На правах рукописи

Шичкина Юлия Александровна

МЕТОДЫ СОЗДАНИЯ И ЭКВИВАЛЕНТНЫХ ПРЕОБРАЗОВАНИЙ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ С УЧЕТОМ ИНФОРМАЦИОННЫХ ЗАВИСИМОСТЕЙ

Специальность 05.13.11. Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
доктора технических наук

Санкт-Петербург – 2014 г.

Работа выполнена на кафедре Вычислительной техники ФГБОУ ВПО «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)» и на кафедре Дискретной математики и защиты информации ФГБОУ ВПО «Братский государственный университет»

Научный консультант	**Куприянов М.С.** – д.т.н., профессор, декан факультета компьютерных технологий и информатики ФГБОУ ВПО «СПбГТЭУ»
Официальные оппоненты:	**Демьянович Ю.К.** – д.ф.-м.н., профессор, заведующий кафедрой параллельных алгоритмов математико-механического факультета ФГБОУ ВПО «СПбГУ» **Болдырев Ю.Я.** – д.т.н., профессор, заведующий лабораторией «Прикладная математика и механика» кафедры прикладной математики физико-механического факультета, директор отделения информационно-вычислительных технологий ФГБОУ ВПО «СПбГПУ» **Марлей В.Е.** – д.т.н., профессор, заведующий кафедрой вычислительных систем и информатики ФГБОУ ВПО «ГУМРФ»
Ведущая организация:	ФГБУН «Санкт-Петербургский институт информатики и автоматизации РАН»

Защита диссертации состоится “25” июня 2014 г. в 15 часов 30 минут на заседании диссертационного совета Д 212.238.01 при Федеральном государственном бюджетном образовательном учреждении высшего профессионального образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)» по адресу: 197376, Россия, Санкт-Петербург, улица Профессора Попова, дом 5.

С диссертацией можно ознакомиться в библиотеке Федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Автореферат разослан «18» февраля 2014 г.

Ученый секретарь
диссертационного совета Д 212.238.01
к.т.н., доцент _____ Щеголева Н.Л.

I. ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы исследования В настоящее время параллельное программирование, несмотря на интенсивное развитие вычислительных систем, новых методов и языков программирования, по-прежнему остается областью, в которой проблема разбиения программы на процессы ложится на разработчика, от квалификации которого, следовательно, зависит качество получаемых в результате решений прикладных задач. Отрицательно сказывается на качестве работы вручную разрабатываемой параллельной программы также рост алгоритмической сложности решаемых задач, что можно объяснить следующими обстоятельствами: чрезвычайно быстрый рост сложности объектов моделирования; необходимость управления сложными промышленными и технологическими процессами в режиме реального времени и в условиях неопределенности; рост числа задач, для решения которых необходимо обрабатывать гигантские объемы информации.

Уменьшить влияние человеческого фактора на процесс разработки параллельных программ возможно, если переложить на ЭВМ функции по расчету числа необходимых для решения прикладной задачи процессоров и построение модели будущей параллельной программы.

Актуальность темы исследования обусловлена следующими факторами.

1. Ограниченность вычислительных ресурсов. Вычислительные системы развиваются очень быстро. Размер оперативной памяти и скорость работы процессора растет год от года, но еще быстрее растут требования к качественной стороне математических моделей, что ведет к увеличению числа параметров моделей. В результате появляется необходимость поиска методов системных переходов от существующих последовательных систем вычислений к распределенным системам обработки данных.

2. Отсутствие аппарата по анализу эффективности параллельных программ. Параллельное программирование, как и обычное программирование, опирается на алгоритмы и структуры данных. Не эффективно работающая программа – это прямые потери производительности компьютера, средств на его приобретение и усилий на решение задач. Таких потерь хотелось бы избежать или, по крайней мере, их минимизировать. Для этого необходимы апробированные методы исследования, разработки и эквивалентных преобразований алгоритмов.

К основным недостаткам существующих методов разработки и эквивалентных преобразований параллельных алгоритмов можно отнести:

– отсутствие этапа анализа структуры алгоритма, для реализации которого проектируется параллельная программа — вместо этого предлагается применять различные эвристики;

– необходимость на каждом шаге поиска оптимальной структуры вычислительной системы решать задачу построения по возможности оптимального расписания организации вычислительного процесса (являющуюся, в общем случае, NP-полной задачей);

– зависимость качества получаемых с помощью эвристик решений от начального приближения и структуры исследуемого алгоритма обработки информации.

3. Неполнота совокупности всех существующих алгоритмов, предназначенных для решения прикладных задач. В процессе длительного использования последовательных компьютеров был накоплен и тщательно отработано огромное количество алгоритмов для самых разных прикладных задач. Среди них, применительно к области параллельного программирования, необходимо выделить следующие алгоритмы и связанные с ними пробелы:

- Алгоритмы на графах имеют пробелы в решении проблем, связанных с реструктуризацией систем при заданных ограничениях на структуру системы и необходимостью промежуточного сбора и анализа метаданных системы.

- Алгоритмы на разреженных матрицах. Отсутствие алгоритмов преобразования прямоугольной разреженной матрицы к новой форме с сохранением первоначальных значений параметров математической модели, несмотря на потребность в таких алгоритмах не только в области параллельных вычислений, но и теории принятия решений, синтеза систем управления производственными процессами и других областях.

4. Неэффективность использования кластеров. Кластерная архитектура занимает лидирующие позиции среди суперкомпьютеров, но современные кластеры, приобретаемые с запасом масштабируемости для решения самых больших задач организации, обычно большую часть времени простаивают, либо загружены незначительно. Повысить эффективность работы кластера возможно, если автоматизировать процесс распределения нагрузки между процессорами.

5. Отсутствие автоматизированных систем построения нормальных реляционных схем. Разработка эффективных алгоритмов обработки данных не может исключить этап построения качественной структуры хранения данных. От того, насколько удачно спроектирована система хранения данных напрямую зависит и скорость доступа к данным и работа приложений по обработке данных. На сегодняшний день требуется высококачественно проектировать и внедрять базы данных за весьма небольшое время. Необходимость управления сложными промышленными и технологическими процессами в режиме реального времени и в условиях неопределенности требует высокой степени надежности хранения информации и высокой скорости ее обработки. Ни одна из современных СУБД не проверяет схему базы данных на наличие аномалий, не смотря на

то, что современный аппарат прикладной математики позволяет автоматизировать этот процесс.

Разработка научных основ анализа и синтеза систем управления сложными многосвязными системами на основе метаданных об информационных зависимостях в них, с целью повышения качества работы в условиях повышенных требований ко времени отклика систем и объему обрабатываемых данных является своевременной и актуальной.

Объектом диссертационного исследования являются параллельные вычисления на кластерных системах.

Предметом диссертационного исследования являются модели, методы и алгоритмы эффективных параллельных вычислений на кластерных системах.

Целью диссертационного исследования является построение, теоретическое обоснование и практическое подтверждение применимости системы методов проектирования оптимальных по числу процессоров и времени исполнения параллельных программ и реляционных схем, а также методов их анализа и преобразований с сохранением первоначальных информационных зависимостей.

Основные задачи исследования

1. Анализ существующих методов и инструментальных средств, предназначенных для формализованного представления параллельных алгоритмов и реляционных схем, а также расчета количественных и качественных характеристик.
2. Получение оценки минимального количества необходимых процессоров для решения вычислительной задачи при фиксированном времени решения с помощью параллельного алгоритма.
3. Разработка методов проектирования параллельных алгоритмов, оптимальных по времени выполнения и числу процессоров, задействованных в решении поставленной задачи.
4. Разработка матричного представления операций реляционной алгебры и методов создания на их основе реляционных схем по декларированным функциональным зависимостям.
5. Разработка программного обеспечения для тестирования полученных методов и методики их применения в зависимости от количественных характеристик параллельного алгоритма и параметров очередного шага проектирования программы.
6. Применение разработанных методов и программного обеспечения к задачам принятия решений на основе эмпирических данных, масштабирования математической модели управления процессом электролиза алюминия, реструктуризации произвольно заданной базы данных и построения математической

модели выполнения параллельных программ на ЭВМ с кластерной архитектурой.

Методы исследования. Для решения поставленных задач использовались методы теории графов; параллельных вычислений, матричной алгебры, баз данных, синергетики, методы анализа алгоритмов и программ и численные методы.

Основные положения, выносимые на защиту

1. Методы обработки разреженных матриц с сохранением первоначальных значений математической модели;
2. Методы разработки и эквивалентных преобразований параллельных алгоритмов с учетом:
   - времени выполнения;
   - количества процессоров;
   - времени выполнения и числа процессоров.
3. Формулы расчета количества процессоров, необходимых для решения вычислительной задачи с помощью параллельного алгоритма;
4. Методы преобразования ориентированного графа в дерево функциональных зависимостей и создания реляционных схем на основе атрибутов предметной области и декларированных функциональных зависимостей;
5. Методы распределения задач на вычислительной системе кластерного типа;
6. Матричный метод построения дерева решений и формула, позволяющая оценить вероятность срабатывания каждого правила.

Научная новизна работы

К наиболее существенным научным результатам относятся следующие:

1. Разработан метод приведения разреженной матрицы к треугольному виду, отличающийся сохранением первоначальных значений элементов, что является необходимым требованием при задачах оптимизации параллельных алгоритмов.
2. Разработаны методы построения и оптимизации информационного графа параллельного алгоритма по ширине. Методы состоят из двух частей, которые могут применяться независимо друг от друга в зависимости от результатов исследования на промежуточном этапе метаданных. Методы основаны на матрице смежности, списках смежности и списках следования и отличаются.
3. Получены и обоснованы граничные условия числа задействованных в параллельных вычислениях процессоров, по которым можно принимать решение о дальнейшей разработке или эквивалентном преобразовании параллельной программы.

