ell], \quad t \geq 0. \tag{5}$$

Найдя решение $u(x,t)$ этой задачи, и, выполнив затем соответствующие преобразования, получим уравнение, связывающее $f(t)$ и $u_x(0,t) = g(t)$.

Полагаем, что существуют $C, \bar{C} > 0$ и $\beta_0, \beta \geq 0$ такие, что $|u(x,t)| \leq Ce^{\beta_0 t}$ и $|f(t)| \leq \bar{C}e^{\beta t}$ выполнены при $x \in [0,\ell]$ и при всех $t \in [0,T]$ для любого $T > 0$. Дополнительно полагаем, что функция $f(t)$ удовлетворяет условиям Дирихле для любого $t \in [0,T]$ при всех $T > 0$. Тогда, следуя результатам, представленным в работе [19], применим прямое преобразование Лапласа в задаче (4), (5). Обозначив изображения функций $u(x,t)$ и $f(t)$ как $\bar{u}(x,p)$ и $\bar{f}(p)$ соответственно, получаем, что операторное изображение прямой задачи (4), (5) имеет вид:

$$\frac{d^2\bar{u}}{dx^2} - \frac{p}{a}\bar{u} = -\bar{f}(p),$$
$$\bar{u}(0,p) = \bar{u}(\ell,p) = 0.$$

Решение этой задачи определяется формулой:

$$\bar{u}(x,p) = \frac{\bar{f}(p)}{p} \frac{sh\left(\sqrt{\frac{p}{a}}\ell\right) - sh\left(\sqrt{\frac{p}{a}}x\right) - sh\left(\sqrt{\frac{p}{a}}(\ell - x)\right)}{sh\left(\sqrt{\frac{p}{a}}\ell\right)}. \tag{6}$$

Лемма 1. Решение $u(x,t)$ задачи (4), (5), имеет вид:

$$u(x,t) = \frac{4}{\pi}\sum_{m=0}^{\infty} \frac{\sin\left(\frac{2m+1}{\ell}\pi x\right)}{2m+1} e^{-\frac{\pi^2(2m+1)^2 at}{\ell^2}} \int_0^t f(\tau) e^{\frac{\pi^2(2m+1)^2 a\tau}{\ell^2}} d\tau. \tag{7}$$

Доказательство. Из теоремы , доказанной в [13] следует, что функцию $\bar{\psi}(x,p) = \dfrac{sh\left(\sqrt{p}x\right)}{sh\left(\sqrt{p}\right)}$ можно представить в виде:

$$\bar{\psi}(x,p) = x + \frac{2}{\pi}\sum_{m=1}^{\infty} \frac{(-1)^m}{m} sin(\pi m x) \frac{p}{p + m^2\pi^2}.$$

Тогда имеют место следующие соотношения:

$$\frac{sh\left(\sqrt{\frac{p}{a}}x\right)}{sh\left(\sqrt{\frac{p}{a}}\ell\right)} = \frac{x}{\ell} + \frac{2}{\pi}\sum_{m=1}^{\infty} \frac{(-1)^m}{m} sin\left(\frac{\pi m x}{\ell}\right) \frac{p}{p + \frac{m^2\pi^2 a}{\ell^2}} \tag{8}$$

$$\frac{sh\left(\sqrt{\frac{p}{a}}(\ell - x)\right)}{sh\left(\sqrt{\frac{p}{a}}\ell\right)} = \frac{\ell - x}{\ell} + \frac{2}{\pi}\sum_{m=1}^{\infty} \frac{(-1)^m}{m} \sin\left(\frac{\pi m(\ell - x)}{\ell}\right) \frac{p}{p + \frac{m^2\pi^2 a}{\ell^2}}.$$

Учитывая сходимость рядов в (8), выполним соответствующие преобразования в (6), получаем:

$$\bar{u}\,(x,p)=\frac{4}{\pi}\frac{\bar{f}(p)}{p}\sum_{m=0}^{\infty}\frac{sin\left(\frac{2m+1}{\ell}\pi x\right)}{2m+1}\frac{p}{p+\frac{(2m+1)^2\pi^2 a}{\ell^2}}. \tag{9}$$

Принимая во внимание свойства функции $f(t)$, применим обратное преобразование Лапласа к обеим частям (9). Далее, используя теорему о свертке [19], получаем:

$$u(x,t)=\frac{4}{\pi}\sum_{m=0}^{\infty}\frac{\sin\left(\frac{2m+1}{\ell}\pi x\right)}{2m+1}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau.$$

Сходимость ряда (7) для любых $x\in[0,\ell)$ и $t\in[0,T)$ следует из теоремы Вейерштрасса и следующей оценки:

$$\frac{\left|\sin\left(\frac{2m+1}{\ell}\pi x\right)\right|}{2m+1}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\left|\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau\right|\le$$

$$\le\frac{1}{(2m+1)}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\frac{\bar{C}\ell^2e^{\frac{\pi^2(2m+1)^2at}{\ell^2}+\beta t}}{(\pi^2(2m+1)^2a+\ell^2\beta)}\le\frac{\bar{C}\ell^2e^{\beta t}}{\pi^2a(2m+1)^3}.$$

□

Обозначим $Q_T=(0,\ell)\times(0,T)$ при $T>0$. В силу того, что численное решение задачи(1)–(3) может быть найдено только в ограниченной области $\overline{Q}_T$, и, учитывая, что для его построения нам потребуются только значения $u_x(0,t)$, мы можем сформулировать следующий результат.

Теорема 1. Пусть решение $u(x,t)$ задачи (4), (5) определено формулой (7), тогда имеет место следующее представление

$$u_x(0,t)=\frac{4}{\ell}\sum_{m=0}^{\infty}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau$$

при всех $t\in[0,T]$ для каждого $T>0$, и для любых $\delta>0$ и $\varepsilon>0$ найдется N такое, что

$$\left\|\frac{4}{\ell}\sum_{m=0}^{N}e^{-\frac{\pi^2(2m+1)^2at)}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau-g_\delta\right\|_{C(0,T)}\le\varepsilon+\delta.$$

Доказательство. Рассмотрим ряд

$$\frac{4}{\ell}\sum_{m=0}^{\infty}\cos\left(\frac{2m+1}{\ell}\pi x\right)e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau. \tag{10}$$

Применяя рассуждения, аналогичные приведенным в лемме 1, получаем:

$$\left|\cos\left(\frac{2m+1}{\ell}\pi x\right)\right|e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\left|\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau\right|\le\frac{\bar{C}\ell^2e^{\beta t}}{\pi^2a(2m+1)^2}.$$

Из сходимости ряда $\sum\limits_{m=0}^{\infty}\frac{1}{(2m+1)^2}$, а также из теоремы Вейерштрасса следует сходимость ряда (10) при всех $x\in[0,\ell)$. Таким образом, используя свойства сходящихся рядов, получаем:

$$u_x(x,t)=\frac{4}{\ell}\sum_{m=0}^{\infty}\cos\left(\frac{2m+1}{\ell}\pi x\right)e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau.$$

Далее аппроксимируем функцию $u_x(0,t)$ конечным рядом. Тогда, в силу сходимости ряда (10) для любого $\varepsilon>0$ найдется N такое, что

$$\left|\frac{4}{\ell}\sum_{m=0}^{N}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau-u_x(0,t)\right|<\varepsilon. \tag{11}$$

Принимая во внимание условие (3), получаем требуемое неравенство. □

Таким образом, мы можем сформулировать основной результат. Из теоремы 1 и условия (3) следует, что неизвестная функция источника $f(t)$ может быть определена при решении следующего уравнения:

$$\frac{4}{\ell}\sum_{m=0}^{N}e^{-\frac{\pi^2(2m+1)^2at}{\ell^2}}\int\limits_0^t f(\tau)e^{\frac{\pi^2(2m+1)^2a\tau}{\ell^2}}d\tau=g\left(t\right) \tag{12}$$

при условии, что вместо точно заданной функции $g\left(t\right)$ известны $g_\delta\left(t\right)$ и $\delta>0$ такие, что $\|g\left(t\right)-g_0\left(t\right)\|\leq\delta$ при всех $t\in[0,T]$.

3. Численный метод

Определив $K(t,\tau)$ при всех $t\in[0;T]$ соотношением

$$K_N\left(t-\tau\right)=\frac{4}{\ell}\sum_{m=0}^{N}e^{-\frac{\pi^2(2m+1)^2a}{\ell^2}(t-\tau)},$$

и при условии, что $g\left(0\right)=0$, мы можем рассматривать уравнение (12) как уравнение Вольтерра первого рода:

$$Af=\int\limits_0^t K_N(t-\tau)f(\tau)d\tau=g(t). \tag{13}$$

Единственность решения этого уравнения при всех $t\in[0,T]$ для любых $T>0$ была доказана в [20]. Решение уравнения (13) возможно получить, применяя различные численные методы, но выбор наиболее оптимального метода решения уравнения (13) не является целью даннного исследования, поэтому нам достаточно рассмотреть один из возможных методов решения задачи идентификации. В качестве такого метода мы используем вычислительную схему, основанную на методе регуляризации Лаврентьева.

Построение численного решения задачи (1)–(3) осуществим в два этапа. На первом выбирем количество слагаемых в (11). Для этого ортогонализуем систему функций $\left\{e^{-\frac{\pi^2(2m+1)^2a}{\ell^2}t}\right\}_0^N$ для каждого $N\geqslant 2$ и $t\in[0,T]$. Обозначим через $\{\psi_m(t)\}_0^N$ полученную ор-

тогонализованную систему функций. Найдем коэффициенты $\{g_m(t)\}_0^N$ в разложение функции $g_\delta(t)$ по этой системе, используя формулу:

$$g_{\delta m} = \int_0^T g_\delta(\tau)\,\psi_m(\tau)\,d\tau.$$

Полагаем $g_{N\delta} = \sum\limits_{m=0}^{N} g_{\delta m}\psi(t)$. Перегруппировав слагаемые в последнем соотношении, получаем $g_{N\delta} = \sum\limits_{m=0}^{N} c_m e^{-\frac{\pi^2(2m+1)^2 a}{\ell^2}t}$, где $\{c_m(t)\}_0^N$ — коэффициенты, полученные после перегруппировки.

Далее при каждом значения N вычисляем величину h_N по формуле

$$h_N = \left\| \frac{4}{\ell}\sum_{m=0}^{N} c_m e^{-\frac{\pi^2(2m+1)^2 a}{\ell^2}t} - g_\delta \right\|_{C([0,T])}. \tag{14}$$

Используя условие $\delta \le h_N \le \delta + \varepsilon$ при $\varepsilon \to 0$, находим N. Заметим, что при таком подходе величина N может принимать только небольшие значения.

На втором этапе получаем численные решения уравнения (13), используя вычислительную схему, основанную на методе регуляризации, предложенном в [21]. Согласно этому подходу, регуляризованное решение поставленной задачи определяется из уравнения:

$$Af + \alpha f = g_\delta, \tag{15}$$

при соблюдении условия $\|Af - g_\delta\| \le \delta + \varepsilon$ при $\varepsilon \to 0$. Выбор параметра α можно осуществить с помощью различных подходов. В данном исследовании параметр α выберем следующим образом. Предположим, что для некоторого α_0 было получено решение f_δ^α уравнения (15). Вычислим $P = \|Af_\delta^\alpha\|$ и $M = \dfrac{P}{\|f_\delta^\alpha\|}$. Затем используем идею, предложенную в работе [22], согласно которой параметр регуляризации определим с помощью функции

$$\Psi(\alpha) = \|\alpha(\alpha E + A_{h_N})^{-2} g_\delta - f\| + \frac{\delta}{\alpha} + \frac{\|g_\delta\| h_N}{\alpha^2},$$

где $0 < \alpha \le M$. Для выбора параметра α мы на каждом этапе при фиксированном N определяем наименьшее по α значение функции $\Psi(\alpha)$. В работе доказано [22], что при таком подходе, мы получаем единственное обобщенное квазиоптимальное значение α. Таким образом, для численного решения уравнения Вольтерра мы получаем итерационную вычислительную схему, где параметрами регуляризации являются количество слагаемых N в ряде (11) и величина α.

4. Вычислительный эксперимент

Подход, позволяющий с помощью преобразований Лапласа свести решение задачи идентификации источника к интегральному уравнению, послужил основой для разработки численного метода и проведения вычислительного эксперимента. Основными целями эксперимента являлись оценка эффективности данного подхода, а также получение экспериментальных оценок погрешностей численных решений. Эксперимент проводился для тестовых функций с использованием равномерной сетки (x_i, t_j) из $(n+1)\times(r+1)$ узлов. В качестве

Таблица 1

Экспериментальные погрешности $\Delta_{f_{opt}}$

δ	f_1	f_2	f_3	f_4	f_5	f_6
0,02	0,0367	0,7471	0,0412	0,0522	0,1112	0,0811
0,05	0,0749	0,8551	0,0838	0,0732	0,1562	0,1672
0,1	0,1878	0,7878	—	0,1723	0,1885	0,2060

тестовых были использованы следующие функции:

$$f_1 = te^t, \quad f_2 = (1-t)\,e^{-2t}, \quad f_3 = t\left(e^{-t} - e^{-1}\right), \quad f_4 = sin\,(3\pi t)\,e^{-t},$$

$$f_5 = sin(10\pi t)e^{-t}, \quad f_6 = \begin{cases} sin(5\pi t) - 2.5t^2, \ t \in [0; 0.5), \\ cos(5\pi t), \quad t \in [0.5; 1] \end{cases}$$

Основные этапы вычислительного эксперимента состояли в следующем.

Этап 1. Для тестовой функции $f_k(t)$ находим решение $u_k(x,t)$ прямой задачи (4), (5), используя конечно-разностные уравнения. Затем моделируем значения $g(t_j)$ в узлах (x_1, t_j), с помощью конечно-разностных аппроксимаций частных производных. Далее задаем возмущение правой части уравнения (13) так, что $g_\delta(t_j) = g(t_j) + e_\delta(t_j)$, где $e_\delta(t_j)$ является случайной величиной равномерно распределенной на $[g(t_j) - \delta; g(t_j) + \delta]$.

Этап 2. Находим решение f_δ^α интегрального уравнения (13) с помощью предложенного численного метода. Для аппроксимации интегралов используем квадратуру правых прямоугольников. После построения f_δ^α вычисляем величину $\Delta_f = \|f_\delta^\alpha - f_k\|_{C(0,T)}$.

Этап 3. Подставляя найденые f_δ^α в (4) и используя (5), находим u_δ^α и вычисляем $\Delta_u = \|u_\delta^\alpha(x,t) - u_k(x,t)\|_{L_1(Q_T)}$.

Этап 4. Для исследования влияния параметров N и α на устойчивость метода повторяем этапы 2, 3 для различных N и α и вычисляем Δ_f при каждом повторе.

Заметим, что подходы, используемые для моделирования возмущенных данных и вычисления Δ_u обусловлены физическим смыслом задачи и спецификой проведения расчетов, основанных на экспериментальных данных, когда известен только максимальный уровень шума, а другие характеристики шума носят случайный характер и требуется, с одной стороны, избежать излишнего усложнения вычислительной процедуры, а с другой стороны, получить численное решение, имеющее приемлимый уровень погрешности.

Результаты вычислительного эксперимента приведены в таблицах и представлены на рисунках. Средние значения $\Delta_{f_{opt}}$ и $\Delta_{u_{opt}}$, полученные при различных δ и оптимальных значениях параметров $N(\delta)$ и $\alpha(\delta)$, представлены в таблицах 1 и 2, а соответствующие им графики и поверхности представлены на рис. 1–6. Таблицы 3–8 содержат результаты исследования влияния параметров N и α на погрешность решений при выбранном подходе к моделированию возмущенных данных. В статье представлены результаты этих исследований, полученные при $\delta = 0,02$.

На всех рисунках используются одинаковые обозначения. Одномерные рисунки иллюстрируют графики функций источника f. Обозначение «model» соответствует графикам тестовых функций f_k, $k = \overline{1,6}$ а графики решений f_δ^α уравнения (15) обозначены как f_δ.

Таблица 2

Экспериментальные погрешности $\Delta_{u_{opt}}$

δ	f_1	f_2	f_3	f_4	f_5	f_6
0,02	0,0076	0,0174	0,0125	0,0169	0,0188	0,0215
0,05	0,0218	0,0230	0,0251	0,0198	0,0202	0,0243
0,1	0,0267	0,0272	—	0,0274	0,0291	0,0304

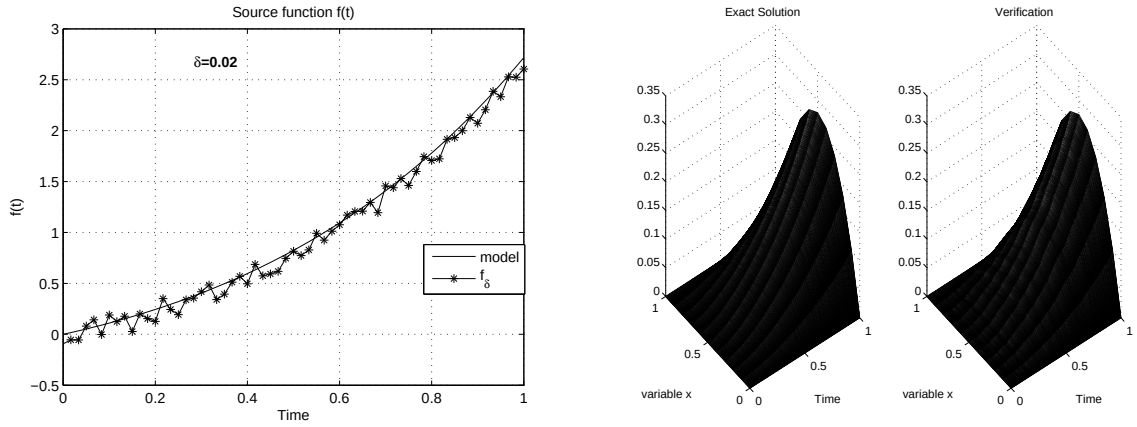

Рис. 1. Результаты численного решения задачи (1)–(3) для функции $f_1(t)$

Величина погрешности, при которой проводились расчеты, обозначена δ. Двумерные поверхности, имеющие обозначение «Exact Solution», соответствуют решениям u_k прямой задачи (4), (5), построенным для тестовых функций. Поверхности, соответствующие решению $u_\delta^\alpha(x,t)$, полученному с использованием функции f_δ^α, обозначены как «Verification».

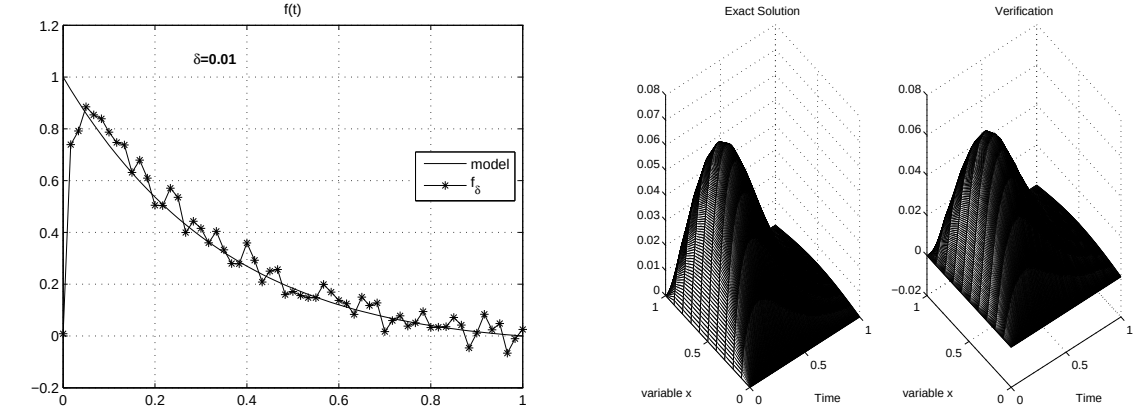

Рис. 2. Результаты численного решения задачи (1)–(3) для функции $f_2(t)$

На основании полученных результатов эксперимента можно сделать следующие выводы. Предложенный метод решения задачи идентификации функции источника позволяет получать регуляризованные решения с удовлетворительной точностью. При этом уровень погрешности исходных данных оказывают наиболее существенное влияние на величину погрешности приближенного решения f_δ^α. Влияние значений N и α на величину уклонения f_δ^α

Таблица 3

Экспериментальная погрешность Δ_{f_1} для функций f_1

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,0557	0,0536	0,0510	0,0524	0,0529	0,0538
4	0,0548	0,0512	0,0462	0,0474	0,0480	0,0491
5	0,0425	0,0386	0,0367	0,0381	0,0402	0,0408
6	0,0478	0,0433	0,0424	0,0442	0,0455	0,0461
7	0,0521	0,0508	0,0515	0,0518	0,0522	0,0524

Таблица 4

Экспериментальная погрешность Δ_{f_2} для функций f_2

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,8210	0,8115	0,7934	0,8024	0,8212	0,8234
4	0,8181	0,8023	0,7732	0,7874	0,8034	0,8177
5	0,8052	0,7756	0,7471	0,7618	0,7945	0,8067
6	0,8137	0,7893	0,7723	0,7856	0,8076	0,8111
7	0,8164	0,8005	0,7802	0,7912	0,8085	0,8123

Таблица 5

Экспериментальная погрешность Δ_{f_3} для функций f_3

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,0490	0,0478	0,0463	0,0468	0,0472	0,0478
4	0,0474	0,0467	0,0437	0,0445	0,0448	0,0451
5	0,0453	0,0438	0,0412	0,0418	0,0421	0,0423
6	0,0464	0,0452	0,0441	0,0445	0,0451	0,0452
7	0,0470	0,0462	0,0450	0,0458	0,0462	0,0465

Таблица 6

Экспериментальная погрешность Δ_{f_4} для функций f_4

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,0549	0,0543	0,0538	0,0542	0,0544	0,0546
4	0,0545	0,0539	0,0531	0,0535	0,0540	0,0542
5	0,0536	0,0529	0,0522	0,0533	0,0537	0,0539
6	0,0542	0,0536	0,0531	0,0535	0,0539	0,0542
7	0,0547	0,0542	0,0536	0,0540	0,0545	0,0549

Таблица 7

Экспериментальная погрешность Δ_{f_5} для функций f_5

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,1157	0,1154	0,1141	0,1143	0,1145	0,1146
4	0,1153	0,1115	0,1132	0,1134	0,1137	0,1141
5	0,1147	0,1143	0,1112	0,1118	0,1122	0,1127
6	0,1149	0,1146	0,1127	0,1129	0,1131	0,1133
7	0,1151	0,1147	0,1135	0,1136	0,1136	0,1137

Таблица 8

Экспериментальная погрешность Δ_{f_6} для функций f_6

N	$\alpha = 10^{-3}$	$\alpha = 10^{-4}$	$\alpha = 10^{-5}$	$\alpha = 10^{-6}$	$\alpha = 10^{-7}$	$\alpha = 10^{-8}$
3	0,0842	0,0838	0,0836	0,0839	0,0841	0,0840
4	0,0834	0,0832	0,0826	0,0830	0,0837	0,0839
5	0,0828	0,0817	0,0811	0,0819	0,0825	0,0827
6	0,0830	0,0825	0,0819	0,0827	0,0828	0,0829
7	0,0832	0,0831	0,0827	0,0829	0,0830	0,0831

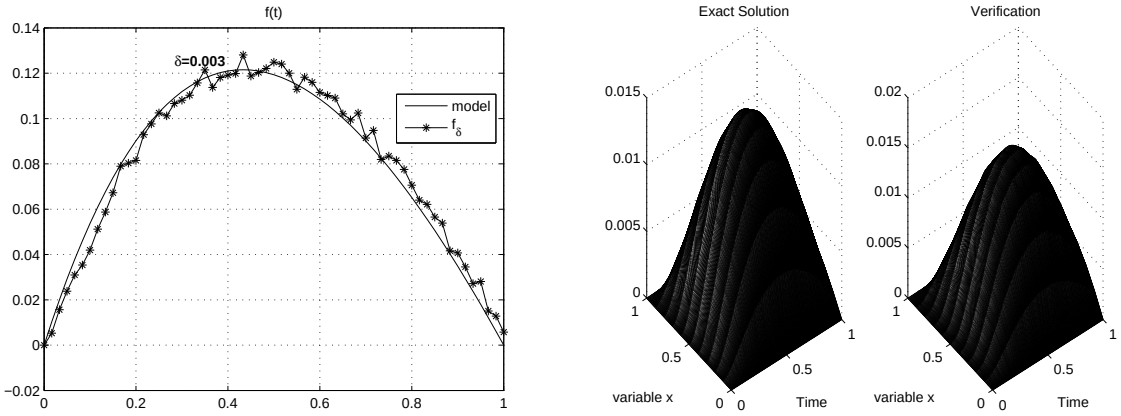

Рис. 3. Результаты численного решения задачи (1)–(3) для функции $f_3(t)$

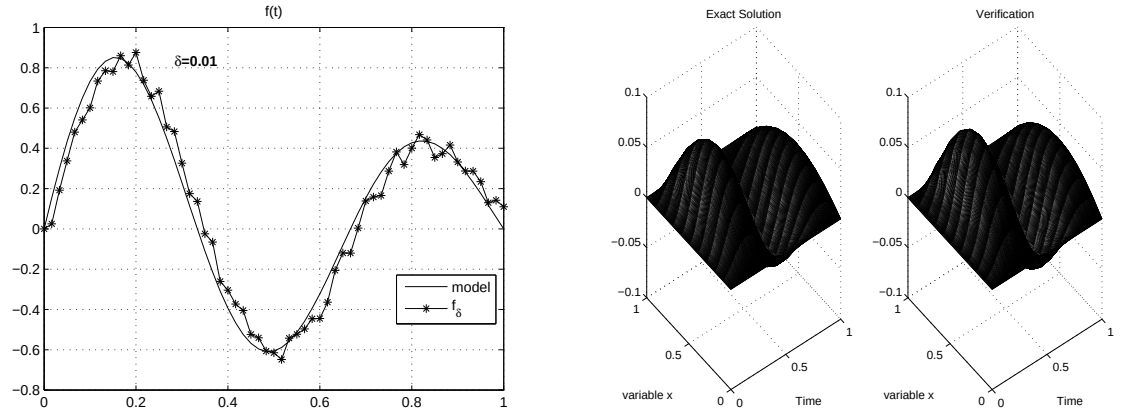

Рис. 4. Результаты численного решения задачи (1)–(3) для функции $f_4(t)$

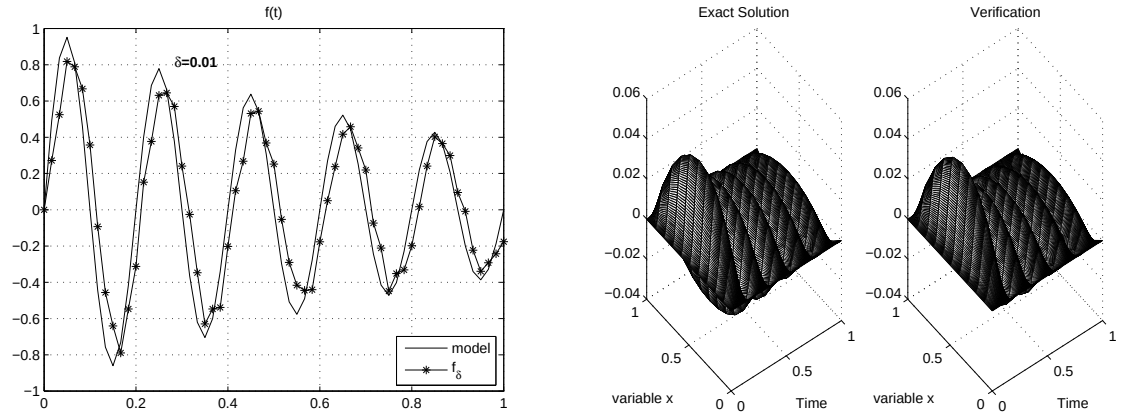

Рис. 5. Результаты численного решения задачи (1)–(3) для функции $f_5(t)$

от f_k при рассматриваемом подходе к моделированию возмущенных данных имеет гораздо меньший эффект, что позволяет ограничиваться небольшими значениями этих параметров при проведении расчетов, основанных на результатах реальных измерений.

Заключение

В статье рассмотрена обратная задача идентификации функции источника для параболического уравнения. С помощью применения прямого и обратного перобразований Лапла-

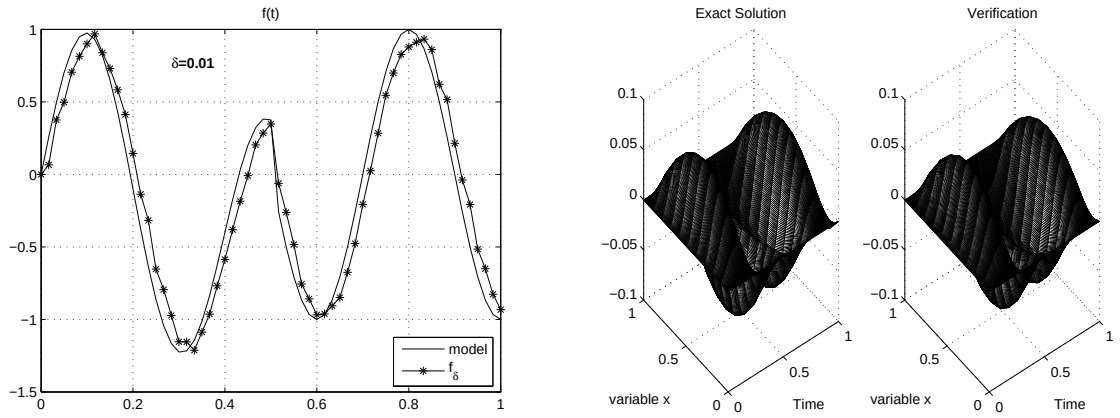

Рис. 6. Результаты численного решения задачи (1)–(3) для функции $f_6(t)$

са эта задача сводится к решению уравнения Вольтерра первого рода, характеризующее прямую зависимость неизвестной функции источника от известных граничных условий. Такой подход позволил исключить неустойчивую процедуру численного обращения преобразования Лапласа из вычислительной схемы. В статье предложен численный метод решения полученного интегрального уравнения, основанный на применении регуляризующего алгоритма с апостериорным выбором параметров регуляризации и послуживший основой для проведения вычислительного эксперимента. Результаты эксперимента свидетельствуют об устойчивости регуляризованных решений и достаточной эффективности предложенного подхода к решению задачи идентификации функции источника. Для более строгого исследования влияния параметра N на точность решений требуется провести дополнительные исследования, в которых возмущения в правой части (13) задаются специальным образом. Результаты данного исследования могут быть использованы при решении прикладных задач, возникающих, например, в металлургии, машиностроении, а также при исследовании различных диффузионных прцессов.

С целью уточнения оценки погрешности решений, полученных при использовании этого подхода, и улучшения сходимости построенных решений в дальнейшем планируется рассмотреть численное решение уравнения такого типа с помощью подходов, основанных на устойчивых разностных методах, где в качестве параметра регуляризации выступает шаг сетки по аналогии с результатами, представленными в работах [12], [14].

Литература

1. Алифанов О.М. Обратные задачи теплообмена. М.: Машиностроение, 1988. 280 с.

2. Erdogan A.S., Sazaklioglu A.U. A note on the numerical solution of an identification problem for observing two-phase flow in capillaries // Mathematical method in the Applied Sciences. 2014. Vol. 37, No. 16. P. 2393–2405.

3. Zenkour A.M., Abouelregal A.E. Vibration of FG nanobeams induced by sinusoidal pulse-heating via a nonlocal thermoelastic model // ACTA Mechanica. 2014. Vol. 225, No. 12. P. 3409–3421.

4. Вабищевич П.Н. Численное решение задачи идентификации правой части параболического уравнения // Известия вузов. Серия: Математика. 2003. № 1(488). С. 9–36.

5. Гольдман Н.Л. Однозначность определения функции источника в квазилинейной обрат-

ной задаче Стефана с финальным наблюдением // Доклады РАН. 2012. Т. 444, № 6. С. 597–601.

6. Прилепко А.И., Ткаченко Д.С. Корректность обратной задачи об источнике для параболических систем // Дифференциальные уравнения. 2004. Т. 40, № 11. С. 1540–1547.

7. Черепанова О.Н., Шипина Т.Н. Об одной задаче идентификации функции источника в параболическом уравнении // Журнал Северного Федерального Университета. Серия: Математика и физика. 2009. Т. 2, № 3. С. 370–375.

8. Hasanov A., Pektas B. Identification of an unknown time-dependent heat source term from overspecified Dirichlet boundary data by conjugate gradient method // Computers and Mathematics with Applications. 2013. Vol. 65, No. 1. P. 42–57.

9. Cialkowski M., Grysa K. A sequential and global method of solving an inverse problem of heat conduction equation // Journal of Theoretical and Applied Mechanics. 2010. Vol. 48, No. 1. P. 111–134.

10. Monde M., Arima H., Liu W., Mitutake Y., Hammad J.A. An analytical solution for two-dimensional inverse heat conduction problems using Laplace transform // International Journal of Heat and Mass Transfer. 2003. Vol. 46. P. 2135–2148.

11. Япарова Н.М. Метод решения одной обратной задачи идентификации функции источника для систем с распределенными параметрами // Вестник Тамбовского университета. Серия: Естественные и технические науки. 2015. Т. 20, № 5. С. 1549–1552.

12. Солодуша С.В., Япарова Н.М. Численное решение обратной граничной задачи теплопроводности с помощью уравнений Вольтерра I рода // Сибирский журнал вычислительной математики. 2015. Т. 18, № 3. С. 327–335.

13. Yaparova N. Numerical Methods for Solving a Boundary Value Inverse Heat Conduction Problem // Inverse Problems in Science and Engineering. 2014. Vol. 22, No. 5. P. 832–847.

14. Апарцин А.С., Бакушинский А.Б. Приближенное решение интегральных уравнений Вольтерра I рода методом квадратурных сумм // Дифференциальные и интегральные уравнения. Иркутск: Иркут. гос. ун-т, 1972. Вып. 1. С. 248–258.

15. Васин В.В., Сережникова Т.И. Регулярный алгоритм аппроксимации негладких решений для интегральных уравнений Фредгольма первого рода // Вычислительные технологии. 2010. Т. 15, № 2. С. 15–23.

16. Кабанихин С.И. Обратные и некорректные задачи. Новосибирск: Сибирское научное издание, 2009. 457 с.

17. Королев Ю.М., Ягола А.Г. Оценка погрешностив линейных обратных задачах при наличии априорной информации // Вычислительны методы и программирование: новые вычислительные технологии. 2012. Т. 13, № 1(25). С. 14–18.

