

Федеральное агентство по образованию
Нижегородский государственный университет
им. Н.И. Лобачевского

Технологии Microsoft

в теории и практике программирования

Материалы конференции

(Нижний Новгород, 3–4 апреля 2007 г.)

Нижний Новгород
Издательство Нижегородского госуниверситета
2007

УДК 681.3.06
ББК 32.973.26–018.2:22
T38

T38 **Технологии Microsoft в теории и практике программирования.** Материалы конференции / Под ред. проф. Р.Г. Стронгина. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2007. – 379 с.

Отв. за выпуск В.П. Гергель

ISBN 978-5-85746-983-5

Материалы конференции, состоявшейся 3–4 апреля 2007 г. на базе Нижегородского государственного университета им. Н.И. Лобачевского, посвящены новейшим технологиям Microsoft в теории и практике программирования. В конференции приняли участие студенты, аспиранты и молодые ученые вузов Приволжского региона Российской Федерации.

Для преподавателей и научных сотрудников, а также аспирантов и студентов высших учебных заведений.

ISBN 978-5-85746-983-5

ББК 32.973.26–018.2:22

Материалы конференции изданы при поддержке:

Корпорации Microsoft
Компании Autodesk



© Нижегородский госуниверситет
им. Н.И. Лобачевского, 2007



2007

3–4 апреля 2007 года в Нижегородском государственном университете им. Н.И. Лобачевского проходит конференция «**Технологии Microsoft в теории и практике программирования**», в которой участвуют студенты, аспиранты и молодые ученые вузов Приволжского региона России.

На конференции рассматриваются:

- теоретические работы в области Software Engineering и Computer Science;
- технологические разработки в области программирования, проектирования и создания программных продуктов;
- практические работы с использованием технологий Microsoft;
- практические и теоретические работы в области современных систем автоматизации проектирования и их применения;
- работы с использованием технологий Autodesk.

ПРОГРАММНЫЙ КОМИТЕТ КОНФЕРЕНЦИИ

Стронгин Р.Г.

председатель, ректор Нижегородского государственного университета им. Н.И. Лобачевского (ННГУ), профессор, д.ф.-м.н.

Гергель В.П.

зам. председателя, декан факультета вычислительной математики и кибернетики (ВМК) ННГУ, профессор, д.т.н.

Швецов В.И.

зам. председателя, проректор ННГУ по информатизации, профессор, д.т.н.

Бажанов Ю.С.

заместитель директора Института радиоэлектроники и информационных технологий (ИРИТ) Нижегородского государственного технического университета (НГТУ), профессор, д.т.н.

Васин Ю.Г.

директор НИИ прикладной математики и кибернетики, профессор, д.т.н.

Гаврилов А.В.

менеджер по связям с университетами Microsoft Russia, к.ф.-м.н.

Калинина И.Ю.

представитель компании Autodesk.

Кузнецов Ю.А.

заведующий кафедрой математического моделирования экономических систем механико-математического факультета ННГУ, профессор, д.ф.-м.н.

Крылов В.В.

директор по региональному развитию департамента стратегических технологий Microsoft Russia, профессор, д.т.н.

Латыпов Р.Ф.

декан факультета вычислительной математики и кибернетики Казанского государственного университета, профессор, д.т.н.

Любимов А.К.

декан механико-математического факультета ННГУ, профессор, д.ф.-м.н.

Моругин С.Л.

заведующий кафедрой компьютерных технологий в проектировании и производстве НГТУ, профессор, д.т.н.

Петрухин Н.С.

заслуженный деятель науки РФ, руководитель Центра бизнес-информатики и математической экономики НФ ГУ Высшей школы экономики, профессор, д.ф.-м.н.

Прилуцкий М.Х.

заместитель заведующего кафедрой информатики и автоматизации научных исследований ВМК ННГУ, профессор, д.т.н.

Ротков С.И.

заведующий кафедрой машинной графики и САПР Нижегородского государственного архитектурно-строительного университета (НГАСУ), профессор, д.т.н.

Савинский С.С.

проректор по академическим связям и высоким технологиям Удмуртского государственного университета, доцент, к.ф.-м.н.

Семенов В.А.

проректор по информатизации и новым технологиям обучения Пермского государственного университета, доцент, к.ф.-м.н.

Сидоркина И.Г.

декан факультета информатики и вычислительной техники Марийского государственного технического университета, доцент, д.т.н.

Сидорук Р.М.

директор Нижегородского областного центра новых информационных технологий, с.н.с., к.т.н.

Супрун А.Н.

заслуженный работник высшей школы РФ, проректор по информационным технологиям НГАСУ, профессор, д.ф.-м.н.

Сытник А.А.

проректор по информационным технологиям и открытому образованию Саратовского государственного социально-экономического университета, профессор, академик РАЕН, д.т.н.

Терехов А.А.

менеджер по связям с университетами Microsoft Russia, к.т.н.

Турлапов В.Е.

профессор кафедры математического обеспечения ЭВМ факультета ВМК ННГУ, д.т.н.

Федосин С.А.

проректор по информатизации Мордовского государственного университета, доцент, к.т.н.

Хисамутдинов Р.А.

проректор Уфимского государственного авиационного технического университета, к.т.н.

Чупрунов Е.В.

проректор ННГУ по научной работе, профессор, д.ф.-м.н.

Шабалкин Д.Ю.

проректор по информационным технологиям Ульяновского государственного университета, доцент, к.ф.-м.н.

Шашков Б.Д.

директор Института информатики и вычислительной техники Пензенского государственного университета, заведующий кафедрой МОиПЭВМ., профессор, к.т.н.

Якимов А.В.

декан радиофизического факультета ННГУ, профессор, д.ф.-м.н.

МОДЕЛИРОВАНИЕ ЭВОЛЮЦИИ ПУЧКА ЧАСТИЦ С УЧЕТОМ СОБСТВЕННОГО ЗАРЯДА

E.C. Abramov

Санкт-Петербургский госуниверситет

Целью работы является построение компьютерной модели движения пучка заряженных частиц через систему управляющих магнитных линз. Данная модель должна позволить исследователю наблюдать за поведением пучка при изменении параметров управления. Результатом работы является компьютерная программа, позволяющая рассчитывать движение пучка при различных характеристиках самого пучка и управляющих линз.

Математическая модель

Для моделирования пучка используется его представление в фазовом пространстве координат и скоростей каждой частицы, которое задается вектором $X = (x, x', y, y')^T$. Начальное состояние пучка характеризуется его размерами и распределением частиц в пространстве. В данной работе рассматривается нормальное распределение частиц в фазовом пространстве. Уравнения движения частицы через систему линз выглядят следующим образом:

$$\begin{aligned} \frac{d^2x}{dt^2} + \left(Q_x - \frac{2R^2}{r_x(r_x + r_y)} \right) x &= 0, \\ \frac{d^2y}{dt^2} + \left(Q_y - \frac{2R^2}{r_y(r_x + r_y)} \right) y &= 0. \end{aligned}$$

В рассматриваемой модели поведение пучка определяется величиной его огибающих r_x, r_y , величиной тока пучка R^2 и величиной внешнего магнитного поля Q . Поскольку коэффициенты этих уравнений зависят от t , то численное моделирование движения необходимо производить с малым шагом, производя вычисление новых огибающих пучка и величины внешнего поля на каждом шаге. Для краткости обозначим коэффициенты при x, y в уравнениях k_x, k_y соответственно.

В данной работе для вычисления траекторий движения частиц используется матричный формализм. При таком подходе для вычисления состояния частицы на следующем шаге используется матричный оператор эволюции, который можно записать следующим образом:

$$M = \begin{pmatrix} M^{11} & 0 \\ 0 & M^{22} \end{pmatrix}, \text{ где } M^{11} = \begin{pmatrix} \cos w\Delta t & \frac{\sin w\Delta t}{w} \\ -w\sin w\Delta t & \cos w\Delta t \end{pmatrix}, \text{ при } k_x \geq 0, w = \sqrt{k_x},$$

$$M^{11} = \begin{pmatrix} \cosh w\Delta t & \frac{\sinh w\Delta t}{w} \\ -w \sinh w\Delta t & \cosh w\Delta t \end{pmatrix}, \text{ при } k_x < 0, w = \sqrt{-k_x}.$$

Аналогично для матрицы M^{22} , которая зависит от k_y .

Выражение для вычисления координат частицы на следующем шаге имеет вид:

$$X_{n+1} = M \times X_n$$

Программная реализация

Компьютерное моделирование движения пучка через систему линз в рамках рассмотренной математической модели может быть реализовано в терминах матричных операций. Поскольку для адекватного моделирования движения пучка необходимо рассмотреть поведение большого количества отдельных частиц, то реализация модели требует значительных вычислительных затрат. Вместе с этим существует возможность параллельной обработки отдельных частиц на разных вычислительных узлах, поскольку поведение каждой частицы зависит только от величины огибающих пучка и величины внешнего поля. Построенная программная модель включает в себя:

1. Генератор начального распределения частиц в пучке.
2. Модель внешнего поля, создаваемого управляющими магнитными линзами.
3. Модель прохождения пучка через систему линз.
4. Процедура определения точки максимальной фокусировки пучка после прохождения линз.
5. Отдельный модуль визуализации фазового портрета пучка.

Компьютерная модель строилась с расчетом на параллельную реализацию на кластерной системе, каждый узел которой сам по себе является многоядерным или много-процессорным. При этом один из узлов кластера является главным.