4. Разработаны методы оптимизации параллельного алгоритма по времени выполнения, позволяющие осуществлять равномерную загрузку процессоров, сокращать время решения поставленной задачи и уменьшать ширину информационного графа, соответствующего данному алгоритму. Методы основаны на матрицах, списках смежности и списках следования.

5. Предложено представление функциональных зависимостей реляционных схем в матричном виде и получены матричные аналоги свойств функциональных зависимостей. В отличие от известных ранее способов представления и обработки функциональных зависимостей, предложенные матричные свойства позволяют дополнить современные СУБД отсутствующей в них функцией проверки реализуемых физических баз данных на корректность структуры и отсутствие аномалии избыточности.

6. Разработан метод построения реляционной схемы на основе атрибутов предметной области и декларированных функциональных зависимостей. Все получаемые схемы удовлетворяют нормальной форме Бойса-Кодда и имеют минимум аномалии избыточности.

7. Разработаны матричные методы построения замыкания функциональных зависимостей и нахождения первичного ключа реляционного отношения на основе функциональных зависимостей.

8. Разработан матричный метод построения дерева решений, который позволяет получать более широкую совокупность правил по сравнению с эвристическими методами. Совокупность правил может быть отрегулирована с помощью выведенных формул, позволяющих также оценить вероятность срабатывания правила. Разработанный матричный метод предпочтительнее эвристических методов за счет более низкой трудоемкости, компактной формы хранения данных и полноты получаемой совокупности правил.

9. Разработан программный комплекс для тестирования полученных методов и проведения дальнейших исследований в области параллельных вычислений, реляционных баз данных и теории принятия решений.

Обоснованность и достоверность полученных результатов обеспечивается корректностью применяемого математического аппарата, строгими доказательствами предложенных утверждений, результатами вычислительного эксперимента.

Практическая значимость результатов исследования

1. Разработанные методы проектирования реляционных отношений можно применять для автоматизированного построения новой и проверки существующей схемы базы данных на оптимальность по скорости доступа к данным и компактности хранения данных.

2. Разработанные методы построения и эквивалентного преобразования параллельных алгоритмов можно применять в целях упрощения и ускорения

процессов разработки, анализа и сравнения параллельных программ и принятия решения о необходимости преобразования исходной программы.

3. Разработанное программное обеспечение можно применять для получения заключения о возможности распараллеливания вычислительной задачи, получения модели параллельной программы, проведения дальнейших исследований в области параллельных вычислений с применением информационных графов, строить реляционные схемы, удовлетворяющие нормальным формам, находить эвристические зависимости на основе большого объема эмпирических данных.

4. Разработанные методы преобразования разреженных матриц можно применять в любой прикладной задаче по преобразованию сложной многосвязной системы, представимой в виде графа.

5. Разработанный метод построения дерева решений может быть применен как в задачах принятия решений, так и в задачах установления связи между статистическими выборками, значения которых могут быть одновременно как непрерывными, так и дискретными и состоящими всего из нескольких значений, что делает применение классических методов статистического корреляционного анализа невозможным.

Реализация результатов исследования

Теоретические и практические результаты диссертационной работы используются в программных средствах ОАО «РУСАЛ-Братск» (Братский алюминиевый завод), ОАО «Концерн «Океанприбор» и ГУ «Санкт-Петербургский центр по гидрометеорологии и мониторингу окружающей среды с региональными функциями», что подтверждено актами о применении. Результаты работы используются в учебном процессе кафедры дискретной математики и защиты информации ФГБОУ ВПО «Братский государственный университет» при преподавании курсов «Теория алгоритмов», «Параллельное программирование», «Численные методы», матричные методы принятия решений и разработанное программное применяются при формализованном подходе к формированию заявки на контрольные цифры приема в ФГБОУ ВПО «Братский государственный университет».

Материалы работы использованы в Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы в рамках проекта «Создание высокопроизводительных вычислительных технологий для интеллектуальных систем оперативной обработки и визуализации гидроакустической информации» (шифр «2012-1.1-12-000-2010-011» соглашение №14.B37.21.0589) и в проекте 4.3 этапа 2013 года программы фундаментальных исследований Президиума РАН №14 «Проблемы создания национальной научной распределенной информационно-вычислительной среды на

основе GRID технологий, облачных вычислений и современных телекоммуникационных сетей».

Апробация результатов исследования

Результаты работы обсуждались на конференциях: научно-технические конференции в г.Братске «Естественные и инженерные науки – развитию регионов» (БрГУ, 2004, 2005, 2009гг.), всероссийская научная конференция молодых ученых в г.Новосибирске (НГТУ, 2004г.), конференция «Математическое моделирование, численные методы и комплексы программы» в г.Санкт-Петербург (ФГБОУ ВПО «СПбГАСУ», 2004, 2009, 2011гг.), международной конференции «Наука в информационном пространстве» в г.Днепропетровск (2009г.), Научно-технических семинарах ФГБОУ ВПО «НовГУ» (2011г.), ФГБУН СПИИРАН, ФГБОУ ВПО «ИТМО», ФГБОУ ВПО «СПбГЭТУ» (2012), II национальный суперкомпьютерный форум «НСКФ» в г.Переславль-Залесский (2013г) и др.

Публикации

Основное содержание диссертационной работы опубликовано в 31 печатных работах общим объемом 13 п.л. (личный вклад автора – 12 п.л.), из них 16 статей в изданиях из перечня ведущих рецензируемых научных журналов и изданий, рекомендованных ВАК РФ (из них 2 в соавторстве).

Структура и объем

Диссертация состоит из введения, 5 глав с выводами по каждой главе, заключения, 20 приложений и списка литературы, содержащего 171 наименование. Общий объем работы составляет 359 страниц машинописного текста, включая 114 рисунков, 21 таблицу.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность исследования, определены направления исследований, сформулирована цель работы, научная новизна и практическая ценность диссертации, сформулированы положения, выносимые на защиту.

В первой главе выполнена постановка задач исследования и обоснован выбор в качестве инструментария методов теории графов, матричной алгебры и параллельных вычислений. Проведен анализ терминов *сложная многосвязная распределенная система* и *матричные алгоритмы на графах*. Установлено, что очень большое число прикладных задач сводится к построению и исследованию сложных многосвязных распределенных систем, для которых не существует матричных алгоритмов на графах. Хороший внутренний параллелизм матричных алгоритмов и сравнительно низкая временная трудоемкость обусловливают необходимость разработки матричных алгоритмов на графах для проведения метавычислений над сложными многосвязными распределенными системами.

Установлено отсутствие методов преобразования разреженных матриц произвольной формы с сохранением первоначальных значений их элементов,

что является одним из основных требований в обработке сложных многосвязных распределенных систем, для которых важна сохранность информационных зависимостей и адекватность отображения входной информации в выходную.

Проведен анализ способов формализованного представления графов, который показал, что ориентированные графы, сводимые к разреженным матрицам, эффективнее всего обрабатывать с помощью списков связности. Методов обработки графов с помощью списков очень мало, в то время как эти методы являются более эффективными по скорости выполнения по сравнению с остальными методами на графах. В связи с этим разработка списковых методов является необходимой для усовершенствования процессов метавычислений над сложными многосвязными распределенными системами.

Рассмотрены основные положения по определению времени выполнения программы, как наиболее важной характеристике любого алгоритма. Представлены различные способы вычисления временной сложности алгоритмов, оценка которой необходима не только при сравнении двух алгоритмов между собой, но и при анализе отдельно взятого алгоритма.

Проведен анализ характеристик параллельных программ и предложены формулы оценки плотности вычислений и эффективности распараллеливания алгоритма: $P = \frac{\sum_{i=1}^{s} N_i}{N \cdot s}$, где N_i – число задействованных процессоров на каждом уровне; N – общее число процессоров, задействованных в вычислении; s – высота графа алгоритма.

За эффективность, как оптимальное соотношение между временем выполнения алгоритма и плотностью вычислений, можно взять тогда: $E = \frac{P}{s}$.

Показано, что параллельные алгоритмы являются в свою очередь также сложными многосвязными распределенными системами представимыми с помощью графов.

Установлено, что существует целый ряд областей, в которых для моделей, представляющих собой сложные многосвязные распределенные системы, отсутствуют матричные методы на графах или существуют, но являются недостаточно эффективными (табл.1).

Установлено, что для построения эффективных последовательных и параллельных алгоритмов необходимо проведение исследований в областях:

– методы представления и обработки разреженных матриц с сохранением первоначальных элементов;

– методы представления, разработки и эквивалентных преобразований реляционных баз данных;

Таблица 1.

Характеристики различных прикладных областей, сводимых к сложным многосвязным распределенным системам

Области	Возможность представления с помощью графов	Наличие методов построения	Наличие матричных методов построения	Наличие матричных методов преобразования	Потребность в матричных методах
Базы данных: реляционные и иерархические	+	-	-	-	+
Параллельные вычисления	+	+	-	-	+
Вычисления на кластерах	+	+	-	-	+
Системный анализ технологических процессов	+	+	+	-	+
Принятие решений	+	+	-	-	+

– методы представления, разработки и эквивалентных преобразований последовательных и параллельных алгоритмов;

– методы получения количественных характеристик сложных многосвязных распределенных систем в предметных областях для сравнения систем между собой и самих с собой на различных стадиях преобразования с целью принятия решения о направлении изменения системы с целью ее улучшения.