18. Bushuev I. Global uniqueness for inverse parabolic problems with final observation // Inverse Problems. 1995. Vol. 11, No. 4. P. L11–L16.

19. Дёч Г. Руководство к практическому применению преобразования Лапласа и Z-преобразования. М.: Наука, 1971. 291 с.

20. Краснов М.Л. Интегральные уравнения. Введение в теорию. М.: Наука, 1975. 302 с.

21. Лаврентьев М.М. Условно-корректные задачи для дифференциальных уравнений. Новосибирск: НГУ, 1973. 71 с.

22. Леонов А.С. О квазиоптимальном выборе параметра регуляризации в методе Лаврентьева // Сибирский математический журнал. 1993. Т. 4, № 4. С. 695–703.

DOI: 10.14529/cmse160302

METHOD FOR SOLVING AN INVERSE TERM SOURCE PROBLEM BASED ON THE LAPLACE TRANSFORM

© 2016 N.M. Yaparova

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: ddjy@math.susu.ac.ru

Received: 09.08.2016

In this contribution, the method for solving the inverse source problem for parabolic partial differential equations with Dirichlet and Neumann boundary conditions is proposed. We reduce this problem to solving Volterra integral equation of the first kind via applications of the Laplace transforms. Then we use the regularization technique for numerical solving for the obtained equation. The integral equation describe the explicit dependence of the unknown source term on the Neumann boundary condition. The proposed approach allows to eliminate the unstable inverse Laplace transform from numerical scheme and simplify the computational procedure provided the basis to develop method for solving an inverse source problem. This approach to solving the inverse source problem is used for the first time. The efficiency of method and accuracy of the numerical solutions were evaluated by means of computational experiment.

Keywords: inverse source problem, Laplace transform, Volterra integral equation, numerical method

FOR CITATION

Yaparova N.M. Method for Solving an Inverse Term Source Problem Based on the Laplace Transform. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 20–35. (in Russian) DOI: 10.14529/cmse160302.

References

1. Alifanov O.M. *Obratnye zadachi teploobmena* [Inverse Heat Transfer Problems]. Moscow, Mashinostroenie, 1988. 280 p.

2. Erdogan A.S., Sazaklioglu A.U. A Note on the Numerical Solution of an Identification Problem for Observing Two-Phase Flow in Capillaries. *Mathematical Method in the Applied Sciences.* 2014. vol. 37, no. 16. pp. 2393–2405. DOI: 10.1002/mma.2985.

3. Zenkour A.M., Abouelregal A.E. Vibration of FG Nanobeams Induced by Sinusoidal Pulse-Heating via a Nonlocal Thermoelastic Model. *ACTA Mechanica.* 2014. vol. 225, no. 12. pp. 3409–3421. DOI: 10.1007/s00707-014-1146-9.

4. Vabishchevich P.N. Numerical Solution of the Problem Identifying the Right Part a Parabolic Equation. *Izvestiya vuzov. Seriya: Matematika* [Russian Mathematics]. 2003. vol. 47, no. 1. pp. 27—35. (in Russian)

5. Gol'dman N.L. Uniqueness of Determination of a Source Function in a Quasilinear Inverse Stefan Problem with Final Observation. *Doklady RAN* [Doklady Mathematics]. 2012. vol. 444, no. 6. pp. 597–601. (in Russian) DOI: 10.1134/s1064562412030337.

6. Prilepko A.I., Tkachenko D.S. Well-Posedness of the Inverse Source Problem for Parabolic Systems. *Differentsial'nye uravneniya* [Differential Equations]. 2004. vol. 40, no. 11. pp. 1619–1626. (in Russian) DOI: 10.1007/s10625-005-0080-y.

7. Cherepanova O.N., Shipina T.N. On a Problem of Identification of the Source Function in the Parabolic Equation. *Zhurnal Severnogo Federal'nogo Universiteta, Seriya: Matematika i fizika* [J. Noth. Federal Univ]. 2009. vol. 2, no. 3. pp. 370–375. (in Russian)

8. Hasanov A., Pektas B. Identification of an Unknown Time-Dependent Heat Source Term from Overspecified Dirichlet Boundary Data by Conjugate Gradient Method. *Computers and Mathematics with Applications.* 2013. vol. 65, no. 1. pp. 42–57. DOI: 10.1016/j.camwa.2012.10.009.

9. Cialkowski M., Grysa K. A Sequential and Global Method of Solving an Inverse Problem of Heat Conduction Equation. *Journal of Theoretical and Applied Mechanics.* 2010. vol. 48, no. 1. pp. 111–134.

10. Monde M., Arima H., Liu W., Mitutake Y., Hammad J.A. An Analytical Solution for Two-Dimensional Inverse Heat Conduction Problems Using Laplace Transform. *International Journal of Heat and Mass Transfer.* 2003. vol. 46. pp. 2135–2148. DOI: 10.1016/s0017-9310(02)00510-0.

11. Yaparova N.M. A Method for Solving the Inverse Problem of Identifying the Source Function for System with Distributed Parameters. *Vestnik tambovskogo universiteta. Seriya "Estestvennye i tekhnicheskie nauki"* [Tambov university reports. Series: Natural and Technical Sciences]. 2015. vol. 20, no. 5. pp. 1549–1552. (in Russian)

12. Solodusha S.V., Yaparova N.M. Numerical Solving an Inverse Boundary Problem of Heat Conduction Using Volterra Equations of the First Kind. *Sibirskiy zhurnal vychislitel'noy matematiki* [Numerical Analysis and Applications]. 2015. vol. 8, no. 3. pp. 267–274. (in Russian) DOI: 10.1134/s1995423915030076.

13. Yaparova N. Numerical Methods for Solving a Boundary Value Inverse Heat Conduction Problem. *Inverse Problems in Science and Engineering.* 2014. vol. 22, no. 5. pp. 832–847. DOI: 10.1080/17415977.2013.830614.

14. Apartsyn A.S., Bakushinskii A.B. Approximate Solution of Volterra Integral Equations of the First Kind by the Quadratures Method. *Differencialnye i integralnye uravneniya* [Differential and Integral Equations]. 1972. vol. 1. pp. 248–258. (in Russian)

15. Vasin V.V., Serezhnikova T.I. A Regularizing Algorithm for Approximation of a Non-Smoth Solution of Fredholm Integral Equations of the First Kind. *Vychislitel'nye tekhnologii* [Computatuional Technologies]. 2010. vol. 15, no. 2. pp. 15–23. (in Russian)

16. Kabanikhin S.I. *Obratnye i nekorrektnye zadachi* [Inverse and Ill-Posed Problems. Theory and Applications]. De Gruyter, Germany. 2011. 457 p. DOI: 10.1515/9783110224016.

17. Korolev Yu.M., Yagola A.G. Error Estimation in Linear Inverse Problems with Prior Information. *Vychislitel'ny metody i programmirovanie: novye vychislitel'nye tekhnologii* [Numerical Methods and Programming]. 2012. vol. 13, no. 1(25). pp. 14–18. (in Russian)

18. Bushuev I. Global Uniqueness for Inverse Parabolic Problems with Final Observation. *Inverse Problems.* 1995. vol. 11, no. 4. pp. L11–L16. DOI: 10.1088/0266-5611/11/4/001.

19. Doetsch G. *Rukovodstvo k prakticheskomu primeneniyu preobrazovaniya Laplasa i Zpreobrazovanij* [Guide to the Applications of the Laplace and Z Transforms]. Moscow, Nauka, 1971. 291 p.

20. Krasnov M.L. *Integralnye uravneniya vvedenie v teoriyu* [Integral Equations. Introduction to the Theory]. Moscow, Nauka, 1975. 302 p.

21. Lavrentiev M.M. *Uslovno korrektnye zadachi dlya differencialnyh uravnenij* [Conditionally Correct Problems for Differential Equations]. Novosibirsk, Publishing of Novosibirsk Goverment University, 1973. 71 p.

22. Leonov A.S. On Quasioptimum Selection of the Regularization Parameter in M.M. Lavreny'ev's Method. *Sibirskiy matematicheskiy zhurnal* [Siberian Mathemathical Journal]. 1993. vol. 34, no. 4. pp. 695–703. (in Russian) DOI: 10.1007/bf00975172.

УДК 519.6 DOI: 10.14529/cmse160303

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ЭВТРОФИКАЦИИ В МЕЛКОВОДНЫХ ВОДОЕМАХ НА МНОГОПРОЦЕССОРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ *

© 2016 г. А.И. Сухинов[1], А.В. Никитина[2], А.Е. Чистяков[2], И.С. Семенов[3], А.А. Семенякина[3], Д.С. Хачунц[2]

[1]*Донской государственный технический университет (344000 Ростов-на-Дону, пл. Гагарина, д. 1),*

[2]*Научно-исследовательский институт многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета (347928 Таганрог, ул. Чехова, д. 2),*

[3]*ОАО «Научно-исследовательский центр суперЭВМ и нейрокомпьютеров» (347900 Таганрог, пер. Итальянский, д. 106)*

E-mail: sukhin@gmail.com, nikitina.vm@gmail.com, cheese_05@mail.ru, flanker555@yandex.ru, j.a.s.s.y@mail.ru, diana-hachunts@mail.ru

Поступила в редакцию: 25.02.2016

Работа посвящена разработке методов решения модельной задачи эвтрофикации вод мелководного водоема, учитывающей движение водного потока, микротурбулентную диффузию, гравитационное оседание, пространственно-неравномерное распределение температуры и солености, а также загрязняющих биогенных веществ, кислорода, фито- и зоопланктона и др. В качестве объекта моделирования были выбраны мелководные водоемы — Азовское море и Таганрогский залив. Численное решение задачи основывается на градиентном методе вариационного типа — методе минимальных поправок. Ускорение и эффективность расчетов достигается при использовании многопроцессорной вычислительной системы Южного федерального университета. Решение поставленной задачи водной экологии позволит прогнозировать изменения качества вод мелководных водоемов, а также изучать механизмы формирования в них зон с пониженным содержанием кислорода.

Ключевые слова: математическая модель, эвтрофикация, метод минимальных поправок, параллельные вычисления, Азовское море.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Сухинов А.И., Никитина А.В., Чистяков А.Е., Семенов И.С., Семенякина А.А., Хачунц Д.С. Математическое моделирование процессов эвтрофикации в мелководных водоемах на многопроцессорной вычислительной системе // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 36–53. DOI: 10.14529/cmse160303.

Введение

Изменение природно-климатических условий и антропогенное воздействие приводят к эвтрофикации вод мелководных водоемов, приводящей к значительному росту популяций фитопланктона, многие виды которого являются токсичными, вызывают тяжелые заболевания у людей. Процессы эвтрофикации влияют на качество вод, состояние рыб-

* Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2016».

ных запасов водоемов, в связи с этим разработка программно-алгоритмического инструментария имеет не только научное, но и важное народнохозяйственное значение, она позволит осуществлять предсказательное моделирование взаимосвязанных процессов эвтрофикации на основе натурных экспериментов, полученных в ходе экспедиционных работ в акватории мелководных водоемов.

Целью работы являлось построение и численная реализация математической модели эвтрофикации вод мелководного водоема, обладающей предсказательной ценностью. В соответствии с поставленной целью решены следующие задачи: изучена предложенная модель эвтрофикации вод мелководного водоема, получены достаточные условия существования и единственности ее решения; разработан и исследован дискретный аналог этой модели. В работе описана построенная библиотека двухслойных методов вариационного типа, предназначенных для решения сеточных уравнений с самосопряженными и несамосопряженными операторами, возникающими при дискретизации разработанной модели, на многопроцессорной вычислительной системе. Для равномерного распределения вычислительной работы между процессорами использовался k-means алгоритм.

Предложенные модели и методы были применены при разработке программного комплекса, предназначенного для расчета значений концентраций планктона и загрязняющих примесей в Азовском море. Входными данными предложенной выше модели эвтрофикации вод мелководного водоема вида (1) – (3) является поле вектора скорости водной среды, рассчитанное по моделям Сухинова А.И., Чистякова А.Е. [3–6]. Статья организована следующим образом. В разделе 1 приводится разработанная математическая модель эвтрофикации вод мелководного водоема, а также ее исследование на непрерывном и дискретном уровне. Раздел 2 содержит описание разработанных методов вариационного типа для решения, возникающих в процессе дискретизации, сеточных уравнений. Раздел 3 посвящен описанию параллельного алгоритма реализации метода минимальных поправок. Раздел 4 содержит описание программного комплекса, численно реализующего модельную задачу эвтрофикации мелководного водоема. Раздел 5 посвящен описанию и анализу результатов численных экспериментов. В заключении приводятся итоги исследования.

1. Трехмерная математическая модель эвтрофикации мелководного водоема

Азовское море является внутренним морем расположено на востоке Европы. Его глубина не превышает 13,5 метров и является самым мелким морем в мире. Азовское море подвержено эвтрофикационным процессам в виду своей мелководности. Эвтрофикация (др. греч. εὐτροφία — хорошее питание) — это процесс обогащения морей, озер и рек биогенными веществами, сопровождающийся повышением объемов растительности в водоемах. Эвтрофикация может возникать как результат антропогенного воздействия, так и естественного старения водоема. К химическим биогенным элементам, способствующим эвтрофикации водоема, относятся азот и фосфор. Морские течения, зависящие от рельефа водоема, оказывают влияние на биогенный режим.

При построении модели эвтрофикации вод (ЭВ) Азовского моря и Таганрогского залива использовались работы Матишова Г.Г., Ильичева В.Г., Якушева Е.В., Сухинова А.И. [1], Крукиера Л.А., посвященные математическому моделированию гидрохимических процессов. В данных моделях учтено, что при штилях и малых ветрах в придон-

ных слоях Азовского моря возникают анаэробные условия. Отсутствие течений в приповерхностных донных слоях влечет за собой высвобождение в раствор (кроме сероводорода) аммония, сульфатов, органических соединений, силикатов и фосфатов, двухвалентного марганца и железа. Расчетная область G представлена на рис. 1.

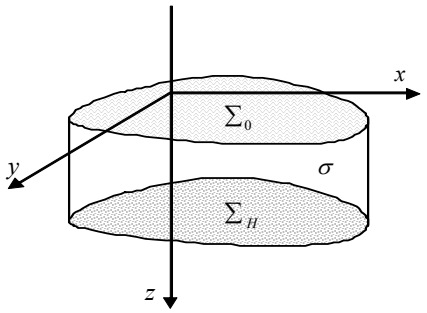

Рис. 1. Схема расчетной области $\overline{G}$

G представляет собой замкнутую область, ограниченную невозмущенной поверхностью водоема Σ_0, дном $\Sigma_H = \Sigma_H(x, y)$ и цилиндрической поверхностью σ для $0 < t \le T_0$. $\Sigma = \Sigma_0 \cup \Sigma_H \cup \sigma$ — кусочно-гладкая граница области G.

Модель будет иметь вид:

$$S'_{i,t} + u\frac{\partial S_i}{\partial x} + v\frac{\partial S_i}{\partial y} + \left(w - w_{gi}\right)\frac{\partial S_i}{\partial z} = \mu_i \Delta S_i + \frac{\partial}{\partial z}\left(\nu_i \frac{\partial S_i}{\partial z}\right) + \psi_i, \tag{1}$$

где S_i — концентрация i-й примеси, $i = \overline{1,17}$, индекс i указывает на вид субстанции, $i = \overline{1,17}$: 1 — сероводород (H_2S); 2 — элементная сера (S); 3 — сульфаты (SO_4); 4 — тиосульфаты (и сульфиты); 5 — общий органический азот (N); 6 — аммоний (NH_4) (аммонийный азот); 7 — нитриты (NO_2); 8 — нитраты (NO_3); 9 — растворенный марганец $(DOMn)$; 10 — взвешенный марганец $(POMn)$; 11 — растворенный кислород (O_2); 12 — силикаты (SiO_3— метасиликат; SiO_4 — ортосиликат); 13 — фосфаты (PO_4); 14 — железо $\left(Fe^{2+}\right)$ 15 — кремнекислота (H_2SiO_3— метакремневая; H_2SiO_4 — ортокремневая); 16 — фитопланктон; 17 — зоопланктон; $\mathbf{u} = (u, v, w)$ — скорость водного потока; w_{g_i} — скорость осаждения i-й компоненты; μ_i, ν_i — коэффициенты турбулентного перемешивания по горизонтальному и вертикальному направлениям соответственно; ψ_i — функция, описывающая химико-биологический источники (сток), а также агрегирование (слипание-разлипание), если соответствующая компонента является взвесью.

Система (1) рассматривается при следующих граничных условиях:

$$S_i = 0 \text{ на } \sigma, \text{ если } U_n < 0; \frac{\partial S_i}{\partial n} = 0 \text{ на } \sigma, \text{ если } U_n \ge 0; S'_{i,z} = 0 \text{ на } \Sigma_0; S'_{i,z} = -\varepsilon_i S_i \text{ на } \Sigma_H, \tag{2}$$

где ε_i — коэффициент поглощения i-й компоненты донным материалом.

Необходимо также добавить начальные условия:

$$S_i\big|_{t=0} = S_{i0}(x, y, z), \; i = \overline{1,17}. \tag{3}$$

Проведено исследование трехмерной модели ЭВ Азовского моря вида (1) – (3), получены достаточные условия существования и единственности ее решения, сформулированные в виде теорем.

Теорема 1. Пусть $S_i(x, y, z, t)$, $\psi_i \in C^2(Ц_t) \cap C(\overline{Ц}_t)$, где $Ц_t = G \times (0 < t < T_0)$; $\mu_i = \text{const} > 0$; $\mathbf{U} = \left(u, v, w - w_{gi}\right)$, $\nu_i(z) \in C^1(\overline{G})$; $S_{io} \in C(\overline{G})$, $i = \overline{1,17}$. Тогда при вы-

полнении неравенств: $\max\limits_G \{\mu_i, \nu_i\} - \frac{1}{\lambda_0} \max\limits_G \{|p_i|\} > 0$ для всех $i = \overline{1,17}$, где $\psi_i = p_i(S_j) S_i + \overline{\psi}_i, i \neq j$; $\overline{\psi}_i = LS_i - DS_i - p_i S_i$, $LS_i = \dfrac{\partial S_i}{\partial t} + div(S_i \mathbf{u}_i)$, $DS_i = \mu_i \Delta S_i + \dfrac{\partial}{\partial z}\left(\nu_i \dfrac{\partial S_i}{\partial z}\right)$, $\lambda_0 = \pi^2 \left(1 / L_x^2 + 1 / L_y^2 + 1 / L_z^2\right)$; L_x, L_y, L_z — пространственные максимальные размеры расчетной области; модельная задача эвтрофикации мелководного водоема ЭВ вида (1) – (3) имеет решение.

Теорема 2. Пусть $S_i(x, y, z, t)$, $\psi_i \in C^2(\Pi_t) \cap C(\overline{\Pi}_t)$, $\mu_i = \text{const} > 0$; $\mathbf{u}$, $\nu_i(z) \in C^1(\overline{G})$, $i = \overline{1,17}$. Тогда при выполнении неравенств: $2\mu_i \left(1 / L_x^2 + 1 / L_y^2\right) + 2\nu_i / L_z^2 \geq \varphi_i$ для всех $i = \overline{1,17}$ (где функции φ_i определяются источниками загрязняющего вещества (ЗВ)) модельная задача ЭВ мелководного водоема вида (1) – (3) имеет единственное решение. Для дискретизации модели (1) – (3) построим в области решения задачи связную сетку $\overline{\omega}_h$. h_x, h_y, h_z — векторные параметры, характеризующие плотность расположения узлов:

$$\overline{\omega}_h = \left\{x_j = jh_x, y_k = kh_y, z_l = lh_z;\ j = \overline{0, N_x}; k = \overline{0, N_y}; l = \overline{0, N_z};\right.$$
$$\left. N_x h_x = L_x, N_y h_y = L_y, N_z h_z = L_z\right\}, \overline{\omega}_{h\tau} = \overline{\omega}_h \times \overline{\omega}_\tau, \overline{\omega}_\tau = \left\{t_n = n\tau, n = \overline{0, N_t}\right\},$$

где i, j, k — индексы по направлениям x, y, z; N_x, N_y, N_z — количество узлов по координатным направлениям.

Расчетная область по пространственным направлениям x, y, z представляет собой объединение параллелепипедов. Дискретизируем модель (1) – (3) с помощью разностной схемы с весами:

$$\frac{S^{n+1}_{(i)j,k,l} - S^n_{(i)j,k,l}}{\tau} + \left(\overline{u}^n_{j+1/2,k,l} \frac{\gamma S^{n+1}_{(i)j+1,k,l} + (1-\gamma) S^n_{(i)j+1,k,l}}{2h_x} - \overline{u}^n_{j-1/2,k,l} \frac{\gamma S^{n+1}_{(i)j-1,k,l} + (1-\gamma) S^n_{(i)j-1,k,l}}{2h_x}\right) +$$
$$+ \left(\overline{v}^n_{j,k+1/2,l} \frac{\gamma S^{n+1}_{(i)j,k+1,l} + (1-\gamma) S^n_{(i)j,k+1,l}}{2h_y} - \overline{v}^n_{j,k-1/2,l} \frac{\gamma S^{n+1}_{(i)j,k-1,l} + (1-\gamma) S^n_{(i)j,k-1,l}}{2h_y}\right) +$$
$$+ \left(\left(\overline{w}^n_{j,k,l+1/2} - \overline{w}^n_{g(i)j,k,l+1/2}\right) \frac{\gamma S^{n+1}_{(i)j,k,l+1} + (1-\gamma) S^n_{(i)j,k,l+1}}{2h_z} -\right.$$
$$\left. - \left(\overline{w}^n_{j,k,l-1/2} - \overline{w}^n_{g(i)j,k,l-1/2}\right) \frac{\gamma S^{n+1}_{(i)j,k,l-1} + (1-\gamma) S^n_{(i)j,k,l-1}}{2h_z}\right) - \tag{4}$$
$$- \frac{\mu_i}{h_x^2}\left(\gamma S^{n+1}_{(i)j+1,k,l} + (1-\gamma) S^n_{(i)j+1,k,l} - 2\gamma S^{n+1}_{(i)j,k,l} - 2(1-\gamma) S^n_{(i)j,k,l}\right) - \frac{\mu_i}{h_x^2}\left(\gamma S^{n+1}_{(i)j-1,k,l} + (1-\gamma) S^n_{(i)j-1,k,l}\right) -$$
$$- \frac{\mu_i}{h_y^2}\left(\gamma S^{n+1}_{(i)j,k+1,l} + (1-\gamma) S^n_{(i)j,k+1,l} - 2\gamma S^{n+1}_{(i)j,k,l} - 2(1-\gamma) S^n_{(i)j,k,l}\right) - \frac{\mu_i}{h_y^2}\left(\gamma S^{n+1}_{(i)j,k-1,l} + (1-\gamma) S^n_{(i)j,k-1,l}\right) -$$
$$- \frac{1}{h_z}\left(\nu^n_{(i)j,k,l+1/2} \frac{\gamma S^{n+1}_{(i)j,k,l+1} + (1-\gamma) S^n_{(i)j,k,l+1} - \gamma S^{n+1}_{(i)j,k,l} - (1-\gamma) S^n_{(i)j,k,l}}{h_z}\right) +$$
$$+ \frac{1}{h_z}\left(\nu^n_{(i)j,k,l-1/2} \frac{\gamma S^{n+1}_{(i)j,k,l} + (1-\gamma) S^n_{(i)j,k,l} - \gamma S^{n+1}_{(i)j,k,l-1} - (1-\gamma) S^n_{(i)j,k,l-1}}{h_z}\right) - \overline{\psi}_i = 0,$$

$1 \leq j \leq N_x - 1, 1 \leq k \leq N_y - 1, 1 \leq l \leq N_z - 1, i = \overline{1,17}$.

К системе (4) добавим аппроксимированные начальные и граничные условия. Исследуем устойчивость разностной схемы (4) для этого запишем ее в каноническом виде:

$$B_1 S^{n+1}_{(i)j+1,k,l} + B_2 S^{n+1}_{(i)j,k+1,l} + B_3 S^{n+1}_{(i)j,k,l+1} + A_1 S^{n+1}_{(i)j,k,l} - B_4 S^{n+1}_{(i)j-1,k,l} -$$
$$-B_5 S^{n+1}_{(i)j,k-1,l} - B_6 S^{n+1}_{(i)j,k,l-1} = -A_2 S^{n}_{(i)j,k,l} + B_7 S^{n}_{(i)j+1,k,l} + B_8 S^{n}_{(i)j,k+1,l} +$$
$$+B_9 S^{n}_{(i)j,k,l+1} - B_{10} S^{n}_{(i)j-1,k,l} - B_{11} S^{n}_{(i)j,k-1,l} - B_{12} S^{n}_{(i)j,k,l-1} - \overline{\psi}_i, \quad (5)$$
$$1 \le j \le N_x - 1, 1 \le k \le N_y - 1, 1 \le l \le N_z - 1, 0 \le n \le N_t - 1, i = \overline{1,17}.$$

Достаточное условие монотонности и устойчивости модели (4) определяется на основе принципа максимума А. А. Самарского при следующих ограничениях:

$$h_x < \left\| 2\mu_i / \overline{u}^n \right\|_{C(\bar{\omega}_h)}, h_y < \left\| 2\mu_i / \overline{v}^n \right\|_{C(\bar{\omega}_h)}, h_z < \left\| 2\nu_i / \left(\overline{w}^n - \overline{w}^n_{g_i} \right) \right\|_{C(\bar{\omega}_h)}, i = \overline{1,17}. \quad (6)$$

Исследование погрешности аппроксимации разработанной схемы с весами вида (4) показало, что при $\gamma = 0.5$ (схема Кранка — Николсон) она имеет наибольший порядок точности по временной и пространственным переменным: $O\left(\tau^2 + |h|^2\right)$, где $|h| = \sqrt{h_x^2 + h_y^2 + h_z^2}$.

2. Метод решения сеточных уравнений

Дискретную модель (4) может быть представлена в виде операторного уравнения:

$$Au = f \quad (7)$$

с положительно определенным оператором A в гильбертовом пространстве H. Рассмотрим неявный двухслойный процесс вида

$$B \frac{y_{k+1} - y_k}{\tau_k} + Ay_k = f, k = 0, 1, \ldots \quad (8)$$

с неким начальным приближением $y_0 \in H$ и оператором — предобуславливателем B [7]. Всякий двухслойный итерационный процесс (8), характеризуется операторами A и B, энергетическим пространством H_D, в котором исследуется сходимость метода, и способом расчета итерационных параметров τ_k. К основным вопросам теории итерационных методов решения сеточных уравнений относится оптимальный выбор параметров τ_k [8–10].

Для вычисления параметров τ_k в методах вариационного типа не требуется априорная информация об операторах задачи (7) (кроме условий общего вида $A = A^* > 0$, $\left(DB^{-1}A\right)^* = DB^{-1}A$). Данные методы основаны на принципе: если задано значение y_k, а y_{k+1} находится по схеме (8), то параметр τ_{k+1} рассчитывается из условия минимизации в H_D энергетической нормы погрешности $z_{k+1} = y_{k+1} - u$, где u — решение уравнения (7). Последовательность y_k, рассчитанная на основе выражения (8), в которой значения τ_k вычисляются из приведенного выше условия, является минимизирующей для квадратичного функционала вида $I(y) = \left(D(y-u), y-u\right)$. Данный функционал имеет ограничения снизу (в силу положительной определенности оператора D), и его значение достигает минимума, равного нулю при $y = u$. Выбор параметра τ_{k+1} из приведенного условия гарантирует локальную минимизацию функционала $I(y)$ при переходе от y_k к y_{k+1}. В случае задания оператора-предобуславливателя методом простой итерации $(B = E)$ переход от y_k к y_{k+1} осуществляется согласно выражению

$$y_{k+1} = y_k - \tau_{k+1} r_k, \; r_k = Ay_k - f.$$

В случае положительно определенного самосопряженного оператора A переход от y_k к y_{k+1} происходит в направлении $-r_k$, которое противоположно направлению гради-

ента для функционала $(A(y-u), y-u)$ в точке y_k. В направлении антиградиента происходит наискорейшее убывание значения функционала. Параметр τ_{k+1} в H_D находится из условия минимизации нормы погрешности $z_{k+1} = y_{k+1} - u$ [11].

Формула для вычисления итерационного параметра τ_{k+1} находится в предположении невырожденности оператора A. Погрешности находятся из уравнений: $z_k = y_k - u$, $k = 0, 1, \ldots$. Схема (8) с учетом $y_k = \tau_k + u$, примет вид: $z_{k+1} = \left(E - \tau_k B^{-1} A\right) z_k$, $k = 0, 1, \ldots$, $z_0 = y_0 - u$. С помощью замены $z_k = D^{-1/2} x_k$ осуществляется переход к выражению, содержащему только один оператор:

$$x_{k+1} = S_{k+1} x_k,\ S_k = E - \tau_k C,\ C = D^{-1/2}\left(DB^{-1}A\right)D^{-1/2}. \tag{9}$$

С помощью равенства $\|z_k\|_D = \|x_k\|$ ($\|\cdot\|_D$ — норма в H_D, $\|\cdot\|$ — норма в H) описанную задачу расчета параметра τ_{k+1} можно описать следующим образом: рассчитывается значение параметр τ_{k+1} из условия минимизации нормы x_{k+1} в пространстве H. Рассмотрим норму:

$$\begin{aligned}\|x_{k+1}\|^2 &= \left(\left(E - \tau_{k+1}C\right)x_k, \left(E - \tau_{k+1}C\right)x_k\right) = \|x_k\|^2 - 2\tau_{k+1}\left(Cx_k, x_k\right) + \tau_{k+1}^2\left(Cx_k, Cx_k\right) = \\ &= \left(Cx_k, Cx_k\right)\left[\tau_{k+1} - \left(Cx_k, x_k\right) / \left(Cx_k, Cx_k\right)\right]^2 + \|x_k\|^2 - \left(Cx_k, x_k\right)^2 / \left(Cx_k, Cx_k\right).\end{aligned} \tag{10}$$

Оператор C не вырожден, поскольку не вырожден оператор A. Таким образом, для любого x_k имеем $\left(Cx_k, Cx_k\right) > 0$, и минимальное значение нормы x_{k+1} достигается при

$$\tau_{k+1} = \left(Cx_k, x_k\right) / \left(Cx_k, Cx_k\right). \tag{11}$$

Подставим (11) в (10) и получим:

$$\|x_{k+1}\| = \rho_{k+1}\|x_k\|, \tag{12}$$

где

$$\rho_{k+1}^2 = 1 - \left(Cx_k, x_k\right)^2 / \left\{\left(Cx_k, Cx_k\right)\left(x_k, x_k\right)\right\}. \tag{13}$$

Выражение (11) описывает оптимальные значения итерационного параметра τ_{k+1}. Выражение (11) с учетом $x_k = D^{-\frac{1}{2}} z_k$ запишется в виде:

$$\tau_{k+1} = \left(DB^{-1}Az_k, z_k x_k\right) / \left(DB^{-1}Az_k, B^{-1}Az_k\right),\ k = 0, 1, \ldots. \tag{14}$$

Принимая во внимание, что $Az_k = Ay_k - Au = Ay_k - f = r_k$ — невязка, а $B^{-1}r_k = \omega_k$ — поправка, выражение для расчета параметра τ_{k+1} запишется в виде:

$$\tau_{k+1} = \left(D\omega_k, z_k\right) / \left(D\omega_k, \omega_k\right),\ k = 0, 1, \ldots, \tag{15}$$

а двухслойная итерационная схема (8) представима в явном виде:

$$y_{k+1} = y_k - \tau_{k+1}\omega_k,\ k = 0, 1, \ldots, \tag{16}$$

то алгоритм для данного метода, запишется в виде:

1) для вектора y_k вычисляется вектор невязки $r_k = Ay_k - f$;

2) вычисляются значения вектора поправки из уравнения $B\omega_k = r_k$;

3) рассчитывается параметр τ_{k+1} из выражения (15);

4) рассчитывается новое приближение y_{k+1} из выражения (16).

Рассмотрим двухслойные градиентные методы, которые использованы при решении модельной задачи эвтрофикации вод мелководного водоема. Каждый метод из данного класса имеет свою область применимости и определяется способом задания

оператора D. Оператор D выбирается таким образом, чтобы выражение (15) для итерационного параметра τ_{k+1} содержало только известные величины.

Если оператор A самосопряжен и положительно определен в H, то для решения (8) можно использовать метод скорейшего спуска (МСС). Если оператор A невырожден и несамосопряжен, а оператор B^*A является положительно определенным, то применим метод минимальных невязок (ММН) [12]. С учетом $Az_k = Ax_k - f = r_k$ и $A = A^*$ и выражения (15) получим формулу для расчета итерационного параметра τ_{k+1} согласно методу скорейшего спуска:

$$\tau_{k+1} = (r_k, \omega_k) / (A\omega_k, A\omega_k), k = 0, 1, \ldots . \tag{17}$$

Для случая $B = E$ получим $\omega_k = B^{-1}r_k = r_k$ и выражение для расчета τ_{k+1} примет вид:

$$\tau_{k+1} = (r_k, r_k) / (Ar_k, Ar_k), k = 0, 1, \ldots . \tag{18}$$

При решении модельной задачи (1) – (3) методом скорейшего спуска по невязке параметр τ_{k+1} рассчитывается по формуле (18), а метод скорейшего спуска по поправке реализуется с помощью расчетной формулы (17).

Метод минимальных поправок (ММП) можно применять для решения модельной задачи (1) – (3) в случае любого несамосопряженного невырожденного оператора A, кроме того требуется положительная определенность оператора B^*A. Для метода минимальных невязок $D = A^*A$.

Формула для итерационного параметра τ_{k+1} в ММП имеет вид:

$$\tau_{k+1} = (A\omega_k, r_k) / (A\omega_k, A\omega_k), k = 0, 1, \ldots . \tag{19}$$

В случае ($B = E$) требуется положительная определенность оператора A, а формула для τ_{k+1} примет вид:

$$\tau_{k+1} = (Ar_k, r_k) / (Ar_k, Ar_k), k = 0, 1, \ldots . \tag{20}$$

В таблице приведены результаты сравнения скоростей сходимости двухслойных градиентных методов вариационного типа, используемых для решения задачи эвтрофикации вод мелководного водоема вида (1) – (3).

В таблице используются следующие обозначения: метод минимальных невязок (ММН); метод минимальных поправок (ММП); метод скорейшего спуска по невязке (МССН); метод скорейшего спуска по поправке (МССП); p — номер итерации; n — номер временного слоя; I_φ — количество итераций, необходимое для сходимости метода решения СЛАУ для уравнения, описывающего изменение концентрации φ, $\varphi \in \{S, X\}$, где $S = S_5$, $X = S_{16}$.