Рассмотрим кратко алгоритм выполнения расчетов.

1. Главный узел генерирует пучок с заданным количеством частиц, делит его на порции (по числу частиц) и рассыпает их всем остальным узлам.
2. Каждый узел самостоятельно строит модель внешнего поля.
3. Каждый узел вычисляет огибающие своей порции пучка и отсылает эти величины главному узлу.
4. Главный узел находит максимальные огибающие (они будут использоваться при расчетах) и посыпает их всем узлам.
5. На всех узлах выполняется расчет состояния пучка на следующем шаге.
6. Если эволюция не завершена, то переход к шагу 3.
7. Определение точки фокусировки и отображение результатов.

Данный алгоритм был реализован на C++ в среде Visual Studio 2005 с использованием библиотеки MPICH 1.2.4 [0]. Распараллеливание вычислений внутри узла было осуществлено при помощи директив OpenMP.

Вычислительный эксперимент

Число частиц	Время расчета 1 комп., 1 ядро, с	Время расчета 1 комп., 2 ядра, с	Время расчета 2 комп., 1 ядро, с	Время расчета 2 комп., 2 ядра, с
50000	605	365	403	253
100000	1211	730	782	470

Области применения программы

Важной задачей в физике пучков является задача фокусировки пучка и построения фокусирующих установок. Проведение экспериментов в этой области является трудно-реализуемой и дорогостоящей процедурой. Рассмотренная в данной работе программная система позволяет анализировать поведение пучка при различном количестве, расположении и параметрах управляющих линз, что может быть полезным при обучении специалистов в области физики пучков. Также после доработки и анализа точности вычислений, программа может быть использована при подборе параметров управляющих линз и определении точки максимальной фокусировки в ходе построения фокусирующих установок.

Список литературы

1. Андрианов С.Н. Динамическое моделирование систем управления пучками частиц. – СПб.: Изд-во С.-Петербургского ун-та, 2002. – 376 с.
2. <http://www-unix.mcs.anl.gov/mpi/mpich1/> — MPICH home page.
3. <http://www.openmp.org/drupal/> — OpenMP home page.

MS VISUAL STUDIO.NET (C#), MS SQL SERVER 2000, GENTLE.NET, MYGENERATION И NUNIT ПРИ ИСПОЛЬЗОВАНИИ РАЗРАБОТКИ ЧЕРЕЗ ТЕСТИРОВАНИЕ ДЛЯ СОЗДАНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДЕКАНАТАМИ ВУЗА

М.Ю. Алтуфьев

Астраханский государственный технический университет

Введение

Разработка через тестирование (Test Driven Development, TDD) – это способ разработки программного обеспечения, при котором написание фактического кода отталкивается от предварительно созданных тестов, которые должны этот код верифицировать.

TDD тесно связана с понятием экстремального программирования, поскольку ориентирована на быструю и качественную разработку программного обеспечения небольшой группой людей.

Экстремальное программирование — это упрощенная методика организации разработки для небольших и средних по размеру команд специалистов, занимающихся разработкой программного продукта в условиях неясных или быстро меняющихся требований. TDD очень гладко укладывается в эту схему, позволяя разработчику, не задумываясь о проектировании как отдельном и сложном процессе, действовать, в связи с определенным проектом. Этот проект — тесты, использующиеся в процессе разработки.

Задача

Необходимость создания в короткие сроки приложения, способного заменить и дополнить функциональность уже существующего при наличии технического задания, хорошо соотносится со схемой экстремального программирования. Наличие такой программы, как NUnit, позволяет применить методику TDD для разработки программного обеспечения в Visual Studio.NET. Задача представляет собой создание автоматизированной системы управления деятельностью деканатов вуза по контролю движения контингента студентов и их успеваемости. Наличие функционирующей системы приема абитуриентов, написанной на C# и использующей в качестве базы данных базу в MS SQL Server 2000, послужило одним из аргументов для разработки АСУ «Деканат» в Visual Studio.NET с использованием СУБД MS SQL Server 2000. Другим важным фактором стало наличие системы Gentle.NET, позволяющей отображать отношения баз данных в классы для среды Visual Studio.NET.

При исследовании различных схем применения экстремального программирования было принято решение, что наибольшим потенциалом для разработки программного продукта средних размеров, такого как АСУ «Деканат», обладает методика TDD.

Целью создания АСУ «Деканат» является автоматизация ввода исходных данных, обработки информации и печати личных дел обучающихся, успеваемости обучающихся, приказов по движению контингента студентов и изменению личной информации; формирования стандартной и нестандартной отчетности деканатов (дирекций).

Назначением системы является повышение эффективности и качества работы деканатов (дирекций) в повседневной работе, связанной с учетом контингента и успеваемости обучающихся, формированием стандартных приказов и нестандартной отчетности.

АСУ «Деканат» должна реализовывать следующие функции:

- Формирование и ведение личного дела студента.
- Учет успеваемости.
- Автоформирование и печать стандартных приказов.
- Управление движением контингента студентов на основе приказов.
- Автоформирование и печать стандартных отчетов (статистической отчетности).
- Автоформирование и печать внешней стандартной отчетности.
- Формирование и печать нестандартной отчетности.

Общая задача разработки – за минимально возможное время разработать программный продукт, реализующий указанные функции и отвечающий всем требованиям надежности и безопасности программного обеспечения, так, чтобы впоследствии можно было отказаться от бумажного документооборота и полностью положиться на электронную форму приказов.

Решение

Стандартная схема TDD заключается в следующем:

- Формируется предварительное задание на разработку (которое в дальнейшем может, и, скорее всего, будет изменяться).
- Формируется набор задач, необходимых для достижения поставленной цели разработки.

- Формируется список задач, которые необходимо разрешить в первую очередь. Величина списка основывается на размере группы разработки (в случае TDD для экстремального программирования – по одной задаче на два человека).
- Для каждой задачи создается набор тестов. Тесты создаются так, чтобы покрыть всю область деятельности: начиная от тестов, рассматривающих реализуемую задачу как черный ящик, и заканчивая тестами конкретных функций, которые (предположительно) будут требоваться от реализуемого модуля.
- В соответствии с набором тестов создаются сначала интерфейсы классов, а потом и их реализация.
- При завершении каждого участка кода, для которого существует тест, весь набор тестов прогоняется заново, чтобы убедиться, что добавление новой функциональности не повлекло изменения существующей.
- В процессе разработки могут корректироваться и сами тесты, и написанные ранее участки кода.

Таким образом, все начинается с первого прогона тестов, на котором ни один тест не работает, и заканчивается наличием некоторого логически завершенного кода, на котором все тесты работают.

Отличие в применении TDD для такой разработки, как АСУ «Деканат», в наличии хорошо описанной предметной области с уже давно установленными параметрами объектов предметной области и потоков передаваемой информации. Поскольку изначально предполагается возможность применения данной разработки в качестве системы электронного документооборота (приказы о контингенте), то имеет смысл отталкиваться при создании классов программы от сущностей в том виде, в каком последние участвуют в процессе бумажного документооборота.

Данная мысль позволяет сразу определиться с большей частью структуры базы данных, а также предложить использование системы Gentle.NET для отображения реляционных отношений в классы и, соответственно, кортежей в объекты.

Gentle.NET – система, позволяющая разработчику избежать подробностей взаимодействия программы с базой данных и работать напрямую с классами, представляющими собой отображения таблиц базы данных. Gentle.NET обеспечивает выборку, добавление, обновление данных; поддерживает создание запросов в конструкторе, представление запросов на языке SQL, но также позволяет полностью отойти от SQL и работать с БД исключительно средствами языка C#.

MyGeneration — программный продукт, позволяющий генерировать классы для работы с таблицами и представлениями баз данных для различных языков программирования. В том числе, MyGeneration поддерживает связку C# – Gentle.NET – MS SQL Server 2000, т.е. генерирует классы на языке C# с метаданными для Gentle.NET на основе имеющейся базы данных SQL Server 2000.

Наличие подобного программного обеспечения привело к тому, что схема работы TDD при разработке АСУ «Деканат» была несколько изменена.

Сначала на основе описания предметной области и структур данных была создана база данных в SQL Server 2000.

Затем были сгенерированы классы для работы с базой данных.

Следующим шагом стало начало применения TDD: были разработаны тесты для определения адекватности взаимодействия сгенерированных классов при работе в комплексе. Тесты были написаны с применением программного продукта NUnit, предназначенного для тестирования проектов, созданных в MS Visual Studio .NET. NUnit позволяет создать классы и статические методы, которые при наличии минимально не-

обходимого количества метаданных способны обеспечить комплексную оценку правильности работы тестируемых объектов, методов, свойств и т.д. при работе приложения в режиме отладки.

После создания вышеупомянутых тестов были написаны классы, предназначенные для инкорпорирования классов Gentle.NET в общую схему работы приложения, что, собственно, и явилось проверкой адекватности их работы.

Когда все тесты были пройдены успешно, наступил следующий этап работы: разбиение на мелкие подзадачи, которые группа из двух программистов способна реализовать в течение одного дня таким образом, чтобы результат реализации представлял собой некоторый законченный элемент программы и позволял проекту компилироваться и успешно проходить все тесты.

Результаты

Общее чистое время работы над кодом проекта составило около 80 рабочих дней. Проект разрабатывался в течение одного года с периодическими перерывами на тестирование бета-версий пользователями и группой по разработке технического задания.