Вторая глава диссертации посвящена описанию примеров математических моделей, требующих элементарных преобразований разреженных матриц с сохранением первоначальных значений элементов, и разработанных методов преобразования матриц к треугольной форме, как наиболее часто применяемой на практике: метода ненулевой диагонали и метода уплотнения вниз. Первый имеет актуальность при решении систем линейных уравнений, второй — при оптимизации алгоритмов с помощью теории графов.

Метод уплотнения вниз, основанный на элементарных преобразованиях первого порядка, позволяет за минимальное число проходов по матрице привести ее к нижнему треугольному методу.

Метод уплотнения вниз. Пусть дана квадратная бинарная матрица $E = (e_{ij})_{n \times n}$. Вектора P – это массив номеров строк (перечисленных сверху вниз), Q – это массив номеров столбцов (перечисленных справа налево), соответствующих нижней треугольной матрице.

1. Просматривая матрицу *E* справа налево, записать номера нулевых столбцов в массив Q. Удалить из матрицы отмеченные столбцы.

2. Просматривая матрицу, записать номера нулевых строк в массив P1. Удалить из матрицы отмеченные строки.

3. В оставшейся ненулевой матрице находим строку, содержащую максимальное количество ненулевых элементов. Заносим номер этой строки в массив P и удаляем ее. Если существуют строки с одинаковым количеством ненулевых элементов, то выбор строки следует производить по следующему правилу:

– определить вес каждого ненулевого элемента – количество ненулевых элементов в данном столбце;

– если ненулевых элементов с минимальным весом будет всего один, то выбираем строку, содержащую этот ненулевой элемент;

– если ненулевых элементов с минимальным весом – несколько, то определяем дополнительный вес каждого ненулевого элемента с минимальным весом. Дополнительный вес вычисляется как сумма ненулевых элементов в строках с единицами из рассматриваемого столбца с ненулевым элементом с минимальным весом. В качестве удаляемой строки выбирается строка, содержащая элемент с минимальным дополнительным весом;

– если элементов с минимальным дополнительным весом несколько, то выбираем любую строку для дальнейшего использования метода.

4. В последней строке матрицы находим столбец с минимальным числом единиц и имеющим единицу в данной строке. Если столбцов, удовлетворяющих данному требованию, несколько, то выбираем любой из них. Перемещаем выбранный столбец (столбцы) в конец матрицы путем перестановки или обмена с другими столбцами. Заносим номера столбцов в массив Q.

5. Если в матрице не осталось ни одной строки и столбца, то записать массив $P=P \cup P1$ и переставить строки и столбцы исходной матрицы в соответствии с номерами, записанными в массивах P и Q. Иначе перейти к шагу 3.

В результате применения метода строки с наибольшим числом ненулевых элементов будут расположены в нижней части матрицы, стянуты к левому нижнему углу матрицы. Это качество позволяет в будущем при оптимизации параллельных алгоритмов применять меньшее число шагов.

В третьей главе приведены результаты исследований в области организации последовательных и параллельных вычислений и получения эффективно работающих алгоритмов, независимо от природы прикладной задачи. Рассмот-

рен метод перебора всех путей, основанный на матрице смежности ориентированного графа. Данный метод лежит в основе методов оптимизации параллельных алгоритмов по различным параметрам и отличается от методов этого класса тем, что не просто позволяет найти все пути по графу, но и в процессе поиска формировать метаданные об исследуемом алгоритме, которые впоследствии будут использованы при построении нового алгоритма или реструктуризации исходного.

Важнейшим и одним из наиболее трудоемких этапов создания программы является разработка алгоритма. При решении этой задачи наиболее рациональным является применение информационного графа и его строгой параллельной формы.

Первый разработанный метод поиска оптимального графа основан на матрице смежности и состоит из двух частей: разбиение совокупности всех операций алгоритма на группы (построение параллельной формы) и перемещение вершин между группами с целью получения уменьшения ширины яруса и получения максимальной плотности.

Хранить и обрабатывать все элементы матрицы информационного графа, большинство из которых нулевые, является не рациональным. Второй разработанный метод основан на списках смежности, что значительно экономит память, уменьшает время преобразования графа, имеет большой запас внутреннего параллелизма.

При построении информационного графа предполагается, что время выполнения любых вычислительных операций является одинаковым и равняется одной условной единице, а передача данных между вычислительными устройствами выполняется мгновенно без каких-либо затрат времени.

Метод эквивалентного преобразования алгоритма

Часть 1. Разбиение совокупности операций по процессорам

1. Для заданного информационного графа построить соответствующий ему список смежности (V_1, V_2), где V_1 – начальная вершина направленного ребра, V_2 – конечная вершина.

2. Найти множество $V = V_1 - V_2$. Вершины, вошедшие в это множество, составят группу вершин, принадлежащих одному ярусу.

3. Удалить из списка смежности все пары, начальная вершина которых совпадает с одной из вершин множества V.

4. Если список не пустой, то вернуться на шаг 2. Если в списке не осталось ни одной пары, то первая часть метода – разбиение совокупности вершин на группы – окончена.

Часть 2. Оптимизация алгоритма по числу процессоров

Вторая часть – равномерное распределение вершин по группам – начинается с предпоследней группы.

5. Рассчитать значение теоретической минимальной ширины d информационного графа. Выбрать первую с конца группу с числом вершин меньшим теоретической минимальной ширины графа d . Обозначим ее M_i.

6. Составить список смежности для каждой из вершин, входящих в группу $M_{i-1}: V_{i-1j}(V_1, V_2)$, где $j = \overline{1,k}$, k – количество вершин в группе M_{i-1}.

7. Найти пересечения множеств: $V = M_i \cap V_2$. Если для некоторой вершины группы M_{i-1} пересечение пустое, то эту вершину можно перенести в группу M_i.

На этом шаге следует переносить вершины из группы M_{i-1} в группу M_i до тех пор, пока число вершин группы M_i не достигнет теоретической минимальной ширины графа d.

8. Если $i > 1$, то перейти к шагу 6, иначе метод окончен.

Третий разработанный метод эквивалентного преобразования алгоритма основан на списках следования, состоит из одной части и применяется в случае отсутствия необходимости расчета промежуточных количественных характеристик алгоритма.

Метод оптимизации информационного графа с помощью списков следования.

1. Рассчитать значение теоретической минимальной ширины d информационного графа.

2. Составить множество $V_к$ – совокупность всех выходных вершин графа.

3. Составить матрицу следования информационного графа и привести ее к нижнему треугольному виду.

4. Построить соответствующий информационному графу список следования (V_1, V_2).

5. Найти множество $V = V_1 - V_2$. Вершины, вошедшие в это множество, составят группу вершин M_i, принадлежащих одному i-му ярусу.

– Если число вершин множества $V = V_1 - V_2$ превышает d, то во множество M_i войдут только d первых в порядке следования слева на-

право (в списке следования сверху вниз) вершин множества V. Остальные вершины необходимо включить в следующую группу M_{i+1}. При повторном проходе этого шага в группу M_{i+1} будут добавлены вершины.

– Если число вершин множества $V = V_1 - V_2$ меньше числа d, то во множество M_i следует перенести вершины из множества $V_к$, удовлетворяющие правилу: $V_{kj} \cap V_1 = \varnothing$, т.е. вершины из $V_к$, отсутствующие во множестве V_1. Эти вершины могут быть взяты в произвольном порядке из множества: $V_k - V_1$.

6. Удалить из списка смежности все пары, конечная вершина которых (V_2) совпадает с одной из вершин множества V.

7. Если список не пустой, то вернуться на шаг 5.

Если в списке не осталось ни одной пары, то определить последнюю группу, которую составят вершины, которые остались нетронутыми и не вошли ни в одну из групп: $M_{k+1} = V_0 - \left(\bigcup_{i=1,k} M_i \right)$, где V_0 – совокупность всех выходных вершин графа.

Далее в главе рассмотрены разработанные методы оценки теоретической и практической минимальной ширины информационного графа, по которым можно принимать решение о дальнейшем преобразовании информационного графа. Зная минимальную высоту графа s, можно рассчитать теоретическое минимальное количество необходимых процессоров – ширину графа: $d = \left\lceil \frac{n}{s} \right\rceil$, где n - количество вершин графа. В этом случае плотность вычислений в идеале должна быть равной $P = \frac{\sum_{i=1}^{s} N_i}{N \cdot s} = \frac{\sum_{i=1}^{s} d}{d \cdot s} = 1$. Коэффициент ускорения при этом: $K = \frac{T_s + T_p}{T_s + T_p / N} = \frac{0 + s}{0 + s/N} = N$. Эффективность: $E = \frac{K}{N} = \frac{N}{N} = 1$.

На практике получить ширину графа, равную найденной в первой части метода списков смежности ширине d, удается крайне редко. Поэтому далее в главе решается вопрос о верхней границе оценки минимальной ширины графа, той границе, которую на практике можно достигнуть всегда.

При оптимизации информационного графа по ширине, передвигать вершины графа можно только в направлении от входных вершин к выходным, поэтому после окончания второй части метода можно рассчитать уточненную оценку минимальной ширины с учетом числа групп: $d' = \max\limits_{1 \le k \le m} \left\lceil \dfrac{\sum\limits_{i=i_k}^{n} d_i}{s-k+1} \right\rceil$, где m – число групп, i_k – номер первой вершины в k-й группе.

Формула для d' является оценкой минимальной ширины графа, более приближенной к практике, но и она не является верхней границей. Если выходная вершина одна, то, как минимум, одна группа (последняя) будет состоять из одной вершины. Следовательно, плотность на других ярусах будет выше. Аналогичный вывод можно сделать для входных вершин. Поэтому рассчитывать верхнюю оценку минимальной ширины графа следует с учетом числа выходных вершин.