Метод минимальных поправок (ММП) был выбран в качестве основного метода в виду его наибольшей скорости сходимости, согласно данным, приведенным в таблице.

Таблица

Сравнение скоростей сходимости методов вариационного типа

ММП				
p \ n	1	2	3	4
1	$I_S = 48, I_X = 44$	$I_S = 47, I_X = 43$	$I_S = 47, I_X = 43$	$I_S = 48, I_X = 42$
2	$I_S = 53, I_X = 48$	$I_S = 52, I_X = 47$	$I_S = 53, I_X = 46$	$I_S = 53, I_X = 45$
3	$I_S = 21, I_X = 21$	$I_S = 22, I_X = 20$	$I_S = 22, I_X = 19$	$I_S = 21, I_X = 19$
4	$I_S = 17, I_X = 15$	$I_S = 14, I_X = 13$		
ММН				
1	$I_S = 83, I_X = 65$	$I_S = 77, I_X = 64$	$I_S = 76, I_X = 63$	$I_S = 76, I_X = 62$
2	$I_S = 87, I_X = 73$	$I_S = 87, I_X = 73$	$I_S = 86, I_X = 72$	$I_S = 85, I_X = 71$
3	$I_S = 51, I_X = 51$	$I_S = 50, I_X = 46$	$I_S = 49, I_X = 43$	$I_S = 48, I_X = 40$
4	$I_S = 40, I_X = 38$	$I_S = 38, I_X = 35$		
МССП				
1	$I_S = 69, I_X = 62$	$I_S = 67, I_X = 61$	$I_S = 67, I_X = 60$	$I_S = 67, I_X = 59$
2	$I_S = 71, I_X = 68$	$I_S = 74, I_X = 67$	$I_S = 74, I_X = 67$	$I_S = 74, I_X = 66$
3	$I_S = 35, I_X = 32$	$I_S = 35, I_X = 30$	$I_S = 35, I_X = 29$	$I_S = 34, I_X = 28$
4	$I_S = 24, I_X = 23$	$I_X = 21, I_X = 20$		
МССН				
1	$I_S = 68, I_X = 64$	$I_S = 79, I_X = 67$	$I_S = 78, I_X = 66$	$I_S = 78, I_X = 65$
2	$I_S = 86, I_X = 74$	$I_S = 87, I_X = 73$	$I_S = 86, I_X = 73$	$I_S = 86, I_X = 72$
3	$I_S = 53, I_S = 49$	$I_S = 52, I_X = 45$	$I_S = 51, I_X = 42$	$I_S = 49, I_X = 41$
4	$I_S = 40, I_X = 38$	$I_S = 36, I_X = 35$		

3. Параллельный вариант метода решения сеточных уравнений

Для геометрического разбиения расчетной области с целью равномерной загрузки вычислителей (процессоров) МВС использовался метод k-means, который позволяет найти центры подобластей: $Q = Q^{(3)}$ на основе на минимизации функционала суммарной выборочной дисперсии функции, описывающей расположение элементов (расчетных узлов сетки). Пусть X_i — множество элементов i – ой подобласти, $i \in \{1, \ldots, m\}$, m — количество подобластей. $Q^{(3)} = \sum\limits_i \frac{1}{|X_i|} \sum\limits_{x \in X_i} d^2(x, c_i) \to \min$, где $c_i = \frac{1}{|X_i|} \sum\limits_{x \in X_i} x$ — центр подобласти X_i, а $d(x, c_i)$ — расстояние между центром подобласти c_i и элементом x в Евклидовой метрике. Метод k-means позволяет разбить исходную область на примерно равные подобласти.

Результат работы метода k-means для модельных двумерной и трехмерной областей представлен на рис. 2.

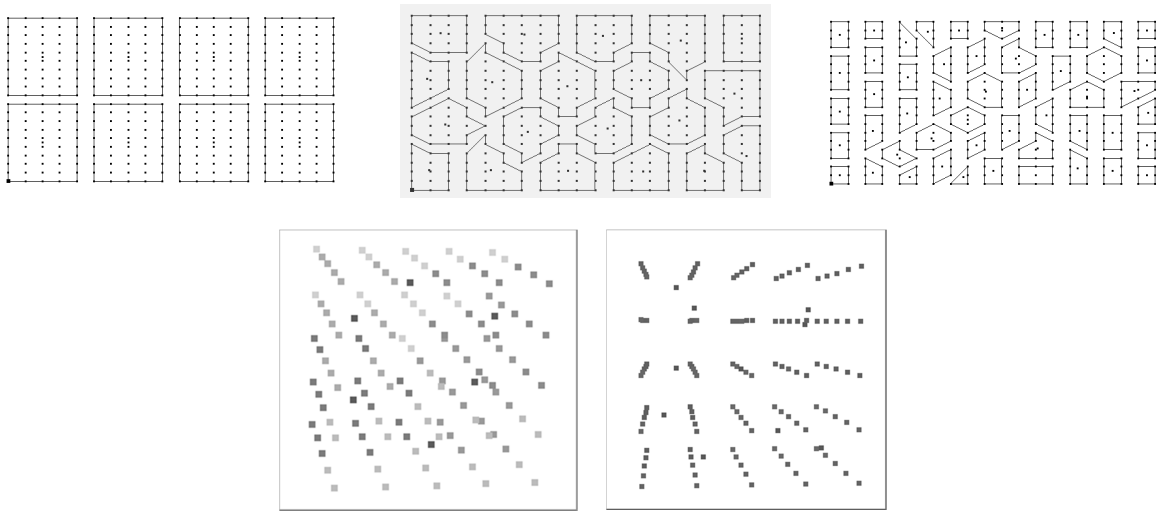

Рис. 2. Декомпозиция области на основе алгоритма k-means: для двумерной (на 9, 38, 150 подобластей); для трехмерной (на 6 и 10)

Опишем работу максиминного алгоритма. Начальное расположение центров подобластей задаются расчетными узлами сетки следующим образом:

1) первый центр подобласти задается первым элементом;
2) второй центр расположен на максимальном удалении от первого центра подобласти;
3) каждый следующий центр подобласти располагается из условия на максимального удаления от ближайшего центра.

Опишем алгоритм работы k-means.

1) Рассчитываются начальное расположение центров подобластей при помощи максиминного алгоритма.
2) Все элементы кластеризуются на m клеток Вороного согласно методу ближайшего соседа, т. е. для любого $x \in X_c$, где X_c — подобласть, должно выполняться условие $\|x - s_c\| = \min\limits_{1 \le i \le m} \|x - s_i\|$, где s_c – центр области X_c.
3) Расчет новых центров осуществляется согласно выражению $s_c^{(k+1)} = \dfrac{1}{\left|X_i^{(k)}\right|} \sum\limits_{x \in X_i^{(k)}} x$.
4) Выполняется проверка условия остановки алгоритма $s_c^{(k+1)} = s_c^{(k)}$ для всех $k = 1, \ldots, m$. Если условие остановки алгоритма не выполняется, то переходим к пункту 2.

Элементы, лежащие на границах подобластей, обмениваются данными при параллельной реализации алгоритмов решения сеточных уравнений. Задача построения выпуклой оболочки решалась с помощью алгоритма Джарвиса. На основе данного алгоритма найдено расположение элементов, участвующих в обменах данными между вычислителями, и номера вычислителей, передающих и принимающих соответствующие данные [13, 14].

Метод сдваивания был использован для расчета итерационного параметра τ при решении сеточных уравнений методом минимальных поправок. Следует отметить, что синхронизация разработанного алгоритма решения задачи (1) – (3) производится только при решении сеточных уравнений ММП на переходах между итерационными слоями.

4. Описание программного комплекса

Для решения модельной задачи эвтрофикации мелководного водоема (1) – (3) был создан комплекс прикладных программ, позволяющий производить расчеты концентраций загрязняющих веществ (ЗВ), фито- и зоопланктона в областях сложной формы (Азовское море и Таганрогский залив) [15].

С помощью созданного комплекса программ можно осуществлять:

- внедрение системы и совершенствование комплексного рыбохозяйственного мониторинга в водоемах (кормовых запасов и базы промысловых объектов наблюдение, прогноз и оценка состояния режима экосистем);
- разработку, согласование предложений и мероприятий по обеспечению оптимального режима, сохранения биоразнообразия промысловых ресурсов, экосистем мелководных водоемов;
- совершенствование методологии природоохранных исследований, разработка новых, апробация и внедрение перспективных методов изучения состояния водных экосистем и отдельных компонентов;
- разработку и совершенствование методов диагностики токсического воздействия ЗВ на гидробионты, в том числе ранней и дифференциальной диагностики токсикоза, а также поиск средств антидотной защиты водных экосистем;
- организацию и проведение исследований по выявлению тенденций и закономерностей изменения состояния водных экосистем под воздействием антропогенных факторов, разработка предложений и мероприятий по снижению и предупреждению таких воздействий;
- оценку ущербов рыбному хозяйству, наносимых разными видами хозяйственной деятельности, разработку предложений по предотвращению, уменьшению и адекватной компенсации ущербов.

Значения поля скоростей движения водной среды, рассчитанные в работах [16], относятся к входной информации для моделей гидробиологических процессов, описанных в первой главе. Для математического моделирования гидробиологических и гидродинамических процессов в трехмерной области сложной формы — Азовское море и Таганрогский залив использовались последовательно сгущающиеся прямоугольные сетки размерностями: $251 \times 351 \times 15$, $502 \times 702 \times 30$,$1004 \times 1404 \times 60$,....

Начальное распределение загрязняющих веществ и планктона было учтено в форме, соответствующей пространственно-временным масштабам моделируемых процессов. Разработанный алгоритм численного решения поставленной задачи позволяет свободно варьировать значениями соответствующих параметров, видами управляющих функций и граничные условиями [17]. Понимание механизма функционирования системы и знание ее основных характеристик позволили для преодоления трудностей при настройке программы использовать феноменологический подход. Эффективность такого подхода связана с тем, что адекватность описания поведение экологической системы часто определяется точностью не отдельных параметров, а их соотношений.

Калибровка и верификация разработанной модели эвтрофикации вод мелководного водоема проводились на основе экологических данных по Азовскому морю, полученных в ходе научно-исследовательских экспедиций, проводимых учеными ЮФУ, начиная с 2000 года. В ходе исследований акватории Азовского моря изучались: виды и концентрации основных загрязняющих воды Азовского моря веществ; пространственные рас-

пределения солености и температуры; кислородный режим; видовой состав фито- и зоопланктона; механизмы возникновения и развития заморов в центрально-восточной части водоема. Обработка экспедиционных данных заключалась в оцифровке, пересчете в стандартные единицы, классификации для использования в разных модельных задачах гидробиологии моря.

На рис. 3 приведена схема разработанного программного комплекса.

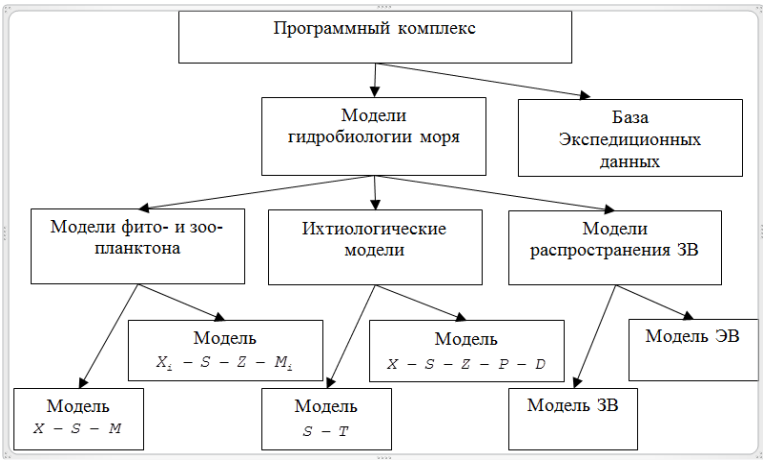

Рис. 3. Схема программного комплекса

Программный комплекс включает: блок управления, базы океанологических и метеорологических данных, системы интерфейсов, системы ввода — вывода и визуализации. Комплекс программ обладает удобным интерфейсом и обеспечивает ввод данных в диалоговом режиме. Построенный программный комплекс имеет универсальный характер и привязка его к условиям моделируемых районов и объектов осуществляется, как правило, на уровне входной информации. Для практического использования модулей требуется создание специальной информационной базы, содержащей сведения о параметрах, определяющих источники примесей, о климатических и физико-географических условиях исследуемых объектов.

При разработке программного комплекса использовался язык высокого уровня C++. Были разработаны две версии параллельного алгоритма решения модельной задачи эвтрофикации вод мелководного водоема. Первая предназначена для ЭВМ с операционной системой Windows, основана на технологии создания в этой системе дополнительных потоков. Вторая может использоваться для кластерных систем, и основана на технологии передачи сообщений (MPI).

5. Результаты численных экспериментов

При моделировании пространственно-неоднородных процессов эвтрофикации вод Азовского моря (1) – (3) учитывалась внешняя периодичность, приводящая к усложнению системы. При этом колебания плотности планктона могут быть столь велики, что не могут быть объяснены случайными флуктуациями, и визуальная картина такова, что сравнительно небольшие площади высокой плотности («облака», «пятна») разделены пространствами с различными плотностями, зачастую не фиксируемыми общепринятыми методами наблюдений. Особенно ярко это явление выражено в тех местах водоема, для которых характерна потребность в биогенных элементах. При моделировании процессов эвтрофикации учитывался вегетационный период фитопланктона.

Диффузные процессы сглаживают пространственное распределение и рассеивают «пятна». Одна из попыток объяснить парадокс стабильности «пятен» с помощью численных экспериментов заключалась в предположении об активном передвижении гетеротрофных организмов (зоопланктона и рыб) в направлении градиента «пищи», что обеспечивает закрепление пространственной неоднородности биогенных веществ в водной среде. Устойчивая пространственная неоднородность распределения может быть, например, связана с диффузионными процессами и наличием у фитопланктона механизма эктокринного регулирования, т.е. регулирования скорости роста водорослей посредством выделения в среду биологически активных метаболитов.

На рисунке 4. показаны результаты расчета концентрации загрязняющего биогенного вещества для модели (1) – (3) (начальное распределение полей течений при северном ветре). Приведенные ниже рисунки отражают влияние структур течений водного потока в Азовском море на распределение загрязняющего биогенного вещества и фитопланктона. Белым цветом выделены максимальные значения концентраций биогенного вещества (азота) и фитопланктона черным — минимальные.

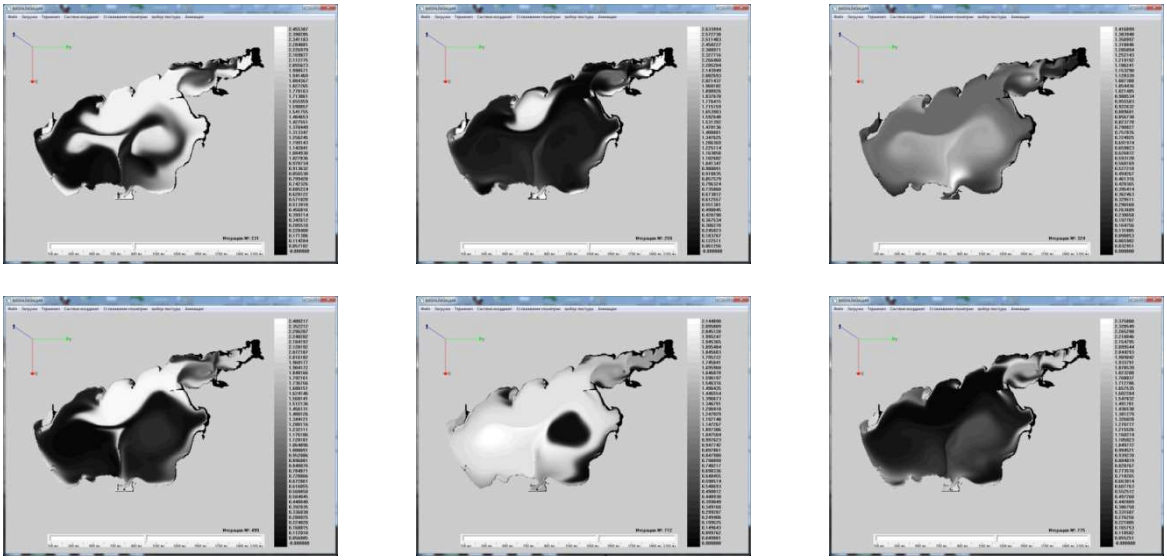

Рис. 4. Распределение концентрации загрязняющих веществ в различные моменты времени

Результаты моделирования динамики фитопланктона в Азовском море представлены на рис. 5. (N — номер итерации, начальное распределение полей течений водного потока при северном направлении ветра).

Критерием проверки адекватности построенных моделей гидробиологии мелководного водоема служила оценка погрешности моделирования с одновременным учетом натурных данных по имеющимся n замерам, которая вычислялась по формуле: $\delta = \sqrt{\sum_{k=1}^{n} \left(S_{k\ nat} - S_k\right)^2} \Big/ \sqrt{\sum_{k=1}^{n} S_{k\ nat}^2}$, где $S_{k\ nat}$ — значение концентрации загрязняющей примеси, полученное с помощью натурных экспедиционных измерений; S_k — значение, рассчитанное с помощью модели (1) – (3).

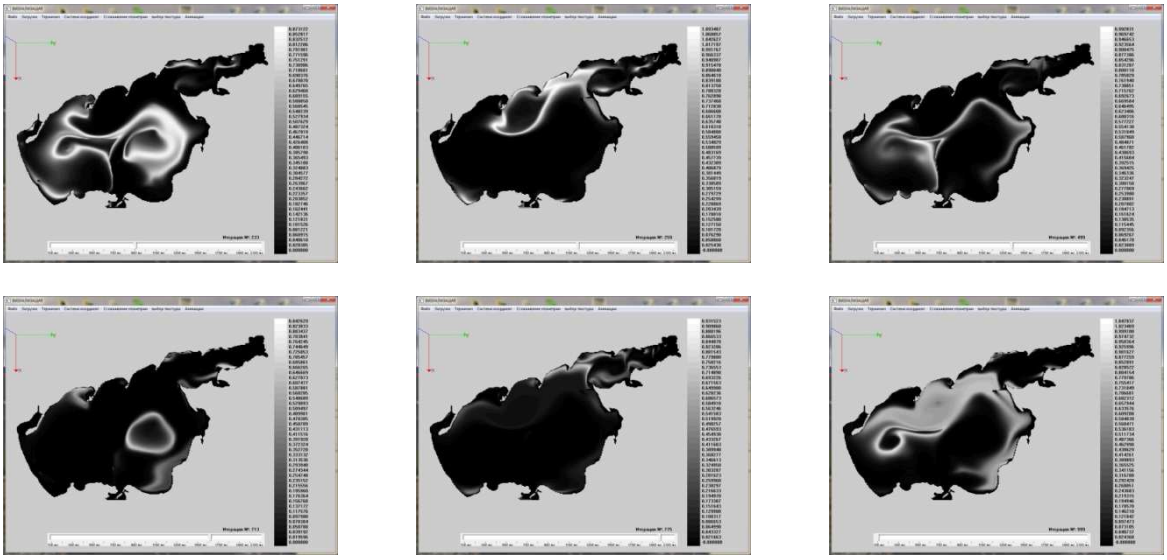

Рис. 5. Распределение концентрации фитопланктона в различные моменты времени

Рассчитанные при различных ветровых ситуациях концентрации загрязняющих веществ и планктона принимались к рассмотрению, если относительная погрешность не превышала 30%.

Заключение

На основе экспедиционных исследований выполнена первичная верификация модели экосистемы Азовского моря. Реализована задача моделирования и прогноза состояния водной экосистемы Азовского моря в условиях антропогенного воздействия и всестороннего изучения уникального водного объекта, который в силу своей мелководности в значительной степени подвергается антропогенному влиянию.

С помощью модели ЭВ (1) – (3) могут быть описаны процессы ассимиляции NH_4, нитратредукции (денитрификации), окисления H_2S, сульфатредукции, нитрификации,окисления и восстановления марганца аммонификации, а также можно изучать механизм условий формирования заморов, прогнозировать изменения кислородного и биогенного режимов в мелководном водоеме. При построении модели были параметризованы процессы биогеохимических циклов химических элементов, ответственных за трансформацию аэробных условий в анаэробные [2].

Созданный программный комплекс объединяет математические модели и базы экологических данных, с помощью разработанного комплекса были изучены условия, при которых мелководные водоемы подвергаются эвтрофитрованию.

Отличительными особенностями разрабатываемых алгоритмов, реализующих поставленные гидробиологические модельные задачи, являются: высокая производительность, достоверность и точность получаемых результатов. Высокая производительность достигается за счет использования эффективных численных методов решения сеточных уравнений, реализованных на параллельных вычислительных системах в реальном и ускоренном масштабах времени. Достоверность достигается за счет учета определяющих физических факторов, таких как: стоки рек, испарение, сила Кориолиса, сложная геометрия береговой линии и дна водоема, турбулентный обмен, трение о дно и ветровые напряжения, динамическое перестроение расчетной области, а также за счет учета

отклонения значения поля давления от гидростатического приближения. Точность достигается применением подробных расчетных сеток, учитывающих степень «заполненности» расчетных ячеек, а также отсутствием неконсервативных диссипативных слагаемых и нефизичных источников (стоков), возникающих в результате конечно-разностных аппроксимаций.

Было проведено сравнение работы созданного программного комплекса, реализующего разработанные сценарии развития экологической обстановки в Азовском море с использованием численной реализации модельной задачи эвтрофикации вод мелководного водоема, с подобными работами в области математического моделирования гидробиологических процессов.

Анализ показал, что в результате разработки программного комплекса удалось повысить точность прогнозов изменения концентраций загрязняющих веществ, фито- и зоопланктона в мелководном водоеме на 10 – 15% в зависимости от решаемой модельной задачи водной экологии.

Работа выполнена при частичной финансовой поддержке Задания № 2014/174 в рамках базовой части государственного задания Минобрнауки России, а также при частичной финансовой поддержке РФФИ по проектам № 15-01-08619, № 15-07-08626, № 15-07-08408.

Литература

1. Якушев Е.В., Сухинов А.И. и др. Комплексные океанологические исследования Азовского моря в 28-м рейсе научно-исследовательского судна «Акванавт» // Океанология. 2003. Т. 43, №1. С. 44–53.
2. Сухинов А.И., Никитина А.В., Чистяков А.Е. Моделирование сценария биологической реабилитации Азовского моря // Математическое моделирование. 2012. Т. 24, №9. С. 3–21.
3. Сухинов А.И., Чистяков А.Е., Алексеенко Е.В. Численная реализация трехмерной модели гидродинамики для мелководных водоемов на супервычислительной системе // Математическое моделирование. 2011. Т. 23, № 3. С. 3–21.
4. Сухинов А.И., Чистяков А.Е. Параллельная реализация трехмерной модели гидродинамики мелководных водоемов на супервычислительной системе // Вычислительные методы и программирование: Новые вычислительные технологии. 2012. Т. 13. С. 290–297.
5. Белоцерковский О.М. Турбулентность: новые подходы. М.: Наука, 2003. 286 с.
6. Самарский А.А. Теория разностных схем. М.: Наука, 1989. 616 с.
7. Самарский А.А., Николаев Е.С. Методы решения сеточных уравнений. М.: Наука, 1978. 592 с.
8. Коновалов А.Н. К теории попеременно-треугольного итерационного метода // Сибирский математический журнал. 2002. 43:3. С. 552–572.
9. Сухинов А.И., Чистяков А.Е. Адаптивный модифицированный попеременно-треугольный итерационный метод для решения сеточных уравнений с несамосопряженным оператором // Математическое моделирование. 2012. Т. 24, № 1. С. 3–20.

10. Чистяков А.Е. Теоретические оценки ускорения и эффективности параллельной реализации ПТМ скорейшего спуска // Известия ЮФУ. Технические науки. 2010. № 6(107). С. 237–249.
11. Чистяков А.Е., Хачунц Д.С., Никитина А.В., Проценко Е.А., Кузнецова И.Ю. Библиотека параллельных итерационных методов решателей СЛАУ для задачи конвекции-диффузии на основе декомпозиции по одному пространственному направлению // Современные проблемы науки и образования. 2015. № 1. URL: http://www.science-education.ru/121-19510 (дата обращения 4.06.2015).
12. Никитина А.В. Численное решение задачи динамики токсичных водорослей в Таганрогском заливе // Известия ЮФУ. Технические науки. 2010. № 6(107). С. 113–116.
13. Никитина А.В. Модели биологической кинетики, стабилизирующие экологическую систему таганрогского залива // Известия ЮФУ. Технические науки. 2009. № 8(97). С. 130–134.
14. Сухинов А.И., Чистяков А.Е., Бондаренко Ю.С. Оценка погрешности решения уравнения диффузии на основе схем с весами // Известия ЮФУ. Технические науки. 2011. № 8(121). С. 6–13.
15. Сухинов А.И., Чистяков А.Е., Семенякина А.А., Никитина А.В. Параллельная реализация задач транспорта веществ и восстановления донной поверхности на основе схем повышенного порядка точности // Вычислительные методы и программирование: новые вычислительные технологии. 2015. Т. 16. С. 256–267.
16. Никитина А.В., Руднева Т.В., Камышникова Т.В., Бокарева Т.А., Дурягина В.В. К вопросу о формировании заморных зон в восточной части Азовского моря // Современные проблемы науки и образования. 2015. № 1. URL: http://www.science-education.ru/121-19509 (дата обращения 4.06.2015).
17. Никитина А.В., Пучкин М.В., Семенов И.С., Сухинов А.И., Угольницкий Г.А., Усов А.Б., Чистяков А.Е. Дифференциально-игровая модель предотвращения заморов в мелководных водоемах // Управление большими системами. М.: ИПУ РАН. 2015. Вып. 55. С. 343–361.

DOI: 10.14529/cmse160303

MATHEMATICAL MODELING OF EUTROPHICATION PROCESSES IN SHALLOW WATERS ON MULTIPROCESSOR COMPUTER SYSTEM

© 2016 A.I. Sukhinov[1], A.V. Nikitina[2], A.E. Chistyakov[2], I.S. Semenov[3], A.A. Semenyakina[3], D.S. Khachunts[2]

[1]*Don State Technical University (Gagarina Sq. 1, Taganrog, 344000 Russia),*

[2]*Kalyaev Scientific Research Institute of Multiprocessor Computer Systems at Southern Federal University (Chekhova 2, Taganrog, 347928 Russia),*

[3]*Scientific Research Center of Supercomputers and Neurocomputers (per. Italyansky 106, Taganrog, 347900 Russia),*

E-mail: sukhin@gmail.com, nikitina.vm@gmail.com, cheese_05@mail.ru, flanker555@yandex.ru, j.a.s.s.y@mail.ru, diana-hachunts@mail.ru

Received: 25.02.2016

The paper covered the development of solution methods for model of eutrophication of shallow water in view the movement of the water flow, microturbulent diffusion, gravitational settling, spatially uneven distribution of temperature and salinity, and pollutants of nutrients, oxygen, phyto- and zooplankton, etc. Shallow waters of the Azov Sea and Taganrog Bay were selected as the object of simulation. Numerical solution of the problem is based on gradient method of variation type – the method of minimal corrections. The acceleration and efficiency of calculations is achieved with using multiprocessor computer system of Southern Federal University. The solution of the problem of water ecology will allow to predict changes of water quality of shallow reservoirs, and to study the mechanisms of formation of zones with low oxygen content.

Keywords: mathematical model, eutrophication, minimum bet, wok, parallel computing, Azov Sea.

FOR CITATION

Sukhinov A.I., Nikitina A.V., Chistyakov A.E., Semenov I.S., Semenyakina A.A., Khachunts D.S. Mathematical Modeling of Eutrophication Processes in Shallow Waters on Multiprocessor Computer System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 36–53. (in Russian) DOI: 10.14529/cmse160303.

References

1. Yakushev E.V., Sukhinov A.I., et al. Complex Oceanographic Research of the Sea of Azov in the 28-th Flight of the Research Vessel «Aquanaut». *Okeanologiya* [Oceanology]. 2003. vol. 43, no. 1. pp. 44–53. (in Russian)
2. Sukhinov A.I., Nikitina A.V., Chistyakov A.E. Simulation Scenario of Biological Rehabilitation of the Azov Sea. *Matematicheskoe modelirovanie* [Mathematical Modeling]. 2012. vol. 24, no. 9. pp. 3–21. (in Russian)
3. Sukhinov A.I., Chistyakov A.E., Alekseenko E.V. Numerical Implementation Three-Dimensional Model of Hydrodynamics for Shallow Water Basins on Superficialities System. *Matematicheskoe modelirovanie* [Mathematical Modeling]. 2011. vol. 23, no. 3. pp. 3–21. (in Russian)
4. Sukhinov A.I., Chistyakov A.E. Parallel Implementation of a Three-Dimensional Model of Hydrodynamics of Shallow Water Bodies on Superficialities System.

Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tekhnologii [Computational Methods and Programming: New Computing Technologies]. 2012. vol. 13. pp. 290–297. (in Russian)

5. Belotserkovsky O.M. *Turbulentnost': novye podhody* [Turbulence: New Approaches]. M.: Science, 2003. 286 p.
6. Samarskii A.A. *Teoriya raznostnyh skhem* [Theory of Difference Schemes]. M.: Science, 1989. 616 p.
7. Samarskii A.A., Nikolaev E.S. *Metody resheniya setochnyh uravneniy* [Methods of Solving Grid Equations]. M.: Science, 1978. 592 p.
8. Konovalov A.N. The Theory of the Alternating-Triangular Iterative Method. *Sibirskiy matematicheskiy zhurnal* [Siberian Mathematical Journal]. 2002. 43:3. pp. 552–572. (in Russian)
9. Sukhinov A.I., Chistyakov A.E. Adaptive Modified Alternating Triangular Iterative Method for Solving Grid Equations with Non-Selfadjoint Operator. *Mathematical Models and Computer Simulations.* 2012. vol. 4, no. 4. pp. 398–409. DOI: 10.1134/s2070048212040084.
10. Chistyakov A.E. Theoretical Estimation of Speed Up and Efficiency of Parallel Implementation of the Steepest Descent PTM. *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences]. 2010. no. 6(107). pp. 237–249. (in Russian)
11. Chistyakov A.E., Hachunts D.S., Nikitina A.V., Protsenko E.A., Kuznetsova I.Yu. A Library of Parallel Iterative Methods Solvers of Linear Algebraic Equation for the Problem of Convection-Diffusion-Based Decomposition in One Spatial Direction. *Sovremennye problemy nauki i obrazovaniya* [Modern Problems of Science and Education]. 2015. no. 1. Available at: http://www.science-education.ru /121-19510 (accessed: 4.06.2015).
12. Nikitina A.V. Numerical Solution of Problems of Dynamics of Toxic Algae in the Taganrog Bay. *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences]. 2010. no. 6(107). pp. 113–116. (in Russian)
13. Nikitina A.V. Modelling of Biological Kinetics, Stabilizing the Ecological System of the Gulf of Taganrog. *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences]. 2009. no. 8(97). pp. 130–134. (in Russian)
14. Sukhinov A.I., Chistyakov A.E., Bondarenko Yu.S. Error Estimate of the Solution of the Diffusion Equation on the Basis of the Schemes with Weights. *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences]. 2011. no. 8(121). pp. 6–13. (in Russian)
15. Sukhinov A.I., Chistyakov A.E., Semenikhina A.A., Nikitina A.V. Parallel Realization of the Tasks of the Transport of Substances and the Recovery of the Bottom Surface on the Basis of the Schemes of High Order of Accuracy. *Vychislitel'nye metody i programmirovanie: novye vychislitel'nye tekhnologii* [Computational Methods and Programming: New Computing Technologies]. 2015. vol. 16. pp. 256–267. (in Russian)

16. Nikitina A.V., Rudneva T.V., Kamyshnikova T.V., Bokareva T.A., Duryagina V.V. To the Formation of Kill Zones in the Eastern Part of Azov Sea Coast. *Sovremennye problemy nauki i obrazovaniya* [Modern Problems of Science and Education]. 2015. no. 1. Available at: http://www.science-education.ru/121-19509 (accessed: 4.06.2015).
17. Nikitina A.V., Puchkin M.V., Semenov I.S., Sukhinov A.I., Ougolnitsky G.A., Usov A.B., Chistyakov A.E. Differential-Game Model of Prevention of Fish Kill in Shallow Ponds. *Upravlenie bol'shimi sistemami* [Managing Large Systems]. M.: IPU Russian Academy of Sciences. 2015. issue 55. pp. 343–361. (in Russian)

УДК 519.1, 004.832.25 DOI: 10.14529/cmse160304

ПРИМЕНЕНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ ДЛЯ ПОИСКА ТРОЕК ВЗАИМНО ЧАСТИЧНО ОРТОГОНАЛЬНЫХ ДИАГОНАЛЬНЫХ ЛАТИНСКИХ КВАДРАТОВ ПОРЯДКА 10*

© 2016 г. О.С. Заикин[1], Э.И. Ватутин[2], А.Д. Журавлев[3], М.О. Манзюк[3]

[1]*Институт динамики систем и теории управления имени В.М. Матросова СО РАН (664033 Иркутск, ул. Лермонтова, д. 134)*

[2]*Юго-Западный государственный университет (305040 Курск, ул. 50 лет Октября, д. 94)*

[3]*Интернет-портал BOINC.ru (Москва)*

E-mail: zaikin.icc@gmail.com, evatutin@rambler.ru, alexone07@mail.ru, hoarfrost@rambler.ru

Поступила в редакцию: 23.05.2016

Статья посвящена поиску троек взаимно частично ортогональных диагональных латинских квадратов порядка 10. Для каждой известной пары ортогональных диагональных латинских квадратов порядка 10 достраивается третий диагональный латинский квадрат таким образом, чтобы условие ортогональности между ним и квадратами из рассматриваемой пары нарушалось в как можно меньшем количестве ячеек. Используются два подхода: первый основан на сведении исходной задачи к задаче о булевой выполнимости, а второй – на использовании метода грубой силы. Построено несколько троек указанного вида с рекордными характеристиками. Эксперименты были проведены в проекте добровольных распределенных вычислений SAT@home, а также на вычислительном кластере.

Ключевые слова: диагональные латинские квадраты, частичная ортогональность, задача о булевой выполнимости, добровольные распределенные вычисления, метод грубой силы, вычислительный кластер.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Заикин О.С., Ватутин Э.И., Журавлев А.Д., Манзюк М.О. Применение высокопроизводительных вычислений для поиска троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 54–68. DOI: 10.14529/cmse160304.