Весь этот промежуток времени происходили не только мелкие изменения в задании, но и (при переосмыслинии некоторых бизнес-процессов, происходящих в деканатах) довольно крупные изменения, требующие, в том числе, изменения структур данных.

База данных была доработана не раз, но, благодаря наличию MyGeneration, Gentle.NET и тестов NUnit, каждое внесение изменений потребовало весьма незначительного времени. Так, изменение структуры двух таблиц, и добавление связанных с ними трех новых было произведено в течение часа и еще около часа ушло на исправление возникших в коде ошибок, показанных после прогона тестов. При этом классы, представляющие новые таблицы, уже оказались готовы к дальнейшей работе с ними.

Всего в рамках проекта было создано более 60 таблиц. Проект занимает 130 Мб и содержит более 200000 строк кода. Разработка велась силами 8 человек, однако не все из них были задействованы в течение всего времени разработки.

АСУ «Деканат» находится в промышленной эксплуатации более одного года и успешно справляется с возложенными на нее задачами.

Выводы

В целом, сдается признать, что связка программных продуктов MS Visual Studio.NET, MS SQL Server 2000, Gentle.NET, MyGeneration и NUnit послужила хорошей основой для быстрой и качественной разработки программного продукта в условиях изменяющихся требований и существенного ограничения временных и человеческих ресурсов.

Недостатком явилось отсутствие возможности автоматизированного тестирования тех частей проекта, которые были написаны на ASP.NET.

Список литературы

1. Кент Бек. Экстремальное программирование. – М.: Питер, 2002.
2. Кент Бек. Экстремальное программирование: разработка через тестирование. – М.: Питер, 2003.
3. Jeffery E. Payne, Roger T. Alexander, Charles D. Hutchinson. Design-for-Testability for Object-Oriented Software. Sigs Publications, INC, 1997.
4. Microsoft ACE Team. Performance Testing Microsoft .Net Web Application. Microsoft Press, 2003.

ПРИМЕНЕНИЕ АЛГОРИТМОВ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ В САПР СЕТЕЙ ИНТЕЛЛЕКТУАЛЬНЫХ ДАТЧИКОВ

И.В. Аппель

Южно-Уральский государственный университет

Введение

В настоящее время в промышленных системах обеспечения технологических процессов широко используются интеллектуальные датчики (датчики, объединяющие в единых интегрированных устройствах первичные сенсоры, электронику аналого-цифровых преобразований, электронику подключения к разделяемой шине данных и автоматику общего управления). Одним из аспектов их применения является построение сетей, функционирующих по специализированным сетевым протоколам ModBus [1], HART [2] и многочисленным аналогам. Такие решения обеспечивают относительно простой сбор информации о протекании технологических процессов в реальном времени.

Важной подзадачей, наряду с конструированием датчиков, реализующих подключение к сенсорным сетям посредством заданных сетевых протоколов и построения систем сбора и обработки данных, является их коммутация в промышленных условиях.

Коммутация включает оптимальную прокладку коммуникаций с целью минимизации стоимости и максимизации надежности полученного индустриального решения, а также задачи физического подключения, не рассматриваемые в данной работе. Все это особенно важно в условиях большого удаления датчиков друг от друга, либо опасных и физико-химически высокоягрессивных технологических процессов, ведущих к быстрому износу систем связи.

Для решения данной задачи пространство промышленного объекта моделируется как взвешенный квазипланарный граф, и для ее решения применяется двухуровневый метод решения задачи Штейнера для графа, описанный в [3] с некоторыми дополнениями, рассмотренными ниже. Метод решения состоит в разделении задачи поиска оптимума на операции оптимального размещения промежуточных вершин при фиксированной топологии и операции перестройки топологии связей фиксированных и промежуточных вершин. Фиксированным вершинам в данной задаче соответствуют датчики, промежуточным – точки соединения связующих линий.

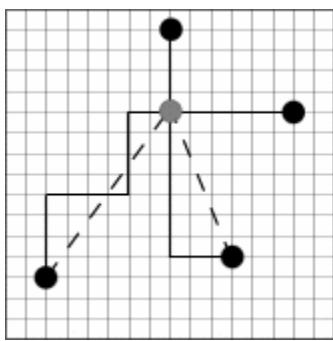


Рис. 1

Математическая модель

Введем понятия топологии размещения и базового графа.

Базовый граф – модель пространства размещения объектов, представляющая собой планарный либо квазипланарный граф. Топология размещения – дерево, включающее фиксированные (заданные) и промежуточные (добавленные) вершины. Вершины дерева топологии отображаются на вершины базового графа, при этом ребра дерева отображаются на кратчайшие пути между спроектированными вершинами.

Для определения целевого функционала для оптимизации размещения введем функцию стоимости раз-

мещения a -той вершины дерева топологии в i -той вершине базового графа:

$$C_a(i) = \sum_{b \in V_a} D_{P_b}(i), \quad (1)$$

где V_a – вершины, связанные с a -той, P_a – проекция a -той вершины на базовый граф, $D_j(i)$ – кратчайшее расстояние между вершинами.

В обычной задаче Штейнера, где стоимость промежуточных вершин считается нулевой, функционал имеет вид

$$F(V, W, P) = \frac{1}{2} \sum_{a \in V} C_a(P_a), \quad (2)$$

где V – топология, W – базовый граф, а P – отображение первого на второе.

Поставленная задача предполагает выполнение условия о ненулевой стоимости размещения единичной промежуточной вершины. Это объясняется тем, что реально эта «промежуточная» вершина представляет собой RS-485, RS-232, HART или аналогичные коммутаторы связи между тремя и более устройствами. Естественно, они имеют ненулевую собственную стоимость и ненулевую стоимость размещения на плоскости. Кроме этого, они имеют конечную емкость. Если предполагать, что все коммутаторы одинаковы и имеют приведенную стоимость H и емкость M , то целевой функционал принимает вид:

$$F(V, W, P) = \frac{1}{2} \sum_{a \in V} C_a(P_a) + H |\tilde{V}|, \quad (3)$$

где \tilde{V} – подмножество промежуточных вершин графа, а функция стоимости размещения

$$C_a(i) = \begin{cases} \sum_{b \in V_a} D_{P_b}(i), & |V_a| \leq M; \\ \infty, & |V_a| > M. \end{cases} \quad (4)$$

Для этой задачи можно описать набор оптимизационных операций.

1. Оптимизация без изменения топологии. Задавая функцию оптимального положения вершины $P_a = \arg \min_{i \in W} C_a(i)$ и комбинируя ее с (4), можно получить систему уравнений, которая решается итерационно. При этом в целях повышения быстродействия, значения необходимых функций D_j вычисляются сразу для всех $i \in W$ и кэшируются.

2. Понижение мощности вершин. Выбираются вершины, пути к которым от связанных вершин пересекаются и в их пересечениях ставятся новые промежуточные вершины. Пример показан на рис. 2.

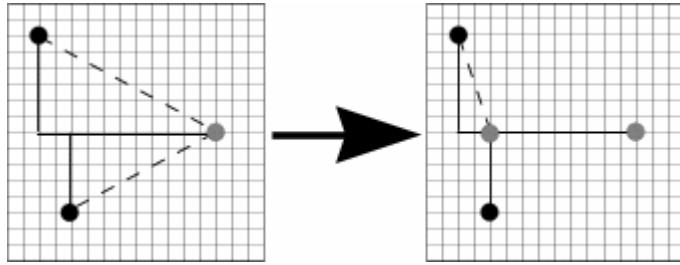


Рис. 2

3. Повышение мощности вершин. Операция, обратная 2. Кроме понижения суммарной стоимости решения за счет уменьшения числа промежуточных вершин, эта операция используется также для вывода решения из локальных минимумов.

4. Локальная оптимизация. Выбираются фиксированные вершины, имеющие соседей, расположенных к ним ближе, чем те, к которым они привязаны и соединяются с ними, при необходимости добавляются промежуточные вершины. Пример показан на рис. 3, при этом топологии на рис. 2 и 3 являются идентичными.

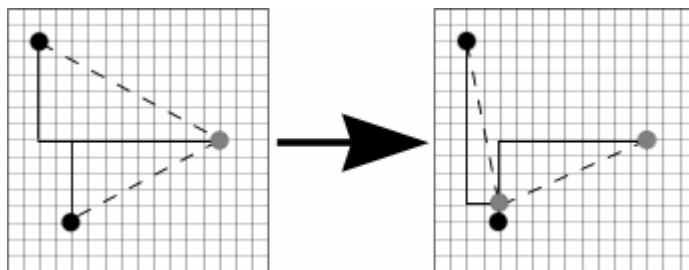


Рис. 3

5. «Сборка мусора». В процессе оптимизации в графе могут появляться избыточные промежуточные вершины с мощностью меньше трех и с совпадающими проекциями. Такие вершины удаляются.

Посредством использования разных сочетаний данных операций можно реализовать аналог метода ветвей и границ и построить дерево возможных решений с отсевом наименее качественных. Лучшее из найденных считается оптимальным.

При построении базового графа используется метод синтеза, рассмотренный в [4], основанный на реальной модели поверхности и отражающий неоднородность пространства размещения и допустимые метрики соединений.

Реализация

Написана бета-версия системы автоматизации проектирования сенсорных сетей. Имеющийся функционал:

- диалоговый ввод положения датчиков на плоскости и структуры поверхности;
- ручное формирование исходной топологии для оптимизации;

- автоматическое построение исходной топологии «звезда» (все фиксированные вершины соединяются с одной промежуточной);
- все операции оптимизации и построение дерева решений;
- ввод данных о неоднородностях поверхности и допустимых метриках.