Пусть $\Delta_{вх} = d' - n_{вх}$ – разность между шириной графа и числом входных вершин, где $n_{вх}$ – количество входных вершин, а $\Delta_{вых} = d' - n_{вых}$, где $n_{вых}$ – количество выходных вершин. Возможны варианты:

1) $\Delta_{вх} > 0, \Delta_{вых} > 0$. Плотность остальных групп повышается на величину: $\Delta = \left\rceil \dfrac{\Delta_{вх} + \Delta_{вых}}{g-2} \right\lceil$, где g – число групп, полученных в результате первой части метода. Следовательно: $d' = d' + \Delta$.

2) $\Delta_{вх} > 0, \Delta_{вых} < 0$. Количество выходных вершин больше минимальной ширины группы. Плотность последней группы уменьшить нельзя, поэтому: $d' = n_{вых}$.

3) $\Delta_{вх} < 0, \Delta_{вых} > 0$. Количество входных вершин больше минимальной ширины группы. Специфика информационного графа такова, что в зависимости от информационных связей между первой и последующими группами, в результате второй части метода ширина первой группы может быть и уменьшена и остаться в первоначальном состоянии. Поэтому поведение оценки минимальной ширины в данном случае остается под вопросом.

4) $\Delta_{вх} < 0, \Delta_{вых} < 0$. Данный случай является композицией 2 и 3-го вариантов. Поэтому: $d' = n_{вых}$.

С учетом того, что число выходных вершин уже учтено в формуле для Δ, то от последних трех вариантов можно отказаться, оставив верхнюю оценку минимальной ширины в следующем виде: $d' = d' + \Delta$, где:

$$\Delta = \begin{cases} \dfrac{\Delta_{вх} + \Delta_{вых}}{g-2}, \left(\Delta_{вх} \succ 0\right) \cap \left(\Delta_{вых} \succ 0\right) \\ 0, \textit{в остальных случаях} \end{cases}$$

Если свести полученные результаты на одну числовую ось d, то можно определить границы практической минимальной ширины информационного графа (рис.1):

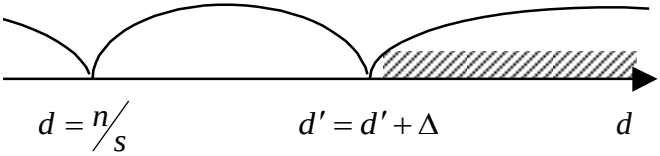

Рисунок 1. – Интервал практической минимальной ширины информационного графа

Таким образом, после применения первой части метода оптимизации параллельного алгоритма по ширине можно определить границы минимальной ширины графа, и уже во время применения второй части метода оптимизации по ширине судить об эффективности его применения. В самом методе во второй части, в зависимости от архитектуры используемого вычислительного кластера, можно применять вместо теоретической оценки минимальной ширины d практическое значение d', или любое другое значение, диктуемое условиями решения задачи.

Далее в главе проводятся исследования по разработке метода оптимизации параллельной программы по времени выполнения при заданном числе процессоров и методике оптимизации параллельной программы одновременно по нескольким параметрам.

При исследовании информационного графа возникают следующие вопросы:

- ✓ Насколько непрерывно работают все три задействованные в вычислительном процессе системе? Сколько времени простаивает каждый процессор в процессе работы алгоритма?
- ✓ Можно ли сократить общее время работы алгоритмы путем уменьшения времени простоев процессоров?
- ✓ Можно ли сократить ширину графа алгоритма путем уменьшения времени простоев процессоров за счет объединения мелких (по времени работы) операций в более крупные?

Как показали исследования, разработанный метод оптимизации алгоритмов по ориентированному графу возможно применить и для оптимизации алгоритмов по взвешенному графу. В главе приведен модифицированный метод оптимизации параллельных алгоритмов по ориентированному взвешенному графу.

Метод оптимизации информационного графа по высоте

1) Разбить вершины графа на группы с помощью метода оптимизации ширины информационного графа.

2) Начиная с первого яруса, найти группу вершин с разными временными значениями. Отметить ее.

3) В найденной группе найти максимальное значение по времени.

4) Отметить вершину, временное значение которой меньше максимального и рассмотреть возможность ее объединения с вершиной из следующей группы по принципу: если в строке матрицы смежности, соответствующей отмеченной вершине есть ненулевой элемент, соответствующий вершине из следующей группы и в столбце, соответствующем этой вершине, отсутствуют ненулевые элементы в строках отмеченной группы, то отмеченную вершину можно объединить с проанализированной вершиной из следующей группы. Формально, это условие выглядит так:

Пусть v_i^1 *— отмеченная вершина группы* G_1, v_j^2 *— вершина из следующей группы* G_2, v_{ij} *— элементы матрицы смежности, стоящие на пересечении строк вершин группы* G_1 *и j-го столбца, тогда:*

$$v_i^1 = v_i^1 + v_j^2, \ \textit{если} \ \left(v_j^1 \neq 0\right) \cap \left(v_{ij} = 0\right)$$

5) Повторить четвертый шаг для всех вершин отмеченной группы.

6) Повторить шаги 3-5 для всех следующих групп, кроме последней.

7) Повторять метод до тех пор, пока не останется ни одного объединения.

Следует отметить, что помимо уменьшения времени выполнения параллельного алгоритма данный метод позволяет в некоторых случаях сократить и ширину графа алгоритма за счет укрупнения операций и заполнения образующихся пустот (простоев) операциями с других процессоров.

Метод поиска оптимального по высоте информационного графа по списку следования заключается в разбиении совокупности вершин на группы (построение параллельной формы) и перемещение вершин между группами с целью уменьшения числа ярусов и достижения максимальной плотности по каждому ярусу.

Метод поиска оптимального по высоте информационного графа

1. Рассчитать значение теоретической минимальной ширины d информационного графа.

2. Добавить в граф вершину. Все выходные вершины соединить с новой вершиной. Таким образом, в графе будет всего одна выходная *n+1*-я вершина.

3. Составить матрицу следования информационного графа и ее замыкания A. Вместо единиц в матрице следования будут стоять значения времени выполнения j-й операции.

4. Построить соответствующий информационному графу временной список следования $V_0 = (V_1, V_2, t)$.

5. Построить множество выходных вершин V_k.

6. Найти множество $V = V_2 - V_1$. Вершины, вошедшие в это множество, составят группу вершин M_i, принадлежащих одному i-му ярусу.

– Если число вершин множества $V = V_2 - V_1$ превышает d, то возможны следующие варианты:

a) во множестве M_i существует ровно d вершин с одинаковым временем выполнения. В этом случае эти d вершин с одинаковым временем выполнения составят множество M_i.

b) Во множестве M_i существует больше чем d вершин с одинаковым временем выполнения. В этом случае любые d вершин с одинаковым временем выполнения составят множество M_i.

c) Во множестве M_i существует меньше d вершин с одинаковым временем выполнения. В этом случае множество M_i составят d вершин с минимальным временем выполнения.

Остальные вершины необходимо включить в следующую группу M_{i+1}. При повторном проходе этого шага в группу M_{i+1} будут добавлены вершины.

– Если число вершин множества $V = V_2 - V_1$ меньше числа d, то во множество M_i следует перенести вершины из множества $V_к$, удовлетворяющие правилу: $V_{kj} \cap V_1 = \varnothing$, т.е. вершины из $V_к$, отсутствующие во множестве

V_1. Эти вершины могут быть взяты в произвольном порядке из множества: $V_k - V_1$.

7. Удалить из списка смежности все пары, конечная вершина которых (V_2) совпадает с одной из вершин множества V, и построить тем самым список V_i.

8. Если i=1, то вернуться на шаг 6.

9. Во множестве M_{i-1} найти вершину с минимальным временем выполнения m_{i-1k}. Если все вершины имеют одинаковое время выполнения, то перейти к шагу 16.

10. Составить множества:

- $S_{m_{i-1k}M_i}$ – совокупность вершин из множества M_i, связанных с вершиной m_{i-1k} одним ребром и являющихся для данного ребра конечными вершинами.
- $S_{(M_{i-1}-m_{i-1k})M_i}$ – совокупность вершин из множества M_i, связанных с вершинами множества M_{i-1} (за исключением вершины m_{i-1k}) одним ребром и являющихся для данного ребра конечными вершинами.

Если во множестве M_{i-1} есть укрупненные операции, то рассматривается всегда только последняя добавленная при укрупнении вершина.

11. Найти разность: $S = S_{m_{i-1k}M_i} - S_{(M_{i-1}-m_{i-1k})M_i}$.

- Если $S = \varnothing, S_{m_{i-1k}M_i} \neq \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то перейти к шагу 5.
- Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} = \varnothing$, то $S = M_i$.
- Если $S_{m_{i-1k}M_i} = \varnothing, S_{(M_{i-1}-m_{i-1k})M_i} \neq \varnothing$, то $S = M_i - S_{(M_{i-1}-m_{i-1k})M_i}$.

12. Найти во множестве вершину с минимальным временем выполнения $s_{\min}$. Удалить эту вершину из множества M_i. Добавить эту вершину во множество M_{i-1}, укрупнив операцию m_{i-1k} путем ее слияния с операцией $s_{\min}$:

$$m_{i-1k} = m_{i-1k} \cup s_{\min}, t_{i-1k} = t_{i-1k} + t_{s_{\min}}$$

13. Если $M_i = \varnothing$, то удалить множество $M_i = \varnothing$ и сдвинуть счетчик множеств $i = i - 1$. Перейти к шагу 5. Если $M_i \neq \varnothing$, то перейти к шагу 10.