Введение

Латинский квадрат порядка n – это квадратная таблица размеров $n \times n$, заполненная элементами некоторого множества $M, |M| = n$ таким образом, что в каждой строке и в каждом столбце таблицы каждый элемент из M встречается в точности один раз [1]. В дальнейшем в настоящей статье в качестве M будет использовано множество $\{0, \ldots, n-1\}$. Латинские квадраты и системы, построенные на их основе, применяются в многих прикладных областях: теории кодирования (см., например, [2]), криптографии (см. [3]) и т.д. Диагональный латинский квадрат порядка n – это латинский квадрат, в котором каждый элемент из $M, |M| = n$ встречается в точности один раз не только в каждой строке и каждом столбце, но и в главной и побочной диагоналях.

* Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2016»

Пара латинских квадратов одинакового порядка называется ортогональной, если различны все упорядоченные пары элементов (a, b), где a – элемент в некоторой ячейке первого латинского квадрата, а b – элемент в ячейке с тем же номером второго латинского квадрата. Другими словами, два латинские квадрата порядка n ортогональны, если при наложении одного квадрата на другой образуется новый квадрат, в котором все n^2 элементов будут различны (такой квадрат называется греко-латинским). Если имеется набор из m различных латинских квадратов, любая пара которых ортогональна, то говорят о системе из m взаимно ортогональных латинских квадратов. Пара латинских квадратов порядка n называется частично ортогональной, если условие ортогональности между ними может выполняться не полностью, т.е. если при наложении одного латинского квадрата на другой среди n^2 элементов получаемого квадрата могут быть повторяющиеся элементы.

Диагональные латинские квадраты порядка 10 пока относительно слабо изучены, то же самое можно сказать и про ортогональные системы, построенные на их основе. Если первая пара ортогональных латинских квадратов порядка 10 была найдена еще в 1959 г. (см. [1]), то первая пара ортогональных диагональных латинских квадратов порядка 10 была найдена только в 1992 г. [4]. Более конкретно, в статье [4] были представлены три такие пары. Количество диагональных латинских квадратов порядка 10 до сих пор неизвестно, при этом количество латинских квадратов порядка 10 было установлено в 1995 г. (см. [1]).

Одной из самых известных открытых задач комбинаторики является следующая: определить, существует ли тройка взаимно ортогональных латинских квадратов порядка 10. Эта задача чрезвычайно сложна, поэтому в последнее время интенсивно развиваются алгоритмы поиска троек взаимно частично ортогональных латинских квадратов порядка 10. В статье [5] приведен текущий рекорд в этом направлении – тройка A, B, C латинских квадратов порядка 10, в которой A ортогонален B и C, а B и C образуют частично ортогональную пару с 91 различными упорядоченными парами элементов из 100 возможных. Под рекордом здесь понимается то, что данная тройка является наиболее близкой к тройке попарно ортогональных латинских квадратов порядка 10 (из опубликованных в открытой печати).

Данная статья посвящена поиску новых троек взаимно частично ортогональных диагональных латинских квадратов порядка 10. Для этого используется подход, согласно которому осуществляется поиск пар ортогональных диагональных латинских квадратов порядка 10, а затем для каждой такой пары достраивается третий диагональный латинский квадрат таким образом, чтобы получаемая тройка удовлетворяла заданным характеристикам. Большая часть экспериментов была проведена в рамках проекта добровольных распределенных вычислений SAT@home и на вычислительном кластере.

Статья имеет следующую структуру. Во втором разделе описаны новые результаты применения SAT-подхода к поиску систем диагональных латинских квадратов порядка 10. Описан эксперимент, в результате которого в проекте добровольных распределенных вычислений SAT@home были найдены новые пары ортогональных диагональных латинских квадратов порядка 10. Приведены результаты поиска троек взаимно частично ортогональных диагональных латинских квадратов порядка 10. Завершает второй раздел доказательство того факта, что на основе всех известных в настоящее время пар ортогональных диагональных латинских квадратов порядка 10 невозможно построить трой-

ку взаимно ортогональных диагональных латинских квадратов порядка 10. В третьем разделе описаны два переборных алгоритма построения диагональных латинских квадратов порядка 10, а также результаты применения этих алгоритмов к поиску троек взаимно частично ортогональных диагональных латинских квадратов порядка 10.

1. Поиск систем диагональных латинских квадратов порядка 10 как задача о булевой выполнимости

Задачи поиска комбинаторных структур можно решать с помощью различных подходов. В данном разделе развивается SAT-подход к решению таких задач, состоящий в сведении исходной задачи к проблеме булевой выполнимости (SAT) [6]. Для использования SAT-подхода необходимо перейти от исходной постановки к булевому уравнению вида «КНФ=1» (здесь КНФ – это конъюнктивная нормальная форма). Такой переход называется пропозициональным кодированием исходной задачи. Все известные алгоритмы решения SAT-задач экспоненциальны в худшем случае, т.к. SAT является NP-трудной задачей (NP-полной в распознавательном варианте). Несмотря на это, современные SAT-решатели (в том числе параллельные и распределенные) успешно справляются с решением многих трудных SAT-задач, кодирующих задачи из различных предметных областей. Большинство из современных SAT-решателей основано на алгоритме Conflict-Driven Clause Learning (CDCL). Базовые принципы этого алгоритма, а также ряд эвристик и структур данных, используемых для его ускорения, рассмотрены в обзорной статье [7]. Применение SAT-подхода к поиску различных систем ортогональных латинских квадратов описано в обзорной статье [8].

1.1. Поиск пар диагональных латинских квадратов порядка 10 как задача о булевой выполнимости

В 2012 г. в проекте добровольных распределенных вычислений SAT@home [9], построенном на платформе BOINC [10], был запущен эксперимент, направленный на поиск новых пар ортогональных диагональных латинских квадратов порядка 10. Результаты этого эксперимента описаны в [11]. Сначала была сделана пропозициональная кодировка указанной задачи. Полученная в результате КНФ состоит из 2000 переменных и 434440 дизъюнктов, файл с КНФ в формате DIMACS занимает около 10 мегабайт. Применялась т.н. «наивная» схема кодирования, при которой каждой ячейке каждого искомого квадрата сопоставляются 10 булевых переменных КНФ. Распараллеливание полученной SAT-задачи на подзадачи проводилось следующим образом. Первая строка первого диагонального латинского квадрата фиксировалось в значении «0 1 2 3 4 5 6 7 8 9», а значения первых 8 ячеек второй и третьей строки варьировались (т.е. перебирались все возможные их значения). Упомянутое фиксирование значения первой строки возможно потому, что любой латинский квадрат можно эффективно привести к такому виду. В итоге в каждой получаемой подзадаче в первом диагональном латинском квадрате из искомой пары были известны значения 26 из 100 ячеек (10 из первой строки и по 8 из второй и третьей строки), это обеспечивалось подстановкой значений 260 из 2000 переменных КНФ. В качестве задания проекта SAT@home выдавался пакет из 20 таких подзадач. На каждую подзадачу из пакета был установлен лимит в 2600 рестартов SAT-решателя Minisat [12] (модифицированный вариант этого решателя используется в качестве расчетного приложения в SAT@home), что примерно соответствует 4 минутам ра-

боты одного ядра современного процессора. На обработку 20 миллионов подзадач, сгенерированных для данного эксперимента, потребовалось около 9 месяцев работы SAT@home (с сентября 2012 года по май 2013 года). В результате были найдены 17 новых пар ортогональных диагональных латинских квадратов порядка 10 (в дополнение к трем ранее известным парам квадратов из статьи [4]).

В апреле 2015 г. в проекте SAT@home нами был запущен эксперимент, направленный на поиск новых пар ортогональных диагональных латинских квадратов порядка 10. Отметим, что разбиение исходной задачи на подзадачи, использованное в 2012-2013 гг. (см. [11]), было выбрано из разумных соображений. В этот раз было принято решение подойти к выбору типа разбиения более систематично. В табл. 1 указаны 8 разбиений и результаты, полученные с их помощью. Всего с 17 апреля 2015 г. по 12 февраля 2016 г. были найдены 32 новые пары ортогональных диагональных латинских квадратов порядка 10 (мы сравнивали их с 3 парами из статьи [4] и 17 парами из статьи [11]), все они выложены в разделе «Найденные решения» проекта SAT@home. При этом было использовано то же расчетное приложение с тем же лимитом на количество рестартов, что и в 2012 г. Во всех разбиениях использовались первые ячейки в строках (т.к. первая строка всегда фиксируется, строки брались начиная со второй).

Таблица 1

Результаты применения различных разбиений для поиска пар ортогональных диагональных латинских квадратов порядка 10 в проекте SAT@home

Вид разбиения	**Найдено пар**	**Дата**	**Статус**
1 строка, 9 ячеек	1	1 месяц в 2015 г.	Завершен
2 строки по 2 ячейки	–	1 день в 2015 г.	Завершен
2 строки по 3 ячейки	–	3 дня в 2015 г.	Завершен
2 строки по 4 ячейки	–	2 недели в 2015 г.	Завершен
2 строки по 5 ячеек	26	6 месяцев в 2015-2016 гг.	В процессе
2 строки по 6 ячеек	5	2 месяца в 2015 г.	Приостановлен
2 строки по 7 ячеек	–	–	Не запущен
2 строки по 8 ячеек	17	9 месяцев в 2012-2013 гг.	Приостановлен

Проанализируем табл. 1. Разбиение «2 строки по 8 ячеек» соответствует разбиению из [11], разбиение «2 строки по 7 ячеек» еще не запускалось, а остальные 6 разбиений были запущены в 2015 г. С помощью разбиений «2 строки по 2 ячейки», «2 строки по 3 ячейки» и «2 строки по 4 ячейки» не было найдено ни одной новой пары. Разбиение «2 строки по 5 ячеек» оказалось более эффективно, чем использованное в [11] разбиение «2 строки по 8 ячеек», т.к. обеспечило нахождение большего количества искомых пар за

единицу времени (даже с учетом того, что в 2015-2016 гг. производительность SAT@home примерно в два раза выше, чем в 2012-2013 гг.).

1.2. Поиск троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 как задача о булевой выполнимости

Рассмотрим тройку диагональных латинских квадратов одного и того же порядка. Характеристикой частично ортогональной тройки далее называется множество различных упорядоченных пар элементов, по которым выполняется условие ортогональности для каждой из трех пар квадратов из тройки.

В статье [11] был описан алгоритм построения троек взаимно частично ортогональных диагональных латинских квадратов порядка 10. Согласно этому алгоритму запускается генерация диагональных латинских квадратов порядка 10, и на основе каждого сгенерированного квадрата строятся частично ортогональные тройки с задействованием каждой из известных ортогональных пар (на основе каждой известной пары строится отдельная тройка). В результате применения этого алгоритма была найдена частично ортогональная тройка с характеристикой 62, тогда как в статье [4] приведена частично ортогональная тройка с характеристикой 60.

В статье [13] предлагается рассматривать задачу построения таких троек как задачу о булевой выполнимости. Конкретнее, была предложена пропозициональная кодировка поиска таких троек, в которой редактированием специального дизъюнкта можно задавать требуемое значение характеристики ортогональности. Если в полученную КНФ подставить значения двух квадратов из некоторой известной пары, то задача сводится к поиску диагонального латинского квадрата, который с подставленной парой квадратов образует частично ортогональную тройку с заданной характеристикой. С помощью данного подхода была найдена тройка взаимно частично ортогональных диагональных латинских квадратов порядка 10 с характеристикой 73 [13]. Для этого был использован многопоточный SAT-решатель treengeling [14], который запускался на одном узле вычислительного кластера и задействовал на нем 32 процессорных ядра. Для каждой известной пары ортогональных диагональных латинских квадратов (на тот момент таковых было 20) строилась отдельная КНФ, подставляя значения двух известных квадратов в описанную выше кодировку.

Т.к. в 2015-2016 гг. в SAT@home были найдены еще 32 пары ортогональных диагональных латинских квадратов порядка 10 (см. раздел 2.1), то на их основе было сделано еще 32 КНФ в дополнение к 20, рассмотренным в [13]. В результате запуска на них SAT-решателя treengeling были найдены еще две частично ортогональные тройки с характеристикой 73. Они были построены на основе 4-й и 15-й пар, найденных в рамках эксперимента, описанного в разделе 2.1. Соответствующие пары в разделе «Найденные решения» сайта проекта SAT@home отмечены как найденные 25 июня и 23 августа 2015 г. Обе найденные тройки выложены на сайте SAT@home в разделе «Ортогональные и частично ортогональные системы латинских квадратов порядка 10».

Предложенная в [13] пропозициональная кодировка позволила получить еще одно семейство результатов. С помощью данной кодировки можно установить требование, чтобы характеристика частично ортогональной тройки порядка 10 была равна 100, т.е. фактически тем самым потребовать нахождение тройки попарно ортогональных диаго-

нальных латинских квадратов порядка 10 без каких-либо ослаблений. При этом подстановка в соответствующую КНФ значений квадратов из известной пары ставит задачу следующим образом: для заданной пары ортогональных диагональных латинских квадратов порядка 10 найти третий диагональный латинский квадрат, образующий с первыми двумя квадратами взаимно ортогональную тройку, либо констатировать факт отсутствия такого диагонального латинского квадрата. Были построены 52 КНФ с подстановкой в каждую из них значений квадратов из соответствующих пар (3 пары из [4], 17 из [11], а еще 32 были найдены в рамках данного исследования, см. раздел 2.1). В каждую из этих КНФ было внесено требование «значение характеристики равно 100». SAT-задачи для всех этих КНФ были решены SAT-решателем plingeling [14] в среднем примерно за одну секунду на 32 процессорных ядрах (были использованы 2 16-ядерных процессора AMD Opteron 6276), и все ответы были «UNSAT». Это означает, что решатель plingeling сделал вывод о невыполнимости всех этих КНФ. Напомним, что КНФ является невыполнимой, если отсутствует такой набор значений переменных из этой КНФ, который бы обращал ее в 1. Отметим, что в [15] была формально доказана корректность алгоритма DPLL (Davis–Putnam–Logemann–Loveland) – т.е. было доказано, что если алгоритму DPLL удается найти решение SAT-задачи, то это решение является правильным. В том числе это касается и случая, когда алгоритм DPLL делает заключение о невыполнимости данной на вход КНФ. В решателе plingeling, основанном на алгоритме DPLL, добавлен ряд усовершенствований (например, добавлена процедура CDCL), которые, тем не менее, не нарушают корректность базового алгоритма DPLL [14]. Из вышеупомянутых фактов следует, что все построенные КНФ оказались действительно невыполнимыми, т.к это было доказано используемым корректным алгоритмом. А из этого факта в свою очередь следует, что на основе всех 52 известных на данный момент пар ортогональных диагональных латинских квадратов порядка 10 невозможно построить тройку взаимно ортогональных диагональных латинских квадратов порядка 10. Дополнительно, на данный момент для всех 52 известных ортогональных пар описанным выше способом удалось провести доказательство невозможности построения взаимно частично ортогональных троек для случая «значение характеристики равно 87». Эти задачи оказались уже довольно сложными – решателю plingeling понадобилось в среднем 8 часов работы для каждой такой КНФ, при этом он работал на 32 ядрах.

2. Применение метода грубой силы для построения диагональных латинских квадратов порядка 10

В статье [11] для построения троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 использовался алгоритм поиска с возвратом, предназначенный для генерации диагональных латинских квадратов порядка 10 (см. раздел 2.2). При этом заполнение происходит слева направо сверху вниз по ячейкам квадрата. Поиск завершается, если найден некоторый диагональный латинский квадрат. После подстановки каждого нового значения в некоторую ячейку осуществляется проверка, нарушает ли текущее заполнение таблицы условия, накладываемые на элементы в диагональном латинском квадрате. Если заполнение не прошло проверку, происходит возврат до ближайшей ячейки, элемент которой нарушает условия, после чего этой ячейке назначается другое допустимое значение. История неудачных использованных значений

хранится для каждой ячейки квадрата. Это нужно для того, чтобы исключить повтор вариантов, которые приводят к недопустимым заполнениям. Если для некоторой ячейки требуется поменять значение, но исчерпано множество допустимых вариантов, то происходит возврат на предыдущую ячейку. Реализация данного алгоритма позволила получать примерно по одному диагональному латинскому квадрату в секунду на одном ядре процессора.

Далее описываются два переборных алгоритма генерации диагональных латинских квадратов порядка 10. Они были разработаны и реализованы, во-первых, для того, чтобы сравнить их по скорости генерации с упомянутым выше алгоритмом поиска с возвратом. Во-вторых, с помощью данных алгоритмов планировалось найти новые тройки частично ортогональных диагональных латинских квадратов.

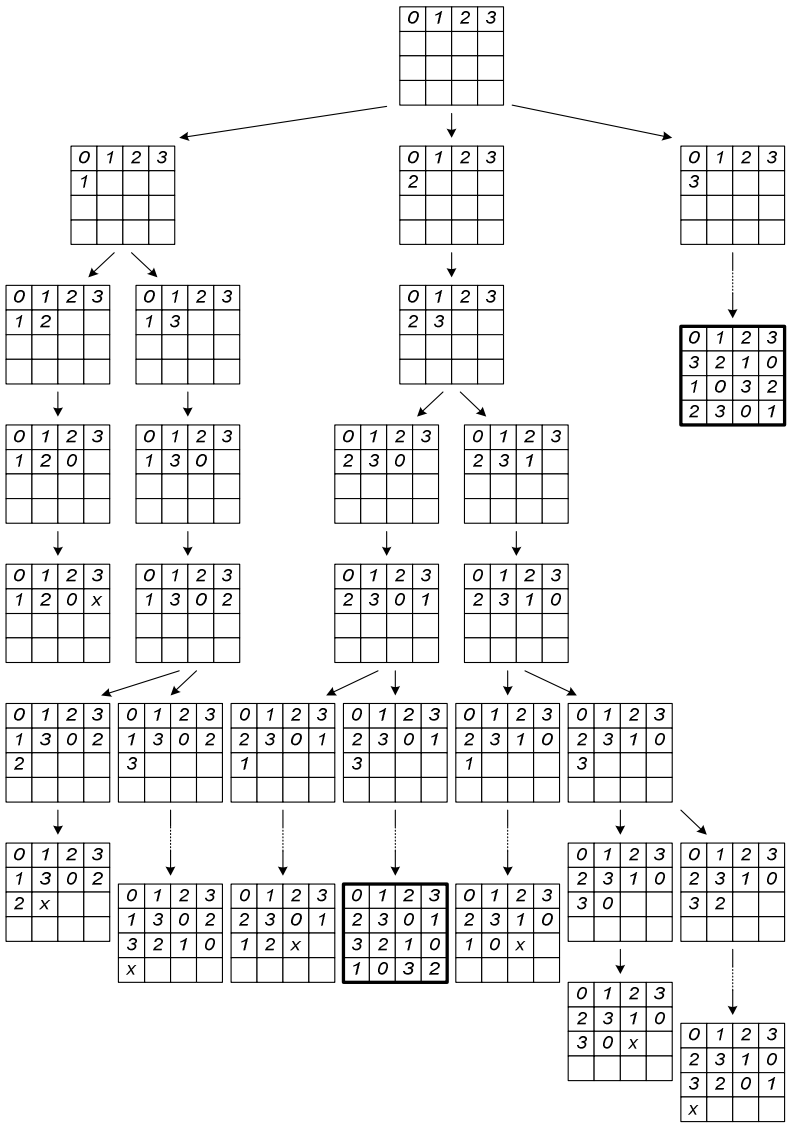

Рис. 1. Пример формирования диагонального латинского квадрата 4-го порядка: крестами отмечены элементы, для которых $\mathbf{S_{ij}} = \emptyset$; жирным выделены найденные решения

Опишем первый переборный алгоритм. Простейшим способом реализации метода грубой силы является последовательное заполнение элементов формируемой структуры

данных (например, латинского квадрата), значениями в соответствии с известным порядком обхода формируемого комбинаторного дерева в глубину. При этом после осуществления комбинаторного спуска для каждого нового элемента формируемой структуры данных (в простейшем случае – элемента a_{ij} латинского квадрата), следуя работе [16], производится построение множества S_{ij} допустимых элементов, которые не нарушают ограничений задачи (например, не встречались в -й строке и j-м столбце). Каждому элементу указанного множества соответствует поддерево в дереве комбинаторного перебора, для них поочередно производится комбинаторный спуск, и процесс заполнения элементов формируемой структуры данных рекуррентно продолжается. Если на каком-либо шаге доступных элементов нет (т.е. $S_{ij} = \emptyset$), то производится рекуррентный возврат на один ярус дерева комбинаторного перебора вверх, что соответствует методу ветвей и границ [17] и приводит к снижению числа анализируемых узлов дерева и, соответственно, к снижению затрат машинного времени при программной реализации. Схематично процесс комбинаторного перебора при построении диагонального латинского квадрата порядка 4 изображен на рис. 1.

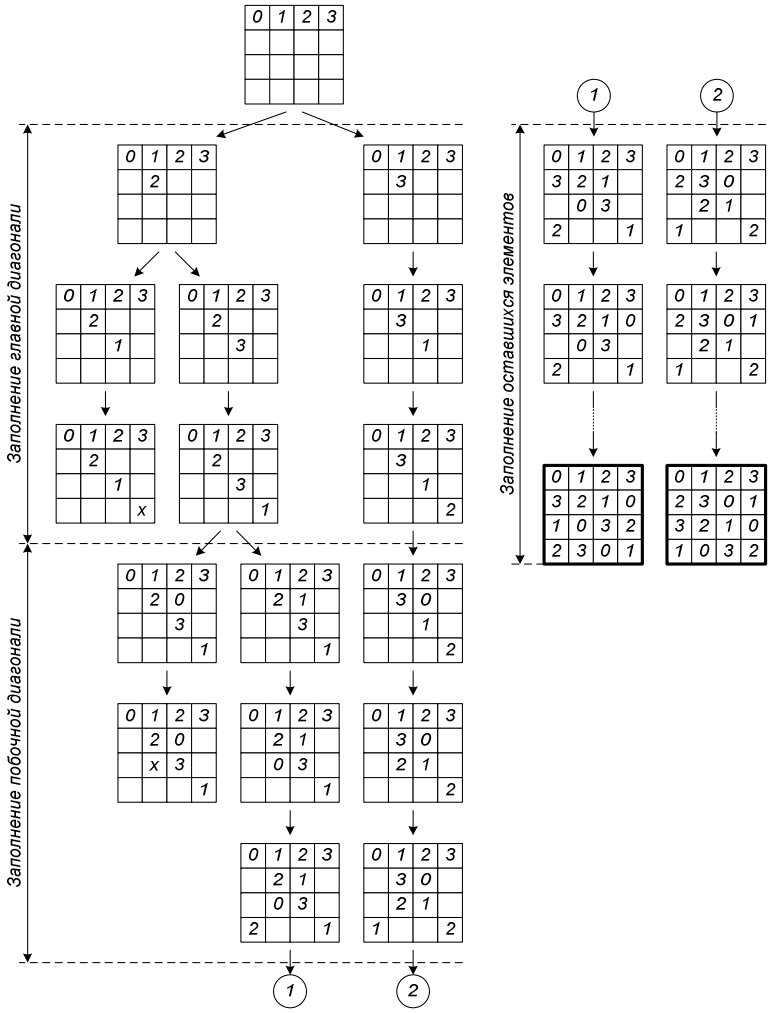

Рис. 2. Дерево комбинаторного перебора, соответствующее диагональному заполнению элементов (по сравнению с деревом, изображенным на рис. 1, число ветвей значительно меньше)

Несложно заметить, что некоторым поддеревьям не соответствует ни одного корректного решения (например, при построении нормализованного диагонального латинского квадрата 4 порядка с $a_{21} = 1$), однако данный факт устанавливается только после перебора всех узлов соответствующего поддерева, на что фактически впустую расходуется машинное время. При формировании диагональных латинских квадратов вероятность появления данной ситуации можно постараться снизить путем изменения порядка заполнения элементов формируемого квадрата (в работах [18, 19] показано, что изменение порядка рассмотрения элементов при формировании решения может оказывать существенное влияние на качество результирующего решения). Можно заметить, что максимальное число ограничений присутствует у элементов, расположенных на главной и побочной диагоналях квадрата (уникальность в строке, в столбце и на соответствующей диагонали), в то время как остальные элементы имеют меньшее число ограничений (уникальность в строке и в столбце). Исходя из данной особенности, заполнение элементов формируемого квадрата с использованием метода полного перебора можно производить в следующем порядке: сперва заполняются элементы главной диагонали, затем производится заполнение элементов побочной диагонали, а уже затем – всех остальных. При этом достигается более раннее возникновение возможных нарушений, что позволяет производить раннее отсечение неперспективных решений без анализа соответствующих поддеревьев (см. рис. 2).

Второй алгоритм построен на работе с т.н. версиями строк квадрата. Версия строки представляет собой номер перестановки её элементов. Например, для строки квадрата 4-го порядка возможные перестановки приведены в табл. 2.

Таблица 2

Возможные перестановки из 4 элементов

№ версии	**Вид строки**
0	0 1 2 3
1	0 1 3 4
...	...
22	3 2 0 1
23	3 2 1 0

Перед работой алгоритма производится построение словаря, включающего в своем составе все возможные перестановки. Наличие подобного словаря не является обязательным, т.к. перестановки можно генерировать и на лету, однако их однократное создание и последующее многократное использование делает поиск быстрее.

В процессе поиска алгоритм оперирует двумя массивами – массивом с данными латинского квадрата и массивом с перечнем версий строк данного квадрата. Пример массива для диагонального латинского квадрата 6-го порядка приведен в табл. 3.

В соответствии с требованием нормализации построение квадрата начинается с первой строки, версии которой задается значение 0, после этого алгоритм производит рекуррентные спуски на следующие строки, подбирая их так, чтобы в уже сформированной части квадрата выполнялись условия несовпадения значений внутри столбцов и диагоналей. При этом уникальность элементов строки автоматически следует из определения перестановки.

Таблица 3

Пример кодирования диагонального латинского квадрата
с использованием номеров версий (перестановок)

Версии строк	**Квадрат**
0	0 1 2 3 4 5
148	1 2 0 5 3 4
719	5 4 3 2 1 0
466	3 5 1 4 0 2
253	2 0 4 1 5 3
571	4 3 5 0 2 1

Описанные выше алгоритмы были реализованы в виде последовательной программы на языке C++. При этом первый из алгоритмов был реализован в двух вариантах очередности обхода ячеек квадрата, описанных выше. Тестирование этих алгоритмов проводилось на ПК с процессором AMD Phenom II X965. Тестирование проводилось в течении 1 часа. С помощью второго алгоритма (оперирующего со строками) не удалось найти ни одного диагонального латинского квадрата порядка 10. Первый алгоритм с вариантом очередности обхода ячеек подряд (слева направо сверху вниз) обеспечил генерацию 422 диагональных латинских квадратов в секунду, а во втором варианте (с первоочередным обходом ячеек главной и побочной диагоналей) – 5120 диагональных латинских квадратов в секунду. Отметим, что алгоритм поиска с возвратом из статьи [11] обеспечивал генерацию примерно 1 диагонального латинского квадрата в секунду на похожей конфигурации ПК.

$$A = \begin{bmatrix} 0&1&2&3&4&5&6&7&8&9\\ 1&2&0&4&5&7&9&8&3&6\\ 4&8&1&5&9&6&2&0&7&3\\ 2&0&6&9&8&4&7&3&5&1\\ 8&3&4&6&7&1&5&9&2&0\\ 7&5&9&1&6&3&0&2&4&8\\ 3&6&5&2&1&9&8&4&0&7\\ 6&9&8&7&3&0&4&5&1&2\\ 9&4&7&8&0&2&3&1&6&5\\ 5&7&3&0&2&8&1&6&9&4 \end{bmatrix} B = \begin{bmatrix} 0&1&2&3&4&5&6&7&8&9\\ 6&8&3&1&7&9&5&4&0&2\\ 2&3&9&0&8&7&4&1&6&5\\ 5&7&0&6&1&3&8&2&9&4\\ 7&9&8&4&5&2&1&0&3&6\\ 1&2&7&8&3&4&9&6&5&0\\ 8&5&6&7&0&1&2&9&4&3\\ 9&4&5&2&6&8&0&3&7&1\\ 3&6&4&9&2&0&7&5&1&8\\ 4&0&1&5&9&6&3&8&2&7 \end{bmatrix} C = \begin{bmatrix} 0&1&2&3&4&5&6&7&8&9\\ 7&4&5&2&1&0&3&9&6&8\\ 1&0&3&4&2&6&5&8&9&7\\ 2&3&0&1&8&9&4&5&7&6\\ 5&8&6&7&9&2&0&3&1&4\\ 6&9&4&5&7&8&1&0&3&2\\ 9&6&8&0&5&3&7&2&4&1\\ 4&2&1&9&0&7&8&6&5&3\\ 8&5&7&6&3&1&9&4&2&0\\ 3&7&9&8&6&4&2&1&0&5 \end{bmatrix}$$

Рис. 3. Тройка взаимно частично ортогональных диагональных латинских квадратов порядка 10 с характеристикой 66

На основе первого алгоритма была сделана MPI-программа на языке C++. Был использован вариант обхода ячеек подряд слева направо сверху вниз. В этой программе исходная задача разбивалась на семейство подзадач путем варьирования первых пяти ячеек второй и третьей строки квадрата (по аналогии с алгоритмом разбиения задачи из раздела 2.1). В рамках этой программы генерация диагональных латинских квадратов была использована для построения троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 таким же образом, как было сделано в [11] (см. раздел 2.2). MPI-программа была запущена на 96 часов на 10 узлах вычислительного кластера «Академик В.М. Матросов» Иркутского суперкомпьютерного центра СО РАН. Каждый узел этого кластера включает в себя 2 16-ядерных процессора AMD Opteron

6276, т.е. суммарно программа работала на 320 ядрах. В результате была найдена частично ортогональная тройка с характеристикой 66 (см. рис. 3).

Отметим, что в [11] подобным подходом была найдена частично ортогональная тройка с характеристикой 62, при этом был задействован примерно такой же объем вычислительных ресурсов. Данный факт объясняется тем, что, как было упомянуто выше, предложенный переборный алгоритм даже в своем более медленном варианте обхода ячеек обеспечивает значительно более высокую скорость генерации, чем алгоритм поиска с возвратом.

Исходя из того, что в разделе 2.2 приведены частично ортогональные тройки с характеристикой 73, можно сделать вывод, что SAT-подход конкретно для этой задачи подходит лучше, чем переборные алгоритмы, предложенные в данном разделе (и тем более лучше, чем алгоритм поиска с возвратом). Тем не менее, дальнейшее усовершенствование предложенных алгоритмов, а также их реализация на GPU (переборные алгоритмы хорошо подходят для GPU, в отличие от CDCL-алгоритмов, лежащих в основе большинства SAT-решателей), может привести к более хорошим результатам. Кроме всего прочего, в ближайшее время мы планируем встроить в MPI-программу переборный алгоритм с первоочередным обходом ячеек из главной и побочной диагоналей, что также может привести к нахождению частично ортогональных троек с более высокими характеристиками.

Заключение

В статье описаны алгоритмы, с помощью которых были найдены новые комбинаторные объекты: 32 пары ортогональных диагональных латинских квадратов порядка 10 (в дополнение к 20 таким парам, известным ранее) и две тройки взаимно частично ортогональных диагональных латинских квадратов порядка 10 с характеристикой 73 (в дополнение к одной такой тройке, известной ранее). Принципиально новым результатом стало доказательство того факта, что на основе всех 52 известных на данный момент пар ортогональных диагональных латинских квадратов порядка 10 (дополняя каждую пару произвольным третьим диагональным латинским квадратом) невозможно построить тройку взаимно ортогональных диагональных латинских квадратов порядка 10. Также были протестированы переборные алгоритмы генерации диагональных латинских квадратов. Результаты показывает, что у этих алгоритмов есть большой потенциал, в том числе и в направлении их реализации на GPU. При получении всех перечисленных результатов были использованы высокопроизводительные вычисления.

Работа была частично поддержана РФФИ (гранты № 14-07-00403-а, 15-07-07891-а и 16-07-00155-а) и советом по грантам Президента РФ (стипендия № СП-1184.2015.5, грант МК-9445.2016.8). Работа выполнена в рамках государственного задания для Юго-Западного государственного университета на 2014–2017 гг., № НИР 2246.

Литература

1. Colbourn C.J., Dinitz J.H. Handbook of Combinatorial Designs. Second Edition. Chapman&Hall, 2006. 984 p.
2. Малых А.Е., Данилова В.И. Об историческом процессе развития теории латинских квадратов и некоторых их приложениях // Вестник Пермского университета. Серия: Математика. Механика. Информатика. 2010. № 4. С. 95–104.

3. Тужилин М.Э. Об истории исследований латинских квадратов // Обозрение прикладной и промышленной математики. 2012. Т. 19, Вып. 2. С. 226–227.
4. Brown J.W., Cherry F., Most L., Most M., Parker E.T., Wallis W.D. Completion of the spectrum of orthogonal diagonal Latin squares // Lecture notes in pure and applied mathematics. 1992. Vol. 139. P. 43–49.
5. Egan J.E., Wanless I.M. Enumeration of MOLS of small order // Mathematics of Computation. 2015. Vol. 85. P. 799–824.
6. Biere A., Heule V., van Maaren H., Walsh T. (eds.) Handbook of Satisfiability. IOS Press. 2009.
7. Семенов А.А., Беспалов Д.В. Технологии решения многомерных задач логического поиска // Вестник Томского государственного университета. 2005. № 14. С. 61–73.
8. Zhang H. Combinatorial Designs by SAT Solvers. In: Biere A., Heule V., van Maaren H., Walsh T. (eds.) Handbook of Satisfiability. IOS Press. 2009. P. 533–568.
9. Заикин О.С., Семенов А.А., Посыпкин М.А. Процедуры построения декомпозиционных множеств для распределенного решения SAT-задач в проекте добровольных вычислений SAT@home // Управление большими системами. М.: ИПУ РАН. Вып. 43. 2013. С. 138–156.
10. Заикин О.С., Посыпкин М.А., Семенов А.А., Храпов Н.П. Опыт организации добровольных вычислений на примере проектов OPTIMA@home и SAT@home // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 5-2. С. 340–347.
11. Заикин О.С., Кочемазов С.Е. Поиск пар ортогональных диагональных латинских квадратов порядка 10 в проекте добровольных распределенных вычислений SAT@home // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2015. Т.4, № 3. С. 95–108.
12. Een N., Sorensson N. An Extensible SAT-solver // Lecture Notes in Computer Science. 2003. Vol. 2919. P. 502–518.
13. Zaikin O., Kochemazov S. The search for systems of diagonal Latin squares using the SAT@home project // Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015), Petrozavodsk, Russia, 2015. CEUR-WS. Vol. 1502. P. 52–63.
14. Biere A. Lingeling, Plingeling and Treengeling Entering the SAT Competition 2013 // Proceedings of SAT Competition 2013. 2013. Vol. B-2013-1. P. 51–52.
15. Maric F., Janicic P. Formal Correctness Proof for DPLL Procedure // Informatica. 2010. Vol. 21, Issue 1. P. 57–78.
16. Ватутин Э.И., Журавлев А.Д., Заикин О.С., Титов В.С. Особенности использования взвешивающих эвристик в задаче поиска диагональных латинских квадратов // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2015. № 3(16). С. 18–30.
17. Land A.H., Doig A.G. An automatic method of solving discrete programming problems // Econometrica. 1960. Vol. 28, No. 3. P. 497–520.
18. Ватутин Э.И., Романченко А.С., Титов В.С. Исследование влияния порядка рассмотрения пар на качество расписаний при использовании жадного подхода // Известия Юго-Западного государственного университета. 2013. № 1(46). С. 58–64.