Планируется внедрение системы в региональном подразделении РАО «Газпром».

Список литературы

1. ModBus Specification and Implementation Guides, <http://www.modbus.org/specs.php>
2. Страница HART Communication Foundation, <http://www.hartcomm2.org/>
3. Панюков А. В. Топологические методы решения задачи Штейнера на графе // Автоматика и телемеханика. 2004. № 3. С. 89–100.
4. Панюков А. В., Пельцвергер Б. В., Шафир А. Ю. Оптимальное размещение точек ветвления транспортной сети на цифровой модели местности // Автоматика и телемеханика. 1990. № 9. С. 153–162.

ОНЛАЙНОВАЯ ИГРА ДЛЯ ОБУЧЕНИЯ СОСТАВЛЕНИЮ АЛГОРИТМОВ

A.I. Ахметзянов

*Елабужский филиал Казанского государственного технического
университета им. А.Н. Туполева*

Целью данной работы является разработка сайта онлайновых игр для программистов, позволяющего в естественной соревновательной среде пользователей изучать принципы создания оптимального алгоритма управления объектом.

Во всем мире постоянно проходят соревнования по программированию. Обычно они представляют собой конкурс на решение поставленной, достаточно сложной, задачи средствами одного из общепринятых языков программирования (Си, Паскаль, Java). Также существуют сервера с большим набором таких задач, где пользователи могут потренироваться перед турниром, набрать рейтинговые баллы, не ограничивая себя по времени нахождения нужного алгоритма. Существует ряд турниров, например Imagine Cup, где участники могут поработать над созданием программы для искусственного интеллекта. Такого рода состязания помогают развивать практические навыки программирования, лучше понять принципы алгоритмизации. Тем не менее, не существует достаточно серьезного русскоязычного ресурса, подобного одновременно и игре (поэтому интересного для широкого круга пользователей), и серьезному образовательному проекту. Для этого в работе проводится создание сайта онлайновых игр для программистов «Кибер-спорт». Такое название не случайно: кибер-спорт – это разминка для ума и творческих способностей программиста.

Пользовательская часть проекта – веб-интерфейс, позволяющий пользователям выставить свою программу-робота на соревнование с программами других пользователей и наблюдать за развитием сражения. Оценка работы алгоритма в условиях соревнования дает учащемуся дополнительный стимул к совершенствованию своих навыков.

Изучение базовых действий робота даже ребенку позволит изучить основные законы составления программ.

Основные правила игры могут меняться и дополняться. С самого начала это – двухмерное игровое поле, на котором в нулевой момент игры случайным образом размещены соревнующиеся роботы. Роботы могут двигаться, накапливать энергию и размножаться, предусмотрены возможности атаки, определения положения, общения с союзниками. Система команд подобна системе команд процедурно-ориентированного языка и адаптирована к ситуации.

Сайт разработан на платформе .NET, что обеспечивает стабильность и быстроту просчета результатов игры.

Вывод: локальная версия разработанной игры успешно применяется на практических занятиях в ЕФ КГТУ им. А.Н.Туполева.

ВЕКТОРНЫЙ ГРАФИЧЕСКИЙ РЕДАКТОР НА ОСНОВЕ ВЕБ-ТЕХНОЛОГИЙ

В.Н. Белов, А.С. Сарайкин

Пензенский госуниверситет

Постановка задачи

В настоящий момент существует большое количество графических редакторов, предоставляющих различные возможности и ориентированных на различные классы пользователей. Но не всегда пользователю необходимо дорогостоящее и многофункциональное программное обеспечение. Возможно, ему просто требуется быстро создать небольшой графический файл. Однако пользователю может не хватать возможностей штатного графического редактора или времени на поиск и установку соответствующего приложения. В этом случае оптимальным решением будет воспользоваться приложением, располагающимся в Интернет и работающим прямо в браузере.

Целью данной работы является реализация полнофункционального графического редактора, доступного через Интернет и предоставляющего пользователю максимальное удобство в работе.

Краткое описание проекта

Перед созданием данного редактора были изучены существующие на данный момент аналоги, написанные на языке Java. Они отличаются бедностью интерфейса пользователя и большим потребляемым трафиком. Также пользователь не может быть полностью уверен, что на его компьютере не исполняется вредоносный код. Векторный редактор, созданный в рамках данного проекта лишен этих недостатков.

Основная часть векторного редактора реализуется с использованием Adobe Flash на языке Action Script 2.0. Выбор в пользу данной технологии был сделан в силу ее ориентированности на векторную графику. Flash также позволяет создавать удобный и интерфейс пользователя, отличающий красотой и изящностью. Кроме того, с точки зрения пользователя Flash явно безопаснее, чем Java, что позволит ему не волноваться по поводу несанкционированного доступа к его компьютеру.

Кроме перечисленных, технология Flash имеет следующие отличительные и немаловажные черты, использованные в данном приложении:

- возможность работы не только через браузер, но и через стандартный проектор;
- наличие средств шифрования исходного кода;
- наличие средств сжатия без абсолютной потери качества;
- объектная ориентированность языка;
- независимость от платформы;
- малый размер приложений;
- отсутствие необходимости постоянного подключения к Интернету во время работы с приложением.

Сразу после загрузки приложения в браузер пользователь может начать работу. Графический редактор предоставляет возможность выбрать размер полотна, а также его цвет или текстуру. Пользователь создает изображение с помощью набора примитивов различной сложности, которые можно вращать, перемещать, масштабировать. Каждый примитив имеет набор свойств, изменение которых влияет на его отображение.

Для вращения любых объектов разработан оригинальный и удобный для пользователя элемент управления. Важно отметить факт, что в приложении реализована операция группировки объектов. Также некоторые примитивы способны выполнять функцию контейнера: при перемещении внутрь них объекты изменяются размер так, чтобы вписаться в границы контейнера. Все операции, произведенные над объектами обратимы. Немаловажно, что в приложении реализован полноценный режим редактирования. Интерфейс приложения показан на рис. 1.

Приложение сохраняет изображение в трех форматах: BMP, JPEG и XML, после чего файл может быть передан пользователю по протоколу FTP. Предпочтительным форматом для сохранения является XML, так как такой файл впоследствии может быть загружен и подвергнут редактированию в данном приложении, а также имеет намного меньший размер.

Выбор в пользу XML был сделан по следующим причинам:

- формат, основанный на международных стандартах;
- иерархическая структура подходит для описания структур векторного редактора;
- представляет собой простой текст, свободный от лицензирования и каких-либо ограничений;
- не зависит от платформы;
- является внутренним форматом представления данных Adobe Flash.

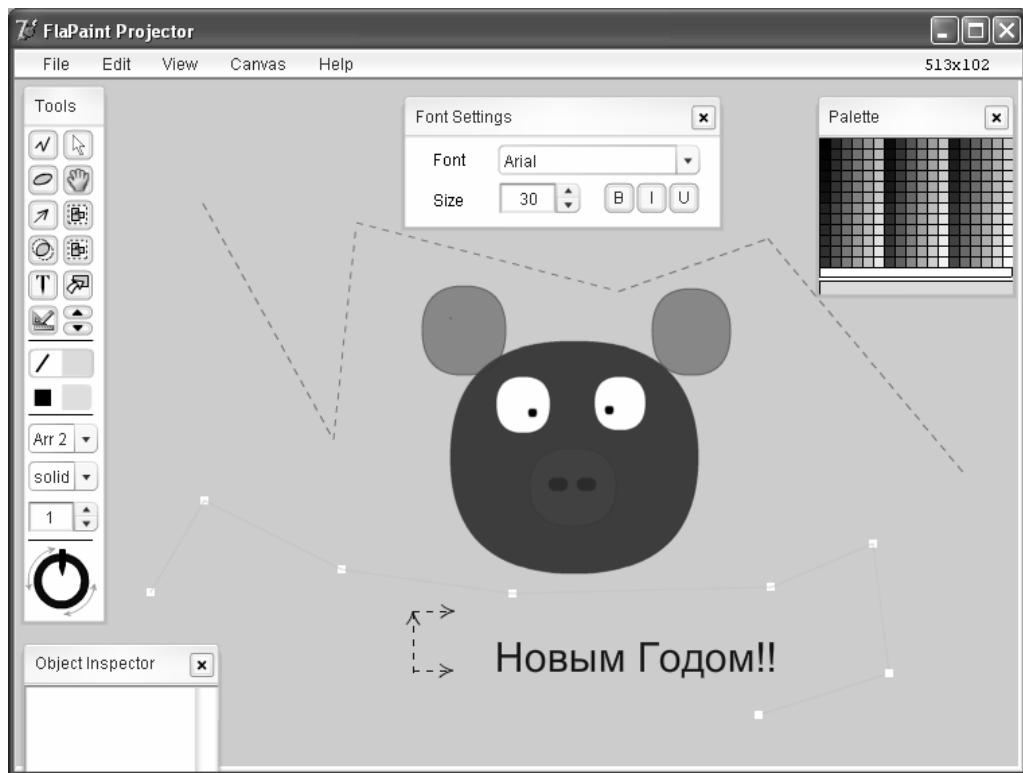


Рис. 1. Интерфейс приложения

Поскольку Flash-приложения в целях безопасности не предоставляют достаточных возможностей для локального сохранения файлов, для реализации этой функции был разработан собственный проектор (элемент ActiveX в среде разработки Delphi). Экспорт растеризованной сцены осуществляется с помощью попиксельной передачи из Flash-приложения в проектор.