14. Если список не пустой, то вернуться на шаг 5.

15. Если в списке не осталось ни одной пары, то метод окончен.

Общее время работы метода составит $T = \sum_{i=1}^{k} t_{i,\max}$, где k – число групп, $t_{i,\max}$ – максимальное время среди операций в i-й группе.

Для тестирования методов оптимизации параллельных программ по ширине и времени выполнения было создано программное обеспечение, позволяющее:

- ✓ строить автоматически ориентированные графы, соответствующие информационным графам в строгой параллельной форме;
- ✓ строить автоматически взвешенные ориентированные графы, соответствующие информационным графам в строгой параллельной форме;
- ✓ строить вручную произвольные ориентированные графы и задавать веса;
- ✓ рассчитывать высоту, ширину, теоретическую и практическую минимальную ширину графа, ранние и поздние строки выполнения операций и др. параметры (рис.3);
- ✓ оптимизировать информационный граф по ширине;
- ✓ оптимизировать информационный граф по времени выполнения;
- ✓ оптимизировать информационный граф по ширине и времени выполнения комбинированным методом, полученным из объединения и модификации методов оптимизации, представленных в главах 3 и 4;
- ✓ строить временные диаграммы;
- ✓ строить матрицы смежности, следования, списков связности;
- ✓ выводить совокупности операций, выполняемых каждым процессором.

По результатам исследования можно предложить следующую методику применения разработанных методов:

1) Разбить совокупность операций на группы с применением первой части метода оптимизации информационного графа по ширине.

2) Вычислить теоретическую оценку минимальной ширины графа *d*.

3) Если $d>n$ (где n – число процессоров), то по результатам применения шага1 добавить дополнительные группы. Если $d \leq n$, то скорректировать значение d с условиями применяемого вычислительного кластера.

4) Провести оптимизацию по ширине с помощью второй части метода оптимизации информационного графа по ширине.

5) Если, в результате, сохранилось по-прежнему много простоев, то применить к результатам шага 1 метод оптимизации информационного графа по времени выполнения.

Тестирование полученных методов показал, что применение метода оптимизации информационного графа по ширине, как правило, сокращая ширину графа, оставляет достаточно много простоев процессоров. Добиться наилучшего результата можно только применив после первого шага оптимизацию по времени или комбинированный метод. В зависимости от структуры информационного графа в разных случаях превалируют или тот или другой метод.

Четвертая глава посвящена исследованиям в области баз данных. Современные базы данных охватывают огромное количество атрибутов предметной области, что значительно усложняет математическую модель. Но развитие вычислительных систем с параллельной архитектурой позволяет значительно ускорить процесс обработки этих моделей. В главе проанализированы проблемы аномалии избыточности в различных моделях данных.

Независимо от вида модели одними из важных характеристик являются зависимости одних атрибутов от других. В реляционной теории существует понятие функциональной зависимости. Для представления функциональных зависимостей и проведения их анализа наиболее полный инструментарий представляет теория графов.

В реляционной модели наличие нескольких путей из одного узла в другой узел означает наличие транзитивных зависимостей в отношении. Устранение транзитивных зависимостей превращает орграф функциональных зависимостей в дерево. Этот процесс основан на следующем правиле: если из узла A в узел B существует несколько путей: $A \to A_{11} \to \ldots \to A_{1n} \to B$ и $A \to A_{21} \to \ldots \to A_{2m} \to B$, то следует удалить из орграфа дугу A_{1n} при $m < n$, или дугу A_{1m} при $n < m$.

После получения дерева функциональных зависимостей следует найти точку декомпозиции отношения, т.е. узел, соответствующий атрибуту (или множеству атрибутов), который является граничным при делении отношения. Другими словами, этот узел входит в качестве первичного ключа в первое новое отношение и внешнего ключа во второе новое отношение.

Чтобы узел орграфа можно было рассматривать как точку декомпозиции, необходимо, чтобы он удовлетворял следующим правилам.

Правило 1. Точка декомпозиции должна быть хотя бы для одной дуги начальным узлом.

Правило 2. Точка декомпозиции должна быть вершиной дерева орграфа функциональных зависимостей.

Правило 3. Поддерево с вершиной в точке декомпозиции является концевым поддеревом орграфа и не содержит в себе других поддеревьев.

При наличии нескольких узлов, удовлетворяющих всем трем правилам, в качестве точки декомпозиции можно брать любой, результат будет всегда один.

На основе проведенных исследований можно сформулировать метод проведения декомпозиции отношения на основе орграфа его функциональных зависимостей.

Метод проведения декомпозиции отношения

1. Провести стягивание подграфов в деревья.
2. Найти все узлы, удовлетворяющие всем трем правилам точки декомпозиции. Допустим это узлы v_1 и v_2 .
3. Выбрать один из найденных узлов в качестве точки декомпозиции.
4. В первое отношение, которое всегда будет удовлетворять BCNF, записать атрибуты, соответствующие узлам-потомкам точки декомпозиции и самой точки декомпозиции.
5. Из орграфа удалить:
    a) все узлы-потомки точки декомпозиции;
    b) все дуги, исходящие из точки декомпозиции.

    Точка декомпозиции и все дуги, входящие в нее, остаются в орграфе.
6. Во второе отношение, которое может и не удовлетворять BCNF, записать все атрибуты, не вошедшие в первое отношение и атрибуты, соответствующие точке декомпозиции.
7. Повторить метод для второго отношения.

Метод является итерационным и повторяется до тех пор, пока в графе не останется ни одного узла, удовлетворяющего всем трем правилам точки декомпозиции.

Данному методу можно поставить в соответствие матричный метод. Пусть имеется матрица смежности $A(m \times n)$. Имена всех атрибутов, а значит и столбцов, хранятся в массиве $T(n)$. Каждый элемент массива $T(n)$ атомарен. Имена атрибутов, соответствующих строкам, хранятся в массиве $S(m)$. Каждый элемент массива $S(m)$ содержит один или несколько атрибутов, соответствующих конкретному узлу графа.

Метод преобразования матрицы смежности

1. Провести стягивание строк.
   a) Найти строку a_{ij}, содержащую всего одну единицу.
   b) Определить столбец t_j, в котором находится эта единица.
   c) Если в столбце t_j больше нет единиц, то логически прибавить ее к строке или строкам, содержащим атрибут с именем, равным имени столбца t_j.
   d) Удалить i-ю строку из матрицы A. Удалить элемент s_i из массива S. Повторить эти шаги для каждой строки с единственной единицей.
2. Найти строку, содержащую единицы только в тех столбцах, имена которых отсутствуют среди элементов массива S. Если такой строки нет, перейти к шагу 6.
3. Включить в новое отношение все атрибуты, соответствующие столбцам с единицами в найденной строке и атрибуты, соответствующие этой строке.
4. Удалить из матрицы столбцы с единицами в найденной строке и саму строку.
5. Повторить действия 2-5 до тех пор пока не останется в матрице строк, содержащих единицы только в тех столбцах, имена которых отсутствуют среди элементов массива S или до тех пор, пока в матрице не останется одна строка, полностью состоящая из единиц.

Все атрибуты, соответствующие столбцам и строкам оставшейся матрицы, включить в последнее отношение.

Далее в главе рассмотрены свойства функциональных зависимостей. Всем свойствам поставлены в соответствие операции над матрицей смежности. В результате сформулированы методы получения новых функциональных зависимостей на основе декларированных и метод определения первичного ключа отношения.

В пятой главе приведены результаты применения разработанных в первых четырех главах методов к четырем прикладным задачам.

Задача 1. Оптимизации параллельных алгоритмов процесса построения реляционной схемы базы данных.

Для того чтобы разработать оптимальный по времени и числу процессоров параллельный алгоритм, проводится декомпозиция последовательного алгоритма. С целью более наглядного представления последовательный алгоритм нормализации реляционной схемы представлен в виде синтаксической диаграммы. Отрицательной чертой параллельной формы синтаксической диаграммы укрупненных операций является неравнозначность временных трудоемко-

стей некоторых блоков. Тем не менее показано, что экономия времени при параллельном выполнении алгоритма будет равна $O(n^2)$ без учета временных затрат на передачу данных. Больший запас внутреннего параллелизма можно выделить при детализации блоков построенной синтаксической диаграммы.

Фактически весь алгоритм был разбит на сравнительно небольшие операции и между ними установлены логические и информационные зависимости. Собрав все информационные подграфы блоков синтаксической диаграммы в единое целое, получаем информационный граф алгоритма нормализации реляционных отношений при $m = 6, n = 11$ (рис.2).

С помощью метода оптимизации параллельной программы по ширине без учета времени и программы, соответствующей этому методу, все вершины графа были разбиты на группы. Для построенного информационного графа была найдена минимальная теоретическая и практическая ширина графа. На основе проведенных исследований построен оптимальный по числу процессоров и времени выполнения параллельный алгоритм проектирования реляционной схемы базы данных.

С помощью метода оптимизации информационного графа по времени сроки выполнения рассматриваемого параллельного алгоритма удалось сократить в 3 раза, сохранив при этом ширину информационного графа в пределах 18 процессоров.

Задача 2. Организация преобразований в системах, основанных на расписании.

Важными требованиями для построения кластера являются высокая степень автоматизации процесса выделения ресурсов по запросу пользователя, обеспечение постоянного запаса вычислительных мощностей для удовлетворения возрастающих нужд заказчиков и высокого уровня доступности используемых систем.