19. Ватутин Э.И., Бобынцев Д.О., Романченко А.С. Исследование влияния частичного упорядочивания пар и локального улучшения окрестности пары на качество расписаний при использовании жадного подхода // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2014. № 1. С. 8–16.

DOI: 10.14529/cmse160304

APPLYING HIGH-PERFORMANCE COMPUTING TO SEARCHING FOR TRIPLES OF PARTIALLY ORTHOGONAL LATIN SQUARES OF ORDER 10

© 2016 O.S. Zaikin[1], E.I. Vatutin[2], A.D. Zhuravlev[3], M.O. Manzyuk[3]

[1]*Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences (Lermontova 134, Irkutsk, 664033 Russia)*

[2]*Southwest State University (50 Let Oktyabrya 94, Kursk, 305040 Russia)*

[3]*Internet-portal BOINC.ru (Moscow, Russia)*

E-mail: zaikin.icc@gmail.com, evatutin@rambler.ru, alexone07@mail.ru, hoarfrost@rambler.ru

Received: 23.05.2016

This paper deals with the search for triples of partially orthogonal Latin squares of order 10. For every known pair of orthogonal Latin squares of order 10 we add a third diagonal Latin square in such a way that the orthogonality condition between it and squares from a considered pair coincides in the maximum possible number of cells. Two approaches are used: the first one is based on reducing an original problem to Boolean satisfiability problem; the second one is based on Brute force method. Several triples of the aforementioned kind with high characteristics were constructed. The experiments were held in the volunteer computing project SAT@home and on a computing cluster.

Keywords: diagonal Latin squares, partial orthogonality, Boolean satisfiability problem, volunteer computing, brute force, computing cluster.

FOR CITATION

Zaikin O.S., Vatutin E.I., Zhuravlev A.D., Manzyuk M.O. Applying High-Performance Computing to Searching for Triples of Partially Orthogonal Latin Squares of Order 10. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 54–68. (in Russian) DOI: 10.14529/cmse160304.

References

1. Colbourn C.J., Dinitz J.H. *Handbook of Combinatorial Designs. Second Edition.* Chapman&Hall, 2006. 984 p. DOI: 10.1201/9781420010541.
2. Malih A.E., Danilova V.I. About Historical Process of the Evolution of Latin squares and Some Their Applications. *Vestnik Permskogo universiteta. Seria: Matematika, Mehanika, Informatika* [Bulletin of Perm University: Mathematics, Mechanics, Computer Science]. 2010. no. 4. pp. 95–104. (in Russian)

3. Tuzhilin M.E. On the History of the Study of Latin Squares. *Obozreniye prikladnoy i promyshlennoy matematiki* [Review of Applied and Industrial Mathematics]. 2012. vol. 19, issue 2. pp. 226–227. (in Russian)
4. Brown J.W., Cherry F., Most L., Most M., Parker E.T., Wallis W.D. Completion of the Spectrum of Orthogonal Diagonal Latin Squares. *Lecture Notes in Pure and Applied Mathematics.* 1992. vol. 139. pp. 43–49.
5. Egan J.E., Wanless I.M. Enumeration of MOLS of Small Order. *Mathematics of Computation.* 2015. vol. 85. pp. 799–824. DOI: 10.1090/mcom/3010.
6. Biere A., Heule V., van Maaren H., Walsh T. (eds.). *Handbook of Satisfiability.* IOS Press, 2009. 980 p.
7. Semenov A.A., Bespalov D.V. Technology for Solving Multidimensional Problems of the Logical Search. *Vestnik Tomskogo gosudarstvennogo universiteta* [Bulletin of Tomsk State University]. 2005. no. 14. pp. 61–73. (in Russian)
8. Zhang H. Combinatorial Designs by SAT Solvers. In: Biere A., Heule V., van Maaren H., Walsh T. (eds.) *Handbook of Satisfiability.* IOS Press, 2009. P. 533–568.
9. Zaikin O.S., Semenov A.A., Posypkin M.A. Constructing Decomposition Sets for Distributed Solution of SAT problems in Volunteer Computing Project SAT@home. *Upravlenie bol'shimi sistemami* [Large-Scale Systems Control]. 2013. issue 43. pp. 138–156. (in Russian)
10. Zaikin O.S., Posypkin M.A., Semenov A.A., Khrapov N.P. The Experience of Organizing Volunteer Computing Projects on the Examples of OPTIMA@home and SAT@home Projects. *Vestnik Nizhegorodskogo universiteta imeni N.I. Lobachevskogo* [Bulletin of the Lobachevsky University of Nizhni Novgorod]. 2012. no. 5-2. pp. 340–347. (in Russian)
11. Zaikin O.S., Kochemazov S.E. The Search for Pairs of Orthogonal Diagonal Latin Squares of Order 10 in the Volunteer Computing Project SAT@home. *Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika* [Bulletin of the South Ural State University: Series "Computational Mathematics and Software Engineering"]. 2015. vol. 4, no. 3. pp. 95–108. (in Russian) DOI: 10.14529/cmse150308.
12. Een N., Sorensson N. An Extensible SAT-solver. *Lecture Notes in Computer Science.* 2003. vol. 2919. pp. 502–518. DOI: 10.1007/978-3-540-24605-3_37.
13. Zaikin O., Kochemazov S. The Search for Systems of Diagonal Latin Squares Using the SAT@home Project. *Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015), Petrozavodsk, Russia, 2015.* CEUR-WS. vol. 1502. pp. 52–63.
14. Biere A. Lingeling, Plingeling and Treengeling Entering the SAT Competition 2013. *Proceedings of SAT Competition 2013.* 2013. vol. B-2013-1. pp. 51–52.
15. Maric F., Janicic P. Formal Correctness Proof for DPLL Procedure. *Informatica.* 2010. vol. 21, issue 1. pp. 57–78.
16. Vatutin E.I., Zhuravlev A.D., Zaikin O.S., Titov V.S. Features of the Use of Weighting Heuristics in the Search for Diagonal Latin Squares. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta. Seriya: "Upravlenie, vychislitel'naya tekhnika, informatika. Meditsinskoe priborostroenie"* [Proceedings of the South-West State University. Series "Control, Computer Engineering, Information Science. Medical Instruments Engineering"]. 2015. no. 3(16). pp. 18–30. (in Russian)

17. Land A.H., Doig A.G. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*. 1960. vol. 28, no. 3. pp. 497–520. DOI: 10.2307/1910129.
18. Vatutin E.I., Romanchenko A.S., Titov V.S. Investigation of the Effect of Pairs Consideration Order on the Quality of Scheduling Using the Greedly Approach. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta* [Proceedings of South-West State University]. 2013. no. 1(46). pp. 58–64. (in Russian)
19. Vatutin E.I., Bobyntcev D.O., Romanchenko A.S. Investigation of the Effect of Partial Ordering of Pairs and the Local Improvement of Pair Neighborhood on Schedule Quality Using the Greedy Approach. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta. Seriya: "Upravlenie, vychislitel'naya tekhnika, informatika. Meditsinskoe priborostroenie"* [Proceedings of South-West State University. Series "Control, Computer Engineering, Information Science. Medical Instruments Engineering"]. 2014. no. 1. pp. 8–16. (in Russian)

УДК 519.6 DOI: 10.14529/cmse160305

ВЫЧИСЛЕНИЕ ОБЛАСТЕЙ УСТОЙЧИВОСТИ ДИСКРЕТНЫХ МОДЕЛЕЙ БОЛЬШИХ НЕЙРОННЫХ СЕТЕЙ ТИПА SMALL WORLD

© 2016 г. С.А. Иванов[1], М.М. Кипнис[2]

[1]*Южно-Уральский государственный университет*
(454080 Челябинск, пр. им. В.И. Ленина, д. 76),
[2]*Южно-Уральский государственный гуманитарно-педагогический университет*
(454080 Челябинск, пр. им. В.И. Ленина, д. 69)
E-mail: saivanov@susu.ru, mmkipnis@gmail.com

Поступила в редакцию: 11.07.2016

Представлено описание дискретных моделей нейронных сетей типа small world с большим числом нейронов с некоторым параметром p, изменяющимся от 0 до 1. При $p = 0$ имеем модель, регулярной нейронной сети, представляющей собой кольцевую сеть, в которой каждый нейрон взаимодействует с несколькими соседями по кольцу. В случае $p = 1$ имеем модель со случайно расположенными связями. При значениях p, не превосходящих $0,1$, имеем сеть типа small world Ваттса—Строгаца. Подобные нейронные сети могут служить моделями различных нейронных структур в живых организмах, например, гипокамп мозга млекопитающих. Работа посвящена исследованию динамики изменения областей устойчивости таких нейронных сетей при $0 \le p \le 0,1$. Численные эксперименты показывают увеличение области устойчивости при переходе от регулярной сети к сети small world.

Ключевые слова: дискретные модели Ваттса—Строгаца, small world, устойчивость.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Иванов С.А., Кипнис М.М. Вычисление областей устойчивости дискретных моделей больших нейронных сетей типа small world // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 69–75. DOI: 10.14529/cmse160305.

Введение

Искусственная нейронная сеть — математическая модель, построенная по принципу организации и функционирования сетей нервных клеток живых организмов. В качестве нейронной сети можно рассматривать модель [1], которая содержит параметр p, отвечающий за вероятность перенаправления связи между нейронами. При $p = 0$ модель является регулярной, при $p = 1$ она представляет собой полностью случайный граф. В промежутке небольших значений p, близких к нулю, модель Ваттса—Строгаца представляет связи типа small world. Некоторые нейронные структуры внутри живых организмов организованы по типу small world [2, 3]. Моделирование отделов головного мозга с помощью нейронных сетей типа small world имеет применение в нейрохирургии [4]. Представленная в работе модель отличается от моделей [2, 3]. В [2] сила взаимодействия любых двух нейронов одинакова, поэтому анализ устойчивости основан только на анализе графа связей в сети. Такой подход не позволяет дать ответ об устойчивости сетей с несимметричными взаимодействиями между нейронами в сети. Автор статьи [3] исследует поведение нейронной сети, силы взаимодействия которой случайны и имеют Гауссово распределение. Кроме того, в обеих моделях [2, 3] отсутствует запаздывание.

Целью данной работы будет является изучение динамики изменения областей устойчивости больших нейронных сетей со структурой связей гипокампа человеческого мозга [5] при изменении параметра p. Такая модель относится к типу small world. Исследуемая модель нейронной сети имеет несимметричные взаимодействия между нейронами: сила действия одного нейрона на другой равна a, и не обязательно равна силе обратного воздействия, равного b. В исследуемой модели оказалось, что при $ab < 0$ области устойчивости как увеличиваются, так и уменьшаются в процессе роста параметра p. Численные эксперименты показали улучшение устойчивости нейронной сети при переходе от регулярной сети к сети типа small world.

Структура статьи такова. В разделе 1 дано описание модели нейронной сети. В разделе 2 дается описание численных экспериментов построения областей устойчивости, основанных на предыдущих работах автора, и формулируются выводы о динамике устойчивости сети при изменении параметра p. В заключении резюмируется проведенная работа и обозначаются направления будущих исследований.

1. Построение модели

1.1. Общее уравнение нейронной сети

Чтобы понять, что происходит между нейронами внутри сети, нужно разобрать строение «типичного» нейрона [6, 7]. Основная масса биологических нейронов схожа по строению и свойствам с двигательными нейронами спинного мозга млекопитающих. Из сомы (тела нейрона) исходит много ветвей, называемых дендритами; сома и дендриты образуют входную поверхность нейрона. Из аксонного бугра нейрона выходит длинное волокно, называемое «аксон», ветви которого образуют дерево. Концы ветвей аксона, называемые нервными окончаниями, встречаются с другими нейронами или эффекторами. Аксон может иметь большую длину. Например, тело нейрона, который контролирует большой палец ноги человека, лежит в спинном мозге, а его аксон проходит по всей длине ноги. Скорость передачи нервных импульсов по нервным волокнам человека варьируется от долей метра в секунду (сигналы по немиелинезированным волокнам) до 120 м/с (по быстропроводящим сенсорным волокнам). Еще меньше скорость нервных процессов простейших организмов (до 2 м/с). Эти свойства делают обоснованным ввод запаздывания в уравнения нейронных сетей.

Исследуемая модель дискретна и включает в себя запаздывающие взаимодействия. Для описания системы связей модель содержит специальную матрицу. В следующем разделе будет показано, как формируются связи в модели.

Положим, что нейронная сеть содержит n нейронов с номерами $1, 2, \ldots, n$. Состояние нейрона с индексом j $(1 \le j \le n)$ в момент времени s $(s = 1, 2, \ldots)$ характеризуется его сигналом x_s^j. Состояние всей нейронной сети в момент времени s характеризуется вектором $x_s = (x_s^1, \ldots, x_s^n)^T$. Если каждый нейрон будет изолирован, то динамика нейронной сети подчиняется уравнению $x_s = \alpha x_{s-r}$ $(-1 \le \alpha \le 1)$, $r \in \mathbb{Z}_+$. Поэтому при отсутствии связей между нейронами нулевое стационарное состояние нейронной сети является устойчивым. Коэффициент α назовем коэффициентом демпфирования собственных колебаний, число r – запаздыванием нейрона в реакции на свой собственный сигнал. В матрице взаимодействий нейронов $B = (\beta_{ij})_{i,j=1}^n$) значение β_{ij} есть сила действия j-го нейрона на i-й. Граф нейронной сети будет содержать вершины $1, 2, \ldots, n$ и множество E направленных дуг, такое что $(i, j) \in E$ тогда и только тогда, когда $\beta_{ij} \ne 0$. Будем полагать, что взаимодействие

между любыми двумя различными нейронами происходит с запаздыванием на m тактов $s > m > r$. Начальное состояние сети определяется силами взаимодействия между нейронами, реакцией нейрона на самого себя и запаздываниями в сети. В результате получаем рекуррентное уравнение динамики нерйонной сети (см. [8, 9]).

$$x_s = \alpha\, x_{s-r} + B\, x_{s-m}, \quad s = 1, 2, \ldots \tag{1}$$

1.2. Система связей в нейронной сети

В этом разделе дано описание системы связей в нейронной сети с параметром p ($0 \le p \le 1$) и k ($k \ll n$). Величина p характеризует случайность связей в сети, величина k/n характеризует плотность связей.

Случай 1: $p = 0$. Вначале расположим все нейроны $1, 2, \ldots, n$ по кругу и проведем направленные дуги от каждого нейрона к ближайшему к нему k нейронам по часовой стрелке. Каждую такую дугу снабдим весом b. Затем снабдим каждую дугу, направленной по часовой стрелке, дугой в противоположном направлении с весом a, и, таким образом, сформируем матрицу B в уравнении (1).

Случай 2: $p \neq 0$. Первый шаг такой же, что в предыдущем случае: проведем направленные дуги от каждого нейрона в сети к ближайшим к нему k нейронам по часовой стрелке. Каждую такую дугу снабдим весом b. Затем проведем процесс переключения. Каждая дуга (i, j) ($1 \le i, j \le n$) с вероятностью p заменим дугой (i, v) ($1 \le v \le n$). Причем требуем выполнения следующих условий: 1) дуги, построенные ранее, не должны повторяться; 2) все вершины, удовлетворяющие предыдущему пункту, для выбора вершины v равновероятны. После процесса переключения каждую дугу (i, j) дополняем дугой (j, i) с весом a, и, таким образом, формируется матрица B в уравнении (1).

При $p = 0$ сеть назовем регулярной. При $p > 0$ ($0 < p \le 0,1$) сеть принадлежит к типу small world [2] и обладает высоким коэффициентом кластеризации и малой, по сравнению с регулярной сетью, длиной кратчайшего пути в среднем [1]. На рис. 1 представлены регулярная нейронная сеть ($p = 0$) и случайная сеть ($p = 1$).

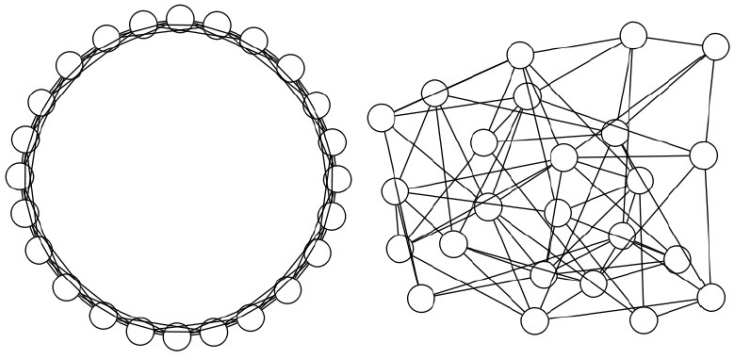

Рис. 1. Пример регулярной и случайной сети

2. Исследование устойчивости нейронной сети

Линейное матричное уравнение (1) зависит от дискретного времени s. Уравнение (1) назовем устойчивым при любом s, если все его решения ограничены, и асимптотически устойчивым, если все его решения стремятся к нулю при $s \to \infty$. Уравнение

$$\det(\lambda^m I - \alpha\lambda^{m-r} I - B) = 0 \tag{2}$$

степени $n \times m$ является характеристическим для (1). В (2) I — единичная матрица размера $n \times n$. Уравнение (1) является асимптотически устойчивым, если и только если все корни его характеристического уравнения лежат строго внутри единичного круга комплексной плоскости. Алгоритмы диагностирования устойчивости уравнения (1) с помощью метода конусов устойчивости изложены в работах [10, 11].

Важно знать, как в исследуемой модели устойчивости зависит от p $(0 \leq p \leq 0,1)$. С помощью метода конусов устойчивости проведены численные эксперименты для определения областей устойчивости в пространстве параметров a, b.

2.1. Алгоритм нахождения границ области устойчивости

Для построения областей устойчивости в виде замкнутой линии в пространстве параметров a, b необходимо найти массив точек, являющихся границей между устойчивым и неустойчивым состоянием сети. Как показано в работе [8], областью устойчивости регулярной сети является замкнутая область. Так как нас интересуют нейронные сети, у которых параметр p, отвечающий за переключения сети, не превосходит $0,1$, то будем полагать, что у сетей small world границей области устойчивости является замкнутая линия и область устойчивости находится внутри нее. Искать границу будем с точностью δ. Применим следующий алгоритм.

ШАГ 1. Находим первую точку границы как точку пересечения границы с осью $0a$. Для этого нам потребуются точки внутри области и одна вне области. В качестве первой точки возьмем точку $(0; 0)$ ввиду того, что она является асимптотически устойчивой точкой для уравнения (1). В качестве второй точки возьмем заведомо неустойчивую при любых параметрах сети. Такой точкой окажется $(1; 0)$. Проводим диагностирование точек принадлежности области устойчивости с помощью алгоритмов, описанных в работах [10, 11]. Уточнение граничной точки области устойчивости проводим методом дихотомии с точностью δ.

ШАГ 2. Все остальные точки находятся следующим образом: 1. С помощью формул поворота осуществляем поворот луча $0a$ на заданный небольшой угол ϕ_0. 2. Ищем следующую точку, как показано в предыдущем шаге. Шаг 2 повторяем до тех пор, пока граница области не замкнется.

Похожий алгоритм поиска границы области устойчивости для непрерывных кольцевых моделей нейронных сетей с двумя соседями был приведен в [12].

2.2. Результаты численных экспериментов

Рассмотренные ранее алгоритмы построения областей устойчивости уравнения (1) были реализованы в среде MATLAB. Численные эксперименты проводились для определения динамики изменения областей устойчивости нейронной сети при изменении параметра p $(0 \leq p \leq 0,1)$ и одинаковыми коэффициентами демпфирования α, запаздываниями в работе сети m, r и числом n нейронов в сети. В проведенных экспериментах согласно [5] число ближайших соседей составляет 2% от общего числа нейронов. Для каждого значения p с помощью алгоритмов, описанных в разделе 2.1, было построено 10 различных матриц B и получены массивы точек на границах устойчивости нейронных сетей. Для визуализации результатов на рисунке области устойчивости являются усредненными для каждого p.

На рис. 2 представлены области устойчивости регулярной нейронной сети и сети с разным значением параметра p. В левой части рисунка представлены области устойчивости

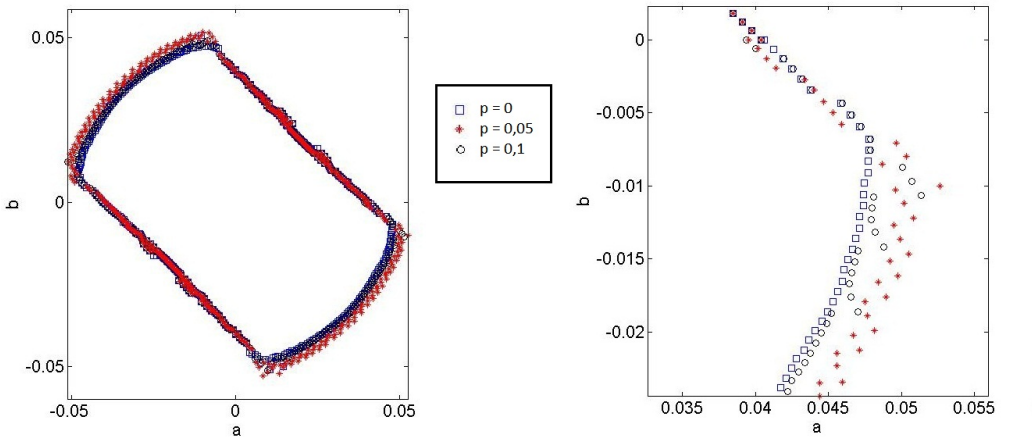

Рис. 2. Границы областей устойчивости нейронной сети при разных значенияхпараметра p

целиком. В правой части рисунка для понимания динамики изменения границ областей представлена часть областей устойчивости. Здесь $n = 1500$, $k = 30$, $m = 3$, $r = 2$, $\alpha = 0,4$. В случае $ab > 0$ граница области устойчивости практически не изменилась при изменении параметра p. В случае $ab < 0$ область устойчивости при $p = 0$ является наименьшей из представленных. При значении параметра $p = 0,1$ область устойчивости больше, чем при $p = 0$, но меньше области устойчивости при $p = 0,05$. Полученные результаты не противоречат результатам для регулярных сетей [8].

Заключение

В рамках исследуемой модели рассмотрены два типа сетей: регулярные ($p = 0$) и сети small world ($0 < p \leq 0,1$). Нейронные сети типа small world имеют большую область устойчивости, чем регулярные. Для регулярной нейронной сети увеличение количества нейронов в сети незначительно влияет на изменение области устойчивости. Это известно из общих формул, задающих спектр матрицы B [8, 13]. Сохраняется ли подобная тенденция для нейронных сетей типа small world будет предметом отдельной статьи.

Работа выполнена при финансовой поддержке РФФИ в рамках научного проекта № 16-31-00343.

Литература

1. Watts D., Strogatz S., Collective dynamics of «small-world» networks. Nature. 1998. Vol. 393. P. 440–442.

2. Gray R.T., Fung C.K.C., Robinson P.A. Stability of small-world networks of neural populations. Neurocomputing. 2009. Vol. 72(7–9). P. 1565–1574.

3. Sinha S. Complexity vs stability in small-world networks. Physica A. 2005. Vol. 346. P. 147–153.

4. Hart M.G., Ypma R.J.F., Romero–Garcia R., Price S.J., Suckling J. Graph theory analysis of complex brain networks: new concepts in brain maping aplied to neurosurgery. Journal of Neurosurgery. 2016. Vol. 124, No. 6. P. 1665–1678.

5. Netoff T.I., Clewley R., Arno S., Keck T., John A. White Epilepsy in Small-World Networks. The Journal of Neuroscience. 2004. Vol. 24(37). P. 8075–8083.

6. Arbib M.A., Érdi P., Szentágothai J. Neural Organization: Structure, Function, and Dynamics. Cambridge. MA: MIT Press, 1998. 420 p.

7. Arbib M. The Handbook of Brain Theory and Neural Networks. Cambridge. MA: MIT Press, 2003. 1308 p.

8. Ivanov S.A., Kipnis M.M. Stability Analysis Discrete-time Neural Networks with Delayed interactions: Torus, Ring, Grid, Line. International Journal of Pure and Aplied Math. 2012. Vol. 78(5). P. 691–709.

9. Ivanov S.A., Kipnis M.M., Medina R. On the stability of the Cartesian product of a neural ring and an arbitrary neural network. Advances in Difference Equations. 2014. Vol. 2014. P. 1–7.

10. Kipnis M.M., Malygina V.V. The Stability Cone for a Matrix Delay Difference Equation. International Journal of Mathematics and Mathematical Sciences. 2011. Vol. 2011. P. 1–15.

11. Ivanov S.A., Kipnis M.M., Malygina V.V. The stability cone for a difference matrix equation with two delays. ISRN Aplied Math. 2011. Vol. 2011. P. 1–19.

12. Хохлова Т.Н. Построение областей устойчивости круговых нейронных сетей. Хроники ОФЭРНиО. 2012. Т. 1(32). С. 4–5.

13. Khokhlova T.N., Kipnis M.M. The breaking of a delayed ring neural network contributes to stability: The rule and exceptions. Neural Networks. 2013. Vol. 48. P. 148–152.

DOI: 10.14529/cmse160305

CALCULATION OF STABILITY DOMAINS OF DISCRETE MODELS OF BIG SIZE SMALL WORLD NETWORKS

© 2016 S.A. Ivanov[1], M.M. Kipnis[2]

[1]*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia),*
[2]*South Ural State Humanitarian Pedagogical University*
(pr. Lenina 69, Chelyabinsk, 454080 Russia)
E-mail: saivanov@susu.ru, mmkipnis@gmail.com

Received: 11.07.2016

The article is devoted to description of discrete models of small world networks with a large number of neurons with a certain parameter p varying from 0 to 1. For $p = 0$ have model, regular neural networks, which is a ring network in which each neuron interacts with several neighbors on the ring. In the case $p = 1$ have a model with randomly distributed connections. When the values of p not exceeding $0,1$ have the Watts–Strogatz small world network. Such a neural network can be models of different neural structures in living organisms, for example, the hipocampus of the mammalian brain. This paper examines the dynamics of change areas of stability of such neural networks when $0 \le p \le 0,1$. Numerical experiments show an increase in sustainability in the transition from a regular network to small world.

Keywords: Watts–Strogatz discrete models, small world, stability.

FOR CITATION

Ivanov S.A., Kipnis M.M. Calculation of Stability Domains of Discrete Models of Big Size Small World Networks. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 69–75. (in Russian) DOI: 10.14529/cmse160305.

References

1. Watts D., Strogatz S. Collective Dynamics of «Small-World» Networks. *Nature*. 1998. vol. 393. pp. 440–442. DOI: 10.1038/30918.
2. Gray R.T., Fung C.K.C., Robinson P.A. Stability of Small-World Networks of Neural Populations. *Neurocomputing*. 2009. vol. 72(7–9). pp. 1565–1574. DOI: 10.1016/j.neucom.2008.09.006.
3. Sinha S. Complexity vs Stability in Small-World Networks. *Physica A*. 2005. vol. 346. pp. 147–153. DOI: 10.1016/j.physa.2004.08.062.
4. Hart M.G., Ypma R.J.F., Romero-Garcia R., Price S.J., Suckling J. Graph Theory Analysis of Complex Brain Networks: New Concepts in Brain Maping Aplied to Neurosurgery. *Journal of Neurosurgery*. 2016. vol. 124, no. 6. pp. 1665–1678. DOI: 10.3171/2015.4.jns142683.
5. Netoff T.I., Clewley R., Arno S., Keck T., John A. White Epilepsy in Small-World Networks. *The Journal of Neuroscience*. 2004. vol. 24(37). pp. 8075–8083.
6. Arbib M.A., Érdi P., Szentágothai J. Neural Organization: Structure, Function, and Dynamics. *Cambridge*. MA: MIT Press, 1998. 420 p.
7. Arbib M. The Handbook of Brain Theory and Neural Networks. *Cambridge*. MA: MIT Press, 2003. 1308 p.
8. Ivanov S.A., Kipnis M.M. Stability Analysis Discrete-Time Neural Networks with Delayed Interactions: Torus, Ring, Grid, Line. *International Journal of Pure and Aplied Math*. 2012. vol. 78(5). pp. 691–709.
9. Ivanov S.A., Kipnis M.M., Medina R. On the Stability of the Cartesian Product of a Neural Ring and an Arbitrary Neural Network. *Advances in Difference Equations*. 2014. vol. 2014. pp. 1–7. DOI: 10.1186/1687-1847-2014-176.
10. Kipnis M.M., Malygina V.V. The Stability Cone for a Matrix Delay Difference Equation. *International Journal of Mathematics and Mathematical Sciences*. 2011. vol. 2011. pp. 1–15.
11. Ivanov S.A., Kipnis M.M., Malygina V.V., The Stability Cone for a Difference Matrix Equation with Two Delays. *ISRN Aplied Math*. 2011. vol. 2011. pp. 1–19. DOI: 10.5402/2011/910936.
12. Khokhlova T.N. *Postroenie oblastey ustoychivosti krugovykh neyronnykh setey. Khroniki OFERNiO* [The Construction of Regions of Stability of a Circular Neural Networks. Chronicles of OFERNiO]. Moscow. 2012. vol. 1(32). pp. 4–5. (in Russian)
13. Khokhlova T.N., Kipnis M.M. The Breaking of a Delayed Ring Neural Network Contributes to Stability: The Rule and Exceptions. *Neural Networks*. 2013. vol. 48. pp. 148–152. DOI: 10.1016/j.neunet.2013.08.001.

УДК 004.457, 004.382.2 DOI: 10.14529/cmse160306

OCTOSHELL: СИСТЕМА ДЛЯ АДМИНИСТРИРОВАНИЯ БОЛЬШИХ СУПЕРКОМПЬЮТЕРНЫХ КОМПЛЕКСОВ*

© 2016 г. Д.А. Никитенко, В.В. Воеводин, С.А. Жуматий

Научно-исследовательский вычислительный центр Московского государственного университета имени М.В. Ломоносова

(119991 Москва, ул. Ленинские Горы, д. 1, стр. 4)

E-mail: voevodin@parallel.ru, serg@parallel.ru, dan@parallel.ru

Поступила в редакцию: 20.10.2015

Управление современными суперкомпьютерными центрами и входящими в их состав вычислительными системами представляет собой сложный и комплексный процесс. Традиционное использование многочисленных инструментов для решения отдельных задач по управлению и администрированию суперкомпьютеров становится ограничивающим фактором эффективного использования вычислительных ресурсов при растущих масштабах систем. Разработанная система поддержки работы суперкомпьютерных центров «Octoshell» призвана решить указанную проблему, реализуя в едином интерфейсе основные инструменты администрирования, и позволяет в значительной мере автоматизировать выполнение типовых задач обеспечения эффективного функционирования больших суперкомпьютерных комплексов.

Ключевые слова: суперкомпьютер, мониторинг, управление суперкомпьютерным центром, администрирование суперкомпьютеров, система поддержки пользователей.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Никитенко Д.А., Воеводин В.В., Жуматий С.А. Octoshell: система для администрирования больших суперкомпьютерных комплексов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 76–95. DOI: 10.14529/cmse160306.

Введение

Современный суперкомпьютер представляет собой большой и сложный вычислительный комплекс, включающий в себя десятки тысяч компонент. Их число даже у рядовых систем уже измеряется десятками тысяч [1], при этом у наиболее крупных систем оно насчитывает сотни тысяч, а в некоторых случаях — миллионы [2]. Растет число вычислительных узлов, число вычислительных ядер процессоров и ускорителей и др. Большинство специалистов сходятся во мнении, что эта тенденция продолжится и далее.

В рамках суперкомпьютерного центра могут функционировать сразу несколько таких систем, и, что очень важно, в работу включается не только управление непосредственно вычислителями, но и множество других сопутствующих процессов. Например, управление лицензионным программным обеспечением, квотами пользователей, техническая поддержка, отслеживание гарантийного и послегарантийного ремонта и многое другое.

Все эти сущности связаны между собой, и, учитывая количество всех составляющих, очень трудно эффективно организовать рабочие процессы. Помимо значительного числа

* Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2015».

аппаратных компонент, немаловажный вклад имеют и программные — системное программное обеспечение, оптимизированные библиотеки и программы для высокопроизводительных вычислений, прикладные пакеты. Сами пользователи суперкомпьютера являются неотъемлемой и важнейшей его частью, имея свои особенности, требования и предпочтения, квоты и привилегии, набор используемого ПО, историю использования суперкомпьютера и др.

Рассмотрим в качестве примеров некоторые задачи, с которыми приходится сталкиваться каждый день при управлении большой вычислительной системой.

Пример 1. Управление лицензионным ПО. На каждом суперкомпьютере может быть установлено десятки пакетов ПО. По каждому такому пакету необходимо контролировать круг пользователей, которым разрешено его использование, сроки лицензии, обновления, контакты с технической поддержкой производителя. На первый взгляд, все просто, но только до тех пор, пока вдруг не окажется, что срок действия лицензии истекает, бюджет на закупку не предусмотрен, и др., а пользователям надо считать... Или пока не поступит заявка на пакет от пользователя с логином «abc123», и не возникнет вопрос, распространяются ли условия лицензии на этого пользователя? Чем он занимается? Где работает? Где условия лицензий и контакты? Зачастую — «в почте», «в какой-то Excel таблице». Это становится большой проблемой, если вспомнить о стоимости лицензий, большом числе пользователей и требуемых компонент ПО. Упорядоченность, полноценный учет становятся вдвойне актуальными, когда встают вопросы о необходимости продления лицензии на тот или иной пакет ПО. Для этого необходимо понимать, насколько он был реально востребован, использовался он одной рабочей группой или несколькими, какие проекты от него зависят?