Для взаимодействия Flash-приложения и проектора используются стандартные функции

```
fscommand(command:String, parameters:String):Void  
и ShockwaveFlash1FlashCall (ASender: TObject; const request: WideString).
```

Можно утверждать, что данное приложение способно удовлетворить потребность большинства пользователей в простом, но эффективном векторном редакторе на основе веб-технологий.

РЕАЛИЗАЦИЯ МОДЕЛИ ЗАДАЧИ О ХАНОЙСКИХ БАШНЯХ С ИСПОЛЬЗОВАНИЕМ СОВРЕМЕННЫХ ТЕХНОЛОГИЙ СОЗДАНИЯ ТРЕХМЕРНОЙ КОМПЬЮТЕРНОЙ ГРАФИКИ

В.Н. Белов, А.С. Сарайкин

Пензенский госуниверситет

Постановка задачи

Постоянный интерес к старым задачам обусловлен множеством причин. Это и красивые легенды, обрамляющие условие, и изящные решения. Несомненно, задача о Ханойских башнях сопровождается одной из самых интригующих легенд: считается, что в Китае есть старый буддийский монастырь, где уже много веков его обитатели поочереди перемещают пирамиду из 64 дисков. По преданию, укладка последнего диска в их пирамиде означает конец света.

Классическая задача о Ханойских башнях была сформулирована французским математиком Эдуардом Лукасом в 1883 году и формулируется следующим образом:

- есть три стержня A, B, и C;
- на стержень A надето N дисков, наверху самый маленький, каждый следующий диск больше предыдущего, а внизу самый большой;
- на другие стержни дисков не надето;
- необходимо перенести диски со стержня A на стержень C, пользуясь стержнем B, как вспомогательным так, чтобы диски на стержне C располагались в том же порядке, в каком они располагаются на диске A перед перемещением;
- при перемещении никогда нельзя класть больший диск на меньший.

Целью данной работы является создание приложения, предоставляющего пользователю возможность самому отыскать алгоритм решения задачи, т.к. существует целый класс приложений, реализующих различные подходы к решению данной задачи, но при этом не предоставляющих пользователю возможность действовать самостоятельно, либо же не обладающих удобным интерфейсом.

Краткое описание проекта

Поскольку приложение ориентировано на рядового пользователя заинтересованного в самостоятельном поиске решения задачи о Ханойских башнях, оно реализовано как игра-головоломка. Стандартный для такого класса приложений подход к сохранению результатов игрока (затраченных ходов и времени) предоставляет пользователю возможность оценить правильность выбранной им стратегии.

Одной из отличительных черт данного приложения является удобство интерфейса: все действия сводятся к перетаскиванию мышкой колец с одного стержня на другой. Другой отличительной чертой является реалистичность и наглядность создаваемого изображения. В приложении реализовано динамическое освещение виртуального пространства, текстурирование и подсветка объектов. Для создания реалистичного изображения использованы средства, предоставляемые библиотекой OpenGL в силу следующих причин:

- библиотека OpenGL разработана как обобщенный, независимый интерфейс, который может быть реализован для различного аппаратного обеспечения;

- высокая производительность версии библиотеки, написанной для ОС Windows;
- наличие большого числа доступной документации;
- при использовании библиотеки OpenGL требуется строить необходимые модели при помощи небольшого набора геометрических примитивов - точек, линий и многоугольников (полигонов), что представляется достаточным в рамках данного приложения.

OpenGL не предоставляет функций для создания окон или для захвата пользовательского ввода. Для реализации этих операций должны использоваться средства ОС Windows, поэтому приложение написано в среде разработки Visual C++ 6.0 с использованием Windows API. Windows API был изначально спроектирован для использования в программах, написанных на языке С (или С++). Работа через Windows API - наиболее близкий к системе способ взаимодействия с ней из прикладных программ. Это позволяет писать небольшие и эффективные приложения.

Разработанное приложение замечательно выполняет свою основную функцию: знакомит пользователя с классической задачей о Ханойских башнях и позволяет ему самостоятельно искать оптимальное решение, не задумываясь о математической сути задачи.

СИСТЕМА УЧЕТА ДЕЯТЕЛЬНОСТИ СТУДЕНТОВ НА УРОВНЕ КАФЕДРЫ

В.Н. Белов, А.С. Сарайкин

Пензенский госуниверситет

Постановка задачи

Традиционная технология планирования расписания занятий и ведения журнала учета деятельности студентов требует серьезных затрат времени, связанных с необходимостью поддержания разнородных документов и сведения их воедино, поиском нужной информации, анализом текущих показателей, подведением итогов и принятием решений. Поэтому представляется необходимым создание автоматизированной системы, которая должна облегчить процесс планирования расписания занятий, формирования и выдачи заданий, приема отчетов и оценки качества выполнения заданий. Это должно частично снизить нагрузку преподавателя, а также упростить для студента получение оперативной информации о текущем состоянии процесса сдачи работ, в том числе и с помощью доступа к системе через Интернет.

Определим основные категории пользователей. Оператор заносит в базу данных основной объем исходной информации: данные о студентах и преподавателях, учебный план, расписание занятий. Преподаватель создает рабочую программу по своим дисциплинам, распределяет студентов по бригадам, заполняет список заданий на семестр для своих групп, добавляет файлы с методическими указаниями, раздает задания, ведет журнал (ставит отметки о посещении лекций, сдаче и защите заданий по занятиям и пишет комментарии). Студент может просматривать доступную ему информацию: бри-

гады, в которых он состоит, список работ на семестр, методические указания преподавателя, комментарии в журнале, которые преподаватель счел необходимым сделать доступными для просмотра студентами, выбирает свой вариант из списка заданий, а также отсылает преподавателю выполненные работы на проверку. И студент, и преподаватель имеют возможность просмотреть расписание в удобной для себя форме. Система создания представлений, реализуемая программно, создает для студента удобное представление журнала преподавателя. К функциям администратора относится начальное развертывание системы, создание учетных записей операторов, а также резервное копирование и восстановление данных.

Краткое описание проекта

Система учета деятельности студентов имеет клиент-серверную архитектуру.

Для работы с базами данных выбран MS SQL Server 2000: он хранит все данные системы, обеспечивает необходимую для системы работу с ролями и превосходно справляется с большим числом пользователей и одновременных обращений. Язык Transact-SQL позволяет писать хранимые процедуры различной степени сложности, которые обеспечивают обработку всей информации, поступающей от клиентов.

Для обработки запросов на передачу файлов написано специальное приложение, реализованное как служба Windows. Механизм передачи файлов максимально упрощен для пользователя: от него скрыта вся ненужная информация, например место хранения его файлов. Для хранения некоторых дополнительных сведений о файлах, например, закачивался ли он уже пользователем, используются средства файловой системы ОС Windows. Серверное приложение для передачи файлов и MS SQL Server 2000 могут располагаться на разных компьютерах.

Система учета деятельности студентов реализуется с учетом возможности работы различных классов пользователей за одним компьютером: клиентская часть представляет собой единое приложение, реализующее функцию начальной проверки учетной записи и пароля, и набор библиотек, каждая из которых представляет собой функционально законченное приложение для определенного класса пользователей. Это гарантирует максимальную простоту добавления в систему новых классов пользователей, а также позволяет исключить несанкционированный доступ к целому классу пользователей с определенного компьютера простым удалением библиотеки с жесткого диска.

В такой области, как учет деятельности студентов, защита от несанкционированного доступа приобретает наиважнейшее значение. Система имеет различные механизмы обеспечения безопасности: шифрование логина и пароля при передаче через сеть, повторная проверка пароля при совершении критических действий пользователем, блокировка исполнения на сервере действий, недоступных классу пользователя. С этой же целью имеется разделение операторов на 2 категории: лишь одна из них имеет право вносить изменения в ранее введенные данные. Администратор при создании учетной записи оператора также может ограничить его функции, составив для него произвольный список доступных действий, сделав для этого всего несколько щелчков мышью. В зависимости от доступных оператору действий, меняется его интерфейс: исчезают или добавляются вкладки, необходимые для их выполнения. Кроме того, для каждого оператора на сервере блокируется выполнение недоступных ему действий. Это позволяет привлекать к выполнению функций оператора любых людей с минимальным риском злоупотребления своими возможностями.

В качестве начальной точки использования системы учета деятельности студентов планируется кафедра «Математическое обеспечение и применение ЭВМ» Пензенского госуниверситета.

СИСТЕМА АВТОМАТИЗИРОВАННОГО КОМПЛЕКСНОГО АНАЛИЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СЕТЕВЫХ КОМПЬЮТЕРНЫХ СИСТЕМ MAXAPT QUICKEYE

A.В. Бочкин, В.Г. Казаков, С.А. Федосин

Мордовский государственный университет им. Н.П. Огарева

Введение

Темой работы является автоматизированный анализ использования программного обеспечения сетевых компьютерных систем с целью дальнейшего повышения эффективности их эксплуатации. В ходе практической и исследовательской работы были созданы алгоритмы функционирования программного комплекса на основе современных передовых технологий. Проект получил название Maxapt QuickEye.

Возможности и назначение системы

Maxapt QuickEye – система мониторинга работы пользователей на компьютерах. Это незаменимый помощник руководителя, предоставляющий отчеты о том, какие программы и сколько времени использовались.

Особенности Maxapt QuickEye.