В главе рассмотрена архитектура кластера, которая включает: «роли», предназначенные для получения и обработки входящих запросов через службы сервера; «сотрудников», получающих данные из хранилища данных и обменивающиеся информацией с «ролями» посредством запросов, размещаемых в очередях; диспетчер очереди, осуществляющий распределение задач из очереди по ролям-сотрудникам; диспетчер программ, необходимый для выбора программы из базы данных и ее параметров и передачи в очередь; а также очередь с информацией о задачах, поступивших на выполнение и базу данных программ.

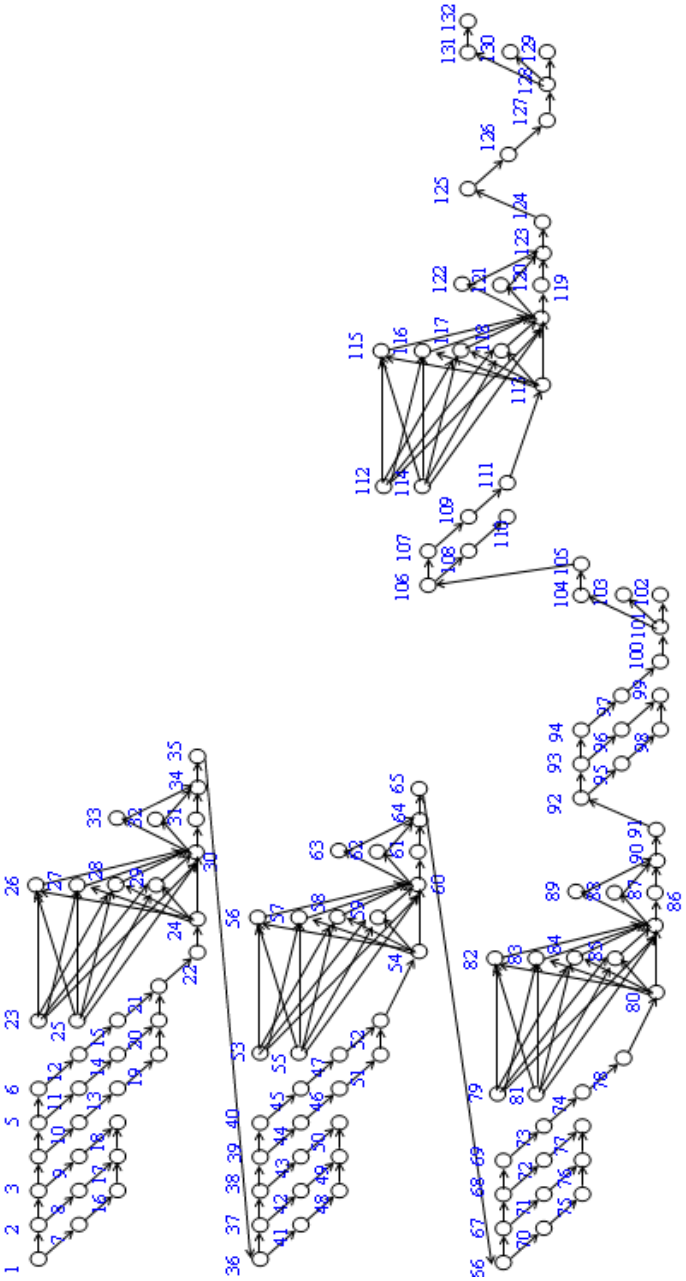

Рисунок 2. – Информационный граф алгоритма нормализации реляционных отношений

В главе показано, что процедура распределения задач по процессорам в целом аналогична распределению операций параллельного алгоритма на распределенных вычислительных системах.

Для наглядного представления взаимосвязей процессоров и задач и поиска формализованного метода управления процессом распределения задач из очереди между ролями-сотрудниками можно применить взвешенные ориентированные графы. Наиболее подходящим видом графа является «детализированный» граф с весами – параметрами: «Процесс» $P\{t_j, t_{11}, \ldots, t_{i1}, \ldots, t_{n1}\}$ и «Сроки» $S\{t_j + t_{11}, \ldots, t_j + t_{i1}, \ldots, t_j + t_{n1}\}$. В работе создан метод построения графа распределения задач.

Метод построения графа распределения задач

1. Первой задаче, поступившей в очередь, ставится в соответствие узел с номером задачи и ребро с параметром «Процесс» *P*.

2. Следующей задаче, поступившей в очередь, ставится в соответствие узел с номером задачи. Если число свободных процессоров N_c больше запрашиваемого Диспетчером программ количества процессоров nj, необходимых для эффективного решения *j*-й задачи ($N_c \geq n_j$), то добавляется ребро с весами *P* и *S*. Если $N_c < n_j$, то добавляем ребро с весом пустого множества *P{}*. Если Nc=0, то ребро не добавляем.

3. Новая задача в очереди изображается в зависимости от числа свободных процессоров и наличия других задач с более ранним временем постановки их в очередь. Если в очереди нет задач, или существующие задачи имеют более поздний срок постановки в очередь, то шаг 2. Если в очереди есть задача, поступившая ранее, то решается локальная задача:

Пусть t_{i4} – время поступления ранней задачи, находящейся в ожидании вычислительных ресурсов; t_{j4} – время поступления новой задачи, m_i – число недостающих вычислительных ресурсов; n_j – число процессоров, необходимых для решения *j*-й задачи; $\tau_{n_j k}$ – время работы *k*-го процессора, задействованного в решении *j*-й задачи. Пусть $R = S_0 \cup S_1 \cup \ldots \cup S_j$ - параметр «Ресурсы», содержащий номера процессоров, отсортированных по времени работы в порядке возрастания.

Запуск *j*-й задачи «вне очереди» (ранее *i*-й задачи) возможен в случае, если выполняется условие: $t_{j4} - t_{i4} + R_{m_i} + \max_{1 \leq k \leq n_i}(\tau_{n_i k}) \leq t_{i3}$. Если запуск *j*-й

задачи невозможен, то на графе для добавления *i*-й задачи следует вычеркнуть из множества *R* первые m_i минимальных числа и провести ребра к узлу с номером *i* от тех узлов во множестве *S*, в которых есть вычеркнутые числа. Время начала решения задачи будет вычисляться по правилу: $t_i = R_{m_i}$.

Реализовать метод распределения задач по ролям-сотрудникам на графах в Диспетчере очереди можно, если перевести его из графической формы в матричную.

Матричный метод распределения задач

Пусть: кластер содержит N процессоров; массив «Очередь» $O(M \times (N+3))$ содержит информацию о номерах задач, поступивших в кластер (первая строка); времени поступления (вторая строка); количестве необходимых процессоров (третья строка) и позиции в списке расписаний (четвертая строка); число M – размер очереди (динамический или статический); массив «Процесс» $P(N \times 3)$ содержит информацию о номерах, вычисляемых в настоящее время на кластере задач (первая строка), расписании (вторая строка) и параметре S (третья строка): $S_j = t + P_{2j}$, где $j = \overline{1; N}$, t – время начала решения задачи (текущее время), которое можно нигде не фиксировать. Величина К – число занятых в текущий момент процессоров. В начале метода все массивы являются пустыми и К=0.

Шаг 1. При поступлении задачи в кластер Диспетчер очереди «ставит ее в очередь», т.е. прописывает параметры задачи в массиве О.

Шаг 2. С интервалом τ Диспетчер очереди проверяет очередь и решает локальную задачу о распределении задач из очереди.

Шаг 2.1.В массиве *O* пройти по третьей строке и найти первую задачу в порядке возрастания номеров задач, для которой выполняется условие: $O_{3j} \le N - K$, где j - задача, извлекаемая из очереди на выполнение.

Шаг 2.2. Если не существует такая j-я задача, для которой выполняется условие $O_{3j} \le N - K$, то шаг 3, иначе шаг 2.3.

Шаг 2.3. Запуск задачи на выполнение. Для этого необходимо занести информацию о задаче в массив Р и передать задачу Диспетчеру программ. Информация в массив Р заносится в нулевые столбцы в порядке следования слева направо.

Шаг 2.4. Увеличить число задействованных процессоров: $K = K + n$, где n – число процессоров, необходимых для решения задачи.

Шаг 2.5. Удалить запись из очереди, обнулив значения.

Шаг 3. С интервалом τ Диспетчер очереди проверяет массив P на наличие освободившихся процессоров. Для этого необходимо посчитать и обнулить все столбцы массива P, для которых $P_{3j} = t$, где t – текущее время. Количество этих столбцов k вычесть из K: $K = K - k$. Если $k \neq 0$ и очередь не пустая, то шаг 2, иначе шаг 1.

С помощью программы Protege 4.0 построена онтологическая модель процесса распределения задач на кластере, которая отражает взаимосвязи между составляющими кластера. С помощью dl-запросов к owl-файлу построенной модели можно получить сведения о наличии свободных процессоров в текущий момент времени, выполняемых задачах, времени окончания решения задач и другую информацию.

Задача 3. Построение и оптимизация параллельных алгоритмов приведения разреженной матрицы к форме SBBDF на примере модели управления процессом электролиза алюминия.

Процесс электролиза алюминия представляет собой взаимосвязь множества параметров. Это процесс, который является сложной многомерной многосвязной системой. Применение единого топологического структурного метода для представления исследуемого объекта в целом приводит к системе линейных уравнений большой размерности. Анализ получаемой системы линейных уравнений показал, что система является разреженной и, следовательно, применение к ней специальных методов может позволить уменьшить трудоемкость по ее решению. С-граф системы является связным и не может быть разбит на компоненты. Матрица структуры, соответствующая С-графу, несимметрична и незнакоположительна, прямоугольна. Поэтому, методы, существующие для разреженных систем, требуют модификации с целью успешного применения к данной системе линейных уравнений. С этой целью разработан метод приведения разреженной матрицы к форме BDF. Легкость метода в понимании, минимальная трудоемкость и возможность использования компактной схемы хранения разреженной матрицы позволяет использовать его в дальнейших исследованиях.