Пример 2. В состав центра входят тысячи единиц техники (вычислительные узлы, коммутаторы, диски и многое другое), которые не могут не ломаться. Сколько времени уходит на типичный ремонт одного узла? Сколько процессорочасов было потеряно за текущий год по причине ремонта? По какой причине выключен конкретный узел? Просто выключен, отдан в ремонт или уже вернулся из ремонта, но не введен в счетное поле? Эти вопросы незаметны и незначительны для одной лаборатории, офиса, а для большого центра становятся критичными.

Есть и множество иных задач, которые требуют комплексных знаний о составе центра, структуре процессов, происходящих внутри него. От эффективности решения всех этих задач напрямую зависит эффективность работы центра в целом.

Для того, чтобы добиться этого, необходимо четко описать все операции, которые выполняются в центре, формализовать их, описать процессы. Несмотря на актуальность и значительную предысторию, предметная область описана в недостаточной мере. Отсюда вытекала необходимость создания модели работы СКЦ. Составление модели работы суперкомпьютерного центра может помочь изменить и его работу, так как можно найти более эффективные способы организации внутренних процессов. Например, более эффективно построить систему квот и приоритетов. В традиционном подходе квота или допуск к ПО выделяется отдельному человеку, в то время как он может работать совместно с другими коллективами в рамках нескольких проектов одновременно. Поэтому рациональнее выделять квоты и допуски не на людей, а на проект.

Статья организована следующим образом. В разделе 1 рассмотрены основные объекты и решаемые задачи при организации работы СКЦ. Раздел 2 содержит описание

традиционных подходов к управлению суперкомпьютерными комплексами. В разделе 3 приводится описание основ разработанной системы Octoshell для администрирования больших суперкомпьютерных центров. В заключении подводятся итоги проведенного исследования и намечаются направления будущих работ.

Основные объекты и решаемые задачи в рамках работы СКЦ

Попробуем рассмотреть эти вопросы подробнее, опираясь на опыт администрирования суперкомпьютерного центра МГУ — одного из крупнейших в России [3]. Центр обслуживает несколько сотен рабочих групп, объединяя более 2500 пользователей. В состав центра входят такие суперкомпьютеры как «Ломоносов», «Ломоносов-2», «Чебышёв». Общее число учетных единиц техники составляет десятки тысяч.

Из чего складывается деятельность суперкомпьютерного центра? Что должно найти отражение в «скелете» модели, какие основные сущности необходимо увязать в единый рабочий процесс?

– Прежде всего, пользователи — специалисты разного уровня подготовки, у которых есть потребность в использовании тех или иных вычислительных ресурсов и ПО.
– Проекты — реальные прикладные задачи, которые решаются пользователями самостоятельно или в составе рабочих групп.
– Аппаратура. Те самые разнородные вычислительные ресурсы, необходимые для решения прикладных задач. Вычислительные системы, выделенные их разделы, например, по принципу однородности ресурсов.
– Системное и прикладное программное обеспечение. Множество прикладных пакетов и библиотек, необходимых для эффективного решения вычислительных задач.
– Механизмы организации работы СКЦ.

Каждая из упомянутых выше компонент обладает массой свойств и особенностей. Причем важно понимать, что в динамике свойства даже уже описанных объектов могут претерпевать изменения. Могут добавляться новые свойства, новые взаимосвязи. Это необходимо учитывать наряду с возможно нетривиальной взаимосвязью этих компонент, отражающих разного рода зависимости [4].

В условиях конкуренции за вычислительные ресурсы, необходимо выстраивать систему приоритетов доступа к ним, а также квотирование. Наиболее рациональным представляется подход, при котором ресурсы выделяются на решение определенной задачи (проект), и не важно, каким числом пользователей она решается. Кроме того, не стоит забывать, что один и тот же пользователь может входить в состав различных рабочих групп и участвовать в решении совершенно не связанных друг с другом задач, играя в их решении принципиально разные роли. В одном случае человек может быть руководителем работ, например, студенческого практикума, когда большинство программ запускается студентами. При этом одновременно он может быть единственным исследователем в рамках другого проекта, а в третьем проекте — уже выступать в роли исполнителя наряду с другими коллегами.

Что касается используемого программного обеспечения, то крайне важен уже упомянутый вопрос лицензий. На разных системах может быть установлен различный набор прикладных пакетов или их версий, могут использоваться лицензии с отличающимися условиями. При этом к одному и тому же пользователю могут относиться со-

вершенно разные условия лицензий в зависимости от того, в рамках какого проекта ведется работа.

Помимо учета лицензий необходимо отслеживать статус самого оборудования, чтобы минимизировать время его простоя, повышая потенциальную отдачу от вычислителя в целом.

Отдельно необходимо сказать о процедурах и процессах, непосредственно не связанных с вычислительным этапом работ. Речь идет о комплексе административных процедур и поддержке работы пользователей.

Работа каждого пользователя начинается с заведения учетной записи, при этом учетных записей может быть заведено несколько. Отдельная запись пользователя может использоваться для работы в рамках отдельного проекта. Если в рамках проекта используются ресурсы разных суперкомпьютеров центра, то учетные записи должны создаваться на каждой машине, при этом должна сохраняться информация для идентификации, какие учетные записи используются для работы над какими проектами.

В работе вычислительного центра всегда возникает необходимость связаться с пользователями. Будь то один конкретный человек, участники определенного проекта или, например, сотрудники одной организации. Для адресной связи с пользователем требуются контактные данные, и их требуется поддерживать актуальными. Важно, что все это должно идти в соответствии с действующим законодательством относительно использования персональных данных.

Часто у самих пользователей возникают вопросы, связанные с работой на вычислительной системе. Для корректного описания возникающих проблем администраторам зачастую необходимо «видеть» условия, в которых у пользователя она возникла. Это значит, что от пользователя требуется описание условий возникновения — командная строка, выдачи ошибок, скриншоты, элементы кода и др. Представление этих данных через стандартизованный интерфейс, наличие элементов «ситуационного экрана», всех данных о пользователе, его учетных записях и истории обращений в распоряжении администратора кардинально сказываются на эффективности оказания помощи пользователям.

Если для небольшого кластера еще может сработать схема общения по электронной почте и телефону, то для больших систем это становится неприемлемым. Использование же готовой системы поддержки, подразумевает глубокую ее интеграцию с другими инструментами администрирования, что на практике достаточно сложно осуществимо.

Создание классического раздела с частыми вопросами, что, вообще говоря, само по себе является шагом необходимым, не всегда решает даже уже хорошо известные ранее встречавшиеся проблемы: к сожалению, многие пользователи предпочитают миновать шаг чтения инструкций или правил и обращаться по каждому поводу непосредственно к администраторам суперкомпьютеров. Это приводит как к увеличению интенсивности самих обращений, так и снижению эффективности самой поддержки: тратится намного больше времени на разбор даже простых ситуаций.

К административным же процедурам может добавиться система проведения экспертизы работы пользователей с целью возможной корректировки их квот и приоритетов.

Изначальное предоставление доступа к вычислительным ресурсам может предполагать создание поручительств и гарантийных писем от организаций, гарантирующих использование вычислительных ресурсов пользователем по назначению.

Все это требуется максимально автоматизировать для ускорения предоставления доступа, минимизации ошибок, повышения эффективной отдачи от суперкомпьютерного центра в целом.

Традиционные подходы к управлению СКЦ

Традиционно для организации работы суперкомпьютерных центров используется комбинация систем мониторинга, управления конфигурациями, управления учетными записями, средств организации поддержки пользователей, удаленного запуска программ.

Недостаток такого подхода заключается в том, что решение практически любой частной задачи, например, заведение новой учетной записи или предоставление выделенного времени счета для некоторой группы пользователей, ведет к необходимости выполнения множества действий с разрозненными приложениями, потере времени администратора на избыточные действия. Это, в свою очередь, ведет к ошибкам, потере информации, увеличении времени реакции.

Ситуация усугубляется, когда центр поддерживает несколько суперкомпьютеров, так как вести учет нескольких систем одновременно не всегда становится возможным и приходится дублировать множество информации.

На данный момент популярными средствами, применяемыми в различных суперкомпьютерных центрах, являются [5]:

а) Средства управления пользователями:
   1) LDAP и оболочки для управления им;
   2) NIS;
   3) batch-системы (torque [6], slurm [7], openpbs [8], maui, lsf, cleo и другие);
   4) ручной учет в MS Excel, OO calc и аналогичных пакетах.

б) Средства мониторинга оборудования и сервисов: Ganglia [9], Zabbix [10], Nagios [11].

в) Средства удаленного управления: ssh, pdsh; ipmi, ilo; snmp.

г) Средства организации общения с пользователями:
   1) Ticket Management Systems (OTRS [12], Trak, RT и другие);
   2) mailman;
   3) wiki;
   4) форумы.

Использование всех этих средств без интеграции друг с другом приводит к множеству ошибок, несогласованности, дублированию информации. Это связано с тем, что невозможно провести интеграцию подобных средств без четкого понимания состава и взаимосвязи того, что собственно необходимо учитывать, чем и как управлять.

На основании опыта администрирования и поддержки работы суперкомпьютерного центра МГУ, а также на основании многочисленных обсуждений с административным и техническим составом других суперкомпьютерных центров, включая крупнейшие европейские центры, можно сформулировать следующую проблему.

Существуют инструменты, позволяющие решать отдельные задачи для обеспечения функционирования суперкомпьютерного центра, но при увеличении масштабов вычислительных систем, роста числа обслуживаемых пользователей и др. число ошибок несогласования, возникающих при использовании отдельных инструментов, значительно

возрастает, что существенно сказывается на отдаче вычислительных комплексов в целом.

Система Octoshell

Ежедневное решение подобных задач в рамках СКЦ МГУ привело к пониманию необходимости создания системы, интегрирующей все основные процессы управления и администрирования суперкомпьютерным центром в единый комплекс программных средств. Система получила название Octoshell, т.к. может рассматриваться как оболочка, объединяющая целый ряд отдельных инструментов/модулей. О некоторых деталях реализации будет сказано далее, сейчас же приведем перечень основных сущностей, которыми оперирует разработанная система.

1.1. Основные сущности

3.1.1 Пользователь

Пользователи — те самые реальные исследователи, для которых и существуют высокопроизводительные вычислительные системы. В рамках каждой вычислительной системы СКЦ каждый пользователь может входить в один или несколько проектов. Он может выступать в качестве участника или руководителя проекта. Участие пользователя в проекте должно быть отражено в виде учетной записи на вычислительной системе. Пользователь обязан принадлежать одной или более организаций. Дополнительно пользователь может выступать в роли эксперта и/или администратора.

Каждый пользователь изначально обязан пройти процедуру регистрации для начала работы. В процессе регистрации он обязан указать необходимые персональные данные, включая контактные данные для экстренной связи, и в дальнейшем он должен поддерживать актуальность этих данных.

Пользователь должен иметь доступ к данным статистики, касающимся его проектов или учетных записей, а также к информации о программном обеспечении, текущим или планирующимся процессам обслуживания систем, если они нарушают обычный порядок работы. Важной возможностью, предоставляемой пользователю, является техническая поддержка, с помощью которой он может получить консультации по работе, сообщить о сбое, возникшей проблеме и др.

В отличие от обычного пользователя, администратор имеет возможность управления параметрами системы. Возможно существование администраторов с ограниченными правами, например, для управления резервным копированием, для создания новых пользователей и др. Администраторы имеют доступ к просмотру и изменению данных других объектов в системе. Для администраторов с ограниченными правами могут быть доступны только необходимые для них объекты.

Администратор, в отличие от обычного пользователя, имеет доступ не только к собственным данным определенного типа (например, проекту), а к данным всех пользователей. Важные дополнительные возможности, которые появляются у администратора в дополнении к возможностям пользователя: построение и просмотр отчетов, управление программным обеспечением, квотами и приоритетами, а также управление процессами обслуживания и сопровождения аппаратных компонент.

Наиболее типичные виды администраторов:

– суперадминистратор, обладающий полными правами на управление объектами информационной системы;
– администратор с ограниченными правами — для сопровождения документооборота, решения простых задач пользователей и других типовых работ;
– эксперт, наделенный правами видеть регистрационные данные пользователей, описания проектов и отчеты по проделанной в их рамках работе, включая их историю, вести анонимную переписку с пользователем с целью доработки пользователями отчетов и приведения пользователями указываемой ими информации к надлежащему виду.

3.1.2 Проект

Этот объект определяет деятельность одного или нескольких пользователей в рамках одного направления исследований. Традиционно проекту может соответствовать один или множество учетных записей в зависимости от специфики суперкомпьютерного центра. Соответственно, и выделение ресурсов, и учет также могут вестись как в терминах учетных записей, так и в терминах проектов.

Проектный подход позволяет описать несколько различных вариантов организации учета пользователей. Если каждый пользователь имеет одну учетную запись и использует ее во всех работах, то полагаем, что каждый пользователь ведет один собственный проект. Если на каждый проект выделяется одна учетная запись, и ее используют все участники, то считаем, что за него ответственен владелец проекта.

Однако, эти подходы, несмотря на свою распространенность, имеют существенные недостатки, и мы предлагаем использовать подход, в котором каждый участник проекта получает отдельную учетную запись, при этом все участники одного проекта входят в одну группу.

При таком подходе один пользователь должен иметь разные учетные записи для разных проектов, что позволяет проще организовать учет и квотирование, а также управлять доступом в терминах проектов, а не отдельных учетных записей. Например, если пользователь ведет свой проект и одновременно участвует в коллективном проекте, то в случае, если коллективный проект будет закрыт или заблокирован, вход для личного проекта останется открытым.

Таким образом, ориентация модели на проекты позволяет как описать традиционные схемы предоставления доступа, так и предложить более эффективную схему. Каждый проект представлен в системе в виде UNIX-группы и одной или более учетной записи. Каждая учетная запись соответствует одному или всем участникам проекта (в зависимости от конечной реализации проектов в системе). Рекомендуется использовать отдельную учетную запись для каждого участника.

Каждый участник проекта должен указать организацию, от имени которой он участвует в проекте. Организация, указанная руководителем проекта, считается ведущей организацией проекта. Такого рода сведения важны, например, для применимости некоторых лицензий на прикладное ПО.

Проекты могут иметь потребность в особом типе ресурсов, ПО, режиме запуска задач и др.

По каждому проекту собирается статистика использования, например, число затраченных процессорочасов, число выполненных задач, объем использованного дискового пространства и др.

3.1.3 Организация

Организации представляют интересы исследовательских групп и выступают гарантами соблюдения пользователями правил предоставления ресурсов. Это может быть реализовано следующим образом. От имени руководителей организаций или подразделений создаются поручительства для получения доступа, с поручительствами соотносятся пользователи и конкретные проекты. Организации могут быть различных типов. Например, в СКЦ МГУ практикуется следующая градация:

– МГУ и его филиалы;
– российское высшее учебное заведение;
– зарубежное высшее учебное заведение;
– институты РАН;
– российская научная организация (кроме РАН);
– зарубежный исследовательский центр;
– и др.

3.1.4 Вычислительная система

Основная цель при администрировании суперкомпьютерных центров — обеспечить возможность использования вычислительных ресурсов пользователями. Вычислительные системы зачастую являются гетерогенными, кроме того, нередко добавляются узлы нового типа в рамках расширения/обновления вычислительной системы. В рамках одного СКЦ таких систем может быть несколько, и одним из основных объектов модели является вычислительная система. Среди своих атрибутов она имеет набор учетных записей, установленного прикладного программного обеспечения.

Приоритеты

Отдельным проектам и пользователям или учетным записям могут быть предоставлены различные приоритеты на вычислительной системе. Приоритеты могут быть реализованы разными способами в зависимости от особенностей систем. Например, повышенный приоритет может ускорять продвижение задач в очереди или автоматически ставить их выше задач с более низким приоритетом.

Квоты

Квоты отчасти схожи с приоритетами, но имеют другой смысл. С помощью квот можно ограничить объем ресурсов для проекта или учетной записи, например, число процессорочасов или объем дискового пространства. Кроме глобального ограничения ресурсов можно ограничивать «мгновенное» потребление ресурсов, например, число одновременно запущенных задач или число занятых задачами процессоров.

При наличии возможности в планировщике и менеджере ресурсов можно использовать квоты на ресурсы за временные отрезки, например, за месяц или неделю. Подобные квоты имеет смысл применять к таким ресурсам, как процессорочасы или число задач. В этом случае пользователь сможет использовать ресурс в нужном ему режиме, не «упираясь» в квоту постоянно: можно запустить несколько задач подряд и выбрать квоту за короткое время, а потом не производить запусков до нового периода квотирования. Это дает остальным пользователям возможность свободнее работать.

Важно, что на разных вычислительных системах механизмы квотирования и привилегий могут существенно отличаться, при этом требуется их согласование при организа-

ции доступа к таким разнородным ресурсам, в том числе, и в рамках отдельного проекта.

Непосредственно реализация механизмов квот и приоритетов не является частью Octoshell, но с его помощью администратор может управлять ими из единой точки, получать данные обо всех квотах определенного проекта, пользователя и др.

1.2. Администрирование

В работе любого суперкомпьютерного центра есть некоторый общий набор ключевых процессов администрирования. Схемы их реализации могут быть кардинально отличными, в каждом конкретном случае используется своя логика, свои правила. И в рамках отдельно взятого СКЦ изменения в организации работ — не редкость. Именно поэтому необходим «конструктор», позволяющий связать основные объекты и процессы, присущие большинству СКЦ в том логическом виде, который является актуальным. Рассмотрим ключевые процессы в Octoshell.

3.2.1 Регистрация и предоставление доступа

Получение доступа к вычислительным ресурсам начинается с регистрации пользователя в системе Octoshell. Регистрация пользователя включает в себя:

– создание базового объекта пользователя;
– заполнение обязательных данных (контактных и др.);
– привязка пользователя к организации и при необходимости, создание новой организации.

Следует помнить, что в соответствии с законодательством РФ не все данные о пользователе могут быть сохранены без применения специальных средств (хотя для работы с суперкомпьютером такие данные, как правило, не требуются), и для сохранения любых персональных данных пользователь должен явно дать разрешение.

После того, как пользователь создан в Octoshell, он получает возможности создания новых проектов или принятия приглашений на участие в уже созданных.

В зависимости от внутренних регламентов суперкомпьютерного центра статус проекта может быть изменен на активный (т.е. создание учетных записей всем участникам) при соблюдении некоторых правил, таких как:

– получение и одобрение официального поручительства от имени организации;
– определенный статус пользователя;
– прямое указание руководства и др.

Сами правила перевода объектов из одного состояния в другое, условия их создания или удаления, а также обязательность и формат некоторых характеристик определяются исходя из политики работы конкретного центра.

Если все условия для предоставления доступа выполнены, то на вычислительных системах выполняются скрипты и создаются учетные записи для всех пользователей, входящих в проект. При этом пользователь может использовать один ключ для работы со всеми своими учетными записями, относящимися к различным проектам.

3.2.2 Рассылки и уведомления

В ходе работы суперкомпьютерного центра часто возникают ситуации, затрагивающие определенные группы пользователей или проектов или даже всех пользователей.

Это может быть выведение машины из работы на время профилактики, изменение системного или прикладного ПО и другие изменения.

Для доведения подобной информации до пользователей существуют механизмы рассылок и уведомлений. В рамках Octoshell создан модуль почтовых рассылок, позволяющий создавать и использовать шаблоны рассылок по определенному набору адресатов. Выборка получателей может быть сделана на основании статусов проектов, организаций и др.

Рассылки разделены на две группы: информационные и общие. При этом подписка на общую часть является обязательной для пользователей, т.е. если пользователь попадает под целевую группу рассылки, то от получения рассылки нельзя отказаться. Через данный тип рассылок ведется уведомление о важных событиях, связанных с работой СКЦ. Второй тип рассылки используется для уведомления о событиях, не затрагивающих работу пользователей напрямую. Например, объявления о проведении в центре мастер-классов по некоторому прикладному пакету ПО.

Механизм уведомлений является скорее надстройкой над другими модулями. Например, над модулями поддержки и перерегистрации. При возникновении события, требующего реакции от пользователя, соответствующие яркие указатели на вкладках интерфейса системы будут сигнализировать о необходимости реакции от пользователя.

3.2.3 FAQ

Типовые вопросы, такие как правила получения доступа, основные принципы работы на вычислительных системах и их конфигурации и др. необходимо должны быть описаны в разделе часто задаваемых вопросов. Чем нагляднее и подробнее будут описания, тем меньше администраторам придется тратить время на такие вопросы, для которых заранее готов ответ.

Конечно, всегда находятся пользователи, которые предпочитают сразу задавать вопрос, не изучая FAQ. Тем не менее, при хорошей структуризации и подробности раздела частых вопросов вероятность того, что пользователь, будучи отправленным к конкретной части раздела, найдет ответы на свои вопросы и в следующий раз сначала проведет поиск ответа на свой вопрос самостоятельно, возрастает.

3.2.4 Поддержка

Техническая поддержка пользователей суперкомпьютерного центра является одной из ключевых составляющих в администрировании. При обращении в техническую поддержку крайне важной является максимально подробное описание возникшей проблемы. При этом имеют значение все детали, которые имеют хотя бы малейшее отношение к обращению: проект, система, прикладное ПО, текущие квоты, последние данные отчетов и др. Наличие этой информации позволяет администратору резко сократить время решения проблемы. Значительная часть этих данных может быть получена автоматически или с помощью пользователя при составлении обращения.

При составлении обращения пользователя в поддержку целесообразно предлагать пользователю ответить на сопутствующие вопросы о типе проблемы и ее обстоятельствах. По ответам можно группировать вопросы по типам обращений, что позволяет, например, объединять однотипные проблемы в одну, исследовать успешные решения подобных проблем и в целом более эффективно решать проблемы.

Все имеющие соответствующий доступ администраторы видят список поступивших обращений и их детали. Если понятно, кто из администраторов ответственный за решение проблемы, его можно назначить. В противном случае ответственным становится первый ответивший на обращение. При уточнении проблемы ее можно переадресовать, например, более компетентному специалисту.

Все обращения должны сохраняться и дублироваться через оперативные каналы: email, jabber, и др. В каждый момент времени каждое обращение должно иметь фиксированный статус: «новое», «есть ответ», «решено», «обновление проблемы» и др. (см. рис. 1). Пользователь и администратор должны иметь возможность в любой момент написать сообщение по обращению, которое обновит статус обращения и будет оперативно отправлено всем участникам переписки.

Важно, что все обращения в поддержку производятся через один интерфейс. Это позволяет сохранять и видеть всю историю обращений и решений проблем со всеми деталями.

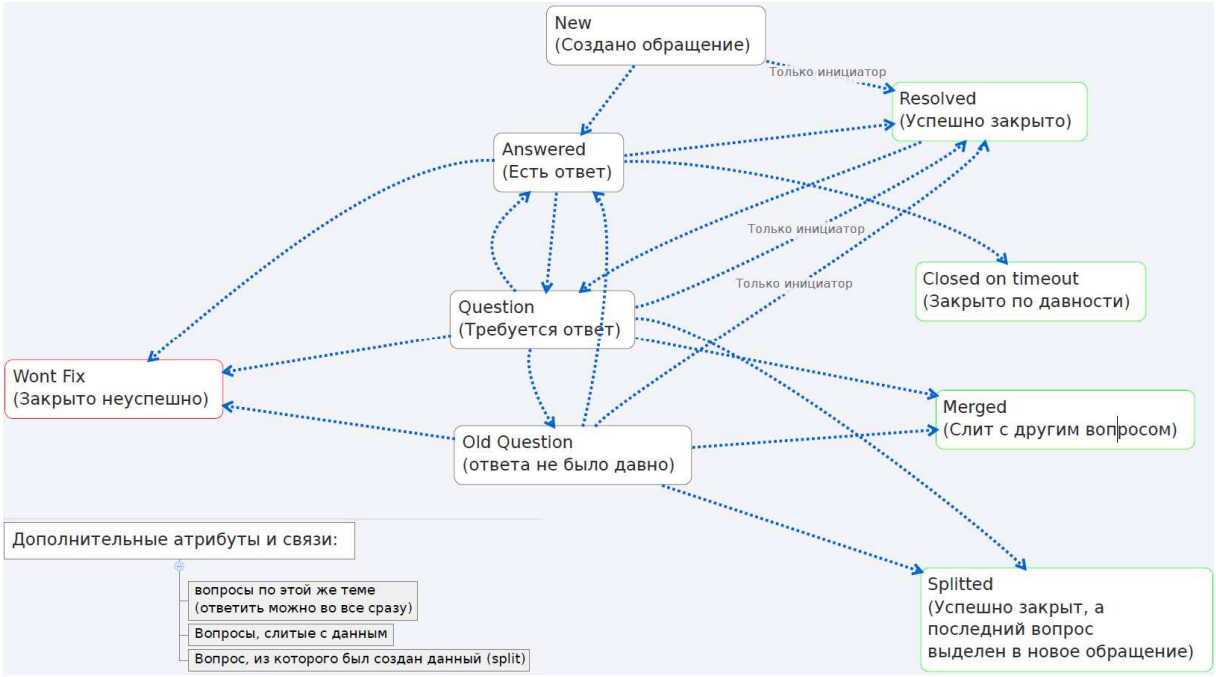

Рис. 1. Диаграмма состояний обращений в пользовательской поддержке

Интерфейс позволяет производить выборку и сортировку по новизне обращений, по ответственному администратору, по инициировавшему пользователю, типу обращения, его статусу и другим параметрам.

3.2.5 Процессы обслуживания и сопровождения аппаратных компонент

Так как оборудование и программное обеспечение вычислительных систем центра постоянно нуждается в периодическом и внеплановом обслуживании и обновлении, для обеспечения своевременного принятия каких-либо мер необходимо отслеживать статус компонент. Прежде всего, это касается больших вычислительных комплексов, где отдельные элементы оборудования даже в надежных системах могут ежедневно выходить из строя просто из-за колоссального числа компонент.

В предлагаемом подходе каждая запись о проведении работ или отправке оборудования в гарантийный ремонт сопровождается следующей информацией:

– состав оборудования или ПО;
– время обнаружения отказа или наступления срока профилактики;
– диагностика сбоя (если таковой был);
– время выведения оборудования или ПО из использования;
– время передачи оборудования в ремонт или обращения в службу поддержки;
– контактные данные по гарантийному ремонту или службы поддержки;
– ожидаемый срок окончания профилактики, ремонта, обновления;
– время возвращения оборудования из ремонта или обновления ПО;
– время передачи ПО или оборудования в использование;
– список изменений, если таковые были (обновление версии, замена блоков и др.);
– текущий статус.

Каждая запись дополняется по мере изменения статуса, при этом старые данные остаются. Это дает возможность проследить историю решения проблемы по каждой записи.

Такие данные вносятся вручную техническим персоналом. В некоторых случаях данные о сбое и выводе оборудования из работы могут быть получены автоматически (от системы мониторинга, например), и тогда можно заносить эту информацию соответствующими средствами с обязательным контролем ответственным персоналом.

3.2.6 Компоненты прикладного программного обеспечения

Установка, обновление и поддержка прикладного ПО требуется на каждой суперкомпьютерной системе. Именно поэтому в модели функционирования суперкомпьютерного центра присутствуют эти компоненты. К сожалению, далеко не всегда можно обеспечить корректный автоматический сбор информации о наличии и версиях установленного прикладного программного обеспечения с систем. Поэтому рекомендуется иметь возможность ручного ввода этих данных. Также следует помнить, что некоторые прикладные пакеты могут быть установлены одновременно в разных конфигурациях, например, библиотека FFTW может быть одновременно представлена версиями 2 и 3, которые не совместимы друг с другом.

При наличии технической возможности следует обеспечить сбор статистики по использованию установленного прикладного программного обеспечения. Если такой технической возможности нет, стоит получать эти данные от пользователей через опросы наряду с данными о востребованности тех или иных пакетов для оценки целесообразности их установки. Эта информация помогает понять реальную потребность в установленном ПО, а, следовательно, и потребность в его поддержке и обновлении.

Информация об установленном прикладном программном обеспечении должна включать в себя не только название и версию. Целесообразно указать по каждой системе следующие данные по установленному ПО:
– название и версия ПО;
– контакты по вопросам лицензирования;
– контакты для технической поддержки;
– срок лицензии;
– срок поддержки;
– специальные условия лицензии (ограничения);
– стоимость обновления лицензии/поддержки;
– путь в файловой системе, куда установлен пакет;

– описание процедуры установки и настройки пакета с указание конкретных параметров, использованных на системе;
– описание процедуры тестирования пакета;
– краткая инструкция для пользователей по использованию пакета;
– ответственный за установку и обновление пакета;
– контакты локального технического консультанта по пакету.

В зависимости от условий лицензии, доступ к пакету может быть предоставлен не всем, а только части пользователей. Например, академические лицензии позволяют работать только в рамках образовательных проектов.

Некоторые лицензии могут быть ограничены только сотрудниками отдельной организации. Каждый прикладной пакет должен содержать набор отношений к проектам и организациям, которые имеют право на доступ к нему.

3.2.7 Статистика

Возможность сбора и анализа разного рода статистики являются важной составляющей эффективного управления вычислительным центром. Статистические отчеты могут строиться автоматически и в ручном режиме, а также по уже имеющимся данным в системе либо по результатам опросов пользователей.

Автоматически создаваемые статистические отчеты. Это могут быть данные по числу проектов, пользователей, организаций и других объектов в системе управления и их распределениям по различным критериям. При наличии регулярного импорта статистических данных с подключенных систем, в автоматические отчеты могут быть включены такие данные, как количество запусков программ, число использованных процессорочасов и др., сгруппированных по пользователям, проектам или организациям. Автоматические отчеты целесообразно составлять за регулярные периоды времени, например, сутки или месяц.

Отчеты в ручном режиме могут быть составлены по тем же данным, что и для автоматических, но за произвольный период времени, по специфическим критериям выбора проектов, пользователей или организаций. Это удобно для получения отчетов по отдельным проектам, организациям и их типам, для генерации годовых или промежуточных отчетов и др.

Данные для отчетов можно получать, как упоминалось выше, из имеющихся данных и из импортированных данных от систем. Кроме того, можно получать важную информацию от самих пользователей, используя механизм опросов. Такие данные позволяют оценить уровень удовлетворенности пользователей, их потребности, спланировать дальнейшую работу.

1.3. Перерегистрация и экспертиза

3.3.1 Процедура перерегистрации

Возникновение процедуры перерегистрации как таковой обусловлено следующими предпосылками:

– Требуется поддерживать актуальность представленных пользователями персональных данных и данных по проводимым исследованиям.

– Требуется контроль использования вычислительных ресурсов. Действительно, при безвозмездном предоставлении доступа к ресурсам целесообразно стимулировать эффективное использование суперкомпьютеров.
– Требуется «держать руку на пульсе» реальных потребностей пользователей, оценивая как динамику объемов запрашиваемых ресурсов отдельными проектами, так и потребность в установке прикладного ПО, закупке лицензий, организации обучения и др.
– Вообще говоря, требуется понимание, подкрепленное реальными показателями, подтверждающими, на что именно были потрачены дорогостоящие ресурсы. Вместе с тем, что ресурсы не стоят реальных денег для пользователей сейчас, для держателя системы в лице Университета содержание выливается в очень значимые деньги. Речь идет не только о самих полученных результатах, но и опубликованных книгах, выступления на конференциях, публикациях в высокорейтинговых изданиях, студентах, привлеченных к исследовательскому процессу или прошедших через практикум, и др.

В связи с этим ежегодно в СКЦ МГУ проводится перерегистрация. В ее рамках каждый руководитель проекта представляет краткий иллюстрированный отчет о полученных результатах с использованием вычислительных ресурсов суперкомпьютерного комплекса. Вместе с тем заполняются традиционно два опроса.

«Опрос по проектам» адресован руководителям всех активных проектов для оценки потребностей с точки зрения организации проведения исследований и подтверждения актуальности данных по проектам. Опрос также включает форму запроса ресурсов на следующий учетный период.

«Опрос по пользователям» адресован всем активным пользователям. С одной стороны, пользователи подтверждают корректность указанных в профиле данных, с другой — имеют возможность высказать свои пожелания по работе, указав, например, интересующие темы для обучения или потребность в специфическом ПО.

3.3.2 Экспертиза

Направлений исследований, проводящихся с использованием ресурсов СКЦ, столь много, что вряд ли найдется отдельный эксперт, способный провести достойную оценку всех работ. В рамках системы создан механизм экспертиз. Каждый эксперт может выделить работы, наиболее близкие по направлению исследований, либо работы могут быть распределены по экспертам принудительно. Если при анализе работы эксперт видит, что целесообразно привлечь эксперта смежной области, есть возможность переназначить эксперта. Эксперты могут видеть представленные отчетные материалы и просить, при необходимости, их доработать. Пользователи, в свою очередь, могут доработать и обновить представленную в системе версию отчета. Эксперты остаются анонимными для пользователей. По результатам экспертизы отчеты оцениваются по ряду критериев.

Неудовлетворительные оценки экспертов дают основание о сокращении объема выделенных ресурсов вплоть до полной блокировки доступа.

Информация о лучших работах представляется на конференциях, авторы приглашаются на специализированные семинары, получают определенные льготы, например, при организации выделенных вычислений.

Реализация разработанного программного комплекса управления СКЦ

Вся работа пользователей, администраторов и экспертов организована через единую точку входа и осуществляется через web. Система Octoshell создана на основе программной платформы Ruby On Rails. Работа прототипа программного комплекса организована следующим образом:

- система реализована по модульному принципу и может быть доработана до требуемого функционала уже по специфике организации работ в конкретном СКЦ;
- действия на целевой системе выполняются через модуль асинхронных задач, выполняющихся независимо от основного модуля и не блокирующих его работу;
- через аналогичный модуль осуществляется рассылка уведомлений пользователям;
- на целевой системе устанавливается набор скриптов, позволяющих выполнять необходимые действия:
  - создание и удаление пользователей и групп;
  - добавление и удаление ключей доступа;
  - блокирование и разблокирование доступа пользователей;
- создается пользователь для выполнения команд на удаленной системе, вход для него разрешается только с помощью ключа, который указывается в модуле для выполнения действий; созданному пользователю на системе предоставляются права выполнения установленных скриптов с привилегиями суперпользователя (через пакет sudo).

Таким образом, достигается максимальный уровень защищенности системы — даже похитив ключ доступа, на системе можно выполнить только заданный набор скриптов.

При необходимости набор действий на целевой системе легко расширяется добавлением новых скриптов. Функциональность комплекса может быть относительно просто расширена.

В качестве хранилища данных используется свободная база данных Postgresql, для реализации модуля асинхронных задач — свободный пакет Sidekiq.