- 1) Простота и наглядность в эксплуатации. Надежность и стабильность в работе.
- 2) Отчеты об использовании программ/компьютеров по времени их работы и по времени работы пользователя с ними.
- 3) Группировка программ/компьютеров для формирования групповых отчетов.
- 4) Автоматическая классификация программ по группам (средства разработки, офис, мультимедиа, игры и т.д.).
- 5) Формирование суммированных отчетов и отчетов по динамике.
- 6) Экспорт отчетов в популярные форматы данных.
- 7) Удобная удаленная установка и администрирование модулей слежения, распределение прав доступа пользователей.
- 8) Защита данных, передаваемых программой по сети, с помощью надежных криптографических алгоритмов.
- 9) Компактный размер и высокая скорость обработки баз данных, устойчивость к сбоям и повреждениям.
- 10) Широкий выбор стильных скинов.

Научно-техническая составляющая

В ходе исследовательской работы был проведен анализ большого количества передовых технологий и разработаны алгоритмы функционирования программного комплекса. При этом высокой сложностью математико-логической реализации характеризуются:

- алгоритмы функционирования СУБД программного комплекса, специально разработанной для обеспечения высокой компактности хранимых данных, высокой скорости обработки данных, а также способной работать со структурами данных высокой сложности организации;

- организация структуры хранения *данных об использовании программного обеспечения сетевых компьютерных систем*, ориентированная на высокую компактность, а также на высокую скорость последующего анализа;
- алгоритмы анализа *данных об использовании программного обеспечения сетевых компьютерных систем*, оптимизированные на высокую скорость работы и малый расход оперативной памяти;
- алгоритмы обеспечения безопасности сетевого взаимодействия модулей программного комплекса (с применением надежных криптографических алгоритмов RSA, AES, SHA256) для предотвращения несанкционированного доступа;
- алгоритмы удаленного администрирования системы по сети;
- алгоритмы защиты программы от нелегального копирования с привязкой к заводским параметрам используемого оборудования, а также с использованием криптографической составляющей, исключающей возможность взлома программы посредством написания генератора ключей.

Модульная структура комплекса

Программный комплекс состоит из *модулей сбора данных* и *модуля обозревателя*. *Модуль сбора данных* устанавливается на каждом компьютере, использование программного обеспечения которого необходимо анализировать. На те компьютеры, при помощи которых планируется проводить анализ использования программного обеспечения определенного набора компьютеров, устанавливаются *модули обозревателя*. Так, например, *модули сбора данных* могут быть установлены на рабочих местах в отделах какого-либо подразделения, а *модули обозревателя* могут быть установлены у начальника каждого отдела и у начальника всего подразделения. Причем начальник конкретного отдела будет иметь доступ только к компьютерам своего отдела, а начальник всего подразделения будет иметь доступ ко всем компьютерам всех отделов подразделения.

Технологическая составляющая

Среди технологий, использованных при создании программного комплекса можно выделить следующие.

1. Каждый модуль *сбора данных* включает в себя специальный *Kernel-Mode драйвер* [1], осуществляющий функции перехвата моментов запуска и останова процессов с помощью технологии *Process Structure Routines (PsXxx routines)* [2]. Драйвер разработан в среде *Microsoft Driver Development Kit* на языке *C*.
2. Каждый модуль *сбора данных* включает в себя специальный системный *сервис* [3], запускаемый в отдельном адресном пространстве. Сервис разработан в среде *Microsoft Visual Studio* на языке *C*.
3. Драйвер оперативно оповещает сервис о запусках и остановах процессов. Для этого используется технология синхронизации *named notification event* [4]. Далее сервис получает от драйвера данные через интерфейс *Device Input and Output Control (IOCTL)* [5].
4. Для перехвата моментов переключения активных окон, сообщений мыши и клавиатуры используется технология *Global Hook* [6]. При этом механизмы *WH_MOUSE_LL Hook* и *WH_KEYBOARD_LL Hook* используются для перехвата всех сообщений от мыши и клавиатуры, а комбинация *WH_SHELL Hook* и *WH_CBT Hook* используются для перехвата моментов переключения активных окон. При использовании этих механизмов необходима синхронизация выполнения специальной *библиотеки перехвата*, внедряющейся в адресное пространство каждого процесса, зарегистрировавшего хотя бы одно окно, с сервисом, который накапливает получаемые данные.

Для синхронизации выполнения кода в адресных пространствах различных процессов используются *named mutex* [7] и *named event* [8]. Для быстрого обмена данными между процессами используется буфер специальным образом созданной *секции данных* в РЕ образе библиотеки [9].

5. Сервис работает в *контексте безопасности системы*, но т.к. процессы иногда запускаются по сети, то для получения доступа к файлам этих процессов *сервису* необходимо работать в *сетевом контексте безопасности пользователя*, от имени которого запускается программа. Переключение контекстов безопасности выполняемого кода происходит с использованием технологии *impersonation* [10].

6. Для обеспечения взаимодействия модулей программного комплекса по сети используется механизм *Windows Sockets 2* [11]. Взаимодействие происходит по *сетевому протоколу TCP/IP*.

7. Для ускоренного удаленного автоматизированного развертывания системы в сети используются технологии *Network Share Management* [12], *Network Management* [13], *Service Management* [14].

8. В программном комплексе также применены механизмы *Low-level Access Control* [15] для работы с *security descriptors* [16].

Исходный код системы составляет около 40 тысяч строк, а на его создание было затрачено более 3 лет.

Опыт внедрения

Программный комплекс доступен для загрузки и покупки на сайте www.maxapt.ru.

Программу приобрели: ЗАО «ПИТЦ Геофизика» (г. Пермь), компания ИНСЭЛ (г. Санкт-Петербург), ОАО «Московский Метрострой» (г. Москва), ООО «Авто-мобил» (г. Москва), ЗАО «Торговая компания DOMO» (г. Казань), НИИ кардиологии имени В. А. Алмазова Минздрава РФ (г. Санкт-Петербург), ОАО «Завод Автоприбор» (г. Владимир), ОАО "Гипросинтез" (г. Волгоград), МП «Салехардэнерго» (г. Салехард), ООО «ИКФ АЛЬТ» (г. Санкт-Петербург), ООО «Жилэнергострой» (Вологодская обл., г. Череповец), ОАО АКБ «ЕВРОФИНАНС МОСНАРБАНК» (г. Москва), ООО Завод «Инструмент, Техобслуживание, Ремонт» (Свердловская обл., г. Ревда), ЗАО «Пластик» (г. Челябинск), ООО Научно-исследовательский центр «ФАПРОКС» (г. Барнаул).

В настоящий момент данным программным комплексом оснащены факультет электронной техники и математический факультет Мордовского ГУ им. Н. П. Огарева.

Список литературы

1. Kernel-Mode Driver Architecture. <http://msdn2.microsoft.com/en-us/library/aa973498.aspx>.
2. Process Structure Routines. <http://msdn2.microsoft.com/en-us/library/ms796700.aspx>.
3. Service Reference. <http://msdn2.microsoft.com/en-us/library/ms685974.aspx>.
4. Named notification event. <http://msdn2.microsoft.com/en-us/library/aa490498.aspx>.
5. Device Input and Output Control (IOCTL). <http://msdn2.microsoft.com/en-us/library/aa363219.aspx>.
6. Hooks. <http://msdn2.microsoft.com/en-us/library/ms632589.aspx>.
7. Named mutex. <http://msdn2.microsoft.com/en-us/library/ms682411.aspx>.
8. Named event. <http://msdn2.microsoft.com/en-us/library/ms682396.aspx>.
9. Data segments. <http://msdn2.microsoft.com/en-us/library/thflx4st.aspx>.
10. Impersonation. <http://msdn2.microsoft.com/en-us/library/aa376391.aspx>.
11. Windows Sockets 2. <http://msdn2.microsoft.com/en-us/library/ms740673.aspx>.
12. Network Share Management. <http://msdn2.microsoft.com/en-us/library/aa380486.aspx>.
13. Network Management. <http://msdn2.microsoft.com/en-us/library/aa370672.aspx>.
14. Service Functions. <http://msdn2.microsoft.com/en-us/library/ms685141.aspx>.
15. Low-level Access Control. <http://msdn2.microsoft.com/en-us/library/aa379189.aspx>.
16. Security descriptor. <http://msdn2.microsoft.com/en-us/library/ms721625.aspx>.

ОПЫТ РАЗРАБОТКИ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ МЕДИЦИНСКОЙ ДИАГНОСТИКИ НА ПЛАТФОРМЕ .NET FRAMEWORK

C.B. Винокуров

Марийский государственный технический университет

Развитие медицинских экспертных систем определяется требованиями к более точной и своевременной диагностике патологических заболеваний. Наряду с усложнением тестов, проводимых аппаратными средствами, в медицинской области вводятся электронное представление истории болезни пациента, что может позволить выявлять заболевания на ранних стадиях развития. Вместе с тем, наблюдается узкая специализация экспертных систем по областям использования, что ограничивает возможности диагностирования заболевания.

Задача медицинской диагностики

Задача медицинской диагностики состоит в определении состояния больного (диагноз), при котором наблюдаются определенные признаки (симптомы). На основании имеющихся знаний (опыта лечащего врача, других врачей или рекомендаций органов здравоохранения) система должна, в том случае если пациент болен, указать этиологию болезни и предложить оптимальный курс лечения.