В главе представлены результаты построения синтаксической диаграммы последовательного алгоритма приведения разреженной матрицы к форме BDF на примере модели управления процессом электролиза алюминия и информационного графа алгоритма с учетом всех блоков синтаксической диаграммы и проведенной детализации.

С помощью метода оптимизации параллельной программы по ширине без учета времени и программы, соответствующей этому методу, все вершины графа были разбиты на группы.

С помощью метода оптимизации информационного графа по времени сроки выполнения рассматриваемого параллельного алгоритма удалось сокра-

тить в 7 раз, сохранив при этом ширину информационного графа в пределах 9 процессоров.

Задача 4. Разработка матричного метода построения дерева решений.

Обнаружение знаний в базах данных является стремительно увеличивающейся областью, развитие которой вызвано большим интересом к настоятельным практическим, социальным и экономическим нуждам. Современные базы данных содержат так много данных, что практически невозможно вручную проанализировать их для извлечения ценной информации, помогающей принимать важные решения. В связи с этим спрос на методы автоматического исследования данных с каждым годом постоянно увеличивается.

Деревья решений – один из таких методов автоматического анализа данных, основанный на эвристических алгоритмах. Большинство из известных эвристических алгоритмов являются «жадными алгоритмами». Если один раз был выбран атрибут, и по нему было произведено разбиение на подмножества, то алгоритм не может вернуться назад и выбрать другой атрибут, который дал бы лучшее разбиение. И поэтому на этапе построения нельзя сказать даст ли выбранный атрибут, в конечном итоге, оптимальное разбиение. Еще одним недостатком эвристических алгоритмов является небольшой запас внутреннего параллелизма.

Устранить все недостатки эвристических алгоритмов возможно при их замене матричными прямыми алгоритмами.

Пусть обучаемое множество представляют собой набор объектов C = C1, C2, ... Каждый объект Ci = x1, x2, ... является вектором, где x1, x2, ... представляют собой атрибуты объекта.

Обозначим: m – число атрибутов, n – объем выборки, i – номер атрибута, i=1..m, m_i – число значений i-го атрибута, j – номер значения в атрибуте, j=1..m_i, cij – количество значений в выборке, соответствующих j-му значению i-го атрибута. Например, если C1 – первый атрибут и C1={sunny, overcast, rain}, то c13=количество в выборке значений «rain».

На основе данных об атрибутах объектов и множестве решений S можно построить граф взаимосвязей значений атрибутов G(V,E), где $V = \{v_1, v_2, \ldots, v_p\}$ и $E = \{e_1, e_2, \ldots, e_q\}$ - соответственно множества вершин и ребер графа. Множество V – совокупность всех значений атрибутов обучаемого множества, E – совокупность всевозможных в обучаемом наборе связей между атрибутами. В полученном графе каждая вершина, соответствующая одному значению своего атрибута, будет связана с каждой вершиной, соответствующей отдельному значению другого атрибута.

Граф взаимосвязей значений атрибутов не является ориентированным и по своей сути характеризует многосвязную сложную систему.

Систему взаимосвязанных значений атрибутов можно в соответствии с определением сложной системы разбить на две подсистемы по принципу их принадлежности тому или иному значению из множества решений S.

Собственно на этом разбиении системы значений атрибутов на несколько подсистем и основан метод построения дерева решений, представляющий собой прямой метод.

Матричный метод построения дерева решений

1. Разбить обучаемое множество на два непересекающихся подмножества S1 и S2, в соответствии с принципом принадлежности отдельного объекта той или иной группе множества решений S.

2. Построить взвешенный граф взаимосвязей значений атрибутов для каждого подмножества.

3. Построить матрицу смежности для каждого графа. Так как матрица будет симметричной относительно главной диагонали, то можно изначально строить верхнее треугольную матрицу смежности.

4. На главной диагонали матрицы смежности разместить значения cij.

5. Рассчитать приоритетность атрибутов в соответствии с формулами энтропии и Gain-индекса:

$$Entropy(S)=-\sum_{i=1}^{2} p_i \log_2 p_i \ ;$$

$$Gain(S,Ci)=Entropy(S)-\sum_{i=1}^{2}\frac{|S_i|}{|S|}Entropy(S_i) \ ;$$

6. Если в матрице смежности Sk существует нулевая строка, то можно сформулировать правило Cij -> Sk.

7. Провести поэлементное сравнение матриц множеств S1 и S2. Если T1(i,j)=0, а T2(i,j)>0, то можно сформулировать правило: Ck,Cr->S2, где Ck – атрибут и его значение, соответствующие i-й строке, а Cr - атрибут и его значение, соответствующие j-му столбцу матрицы смежности.

8. Если T2(i,j)=0, а T1(i,j)>0, то можно сформулировать правило: Ck,Cr->S1, где Ck – атрибут и его значение, соответствующие i-й строке, а Cr - атрибут и его значение, соответствующие j-му столбцу матрицы смежности.

9. Если в i-й строке несколько элементов, удовлетворяющих условию 7 или 8, то выбирается атрибут с наибольшим Gain-индексом.

Один из вопросов, который возникает в эвристических алгоритмах дерева решений – это оптимальный размер конечного дерева. Так, небольшое дерево может не охватить ту или иную важную информацию о выборочном пространстве. Тем не менее, трудно сказать, когда алгоритм должен остановиться, потому что невозможно спрогнозировать, добавление какого узла позволит значительно уменьшить ошибку.

В диссертационной работе для решения задачи глубины дерева предлагается рассчитать вес каждой ветви по формуле: $p_{ij} = \frac{T(i,j)}{Cij} 100\%$.

Следует отметить, что совокупность ветвей с весом, равным 100%, получаемая с помощью описанного прямого матричного метода, полностью соответствует совокупности ветвей, получаемых с помощью эвристического алгоритма ID3.

По разработанному матричному методу построения дерева решений был создан параллельный алгоритм. После применения к параллельному алгоритму методов оптимизации по ширине и высоте были получены результаты, представленные в таблице:

	До оптимизации	После оптимизации по ширине	После оптимизации по ширине и времени
Процессоры	24	16	15
Время	320	340	390

Разработанный метод позволяет получать более широкую совокупность правил по сравнению, к примеру, с методом ID3. Совокупность правил может быть отрегулирована с помощью выведенных формул, позволяющих также оценить вероятность срабатывания правила. С помощью данного метода можно также регулировать глубину искомого дерева решений. Разработанный матричный метод также предпочтительнее эвристических алгоритмов за счет более низкой трудоемкости.

В **заключении** сформулированы основные научные и практические результаты исследований, выполненные в диссертационной работе.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Предложено представление функциональных зависимостей реляционных схем в матричном виде и получены матричные аналоги свойств функциональных зависимостей.
2. Разработан метод построения реляционной схемы на основе атрибутов предметной области и декларированных функциональных зависимостей. Все получаемые схемы удовлетворяют нормальной форме Бойса-Кодда.
3. Разработан метод нахождения первичного ключа отношения на основе функциональных зависимостей.
4. Выведены оценки теоретической и практической минимальной ширины информационного графа, по которым можно принимать решение о дальнейшем преобразовании информационного графа.

5. Разработаны методы оптимизации информационного графа параллельного алгоритма по числу процессоров, задействованных в решении поставленной прикладной задачи. Методы состоят из двух частей, которые могут применяться независимо друг от друга. Методы основаны на матрицах смежности, списках смежности и следования. В силу того, что при реализации методов достаточно хранить номера только ненулевых элементов, то методы являются экономными в отношении используемой памяти и времени выполнения. Методы обеспечивают равномерную загрузку процессоров и построены таким образом, что количество процессоров в каждую единицу времени будет приближено к теоретическому минимальному числу.

6. Разработан метод оптимизации параллельного алгоритма по времени выполнения, позволяющий осуществить равномерную загрузку процессоров, сократить время решения поставленной задачи и уменьшить ширину информационного графа.

7. Разработан метод приведения матрицы к блочной треугольной форме с помощью элементарных преобразований первого типа, т.е. с сохранением первоначальных значений элементов матрицы, что важно при эквивалентных преобразованиях параллельных алгоритмов и реляционных схем.

8. Разработано программное обеспечение для тестирования полученных методов и проведения дальнейших исследований в области параллельных вычислений, баз данных и теории принятия решений.

9. Разработан метод построения детализированного взвешенного ориентированного графа распределения задач по ролям-сотрудникам при организации вычислений на кластере и на его основе построен матричный метод распределения задач по ролям-сотрудникам, обладающий следующими характеристиками: возможность легкого переноса на любой императивный язык программирования и большой запас внутреннего параллелизма.

10. С помощью программы Protege 4.0 построена онтологическая модель процесса распределения задач на кластере, отражающая взаимосвязи между его составляющими и позволяющая с помощью dl-запросов получить информацию о процессорах и задачах, необходимую для дальнейшего принятия решения о запуске новой задачи из очереди.

11. Разработан матричный метод построения дерева решений, который позволяет получать более широкую совокупность правил по сравнению, к примеру, с методом ID3. Совокупность правил может быть отрегулирована с помощью выведенных формул, позволяющих также оценить вероятность срабатывания правила. Разработанный матричный метод также предпочтительнее существующих эвристических алгоритмов за счет более низкой трудоемкости и полноты получаемых правил.