Внедренный и работающий прототип комплекса расположен в Сети по адресу https://users.parallel.ru. Исходный текст разработанного комплекса доступен по адресу https://github.com/octoshell/octoshell-v2.

В разработанном прототипе комплекса поддерживается гибкое разделение прав доступа — отдельной группе пользователей могут быть предоставлены возможности, например, для ответов в технической поддержке, проведении экспертизы отчетов и т. д. При этом каждый пользователь может видеть на странице только те элементы, которые ему доступны.

Подключение новой системы в прототип комплекса производится также через web-интерфейс. Предварительно на системе должен быть заведен пользователь для доступа и настроен набор управляющих скриптов.

Для того чтобы иметь возможность управлять прохождением задач пользователей, например, изменять приоритеты, квоты на процессорочасы и др., требуется поддержка со стороны менеджера ресурсов. Не все менеджеры имеют такую поддержку, поэтому для них требуется создание дополнительных модулей или программ. В рамках данного проекта успешно начата работа по созданию такого модуля для популярного менеджера ресурсов SLURM [13].

Распределение функционала в системе реализовано следующим образом. В базовом приложении реализована минимальная функциональность, большинство действий реализовано на уровне модулей (см. рис. 2):

- Базовое приложение:
  - профиль пользователя;
  - формирование группы пользователей по правам доступа;
  - разграничение прав на различные действия.
- CORE — ключевой модуль, отвечающий специфике организации доступа к вычислительным ресурсам, в его реализацию входит:
  - синхронизация данных с вычислительными системами;
  - управление доступом к ресурсам;
  - описание вычислительных систем;
  - запросы пользователей на предоставление ресурсов;
  - механизм поручительств для предоставления доступа;
  - описание проектов;
  - организации;
  - и др.
- AUTHENTICATION: аутентификация пользователей системы Octoshell.
- FACE: пользовательский интерфейс.
- ANNOUNCEMENT: механизм рассылок.
- SUPPORT: техническая поддержка.
- SESSIONS: механизм проведения перерегистраций.
- STATISTICS: построение разного рода статистики.

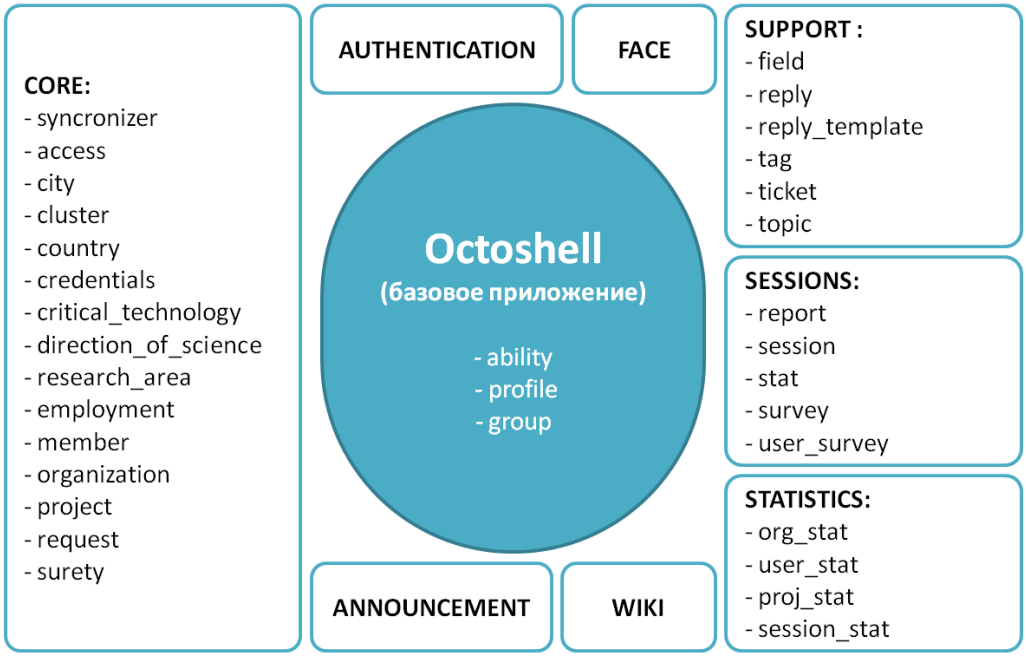

Рис. 2. Модульная структура системы Octoshell

Для самостоятельной установки для скачивания доступна базовая версия, однако, при использовании модели функционирования близкой к СКЦ Московского университета, развертывание системы не потребует серьезных усилий. Дистрибутив устанавливает-

ся на отдельную виртуальную машину, представляя собой, де-факто, решение «из коробки».

Заключение

Помимо уже реализованного функционала, в систему добавляются и новые возможности. Приведем некоторые из них.

В рамках разрабатываемого в данный момент модуля реализуется возможность более гибкого управления приоритетами и квотами на процессорное время, что позволит более полно интегрировать менеджер ресурсов SLURM в разработанную систему и дать возможность администратору выделять отдельным проектам больше (или меньше) ресурсов в зависимости от их приоритетности или ранее полученных результатов непосредственно через интерфейс системы.

С каждой системы, как правило, возможно получение статистической информации об активности пользователей и использовании ресурсов, таких как дисковое пространство, число запущенных задач, процессорочасы. Такие данные позволяют получить информацию о фактическом использовании ресурсов. Такие данные могут быть получены различными способами: от систем мониторинга, с помощью системных скриптов для сбора статистики, с помощью специализированных пакетов. Полученные данные могут быть обработаны с целью соотнесения их с другими объектами системы: пользователями, проектами, организациями.

В ближайшем будущем в личных кабинетах пользователей планируется добавить возможность видеть перечень запускавшихся ими задач. Руководители проектов, соответственно, будут видеть данные о запусках задач всеми участниками проекта.

Следующим шагом станет указание более точных данных по задачам: станут доступны средние показатели использования ресурсов CPU, нагрузки на сеть, интенсивности работы с файловой системой, и другие важные интегральные характеристики [14].

Для отдельных задач на следующем этапе развития модуля станут доступны профили использования ресурсов задачей, откроется возможность изучения поведения ключевых динамических характеристик [15]. Для полноценной реализации данного функционала потребуется провести ряд изменений в используемые методы сбора данных системного мониторинга [16].

Доработанная система мониторинга позволит, в том числе, реализовать модуль ситуационного экрана для системных администраторов, отражая степень мгновенной загрузки ресурсов вычислительного комплекса с историей изменений [17].

Важным функциональным дополнением для администраторов станет интеграция с системой обеспечения автономного функционирования суперкомпьютерного комплекса, получив в рамках единого интерфейса возможность видеть аномалии в поведении компонент СКЦ: аппаратуры, ПО, инфраструктуры [18].

В данный момент Octoshell активно используется в суперкомпьютерном центре Московского университета. Вычислительный парк СКЦ включает такие системы как «Чебышёв» (60 TFlops), «Ломоносов»(1,7 PFlops), «Ломоносов-2» (2,5 PFlops) — всего около 10000 вычислительных узлов. В завершении, приведем немного цифр, отражающих реальные условия использования системы:

– 616 исследовательских активных проектов;
– более чем 350 организаций;
– в Octoshell зарегистрированы и имеют возможность работать 2677 пользователей;

– службой поддержки за год решается около 1000 обращений пользователей.

Полученные результаты в ходе апробации подтверждают целесообразность использования системы. Перспективы системы Octoshell — эксплуатация и дальнейшее расширение функционала в СКЦ МГУ с одной стороны, с другой — как разработчики, так и пользователи системы заинтересованы в использовании данной системы и в других суперкомпьютерных центрах с целью выявления новых направлений дальнейшего развития.

Работа ведется при финансовой поддержке гранта РФФИ, грант N13–07–12206–офи_м.

Литература

1. Топ50 Суперкомпьютеры. URL: http://top50.supercomputers.ru (дата обращения: 02.08.2015).
2. Top500 Supercomputer sites. URL: http://top500.org (дата обращения: 02.08.2015).
3. Воеводин Вл.В., Жуматий С.А., Соболев С.И., Антонов А.С., Брызгалов П.А., Никитенко Д.А., Стефанов К.С., Воеводин Вад.В. Практика суперкомпьютера «Ломоносов» // Открытые системы. 2012. № 7. С. 36–39.
4. Жуматий С.А., Никитенко Д.А. Подход к гибкому управлению суперкомпьютерами // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (Новороссийск, 23–28 сентября 2013 г.). М.: Изд-во МГУ, 2013. С. 296–300.
5. Жуматий С.А., Дацюк О.В. Администрирование суперкомпьютеров и кластерных систем. М.: Изд-во МГУ, 2014. 400 с.
6. Torgue batch system. URL: http://www.adaptivecomputing.com/products/open-source/torque/ (дата обращения: 02.08.2015).
7. SLURM workload manager. URL: http://slurm.schedmd.com/ (дата обращения: 02.08.2015).
8. OpenPBS. URL: http://www.mcs.anl.gov/research/projects/openpbs/ (дата обращения: 02.08.2015).
9. Ganglia Monitoring System. URL: http://ganglia.sourceforge.net/ (дата обращения: 02.08.2015).
10. Zabbix monitoring. URL: http://www.zabbix.com/ru/ (дата обращения: 02.08.2015).
11. Nagios monitoring. URL: https://www.nagios.org/ (дата обращения: 02.08.2015).
12. Open-source Ticket Request System. URL: http://www.otrs.org/ (дата обращения: 02.08.2015).
13. Леоненков С.Н. Расширение функциональности менеджера ресурсов суперкомпьютера SLURM // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (Новороссийск, 22–27 сентября 2014 г.). М.: Изд-во МГУ, 2014. С. 472–476.
14. Никитенко Д.А. Комплексный анализ производительности суперкомпьютерных систем, основанный на данных системного мониторинга // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). 2014. Т. 15. С. 85–97.
15. Антонов А.С., Жуматий С.А., Никитенко Д.А., Стефанов К.С., Теплов А.М., Швец П.А. Исследование динамических характеристик потока задач суперкомпью-

терной системы // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). 2013. Т. 14, № 2. С. 104–108.
16. Стефанов К.С. Система мониторинга производительности суперкомпьютеров // Вестник Пермского Национального исследовательского политехнического университета. Аэрокосмическая техника. 2014. № 39. С. 17–34.
17. Воеводин Вл.В. Ситуационный экран суперкомпьютера // Открытые системы. 2014. № 3. С. 36–39.
18. Антонов А.С., Воеводин Вад В., Даугель-Дауге А.А., Жуматий С.А., Никитенко Д.А., Соболев С.И., Стефанов К.С., Швец П.А. Обеспечение оперативного контроля и эффективной автономной работы Суперкомпьютерного комплекса МГУ // Вестник Южно-Уральского государственного университета. Серия Вычислительная математика и информатика. 2015. Т. 4, № 2. С. 33–43.

DOI: 10.14529/cmse160306

OCTOSHELL: LARGE SUPERCOMPUTER COMPLEX ADMINISTRATION SYSTEM

© 2016 D.A. Nikitenko, V.V. Voevodin, S.A. Zhumatiy

Research Computing Center, Moscow State University (Leninskie Gory 1/4, Moscow, 119991 Russia)

E-mail: voevodin@parallel.ru, serg@parallel.ru, dan@parallel.ru

Received: 20.10.2015

Managing and administering of modern supercomputer centers and HPC systems as a part is a complicated and complex task. The usage of numerous traditional stand-alone tools for administering and management of supercomputers becomes a bottleneck for efficient resource utilization in conditions of growing systems scale. The developed "Octoshell" system for support of running supercomputer centers is aimed at solving this problem. It implements essential tools for administering in a single interface and allows significant automatization of typical management tasks ensuring higher efficiency of large supercomputer complex output as a whole.

Keywords: supercomputer, monitoring, managing HPC center, administering supercomputers, user support.

FOR CITATION

Nikitenko D.A., Voevodin V.V., Zhumatiy S.A. Octoshell: Large Supercomputer Complex Administration System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 76–95. (in Russian) DOI: 10.14529/cmse160306.

References

1. *Top50 superkomputery* [Top50 Supercomputers]. Available at: http://top50.supercomputers.ru (accessed: 02.08.2015).
2. *Top500 Supercomputer Sites*. Available at: http://top500.org (accessed: 02.08.2015).
3. Voevodin V.V., Antonov A.S., Bryzgalov P.A., Nikitenko D.A. , Sobolev S.I., Stefanov K.S., Voevodin Vad.V. , Zhumatij S.A. Practice of «Lomonosov» Supercomputer. *Otkrytye sistemy* [Open Systems]. 2012. no. 7. pp. 36–39. (in Russian)
4. Zhumatiy S.A., Nikitenko D.A. Flexible Approach to the Management of Supercomputers. *Nauchnyj servis v seti Internet: vse grani parallelizma: Trudy Mezhdunarodnoj superkomp'juternoj konferencii* [Scientific Service over Internet: All Shades of Parallel-

ism: Proceedings of the International Supercomputing Conference] (Novorossiisk, 23–28 September 2013). Moscow, Publishing of the MSU, 2013. pp. 296–300. (in Russian)

5. Zhumatiy S.A., Datsyuk O.V. *Administrirovanie superkomp'juterov i klasternyh sistem* [Administering of Supercomputers and Cluster Systems]. Moscow, Publishing of the MSU, 2014. 400 p.
6. *Torgue Batch System*. Available at: http://www.adaptivecomputing.com/products/open-source/torque/ (accessed: 02.08.2015).
7. *SLURM Workload Manager*. Available at: http://slurm.schedmd.com/ (accessed: 02.08.2015).
8. *OpenPBS*. Available at: http://www.mcs.anl.gov/research/projects/openpbs/ (accessed: 02.08.2015).
9. *Ganglia Monitoring System*. Available at: http://ganglia.sourceforge.net/ (accessed: 02.08.2015).
10. *Zabbix Monitoring*. Available at: http://www.zabbix.com/ru/ (accessed: 02.08.2015).
11. *Nagios Monitoring*. Available at: https://www.nagios.org/ (accessed: 02.08.2015).
12. *Open-source Ticket Request System*. Available at: http://www.otrs.org/ (accessed: 02.08.2015).
13. Leonenkov S.N. Extending Functionality of SLURM Supercomputer Resource Manager. *Nauchnyj servis v seti Internet: mnogoobrazie superkomp'juternyh mirov: Trudy Mezhdunarodnoj superkomp'juternoj konferencii* [Scientific Service over Internet: Diversity of Supercomputing Worlds: Proceedings of the International Supercomputing Conference] (Novorossiisk, 22–27 September 2014). Moscow, Publishing of the MSU, 2014. pp. 472–476. (in Russian)
14. Nikitenko D.A. Complex Analysis of Supercomputer Systems' Performance Based on System Monitoring Data. *Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tehnologii* [Computational Methods and Programming: New Computational Technologies]. 2014. vol. 15. pp. 85–97. (in Russian)
15. Antonov A.S., Zhumatiy S.A., Nikitenko D.A., Stefanov K.S., Teplov A.M., Shvets P.A. Dynamic Characteristics Analysis of Jobs Sequence on Supercomputer System. *Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tehnologii* [Computational Methods and Programming: New Computational Technologies]. 2013. vol. 14, no. 2. pp. 104–108. (in Russian)
16. Stefanov K.S. Supercomputer Performance Monitoring System. *Vestnik Permskogo Nacional'nogo issledovatel'skogo politehnicheskogo universiteta. Ajerokosmicheskaja tehnika* [Bulletin of Perm National Research Polytechnical University. Aerospace Technics]. 2014. no. 39. pp. 17–34. (in Russian)
17. Voevodin V.V. Situational Screen of Supercomputer. *Otkrytye sistemy* [Open Systems]. 2014. no. 3. pp. 36–39. (in Russian)
18. Antonov A.S., Voevodin Vad.V., Daugel-Dauge A.A., Zhumatiy S.A., Nikitenko D.A., Sobolev S.I., Stefanov K.S., Shvets P.A. Securing of Active Control and Efficient Autonomous Operating of MSU Supercomputer Center. *Vestnik Juzhno-Ural'skogo gosudarstvennogo universiteta. Serija Vychisli-tel'naja matematika i informatika* [Bulletin of South Ural State University. Series: Computational Mathematics and Informatics]. 2015. vol. 4, no 2. pp. 33–43. (in Russian) DOI: 10.14529/cmse150203.

УДК 519.67 DOI: 10.14529/cmse160307

ОЦЕНКА ЛОКАЛЬНОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ, РЕАЛИЗУЕМЫХ НА ГРАФИЧЕСКИХ ПРОЦЕССОРАХ*

© 2016 г. Н.А. Лиходед, М.А. Полещук
Белорусский государственный университет
(220030 Республика Беларусь, Минск, пр. Независимости, д. 4)
E-mail: likhoded@bsu.by, poleschuma@bsu.by

Поступила в редакцию: 02.03.2016

Исследуется задача получения блоков операций и потоков операций параллельного алгоритма, приводящих к меньшему числу обращений к глобальной памяти и к эффективному использованию параллельными потоками вычислений кэшей и разделяемой памяти графического процессора. Сформулированы и доказаны утверждения, позволяющие оценить объем коммуникационных операций, порождаемых альтернативными вариантами задания размеров блоков вычислений, а также минимизировать число промахов кэша за счет использования временной и пространственной локальности данных с учетом размера и длины строк кэша. Исследования конструктивны и допускают программную реализацию для практического использования.

Ключевые слова: параллельные вычисления, графический процессор, минимизация объема коммуникационных операций, временная локальность, пространственная локальность.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Лиходед Н.А., Полещук М.А. Оценка локальности параллельных алгоритмов, реализуемых на графических процессорах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2016. Т. 5, № 3. С. 96–111. DOI: 10.14529/cmse160307.

Введение

Время решения задачи на современном компьютере во многом определяется локальностью реализуемого алгоритма — степенью использования памяти с быстрым доступом. В этой работе в качестве компьютера, на котором требуется реализовать параллельную версию алгоритма, будем рассматривать графические процессоры (GPU) — мощные многоядерные вычислительные устройства. При вычислениях на GPU быстрым является процесс обращения к разделяемой памяти мультипроцессора и к кэшам, но не обращение к глобальной памяти GPU.

Графический процессор выполняет множество параллельных потоков вычислений. Потоки объединяются в блоки, а блоки объединяются в сетку. Синхронизация между потоками разных блоков не предусмотрена. Каждый блок потоков выполняется атомарно на одном из мультипроцессоров графического процессора. Должны быть указаны блоки, которые могут выполняться мультипроцессорами одновременно и независимо друг от друга.

Целью этой работы является разработка метода, позволяющего получать блоки вычислений с меньшим числом обращений к глобальной памяти. Используется анализ информационных зависимостей, порождающих коммуникационные операции. Сформулиро-

* Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии — 2016».

ваны и доказаны утверждения, позволяющие ранжировать параметры размера блоков на основе асимптотических оценок объема коммуникационных операций. В работе [1] подобные оценки получены с использованием более сложного (но и более приспособленного для автоматизации) математического аппарата. Известен способ получения более точной оценки количества элементов, к которым осуществляется доступ при выполнении операций блока вычислений, но практическое использование известного результата довольно трудоемкое и требует привлечения специализированных средств автоматизации [2, 3].

В работе также указаны условия наличия временной локальности и условия наличия пространственной локальности данных. Существование локальности позволяет эффективно использовать параллельными потоками вычислений кэши и разделяемую память. Исследование проводится аналитическими методами. Другой способ исследования локальности данных — построение профиля работы приложения с памятью [4].

Статья организована следующим образом. В разделе 1 приведены необходимые для дальнейшего изложения сведения о рассматриваемом классе алгоритмов, информационных зависимостях между операциями, получении блоков макроопераций. В разделе 2 оценивается объем коммуникационных операций чтения и операций записи, порождаемых разбиением итераций алгоритма на блоки вычислений, приводится пример установления соотношения размеров блоков для минимизации объема коммуникационных операций. В разделе 3 даются рекомендации по выбору размеров потоков вычислений для достижения наименьшего числа кэш-промахов и организации совместного использования потоками данных. В разделе 4 обсуждаются результаты вычислительных экспериментов. В заключении суммируются основные результаты, намечаются направления дальнейших исследований.

1. Предварительные сведения

Будем считать, что алгоритм задан последовательной программой линейного класса[1] [5]. Основную вычислительную часть такой программы составляют циклические конструкции; границы изменения параметров циклов задаются неоднородными формами, линейными по совокупности параметров циклов и внешних переменных. Предполагается, что в гнезде циклов имеется K выполняемых операторов S_β и используется L массивов a_l размерностей ν_l. Область изменения параметров циклов (область итераций) для оператора S_β и размерность этой области обозначим соответственно V_β и n_β.

Вхождением (l,β,q) будем называть q-е вхождение массива a_l в оператор S_β. Индексы элементов l-го массива, связанных с вхождением (l,β,q), выражаются функцией $\overline{F}_{l,\beta,q}$ вида

$$\overline{F}_{l,\beta,q}(J) = F_{l,\beta,q}J + G_{l,\beta,q}N + f^{l,\beta,q},$$

$$J(j_1,...,j_{n_\beta}) \in V_\beta,\ N \in Z^s,\ F_{l,\beta,q} \in Z^{\nu_l \times n_\beta},\ G_{l,\beta,q} \in Z^{\nu_l \times s},\ f^{l,\beta,q} \in Z^{\nu_l},$$

где $N \in Z^s$ — вектор внешних переменных алгоритма, s — число внешних переменных.

[1] В литературе используются также термины «аффинные гнезда циклов», «алгоритмы с аффинными зависимостями».

Выполнение оператора S_β при конкретных значениях β и вектора параметров циклов J будем называть операцией и обозначать $S_\beta(J)$. Пара вхождений $(l,\alpha,1)$ и (l,β,q) порождает истинную зависимость $S_\alpha(I)\to S_\beta(J)$, если: $S_\alpha(I)$ выполняется раньше $S_\beta(J)$; $S_\alpha(I)$ переопределяет (изменяет) элемент массива a_l, а $S_\beta(J)$ использует на вхождении (l,β,q) в качестве аргумента тот же элемент массива; между операциями $S_\alpha(I)$ и $S_\beta(J)$ этот элемент не переопределяется.

Зависимости (информационные связи) $S_\alpha(I)\to S_\beta(J)$ между операциями будем характеризовать векторами $d^{\alpha,\beta}=J-I$. Если размерности векторов J и I не совпадают, то $J-I$ определяется как разность векторов меньшей из размерностей. Векторы $d^{\alpha,\beta}$ часто называют векторами зависимостей: $d^{\alpha,\beta}$ позволяет для операции $S_\beta(J)$ найти операцию $S_\alpha(I)$, от которой $S_\beta(J)$ зависит.

Пусть в гнезде циклов имеется Θ наборов выполняемых операторов. Под набором операторов будем понимать один или несколько операторов, окруженных одним и тем же множеством циклов. Операторы и наборы операторов линейно упорядочены расположением их в записи алгоритма. Обозначим: V^ϑ, $1\le\vartheta\le\Theta$, — области изменения параметров циклов, окружающих наборы операторов, n^ϑ — размерность области V^ϑ, число циклов, окружающих ϑ-й набор операторов. Будем считать, что если оператор S_β принадлежит набору операторов с номером ϑ^β, то $n^\vartheta=n_\beta$.

Выделим блоки вычислений путем разбиения циклов, окружающих операторы. Пусть $m_\zeta^\vartheta=\min\limits_{J\left(j_1,j_2,..,j_{n^\vartheta}\right)\in V^\vartheta} j_\zeta$, $M_\zeta^\vartheta=\max\limits_{J\left(j_1,j_2,..,j_{n^\vartheta}\right)\in V^\vartheta} j_\zeta$, $1\le\zeta\le n^\vartheta$, — предельные значения изменения параметров циклов. Зададим размеры блоков вычислений натуральными числами $r_1^\vartheta,...,r_{n^\vartheta}^\vartheta$. Параметр r_ζ^ϑ обозначает число значений параметра j_ζ, приходящихся на один блок ϑ-го набора операторов. Число частей Q_ζ^ϑ, на которые при формировании блоков разбивается область значений параметра j_ζ цикла, окружающего ϑ-й набор операторов, находится согласно $Q_\zeta^\vartheta=\left\lceil (M_\zeta^\vartheta-m_\zeta^\vartheta+1)/r_\zeta^\vartheta\right\rceil$ ($\lceil\cdot\rceil$ обозначает ближайшее «сверху» целое число). Нумеровать блоки вычислений будем по каждой координате в пределах от 0 до $Q_\zeta^\vartheta-1$, $1\le\zeta\le n^\vartheta$.

Формализованным способом разбить множество операций алгоритма на блоки и потоки вычислений можно путем применения тайлинга — преобразования алгоритма для получения макроопераций [6, 7].

Пример 1. Рассмотрим алгоритм Флойда—Уоршелла поиска кратчайших путей между всеми парами вершин графа (рис. 1).

```
do k = 1, n
  do i = 1, n
    do j = 1, n
      a(i,j) = min(a(i,j), a(i,k) + a(k,j))
    enddo
  enddo
enddo
```

Рис. 1. Основная часть алгоритма Флойда—Уоршелла

В гнезде циклов имеется один выполняемый оператор S_1 (один набор операторов) и используется один массив a размерности 2. Область изменения параметров циклов (область итераций) $V_1=V^1=\{(k,i,j)\in \mathrm{Z}^3 |\ 1\le k\le n,\ 1\le i\le n,\ 1\le j\le n\}$ для оператора S_1 имеет размерность 3. Для матриц $F_{l,\beta,q}$ на вхождениях (l,β,q) имеем:

$$F_{1,1,1}=F_{1,1,2}=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix},\quad F_{1,1,3}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\end{pmatrix},\quad F_{1,1,4}=\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}.$$

В рассматриваемом примере векторы зависимостей $d^{\alpha,\beta}$ будем для наглядности помечать не номерами операторов, а элементами массивов, фигурирующими на порождающих зависимости вхождениях. Например, вектор $d^{a(i,j),a(i,k)}$ порождается зависимостью между данными $a(i,j)$ на вхождении $(l,\alpha,1)=(1,1,1)$ в левой части оператора S_1, и данными $a(i,k)$ на вхождении $(l,\beta,q)=(1,1,3)$ в правой части оператора. Укажем итерации, порождающие зависимости, и векторы зависимостей:

- $S_1(k–1,i,j)\to S_1(k,i,j)$: данное $a(i,j)$, вычисленное на итерации $(k–1,i,j)$, является аргументом $a(i,j)$ для вычислений на итерации (k,i,j); $d^{a(i,j),a(i,j)}=(1,0,0)$;
- $S_1(k–1,i,k)\to S_1(k,i,j)$: $a(i,k)$, вычисленное на итерации $(k–1,i,k)$, является аргументом для вычислений на итерации (k,i,j); $d^{a(i,j),a(i,k)}=(1,0,j–k)$;
- $S_1(k–1,k,j)\to S_1(k,i,j)$: $a(k,j)$, вычисленное на итерации $(k–1,k,j)$, является аргументом для вычислений на итерации (k,i,j); $d^{a(i,j),a(k,j)}=(1,i–k,0)$.

Отметим, что для фиксированного k все операции не зависят друг от друга. Кроме того, непосредственно из записи алгоритма следует, что данные $a(i,j)$ не обновляются, если $i=k$ или $j=k$.

Выделим блоки вычислений. Напомним, блоки вычислений должны выполняться атомарно, независимо друг от друга. С учетом зависимостей алгоритма, такие блоки проще всего задать, если положить $r_1=1$, r_2 и r_3 — параметры. Всего будет $Q_1\times Q_2\times Q_3$ двумерных (2D) блоков, где $Q_1=n$, $Q_2=\left\lceil\frac{n}{r_2}\right\rceil$, $Q_3=\left\lceil\frac{n}{r_3}\right\rceil$. Блочный алгоритм имеет следующий вид:

```
do k = 1, n
  do i^bl = 0, Q_2 - 1
    do j^bl = 0, Q_3 - 1
      do i = 1 + i^bl*r_2, min((i^bl + 1)*r_2, n)
        do j = 1 + j^bl*r_3, min((j^bl + 1)*r_3, n)
          a(i,j) = min(a(i,j), a(i,k) + a(k,j))
        enddo
      enddo
    enddo
  enddo
enddo
```

Рис. 2. Основная часть алгоритма Флойда—Уоршелла с выделенными 2D блоками

2. Оценка объема коммуникационных операций

2.1. Выражения для оценки объема операций чтения и записи

Обозначим $M^\vartheta=\max\limits_\zeta\left(M_\zeta^\vartheta-m_\zeta^\vartheta\right)+1$ — наибольшее число итераций циклов, участвующих в получении блоков. Для простоты записи будем использовать обозначение M без

индекса ϑ и подразумевать набор операторов ϑ^{β} при упоминании оператора S_{β}. Отметим, что величина $Q_{\zeta}^{\vartheta} r_{\zeta}^{\vartheta}$ имеет порядок M, поэтому r_{ζ}^{ϑ}, Q_{ζ}^{ϑ} могут быть величинами порядка M.

Определим термин «фиксированное данное массива» как конкретное, неизмененное содержимое соответствующей ячейки памяти. Введем в рассмотрение величины $\rho_{l,\beta,q}^{d}$, $\rho_{l,\beta,q}^{d,\zeta}$, которые определим следующим образом: пусть число фиксированных данных, используемых на вхождении (l,β,q), оценивается величиной $O(M^{\rho_{l,\beta,q}^{d}})$, а число фиксированных данных, используемых на вхождении (l,β,q) при фиксированном значении цикла с параметром j_{ζ} оценивается величиной $O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})$. Наличие индекса d в обозначениях подчеркивает, что если вхождение (l,β,q) порождает истинную зависимость $S_{\alpha}(I) \to S_{\beta}(J)$, то $\rho_{l,\beta,q}^{d}$ и $\rho_{l,\beta,q}^{d,\zeta}$ зависят от вектора $d^{\alpha,\beta}$. Найти $\rho_{l,\beta,q}^{d}$ и $\rho_{l,\beta,q}^{d,\zeta}$ не представляет труда, если детально понимать реализуемый алгоритм. Формализованный способ получения величин $\rho_{l,\beta,q}^{d}$, $\rho_{l,\beta,q}^{d,\zeta}$ изложен в работе [1]. Этот способ использует функции $\overline{F}_{l,\beta,q}$, а также функции, описывающие информационную структуру алгоритма — покрывающие функции графа алгоритма [5] (называемые еще h-преобразованиями [8]).

Отметим, что число фиксированных данных, используемых (в пространстве размерности n_{β}–1) на вхождении при фиксированном значении цикла с параметром j_{ζ}, по порядку либо равно ($\rho_{l,\beta,q}^{d,\zeta}$ –1=$\rho_{l,\beta,q}^{d}$), либо на 1 меньше ($\rho_{l,\beta,q}^{d,\zeta}$ –1=$\rho_{l,\beta,q}^{d}$ –1) числа фиксированных данных, используемых (в пространстве размерности n_{β}) при всех значениях цикла с параметром j_{ζ}.

Обозначим через $d_{\zeta}^{\alpha,\beta,\max}$ максимальные по модулю значения компонент векторов $d^{\alpha,\beta}$. Сделаем естественное для практики предположение: компоненты векторов $d^{\alpha,\beta}$ по модулю или не превосходят нескольких единиц, или неограниченно возрастают с ростом размера задачи. В первом случае $d_{\zeta}^{\alpha,\beta,\max}$ также не превосходят нескольких единиц, во втором случае являются большими, значительно превосходящими r_{ζ}^{ϑ}.

Теорема 1. Пусть вхождение (l,β,q) порождает истинную зависимость: определение некоторого данного происходит на вхождении $(l,\alpha,1)$ в левой части оператора S_{α}, а использование — на вхождении (l,β,q) в правой части оператора S_{β}, причем в окружении операторов S_{α} и S_{β} имеется цикл с параметром j_{ζ}. Тогда, при получении блоков вычислений для реализации алгоритма на графическом процессоре, разбиение итераций цикла j_{ζ} порождает коммуникационные операции чтения и записи, объем которых имеет следующие оценки:

- $O(Q_{\zeta}^{\vartheta} M^{\rho_{l,\beta,q}^{d}-1})$ операций чтения и операций записи, если $0 < d_{\zeta}^{\alpha,\beta,\max} < r_{\zeta}^{\vartheta}$;
- $O(Q_{\zeta}^{\vartheta} M^{\rho_{l,\beta,q}^{d}})$ операций чтения и $O(M^{\rho_{l,\beta,q}^{d}})$ операций записи, если $d_{\zeta}^{\alpha,\beta,\max} \ge r_{\zeta}^{\vartheta}$, $\rho_{l,\beta,q}^{d,\zeta} \ne \rho_{l,\beta,q}^{d}$;
- $O(M^{\rho_{l,\beta,q}^{d}})$ операций чтения и операций записи, если $d_{\zeta}^{\alpha,\beta,\max} \ge r_{\zeta}^{\vartheta}$, $\rho_{l,\beta,q}^{d,\zeta} = \rho_{l,\beta,q}^{d}$;

- не требуется ни операций чтения, ни операций записи, если $d_\zeta^{\alpha,\beta,\max}=0$.

В случае, когда вхождение (l,β,q) не порождает истинной зависимости (происходит обращение к входным данным) или цикл с параметром j_ζ имеется в окружении только оператора S_β, оценки следующие:

- $O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^d})$ операций чтения, если $\rho_{l,\beta,q}^{d,\zeta}\neq\rho_{l,\beta,q}^d$;
- $O(M^{\rho_{l,\beta,q}^d})$ операций чтения, если $\rho_{l,\beta,q}^{d,\zeta}=\rho_{l,\beta,q}^d$.

Доказательство. Утверждения теоремы оценивают объем коммуникационных операций, порождаемых разбиением итераций цикла j_ζ, в зависимости от значений ζ-й координаты вектора $d^{\alpha,\beta}$ и величин $\rho_{l,\beta,q}^{d,\zeta}$, $\rho_{l,\beta,q}^d$. Напомним, множество итераций цикла с параметром j_ζ разбивается на Q_ζ^ϑ частей, каждая часть имеет некоторый номер j_ζ^{gl}, $0\le j_\zeta^{gl}\le Q_\zeta^\vartheta-1$, и содержит r_ζ^ϑ итераций (кроме, возможно, части с номером $Q_\zeta^\vartheta-1$).

Пусть $0<d_\zeta^{\alpha,\beta,\max}<r_\zeta^\vartheta$, определение данного происходит при выполнении операции $S_\alpha(I(i_1,...,i_{n_\alpha}))$, а использование — при выполнении операции $S_\beta(J(j_1,...,j_{n_\beta}))$.