В известных экспертных системах [1] процесс определения диагноза осуществляется либо на основе алгоритмов математической статистики, где происходит поиск частичного соответствия наблюдаемых признаков пациента с признаками у ранее наблюдавшихся пациентов, диагноз которых известен, либо на основе правил, разработанных экспертами для определенного вида заболевания, которые выполняются при соответствии состояния больного картине клинического заболевания. К недостаткам систем со статистическими алгоритмами относят то, что они не обеспечивают соответствующего уровня объяснения полученных результатов. Недостатком систем, основанных на правилах, является то, что каждый случай является сугубо индивидуальным и не всегда можно определить ярко выраженные границы пороговых значений наблюдаемых признаков для соответствующего заболевания. Определенное влияние на это оказывают и факторы ухудшающейся экологической обстановки.

Предложена система, в основе которой заложено моделирование состояния пациента с использованием мультиагентной технологии. При этом достигаются преимущества статистического подхода с накопленным ранее опытом и можно отследить контрольные точки развития заболевания, на основе которых выдавать понятные врачу объяснения действий системы.

Принципы разработки системы

В настоящее время в информационной системе лечебного учреждения (ЛУ) обеспечение функций интеллектуальной поддержки врача возложено на программно-аппаратный комплекс, реализуемый в виде автоматизированного рабочего места (АРМ) врача определенной специализации. В состав такого комплекса обычно входит или компонент, определяющий на основе диалога состояние больного и предлагающий соответствующие рекомендации, или набор справочных материалов, зачастую в полно-

текстовом формате. Данные, используемые в приложениях АРМ-ов поступают либо от соответствующего медицинского оборудования, либо непосредственно от врача, и обычно хранятся в централизованных хранилищах информации. Причем доступ к информации, полученной с определенного АРМ, должен иметь только пользователь этого АРМ. Следовательно, особенностью разрабатываемой системы является необходимость взаимодействия с различными приложениями (часто небольшими), выполняющими специализированную функцию, потребность в которых возникает не очень часто. В то же время, подсистема диагностики является неотъемлемой частью подсистемы, обеспечивающей мониторинг текущего состояния пациента, которая кроме этого вычисляет прогноз развития заболевания и предлагает врачу оптимальную траекторию достижения цели лечения. Для этого, она обеспечивает взаимодействие непосредственно с центральной базой данных ЛУ и использует в работе всю доступную информацию. Как только подсистема мониторинга обнаруживает отклонение состояния пациента от нормы (для группы индивидуумов с характерными особенностями), она генерирует гипотезы возможных заболеваний и передает управление подсистеме диагностики. Подсистема диагностики проводит моделирование состояния пациента с учетом известных моделей органов и функциональных систем организма, на основании которых предлагает соответствующие решения (диагнозы). Клиентское приложение получает информацию по состоянию пациента в рамках рассматриваемых текущих моделей заболеваний. Если системе необходимо для определения диагноза заключение врача другой специализации, то она планирует время посещения пациентом этого врача. В случае, если все данные получены, но однозначно определить диагноз не удается, то система запланирует проведение специальных лабораторных исследований, требуемых для наиболее вероятного заболевания.

Итак, к основным принципам построения системы можно отнести следующие.

- 1) Основная функция системы реализуется службой, которая осуществляет мониторинг текущего состояния пациента. К базе данных учреждения имеет доступ только указанная служба, а приложение использует функции объектов, предоставляемых службой. Тем самым достигается необходимый уровень безопасности конфиденциальной медицинской информации.
- 2) Врач может запросить доступную для него информацию об определенном пациенте, а также комментарии о том, как получена эта информация (специфическая информация предоставляется в рамках предположительного заболевания).
- 3) Информация о пациенте формируется как от соответствующего аппаратного обеспечения (количественные данные), так и непосредственно врачом (качественные данные). Врач должен иметь возможность формировать классы наблюдаемых ситуаций, кроме того, данная функция должна присутствовать в основной службе системы и предлагать врачу автоматически обнаруживаемые классы картин заболеваний.
- 4) Все действия работы с системой (получение данных, выявленные отклонения, причины заболевания) должны протоколироваться.
- 5) Разграничение доступа к информации осуществляется операционной системой (информационной системой ЛУ).

Процессы и архитектура системы

На основании вышесказанного основным процессом, функционирующим в системе, является процесс мониторинга, который активизирует процессы диагностики при изменении состояния пациента (рис. 1).

Диагностика осуществляется моделированием состояния пациента, в основе которого используются группы агентов, представляющие характеристики функциональных систем организма. Таким агентам назначаются программы, посредством которых осуществляется стимуляция деятельности отдельных компонентов функциональной системы организма.

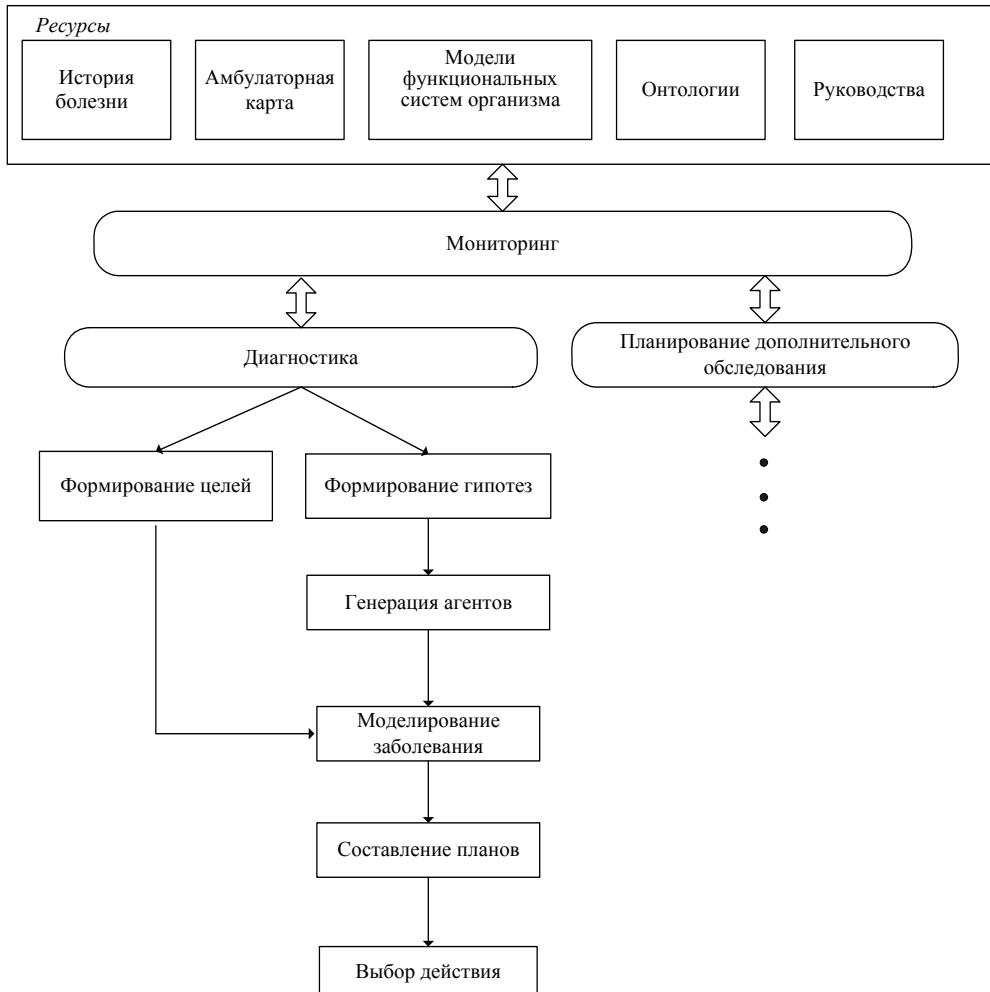


Рис. 1. Процессы, происходящие при диагностировании заболевания

Программы агентов функциональных систем поступают планировщику, который формирует план с использованием библиотек шаблонов. В шаблонах учитываются физиологические данные пациента, анамнез, необходимые инструментальные исследования. Кроме этого, в шаблонах указываются способы изменения программ агентов, которые назначены для данного плана. Далее, выбранные программы, передаются ис-

полнителям, которые выступают в качестве центров управления процессами. При этом исполнители могут повлиять на программу агентов с целью приведения систем организма в равновесие. Внешние проявления заболевания (симптомы) характеризуются работой исполнителей высокого уровня, которые воздействуют на представление функциональных систем организма.

Система выполнена в архитектуре клиент-сервер. В состав функциональных объектов сервера входят:

- подсистема формирования гипотез;
- подсистема генерации агентов (для моделирования изменения состояния пациента);
- подсистема выполнения агентов;
- подсистема руководств;
- подсистема объяснения (с использованием онтологий) [2].

Поскольку в качестве основной функции системы выбрана функция мониторинга текущего состояния пациента, то обоснованным является решение реализации системы как набора утилит в виде служб Windows, отвечающих за действия медицинского работника по получению необходимой ему информации. При работе с данными утилитами формируется запрос на серверную часть системы, которая предоставляет необходимые ресурсы (информация из истории болезни, критерии смоделированного состояния, объяснения полученных результатов).

При реализации системы использовались инструменты Microsoft Office, которые позволили упростить процесс разработки пользовательского интерфейса. Кроме этого, это сократило время обучения персонала поликлиники работе с системой.