СПИСОК РАБОТ, ОПУБЛИКОВАННЫХ ПО ТЕМЕ ДИССЕРТАЦИИ

Публикации в изданиях, рекомендованных ВАК России:

1. Шичкина, Ю.А. Применение параллельной формы информационного графа в задачах распараллеливания с использованием разреженных матриц / Ю.А. Шичкина // Информационно-измерительные и управляющие системы. – 2008. – Т.6, № 4,– С. 63 – 67.

2. Шичкина, Ю.А. Применение матричной алгебры для оптимизации параллельных алгоритмов / Ю.А. Шичкина // Информационно-измерительные и управляющие системы. – 2008. – Т.6, № 10. – С. 72 – 77.

3. Шичкина, Ю.А. Сокращение высоты информационного графа параллельного алгоритма / Ю.А. Шичкина // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. – 2009. – № 3 (80). – С.148– 152.

4. Шичкина, Ю.А. Применение онтологического моделирования для описания параллельных процессов / Ю.А. Шичкина // Информационно-измерительные и управляющие системы. – 2009. – Т.7, № 4.– с.27 – 28.

5. Шичкина, Ю.А., Преобразование информационного графа параллельного алгоритма / Ю.А. Шичкина // Современные технологии. Системный анализ. Моделирование. – 2010 – № 1 (25). – С.133– 137.

6. Шичкина, Ю.А. Оптимизация параллельного алгоритма по числу процессов / Ю.А. Шичкина, В.И. Воробьев // Вестник гражданских инженеров.– 2008. – № 2(15). – С. 92 – 97.

7. Шичкина, Ю.А. Алгоритмизация процесса получения новых функциональных зависимостей на основе декларированных / Ю.А. Шичкина // Автоматизация и современные технологии. Москва, "Издательство Машиностроение". – 2010. – № 8. – С.144 – 150.

8. Шичкина, Ю.А. Преобразование разреженной матрицы к треугольному виду / Ю.А. Шичкина, М.С. Куприянов // Вестник гражданских инженеров. 2013. – № 3 (38). – С. 233 – 236.

9. Шичкина, Ю.А. Применение блочной треугольной формы разреженной матрицы для преобразования информационного графа алгоритма / Ю.А. Шичкина // Вестник гражданских инженеров.– 2012 – № 4(33). – С. 259 – 263.

10. Шичкина, Ю.А. Применение списков следования для преобразования информационного графа по высоте / Ю.А. Шичкина // Системы. Методы. Технологии. 2011. – № 9. - С. 68 – 77.

11. Шичкина, Ю.А. Применение теории графов для разработки прямого метода построения деревьев решений / Ю.А. Шичкина, М.С. Куприянов // Системы. Методы. Технологии. 2012. – № 4(16). – С. 62 – 65.

12. Шичкина, Ю.А. Распределение параллельных программ между ролями-сотрудниками в облачном кластер / Ю.А. Шичкина // Системы. Методы. Технологии. 2012. – № 2. – С. 59 – 63.

13. Шичкина, Ю.А. Применение блочной треугольной формы разреженной матрицы для преобразования информационного графа алгоритма / Ю.А. Шичкина // Современные технологии. Системный анализ. Моделирование. 2010. – № 4. – С. 117 – 120.

14. Шичкина, Ю.А. Применение теории графов для разработки прямого метода построения деревьев решений / Ю.А. Шичкина, М.С. Куприянов // Известия СПбГЭТУ "ЛЭТИ". 2012. – № 9. – С. 68 – 74.

15. Шичкина, Ю.А. Математическое моделирование системы качества как механизм улучшения его показателей. / Ю.А. Шичкина, Ю.В.Планкова // Системы. Методы. Технологии. 2012. – № 3. – С. 53 – 55.

16. Шичкина, Ю.А. Применение методов автоматического анализа при формировании контрольных цифр приема в учреждения профессионального образования / Ю.А. Шичкина, Ю.В.Планкова // Системы. Методы. Технологии. 2013. – № 2 (18). – С. 82 – 87.

Монография

17. Шичкина, Ю.А. Организация вычислений на распределенных системах / Ю.А. Шичкина. – ФГБОУ ВПО БрГУ, 2013. – 176с.

Учебные пособия

18. Шичкина, Ю.А. Управление данными. Учебное пособие. / Ю.А. Шичкина. – Братск: ГОУ ВПО БрГУ, 2007. – 138с.

19. Шичкина, Ю.А. Проектирование баз данных в среде Delphi 7.0 Учебное пособие. / Ю.А. Шичкина. – Братск: ГОУ ВПО БрГУ, 2008. – 132с.

20. Шичкина Ю.А. Проектирование баз данных в среде C# Учебное пособие. / Ю.А. Шичкина, В.С.Кедрин. – Братск: ГОУ ВПО БрГУ, 2011. – 122с.

Научные сборники

21. Шичкина, Ю.А. Определение функции времени выполнения алгоритма / Ю.А. Шичкина // Математическое моделирование, численные методы и комплексы программ: Межвуз.темат.сб.тр. – СПб.:СПбГАСУ – 2004. – Вып.10. – С.218 – 220.

22. Шичкина, Ю.А. Проблема улучшения качества проектирования реляционных баз данных / Ю.А. Шичкина // Математическое моделирование и комплексы программ: Межвузовский тематический сборник трудов. – СПб.:СПбГАСУ. –2005. – Вып.11. – С.124 – 127.

23. Шичкина, Ю.А. Применение формализованного метода неограниченного хаоса в реляционных моделях баз данных / Ю.А. Шичкина // Математическое моделирование и комплексы программ: Межвузовский тематический сборник трудов. – СПб.:СПбГАСУ. –2009. – Вып.15. – С.199 – 204.

24. Шичкина, Ю.А. Декомпозиция реляционного отношения методом ограниченного хаоса / Ю.А. Шичкина // Системы. Методы. Технологии. – Братск:БрГУ,2010. – № 4. – С.152 – 156

25. Шичкина, Ю.А., Построение информационного графа нормализации реляционных отношений. / Ю.А. Шичкина // Системы. Методы. Технологии. . – Братск:БрГУ, 2010. – № 1 (5). – С.74 – 84.

26. Шичкина, Ю.А., Распараллеливание алгоритма нормализации реляционных отношений. / Ю.А. Шичкина // Системы. Методы. Технологии. . – Братск:БрГУ, 2010. – № 2 (6). – С.69-77.

27. Шичкина, Ю.А. Применение онтологического моделирования для описания параллельных процессов. / Ю.А. Шичкина, В.В.Александров, В.А.Сарычев. // Программируемые инфокоммуникационные технологии.– М.:Радиотехника, 2009. – С.27 – 28.

28. Шичкина, Ю.А., Комбинированный метод многопараметрической оптимизации взвешенного информационного графа. / Ю.А. Шичкина // Системы. Методы. Технологии. – Братск:БрГУ, 2010. – № 3 (7). – С.76 – 82.

Тезисы докладов на конференциях и семинарах

29. Шичкина, Ю.А. От метода неограниченного хаоса к матричному методу оптимизации реляционных моделей баз данных / Ю.А. Шичкина // Наука в информационном пространстве: материалы V Междунар.науч.-практ.конф,30 – 31 окт.2009.: В 6 т. – Т.1. – Днепропетровск, 2009. – С.120 – 124.

30. Шичкина, Ю.А. Временные оценки параллельного алгоритма / Ю.А. Шичкина // Труды Братского государственного университета: Сер.: Естественные и инженерные науки - развитию регионов Сибири. – Братск: БрГУ, 2009. – С.86 – 87.

31. Шичкина, Ю.А. Временная сложность алгоритмов / Ю.А. Шичкина // Естественные и инженерные науки - развитию регионов: Материалы межрегиональной НТК. – Братск:БрГТУ,2004. – С.112 – 113.

32. Шичкина, Ю.А. Разреженные матрицы в базах данных / Ю.А. Шичкина // Естественные и инженерные науки – развитию регионов: Материалы межрегиональной НТК. – Братск:БрГТУ,2004. – С113 – 114.

33. Шичкина, Ю.А. Автоматизация процесса нормализации реляционных баз данных / Ю.А. Шичкина // Наука.Технологии.Инновации. Материалы всероссийской научной конференции молодых ученых в 6-ти частях. – Новосибирск:Изд-во НГТУ, 2004. Часть 1. – С.176 – 179.

34. Шичкина, Ю.А. Взвешенные графы в параллельных вычислениях / Ю.А. Шичкина // Современные наукоемкие технологии. – 2009. – № 11. – С. 93 – 95.

Свидетельства о регистрации программ для ЭВМ и баз данных

35. Шичкина, Ю.А. Преобразование разреженных матриц (BDF+) v1.0: программа для ЭВМ. Св. ГР. №2012618115 Рос. Федерация; зарег. в реестре Федер. службы по интеллектуальной собственности, пат. и товарным знакам 10.11.2012.

36. Шичкина, Ю.А. Построитель реляционных схем (Create Scheme) v1.0: программа для ЭВМ. Св. ГР. №2012618116 Рос. Федерация; зарег. в реестре Федер. службы по интеллектуальной собственности, пат. и товарным знакам 10.11.2012.

37. Шичкина, Ю.А. Распараллеливание последовательных алгоритмов v1.0: программа для ЭВМ. Св. ГР. №2012618117 Рос. Федерация; зарег. в реестре Федер. службы по интеллектуальной собственности, пат. и товарным знакам 10.11.2012.

Отпечатано в типографии
ФГБОУ ВПО «Братский государственный университет»
665730, г.Братск, ул.Макаренко, 40
Тел./факс; (3953)32-53-01
Формат 60×80 1\16, объем 2,5 п.л., тираж 120экз.
Заказ № _____ подписано в печать 14.02.14г.