Число фиксированных данных, используемых на вхождении (l,β,q) при фиксированном значении цикла с параметром j_ζ (т.е. в пространстве размерности n_β–1) оценивается величиной $O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})$. Тогда число фиксированных данных, используемых на вхождении (l,β,q) при всех значениях цикла с параметром j_ζ (в пространстве размерности n_β) оценивается величиной $\frac{M}{d_\zeta^{\alpha,\beta,\max}}O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})=O(M^{\rho_{l,\beta,q}^{d,\zeta}})$ (напомним, величина $d_\zeta^{\alpha,\beta,\max}$ не превышает нескольких единиц). С другой стороны, оценка числа таких данных есть $O(M^{\rho_{l,\beta,q}^d})$. Таким образом, в рассматриваемом случае верно $\rho_{l,\beta,q}^{d,\zeta}=\rho_{l,\beta,q}^d$.

В каждой из Q_ζ^ϑ частей итераций цикла j_ζ число фиксированных данных, которые определяются при одном значении j_ζ^{gl}, но используются при другом, оценивается величиной $d_\zeta^{\alpha,\beta,\max}O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})$, причем независимо от значения r_ζ^ϑ величина $d_\zeta^{\alpha,\beta,\max}$, как было условлено, не превышает нескольких единиц. Суммарный объем коммуникационных операций чтения и операций записи по всем Q_ζ^ϑ частям определяется оценкой $Q_\zeta^\vartheta O(M^{\rho_{l,\beta,q}^{d,\zeta}-1})=O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^d-1})$.

Пусть $d_\zeta^{\alpha,\beta,\max}\ge r_\zeta^\vartheta$. Если $\rho_{l,\beta,q}^{d,\zeta}\neq\rho_{l,\beta,q}^d$ (т.е. $\rho_{l,\beta,q}^{d,\zeta}-1=\rho_{l,\beta,q}^d$), то число используемых на вхождении (l,β,q) фиксированных данных в каждой части и во всех Q_ζ^ϑ частях оценивается одинаковой величиной $O(M^{\rho_{l,\beta,q}^d})$. Этой величиной следует оценить объем коммуникационных операций записи. Оценка объема коммуникационных операций чтения по всем Q_ζ^ϑ частям есть $Q_\zeta^\vartheta O(M^{\rho_{l,\beta,q}^d})=O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^d})$. Если $\rho_{l,\beta,q}^{d,\zeta}=\rho_{l,\beta,q}^d$, то число используемых фиксированных данных в каждой из Q_ζ^ϑ частей оценивается величиной $O(r_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d,\zeta}-1})$. Суммарный объем коммуникационных операций чтения и операций запи-

си по всем Q_ζ^ϑ частям определяется оценкой $Q_\zeta^\vartheta O(r_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d,\zeta}-1}) = O(Q_\zeta^\vartheta r_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d}-1}) = O(M^{\rho_{l,\beta,q}^{d}})$.

Если $d_\zeta^{\alpha,\beta,\max}=0$, то любое фиксированное данное определяется и используется при одном и том же значении параметра цикла j_ζ^{gl}. Коммуникационных операций не требуется.

Пусть вхождение (l,β,q) не порождает истинной зависимости (происходит обращение к входным данным) или порождает истинную зависимость, но цикл с параметром j_ζ имеется в окружении только оператора S_β. Если $\rho_{l,\beta,q}^{d,\zeta} \neq \rho_{l,\beta,q}^{d}$ (т.е. $\rho_{l,\beta,q}^{d,\zeta} - 1 = \rho_{l,\beta,q}^{d}$), то объем коммуникационных операций (только чтение) по всем Q_ζ^ϑ частям оценивается величиной $Q_\zeta^\vartheta O(M^{\rho_{l,\beta,q}^{d}}) = O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d}})$. Если $\rho_{l,\beta,q}^{d,\zeta} = \rho_{l,\beta,q}^{d}$, то объем коммуникационных операций (только чтение) по всем Q_ζ^ϑ частям определяется оценкой $Q_\zeta^\vartheta O(r_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d,\zeta}-1}) = O(M^{\rho_{l,\beta,q}^{d}})$.

□

Доказательство теоремы для случая, когда информационная структура алгоритма представлена покрывающими функциями графа алгоритма, приведено в работе [1].

Пример 1 (продолжение). Напомним векторы зависимостей, порождаемые вхождениями массива a в правую часть оператора, и укажем оценки числа фиксированных данных, используемых на вхождениях.

Для второго вхождения $a(i,j)$ (использование прежнего значения обновляемого элемента) $d^{a(i,j),a(i,j)}=(1,0,0)$; $\rho_{1,1,2}^{d}=3$ — число фиксированных данных, используемых на вхождении, оценивается величиной $O(n^3)$; $\rho_{1,1,2}^{d,1}=\rho_{1,1,2}^{d,2}=\rho_{1,1,2}^{d,3}=3$ — число фиксированных данных, используемых на вхождении при фиксированном значении одного из циклов с параметрами $j_1=i$, $j_2=j$, $j_3=k$, оценивается величинами $O(M^{\rho_{1,1,2}^{d,\zeta}-1}) = O(n^2)$.

Для третьего вхождения $a(i,k)$ (использование при фиксированном k на итерациях (i,j,k) элементов одного столбца) $d^{a(i,j),a(i,k)}=(1,0,j-k)$, $\rho_{1,1,3}^{d}=2$, $\rho_{1,1,3}^{d,1}=2$, $\rho_{1,1,3}^{d,2}=2$, $\rho_{1,1,3}^{d,3}=3$.

Для четвертого вхождения $a(k,j)$ (использование при фиксированном k на итерациях (i,j,k) элементов одной строки) $d^{a(i,j),a(k,j)}=(1,i-k,0)$, $\rho_{1,1,4}^{d}=2$, $\rho_{1,1,4}^{d,1}=2$, $\rho_{1,1,4}^{d,2}=3$, $\rho_{1,1,4}^{d,3}=2$.

Оценим, используя теорему 1, объем коммуникационных операций, порождаемых разбиением итераций циклов.

Пусть $\zeta=1$. Тогда для всех вхождений $d_\zeta^{\alpha,\beta,\max} = d_\zeta^{\alpha,\beta}=1$. Имеет место случай $0 < d_\zeta^{\alpha,\beta,\max} < r_\zeta^\vartheta$, требуется $O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d}-1})$ операций чтения и операций записи: для второго вхождения $O(Q_1 n^2)$ операций, для третьего и четвертого вхождений — $O(Q_1 n)$ операций.

Пусть $\zeta=2$. Для второго и третьего вхождений $d_\zeta^{\alpha,\beta,\max} = d_\zeta^{\alpha,\beta}=0$, поэтому не требуется ни операций чтения, ни операций записи. Для четвертого вхождения (случай $d_\zeta^{\alpha,\beta,\max} \geq r_\zeta^\vartheta$, $\rho_{l,\beta,q}^{d,\zeta} \neq \rho_{l,\beta,q}^{d}$, $(l,\beta,q)=(1,1,4)$) требуется $O(Q_\zeta^\vartheta M^{\rho_{l,\beta,q}^{d}}) = O(Q_2 n^2)$ операций чтения и $O(M^{\rho_{l,\beta,q}^{d}}) = O(n^2)$ операций записи.

Пусть ζ=3. Для второго и четвертого вхождений $d_\zeta^{\alpha,\beta,\max} = d_\zeta^{\alpha,\beta} = 0$, не требуется ни операций чтения, ни операций записи. Для третьего вхождения (случай $d_\zeta^{\alpha,\beta,\max} \geq r_\zeta^\vartheta$, $\rho_{l,\beta,q}^{d,\zeta} \neq \rho_{l,\beta,q}^{d}$, $(l,\beta,q)=(1,1,3)$) требуется $O(Q_3 n^2)$ операций чтения и $O(n^2)$ операций записи.

2.2. Ранжирование размеров блоков вычислений

Утверждения теоремы 1 позволяют найти асимптотику суммарного объема коммуникационных операций, порождаемых разбиением множества итераций j_ζ на Q_ζ^ϑ частей, и, следовательно, дают возможность ранжировать параметры размера блоков вычислений для минимизации объема коммуникационных операций. При ранжировании (установлении соотношений размеров относительно друг друга) параметров размера блоков вычислений параллельного алгоритма, реализуемого на графическом процессоре, следует в оценках объема коммуникационных операций принимать во внимание только оценки, содержащие множитель Q_ζ^ϑ; следует также учитывать, что величины r_ζ^ϑ и Q_ζ^ϑ связаны обратно пропорциональной зависимостью.

Пример 1 (продолжение). Для ранжирования параметров r_1, r_2 и r_3 размера блоков вычислений, выполняемых на графическом процессоре, укажем, следуя результатам предыдущего подраздела, оценки объема коммуникационных операций, содержащие множитель Q_ζ^ϑ — число частей, на которые при формировании блоков разбивается область значений параметра j_ζ.

Если ζ=1, то требуется $O(Q_1 n^2)$ операций чтения и операций записи для второго вхождения и $O(Q_1 n)$ операций чтения и операций записи для третьего и четвертого вхождений. Если ζ=2, то требуется $O(Q_2 n^2)$ операций чтения для четвертого вхождения. Если ζ=3, то требуется $O(Q_3 n^2)$ операций чтения для третьего вхождения.

Таким образом, сравнивая оценки коммуникационных издержек и по чтению, и по записи, приходим к выводу, что параметр размера блока вычислений r_1 увеличивать более выгодно, чем параметры r_2 и r_3. В то же время, наиболее простой блочный алгоритм Флойда—Уоршелла (рис. 2) содержит 2D блоки, для которых r_1=1. Блочный алгоритм с 3D блоками (рис. 3), для которых $r_1=r_2=r_3=r>1$ получен в работе [9]. Представляет также интерес разработка блочного алгоритма с 3D блоками, для которых $r_1>r_2$, $r_1>r_3$.

```
do k = 1 + k^bl*r, min((k^bl + 1)*r, n)
  do i = 1 + i^bl*r, min((i^bl + 1)*r, n)
    do j = 1 + j^bl*r, min((j^bl + 1)*r, n)
      a(i,j) = min(a(i,j), a(i,k) + a(k,j))
    enddo
  enddo
enddo
```

Рис. 3. 3D блоки блочного алгоритма Флойда—Уоршелла

3. Условия наличия локальности данных

Потоки одного блока выполняются на мультипроцессоре частями-пулами, называемыми варп. Варп содержит до 32 потоков (два полуварпа по 16 потоков). Если несколько потоков одного полуварпа обращаются к одному и тому же банку разделяемой памяти, константной памяти или кэша текстурной памяти для доступа к различным данным, то происходит конфликт. Но конфликта не происходит, если несколько потоков одного полуварпа обращаются к одному и тому же данному (broadcast). В этом случае запрашиваемое данное передается только один раз, поэтому трафик сокращается в 16 раз (для архитектуры Fermi в 32 раза, так как объединение запросов в память происходит для 32 потоков).

Текстурная память может быть использована эффективно, если каждый поток полуварпа запрашивает близко расположенные в памяти данные, то есть если доступ к памяти характеризуется пространственной локальностью. Наличие пространственной локальности позволяет эффективно использовать кэш текстурной памяти и разделяемую память параллельными потоками.

Зададим в блоках вычислений потоки вычислений посредством выделения блоков второго уровня. Размеры блоков второго уровня зададим натуральными числами $r_{1,2}$, $r_{2,2}$, ..., $r_{n_\beta,2}$. Параметр $r_{\xi,2}$ обозначает число значений параметра j_ξ, приходящихся на один поток (на один блок второго уровня). Обозначим через $Q_{\xi,2}$ число частей, на которые при формировании потоков разбивается область значений цикла с параметром j_ξ, приходящихся на один блок; $Q_{\xi,2}=\lceil r_\xi / r_{\xi,2} \rceil$. Выбор размеров $r_{\xi,2}$, оптимальных по числу кэш-промахов, определяется в значительной степени наличием временной локальности и пространственной локальности данных.

Введем обозначения:

$F_{l,\beta,q,\xi}$ — столбец с номером ξ матрицы $F_{l,\beta,q}$;

$\xi^{l,\beta,q}$ — номер ненулевого столбца матрицы $F_{l,\beta,q}$, правее которого находятся только нулевые столбцы матрицы; если самый правый столбец F_{l,β,q,n_β} не нулевой, то по определению $\xi^{l,\beta,q}=n_\beta$;

$\Xi^{l,\beta,q}$ — множество номеров линейно независимых столбцов матрицы $F_{l,\beta,q}$ (если таких множеств более одного, то зафиксировать любое);

$F^{\Xi}_{l,\beta,q}$ — матрица, составленная из столбцов матрицы $F_{l,\beta,q}$ с номерами из множества $\Xi^{l,\beta,q}$; матрица $F^{\Xi}_{l,\beta,q}$ имеет размеры $\nu_l \times |\Xi^{l,\beta,q}|$, где ν_l — размерность массива a_l, $|\Xi^{l,\beta,q}|$ — число элементов множества $\Xi^{l,\beta,q}$;

L_l — число элементов массива a_l, помещающихся в строку кэша; предполагается, что строка массива помещается в кэш.

Лемма 1. Пусть для вхождения (l,β,q) массива a_l в правую часть оператора S_β для всех $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$, выполнены условия $r_{\xi,2}=1$ (условие накладывается при наличии указанных ξ) и $\nu_l \ge |\Xi^{l,\beta,q}|$. Тогда в каждом потоке вычислений на разных итерациях циклов с параметрами j_ξ, $1 \le \xi \le \xi^{l,\beta,q}$, используются разные элементы массива a_l.

Доказательство. Предположим противное утверждению леммы: на двух итерациях потока вычислений с различными параметрами циклов j_ξ, $\xi \le \xi^{l,\beta,q}$, используется один и тот же элемент данных. Это означает $F_{l,\beta,q}J = F_{l,\beta,q}\hat{J}$ на итерациях J и $\hat{J}$ потока вычислений с различными параметрами циклов j_ξ, $\xi \in \Xi^{l,\beta,q}$; учтено, что $r_{\xi,2}=1$ для $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$.

Так как матрица $F^{\Xi}_{l,\beta,q}$ имеет $|\Xi^{l,\beta,q}|$ линейно независимых столбцов, то можно выделить столько же линейно независимых строк. Матрицу, составленную из выделенных строк, обозначим F_Ξ. Обозначим через J_Ξ и $\hat{J}_\Xi$ векторы размерности $|\Xi^{l,\beta,q}|$, построенные по векторам J и $\hat{J}$ выбором компонент с номерами из множества $\Xi^{l,\beta,q}$. Тогда векторы J_Ξ и $\hat{J}_\Xi$ различны и $F_{l,\beta,q}J - F_{l,\beta,q}\hat{J} = F^{\Xi}_{l,\beta,q}\, J_\Xi - F^{\Xi}_{l,\beta,q}\, \hat{J}_\Xi = 0$. Так как строки матрицы F_Ξ составлены из строк матрицы $F^{\Xi}_{l,\beta,q}$, то выполняется равенство $F_\Xi J_\Xi = F_\Xi \hat{J}_\Xi$. Из построения матрицы F_Ξ следует, что она является невырожденной. Умножая последнее равенство слева на $(F_\Xi)^{-1}$, получим $J_\Xi = \hat{J}_\Xi$ (противоречие). □

Отметим, что если $F_{l,\beta,q,\xi}=0$, $\xi < \xi^{l,\beta,q}$, и итерации цикла с параметром j_ξ выполняются параллельными потоками, то имеет место бродкаст.

Теорема 2. Пусть выполняются предположения леммы и не более чем один столбец $F_{l,\beta,q,\zeta}$ имеет вид $(0,...,0,b_{l,\beta,q,\zeta})^{\mathrm{T}}$, $0 < |b_{l,\beta,q,\zeta}| r_{\zeta,2} < L_l$. Если $r_{\zeta,2}=1$, цикл с параметром j_ζ выполняется параллельно, циклы с параметрами j_ξ, $\xi \notin \Xi^{l,\beta,q}$, $F_{l,\beta,q,\xi} \ne 0$, выполняются последовательно (с синхронизацией потоков вычислений между итерациями в случае $r_\xi > 1$), то независимо от выбора $r_{\xi,2}$, $\xi > \xi^{l,\beta,q}$, совокупное число кэш-промахов по всем потокам вычислений является наименьшим. Сформулированные условия накладываются при наличии указанных столбцов.

Доказательство. Сделаем естественное предположение, что строка массива a_l выровнена в памяти по размеру строки кэша; в этом случае обращения к одной строке массива не уменьшат число кэш-промахов при обращении к данным, расположенным в начале следующей строки.

Условие леммы $r_{\xi,2}=1$ для всех $\xi \notin \Xi^{l,\beta,q}$, $\xi \le \xi^{l,\beta,q}$, в случае $r_\xi > 1$, цикл с параметром j_ξ, $\xi \le \xi^{l,\beta,q}$, является последовательным и столбец $F_{l,\beta,q,\xi}$ линейно зависим со столбцами $F_{l,\beta,q,\xi}$, $\xi \in \Xi^{l,\beta,q}$, можно заменить условием синхронизации потоков между итерациями цикла с параметром j_ξ. Так как выполняются предположения леммы, то в каждом потоке вычислений разные элементы данных используются на разных итерациях циклов с параметрами j_ξ, $\xi \le \xi^{l,\beta,q}$. В каждом потоке вычислений на итерациях циклов с параметрами j_ξ, $\xi > \xi^{l,\beta,q}$ (если $\xi^{l,\beta,q} \ne n_\beta$), используется один и тот же элемент данных, так как столбцы $F_{l,\beta,q,\xi}$, $\xi > \xi^{l,\beta,q}$, являются нулевыми; итерации указанных циклов в потоке вычислений выполняются одна за другой. Следовательно, в каждом потоке на всех, кроме первой, итерациях, на которых используется элемент данных, он находится в кэше, то есть кэш-промахи, связанные с его использованием, отсутствуют. Если $F_{l,\beta,q,\xi}=0$, $\xi < \xi^{l,\beta,q}$, и итерации цикла с параметром j_ξ выполняются не одним, а параллельными потоками, то осуществляется бродкаст и, следовательно, совокупное по потокам вычислений число кэш-промахов остается неизменным в полуварпе. Совокупное по потокам число кэш-

промахов также не изменяется при произвольном выборе $r_{\xi,2}$, $\xi \in \Xi^{l,\beta,q}$, $\xi \neq \zeta$, так как разные итерации параллельно выполняемых циклов j_ξ имеют доступ к разным строкам массива. Кроме того, циклы по j_ξ, $\xi \notin \Xi^{l,\beta,q}$, $F_{l,\beta,q,\xi} \neq 0$, выполняются последовательно (по условию теоремы).

Рассмотрим цикл с параметром j_ζ (если имеется столбец указанного в формулировке теоремы вида). Использование элемента данных характеризуется пространственной локальностью относительно цикла с параметром j_ζ, так как используемые данные располагаются в одной строке массива на расстоянии $|b_{l,\beta,q,\zeta}|$ элементов. Число используемых строк кэша потоками равно $\left\lceil \frac{|b_{l,\beta,q,\zeta}| r_{\zeta,2} Q_{\zeta,2}}{L_l} \right\rceil$. Тогда в потоках вычислений имеется $Q_{\zeta,2} - \left\lceil \frac{|b_{l,\beta,q,\zeta}| r_{\zeta,2} Q_{\zeta,2}}{L_l} \right\rceil$ использований строк кэша без кэш-промахов при чтении элементов строки массива (с учетом того, что строка массива a_l помещается в кэш и выполнено условие $|b_{l,\beta,q,\zeta}| r_{\zeta,2} < L_l$). Их число наибольшее при наибольшем $Q_{\zeta,2}$, то есть при $r_{\zeta,2}=1$. Тогда совокупное по всем потокам вычислений число кэш-промахов, связанных с использованием элементов строки массива a_l, наименьшее (потребуется один обязательный кэш-промах для строк кэша). □

Теорема дает условия, при которых в потоках вычислений повторное использование элемента данных возникает только на последовательных итерациях алгоритма (следует из доказательства). Указывается выбор $r_{\xi,2}$, при котором число кэш-промахов наименьшее, а размеры $r_{\xi,2}$, для которых $\xi \in \Xi^{l,\beta,q}$ или $\xi > \xi^{l,\beta,q}$, могут быть заданы произвольно, так как число кэш-промахов на этом вхождении будет одинаково по всем потокам вычислений. Выполнение условий теоремы свидетельствует о наличии пространственной локальности на итерациях цикла с параметром j_ζ (в различных потоках вычислений) и о наличии временной локальности для осуществления бродкаста данного по параллельно выполняемым циклам с параметрами j_ξ, для которых $F_{l,\beta,q,\xi}=0$.

Пример 1 (продолжение). Рассмотрим блоки вычислений (рис. 3). Известно, что самый внешний цикл (цикл с параметром j_1, $j_1=k$) является последовательным, а внутренние циклы по $j_2=i$ и по $j_3=j$ могут выполняться параллельно. Имеем: $F_{1,1,2}=\begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix}$, $F_{1,1,3}=\begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\end{pmatrix}$, $F_{1,1,4}=\begin{pmatrix}1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}$, $\xi^{1,1,2}=3$, $\xi^{1,1,3}=2$, $\xi^{1,1,4}=3$. На вхождении (1,1,2) столбец $F_{1,1,2,1}$ является нулевым, поэтому, следуя теореме (предположения леммы справедливы и для теоремы) применительно к вхождению $(l,\beta,q)=(1,1,2)$, положим $r_{1,2}=1$ (или положим $r_{1,2}$ максимально возможным — размеру блока r_1 — и установим синхронизацию потоков между итерациями цикла по k). Далее на вхождении (1,1,2) имеем: $F_{1,1,2,2}=(1\ 0)^{\mathrm{T}}$, $\xi=2$, $\xi \in \Xi^{l,\beta,q}$, поэтому $r_{2,2}$ можно выбрать произвольно (но не больше максимально возможного размера блока); $F_{1,1,2,3}=(0\ 1)^{\mathrm{T}}$, поэтому выберем $r_{3,2}=1$. На вхождении (1,1,3) имеем $F_{1,1,3,1}=(0\ 1)^{\mathrm{T}}$, цикл по j_1 выполняется последовательно, поэтому выбор $r_{1,2}$ не определяется теоремой; $F_{1,1,3,2}=(1\ 0)^{\mathrm{T}}$, поэтому $r_{2,2}$ можно выбрать произволь-

но; так как $F_{1,1,3,3}$=0, ξ=3>$\xi^{1,1,3}$, то $r_{3,2}$ можно выбрать произвольно. На вхождении (1,1,4) имеем $F_{1,1,4,1}$=$(1\ 0)^{\mathrm{T}}$, поэтому $r_{1,2}$ можно выбрать произвольно; $F_{1,1,4,2}$=0, ξ=2≤$\xi^{1,1,4}$, поэтому выберем $r_{2,2}$=1; $F_{1,1,4,3}$=$(0\ 1)^{\mathrm{T}}$, поэтому выберем $r_{3,2}$=1.

Таким образом, суммируя сведения по каждому вхождению, выберем размеры потоков вычислений следующим образом: $r_{1,2}$=1, $r_{2,2}$=1, $r_{3,2}$=1. Выбор размера потоков указанным образом определяет наименьшее в совокупности по потокам вычислений число кэш-промахов для каждого из вхождений (1,1,2), (1,1,3), (1,1,4) в отдельности. Бродкаст данных осуществляется на вхождении (1,1,3) для параллельного цикла по j_3 (так как $F_{1,1,3,3}$=0) и на вхождении (1,1,4) для параллельного цикла по j_2 (так как $F_{1,1,4,2}$=0). Использование данных на итерациях цикла по j_3 (в различных параллельных потоках вычислений) на вхождениях (1,1,2), (1,1,4) ($F_{1,1,2,3}$=$F_{1,1,4,3}$=$(0\ 1)^{\mathrm{T}}$) характеризуется пространственной локальностью. Как уже отмечалось, можно положить $r_{1,2}$=r_1 и установить синхронизацию потоков после каждой итерации k. Если потоки одного блока не имеют общих данных, то синхронизация не требуется.

4. Вычислительные эксперименты

Пример 1 (продолжение). В работе [10] реализован алгоритм Флойда—Уоршелла с 3D блоками. Как показано в подразделе 2.2, блочный алгоритм с 3D блоками обладает улучшенной, по сравнению с 2D-блочной версией, локальностью; это позволило уменьшить время реализации в 5-6 раз. В работе [11] алгоритм Флойда—Уоршелла с 3D блоками модифицирован, в том числе и с целью использования пространственной локальности данных; время реализации уменьшилось еще в 5 раз.

Пример 2. Рассмотрим алгоритм прямого хода метода Гаусса решения систем линейных алгебраических уравнений с использованием расширенной матрицы (рис. 4).

```
o k = 1, n - 1
  do i = k + 1, n
    do j = k + 1, n + 1
      a(i,j) = a(i,j) - (a(i,k) / a(k,k)) * a(k,j)
    enddo
  enddo
enddo
```

Рис. 4. Основная часть алгоритма Гаусса (без выбора ведущего элемента)

Исследование локальности этого алгоритма практически совпадает с приведенным в разделе 2 исследованием локальности алгоритма Флойда—Уоршелла. Некоторое несущественное дополнение к приведенным выкладкам связано с наличием ведущего элемента $a(k,k)$.

В экспериментах использовалась видеокарта GeForce GT 860M со следующими характеристиками: объем глобальной памяти — 4295 МБ; максимальный размер разделяемой памяти в одном блоке — 49 КБ; количество 32-разрядных регистров в одном блоке — 65 536; количество потоков в варпе — 32; мультипроцессоров — 5; всего ядер — 640.

На рис. 5 продемонстрирована зависимость времени реализации алгоритма от r_1 — параметра размера блока, характеризующего число записей в глобальную память. Число записей результатов вычисления одного блока пропорционально $Q_1 = \left\lceil \frac{n-1}{r_1} \right\rceil$, поэтому чем большие r_1, тем меньше затраты времени на запись в глобальную память. Блоки имеют размер $r_1 \times r_2 \times r_3 = r_1 \times 64 \times 64$. В разделяемую память заносились только те элементы массива, которые использовались и для чтения, и для записи. Элементы, которые использовались только для чтения, читались из глобальной памяти. Пространственная локальность данных не использовалась.

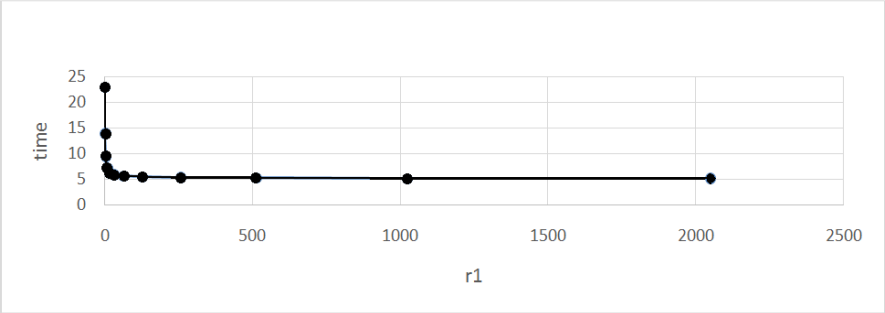

Рис. 5. Зависимость времени реализации блочного алгоритма Гаусса от r_1

Вычислительные эксперименты показали, что время реализации алгоритма уменьшается (до некоторого предела) с ростом r_1, т.е. уменьшается при уменьшении числа операций записи в глобальную память. Стабилизация времени связана с тем, что требуется определенное время на запуски ядра и на планирование потоков.

Заключение

Таким образом, в работе сформулированы и доказаны утверждения, позволяющие оценить объем коммуникационных операций, порождаемых разбиением множества итераций. Асимптотические оценки суммарного объема коммуникационных операций дают возможность ранжировать параметры размера блоков для минимизации объема коммуникационных операций. В работе также получены условия, при выполнении которых достигается наименьшее число кэш-промахов в потоках вычислений при наличии локальности данных.

Направления дальнейших исследований: разработка и программная реализация алгоритмов, позволяющих выбирать соотношения размеров блоков вычислений; получение условий и соотношений, точно характеризующих объем коммуникационных операций; обобщения достаточных условий наличия локальности данных, пригодные для применения в большем числе случаев; исследования, направленные на минимизацию объема коммуникационных операций между параллельными вычислительными процессами, реализуемыми на суперкомпьютерах с распределенной памятью; разработка и программная реализация параллельных алгоритмов для решения прикладных задач с использованием предлагаемого подхода.

Работа выполнена в рамках государственной программы научных исследований Республики Беларусь «Конвергенция-2020», подпрограмма «Методы математического моделирования сложных систем».

Литература

1. Лиходед Н.А. Полещук М.А. Метод ранжирования параметров размера блоков вычислений параллельного алгоритма // Доклады НАН Беларуси. 2015. Т. 59, № 4. С. 25–33.
2. Kandemir M., Ramanujam J., Irwin M., Narayanan V., Kadayif I., Parikh A. A compiler based approach for dynamically managing scratch-pad memories in embedded systems // IEEE Transactions on Computer-Aided Design. 2004. Vol. 23, No. 2. P. 243–260.
3. Baskaran M., Bondhugula U., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic data movement and computation mapping for multi-level parallel architectures with explicitly managed memories // Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming. Salt Lake City, USA, February 20–23, 2008.
4. Воеводин Вл.В., Воеводин Вад.В. Спасительная локальность суперкомпьютеров // Открытые системы. 2013. № 9. С. 12–15.
5. Воеводин В.В., Воеводин В.Вл. Параллельные вычисления. Санкт-Петербург: БХВ-Петербург, 2002. 608 с.
6. Xue J., Cai W. Time-minimal tiling when rise is larger than zero // Parallel Computing. 2002. Vol. 28, No. 5. P. 915–939.
7. Baskaran M., Ramanujam J., Sadayappan P. Automatic C-to-CUDA code generation for affine programs // Proceedings of the Compiler Construction, 19th International Conference. Part of the Joint European Conferences on Theory and Practice of Software. Paphos, Cyprus, March 20–28, 2010.
8. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic transformations for communication-minimized parallelization and locality optimization in the polyhedral model // Lecture notes in computer science. 2008. No. 4959. P. 132–146.
9. Venkataraman G., Sahni S., Mukhopadhyaya S. A blocked all-pairs shortest-paths algorithm // J. Exp. Algorithmics. 2003. Vol. 8. P. 2.2.
10. Katz G.J., Kider J. All-pairs shortest-paths for large graphs on the GPU // Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware. Sarajevo, Bosnia and Herzegovina: Eurographics Association. 2008. P. 47–55.
11. Lund B.D., Smith J.W. A multi-stage cuda kernel for floyd-warshall // CoRR abs/1001.4108. 2010.

DOI: 10.14529/cmse160307

ESTIMATE OF LOCALITY OF PARALLEL ALGORITHMS IMPLEMENTED ON GPUS

© 2016 N.A. Likhoded, M.A. Paliashchuk

Belarusian State University (Nezavisimosti avenue 4, Minsk, 220030 Republic of Belarus)

E-mail: likhoded@bsu.by, poleschuma@bsu.by

Received: 02.03.2016

The problem of obtaining blocks of operations and threads of parallel algorithm resulting in a smaller number of accesses to global memory and resulting in the efficient use of caches and shared memory graphics processor is investigated. We formulated and proved statements to assess the volume of communication transactions generated by alternative sizing of blocks, as well as to minimize the number of cache misses due to the use of temporal and spatial locality of data. The research is constructive and allows software implementation for practical use.

Keywords: parallel computing, GPU, minimization of communications, temporal locality, spatial locality.

FOR CITATION

Likhoded N.A., Paliashchuk M.A. Estimate of Locality of Parallel Algorithms Implemented on GPUs. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 3. pp. 96–111. (in Russian) DOI: 10.14529/cmse160307.

References

1. Likhoded N.A., Poleshchuk M.A. Method of Ranking Tiles Size Parameters of Parallel Algorithm. *Doklady NAN Belarusi* [Proceedings of the National Academy of Sciences of Belarus]. 2015. vol. 59, no. 4. pp. 25–33. (in Russian)
2. Kandemir M., Ramanujam J., Irwin M., Narayanan V., Kadayif I., Parikh A. A Compiler Based Approach for Dynamically Managing Scratch-Pad Memories in Embedded Systems. *IEEE Transactions on Computer-Aided Design*. 2004. vol. 23, no. 2. pp. 243–260. DOI: 10.1109/tcad.2003.822123.
3. Baskaran M., Bondhugula U., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic Data Movement and Computation Mapping for Multi-Level Parallel Architectures with Explicitly Managed Memories. *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. Salt Lake City, USA, February 20–23, 2008. DOI: 10.1145/1345206.1345210.
4. Voevodin Vl.V., Voevodin Vad.V. The Fortunate Locality of Supercomputers. *Otkrytye sistemy* [Open Systems]. 2013. no. 9. pp. 12–15. (in Russian)
5. Voevodin V.V., Voevodin Vl.V. *Parallel'nye vychisleniya* [Parallel Computing]. Sankt-Peterburg: BKhV-Peterburg, 2002. 608 p.
6. Xue J., Cai W. Time-Minimal Tiling when Rise is Larger than Zero. *Parallel Computing*. 2002. vol. 28, no. 5. pp. 915–939. DOI: 10.1016/s0167-8191(02)00098-4.
7. Baskaran M., Ramanujam J., Sadayappan P. Automatic C-to-CUDA Code Generation for Affine Programs. *Proceedings of the Compiler Construction, 19th International Conference. Part of the Joint European Conferences on Theory and Practice of Software*. Paphos, Cyprus, March 20–28, 2010. DOI: 10.1007/978-3-642-11970-5_14.
8. Bondhugula U., Baskaran M., Krishnamoorthy S., Ramanujam J., Rountev A., Sadayappan P. Automatic Transformations for Communication-Minimized Parallelization

and Locality Optimization in the Polyhedral Model. *Lecture Notes in Computer Science*. 2008. no. 4959. pp. 132–146. DOI: 10.1007/978-3-540-78791-4_9.

9. Venkataraman G., Sahni S., Mukhopadhyaya S. A Blocked All-Pairs Shortest-Paths Algorithm. *J. Exp. Algorithmics*. 2003. vol. 8. pp. 2.2. DOI: 10.1145/996546.996553.
10. Katz G.J., Kider J. All-Pairs Shortest-Paths for Large Graphs on the GPU. *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*. Sarajevo, Bosnia and Herzegovina: Eurographics Association. 2008. pp. 47–55.
11. Lund B.D., Smith J.W. A Multi-Stage CUDA Kernel for Floyd-Warshall. *CoRR abs/1001.4108*. 2010.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по следующим отраслям и группам специальностей: 05.13.00 – информатика, вычислительная техника и управление; 01.01.00 – математика; 25.00.00 – науки о Земле (№421).

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год (февраль, май, август и ноябрь).

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**