Заключение

В работе предложен подход к организации диагностического процесса с использованием групп агентов для моделирования функциональных систем организма. Применение такого подхода позволяет комплексное использование информации для проведения диагностики: процесс диагностики выполняется пошагово, обеспечивая понятные медицинскому работнику объяснения принимаемых решений. Реализованы методы диагностики кардиомиопатий. На данный момент в системе используется только модель сердечно-сосудистой системы. В дальнейшем планируется моделирование и других функциональных систем организма.

Список литературы

1. Брейкин Т. В., Камплова Л. З. и др. Проектирование экспертных систем медицинской диагностики на базе нечеткой логики с применением методов системного моделирования // Управление в сложных системах. Уфа, 2000. <http://asu.ugatu.ac.ru/book/collect/pdf/2000/14.pdf>
2. Клещев А. С., Черняховская М. Ю., Москаленко Ф. М. Модель онтологии предметной области «медицинская диагностика». Ч. 2. Формальное описание причинно-следственных связей, причин значений признаков и причин заболеваний. // НТИ. Сер. 2. 2006. № 2. С. 19–30.

ПОРТАЛ СОТРУДНИЧЕСТВА В ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ «ВЗАИМОДЕЙСТВИЕ»

E.A. Гафнер¹, И.В. Аппель²

¹ Челябинская государственная медицинская академия

² Южно-Уральский госуниверситет

В России ведется активная научно-исследовательская работа студентов, существует и работает множество разнообразных лабораторий и исследовательских коллективов. Они разобщены и не имеют возможности взаимодействовать и обмениваться материалами исследований. Они могут устанавливать и поддерживать связи посредством конференций, периодических изданий и личных контактов.

Аналогичная ситуация налаживается и в области преподавания: есть интересные локальные курсы, есть преподаватели, разрабатывающие их и есть студенты, которым бы полезны эти курсы, но нет средств обеспечения взаимодействия между преподавателями и студентами в условиях территориальной удаленности друг от друга.

Исследователи в области программирования и вычислительной техники находятся в несколько более выигрышном положении – они более активно используют компьютеры, имеют определенный профессиональный опыт и могут самостоятельно организовывать сотрудничество посредством форумов, самодельных веб-сайтов и специализированных сервисов и платформ, таких как Wiki и Sourceforge.

Однако, есть химики, биологи, медики, физики – специалисты, не имеющие достаточной экспертизы в использовании современных средств коммуникации, необходимой для построения эффективно работающих профессиональных web-сообществ.

Им нужен готовый развитый сервис.

Задачей проекта «Взаимодействие» является создание портала в Internet и обеспечение на его основе совместной работы аспирантов и студентов различных вузов, занимающихся исследовательской работой. Кроме этого портал допускает размещение в свободном доступе учебных курсов заинтересованных в этом преподавателей и организацию дистанционных курсов.

Система исходно спроектирована для обеспечения функционала, удовлетворяющего потребности работников различных специальностей, в частности для того, чтобы определить круг возможных задач для нее, в качестве консультантов были приглашены специалисты в области физики, химии и медицины.

Все преподаватели могут размещать в сети материалы своих курсов, проводить разнообразные онлайновые тестирования и вести переписку со студентами. Все исследователи могут публиковать свои письменные работы и рецензировать работы друг друга.

При этом, например, специалисты по программированию могут поддерживать на сайте репозитории кода и технической документации своих проектов, как в качестве исследовательских проектов, так и в виде примеров для обучающихся на дистанционных курсах.

Медики и физики-экспериментаторы могут размещать массивы статистических данных, предложения по части совместного проведения исследовательских работ и использования лабораторий, а также доставке, при необходимости, редкого оборудования и лабораторных препаратов.

Наконец, на платформе портала «Взаимодействие» возможно междисциплинарное взаимодействие, например, физиков-теоретиков и специалистов по высокопроизводительным вычислениям. В частности, возможно размещение на портале запросов на совместное проведение исследований.

Таким образом, Collaborator интегрирует возможности таких сервисов, как: MySpace, LiveJournal и «Мой Круг» (социальная сеть); LiveJournal и Slashdot (специализированные форумы); Slashdot (динамические рейтинги компетентности); arXiv.org (публикация материалов); SourceForge (публикация исходного кодов и документации); Rapidshare (публикация массивов данных).

При этом он позволяет создавать сообщества, оптимизированные под интересы конкретных групп пользователей. Базовым понятием объединения пользователей является «лаборатория». Лаборатория является тематическим сообществом. Тематика исследований лаборатории определяется достаточно широкой, чтобы избежать лишней фрагментации, но достаточно узкой, чтобы обеспечить интерес всех участников в работе сообщества. При этом каждое исследование, материалы по которому публикуются на сайте, привязывается к двум-трем лабораториям.

Более широким объединением является «отдел». «Отдел» представляет некую более широкую предметную область, и все лаборатории привязываются к одной-двум предметным областям.

Для пилотного запуска проекта определены следующие отделы и лаборатории.

- 1) Программирование и вычислительная техника (лаборатории веб-программирования, параллельных вычислений, функционального программирования, системного программирования, вычислительной математики).
- 2) Математика (дискретная математика, вычислительная математика).
- 3) Физика (физика твердого тела, наноструктурные материалы).
- 4) Химия (наноструктурные материалы, биохимия).
- 5) Медицина (патологическая физиология, биохимия).

Данный набор определяется сугубо экспертизой людей, задействованных в разработке, и может быть легко расширен по мере появления на портале участников, ведущих исследования и предлагающих учебные курсы в других областях и способных модерировать общение в рамках лабораторий.

Для удовлетворения всего этого многообразия технических требований система построена на плагинной архитектуре:

- базовая часть системы:
 - необходимые высокоуровневые средства абстракции взаимодействия с IIS, SQL Server и файловой системой;
 - базовые функции пользовательского интерфейса;
 - интерфейс интеграции плагинов;
- плагины:
 - загрузка на сервер письменных работ и аннотаций к ним;
 - форумы для обсуждения публикуемых работ с возможностью рецензирования и средствами формирования динамического рейтинга компетентности пользователей (аналогично карме на Slashdot) – карма определяется в рамках одного отдела;
 - форумы для обсуждения междисциплинарного взаимодействия;
 - форумы для межвузовского и межрегионального взаимодействия;

- загрузка и отображение специфических видов информации;
- полнотекстовый поиск по структурным элементам сайта (лабораториям, отделам) и аннотациям материалов, помещенных в систему;
- разграничение доступа к лабораториям (всем желающим, по утверждению модератора)
- разграничение доступа к материалам (всем, только участникам лаборатории, только доступ к аннотациям).

Все это предусмотрено для обеспечения гибкости системы по мере расширения номенклатуры предметных областей. Также для удобства пользователей базовый интерфейс системы максимально упрощен, но может настраиваться за счет подключения тех плагинов, которые интересуют конкретного пользователя. Предусмотрена интеграция с LiveJournal, «Мой Круг» и MySpace с целью загрузки контактной информации пользователей с них.

Для реализации проекта используется ASP.NET 2.0 и SQL Server Express, проект планируется представить на конкурсе программных проектов Imagine Cup 2007.

ПРОГРАММНАЯ СРЕДА ДЛЯ РАЗРАБОТКИ И РАСПАРАЛЛЕЛИВАНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

K.A. Генералов

Пензенский государственный университет

Генетические алгоритмы – это достаточно новый и довольно мощный механизм поиска экстремума многоэкстремальных функций. Много практических задач сводятся к нахождению экстремума, следовательно, данные задачи можно свести к задачам для решения средствами генетических алгоритмов (ГА), которые оперируют такими объектами, как «ген», «хромосома», «популяция». Принцип работы основан на идеях эволюции живой природы.

Разработан инструмент, позволяющий гибко использовать генетические алгоритмы – язык генетического программирования (GPLN – Genetic Programming Language) [1]. Он реализован в составе среды вычислений GPD – Genetic Programming Developer.

Основаниями актуальности введения языка генетического программирования и среды вычислений являются:

- решение задачи с помощью генетических алгоритмов занимает больше времени при использовании существующих языков;
- при применении других языков необходимо знать все тонкости работы ГА;
- разработчик при решении задачи на других языках программирования больше сосредоточен на ГА, чем на задаче;
- решение задач с помощью ГА на других языках обладает низкой эффективностью;
- при использовании существующих языков и ГА высока вероятность программной ошибки из-за большого объема программы и высокой сложности, вследствие чего задача может быть решена неверно;

- полноценного программного обеспечения, позволяющего использовать ГА для решения поставленных задач, не существует.

- при разработке средств автоматизированного использования ГА, язык программирования – наиболее гибкое и мощное средство.

Среда вычислений GPD по сути является распределённым программным комплексом. В составе системы присутствуют серверная и клиентская части, взаимодействующие через Ethernet. Взаимодействие происходит через специально разработанный протокол прикладного уровня.

Структура системы показана на рис. 1.

Среда вычислений GPD реализована как сетевой сервис на базе ОС Linux. Для увеличения скорости вычислений в язык программирования GPLN заложена возможность параллельных вычислений с помощью команд: send(), recv(), parallel(), autoparallel() [1]. В этом случае вычисления происходят на кластере на базе MPI. Сервер, на котором работает сетевой сервис среды GPD, является консолью кластера. Все вычисления происходят на сервере (или на кластере), клиент только отправляет исходный текст программы на сервер принимает результаты обработки: данные о синтаксических/лексических ошибках, текстовая информация, графическая информация.

Клиентские части среды вычислений содержат: области для ввода и редактирования программы на языке GPLN, реализующие некоторый генетический алгоритм; область для вывода текстовой информации, область для вывода графиков.

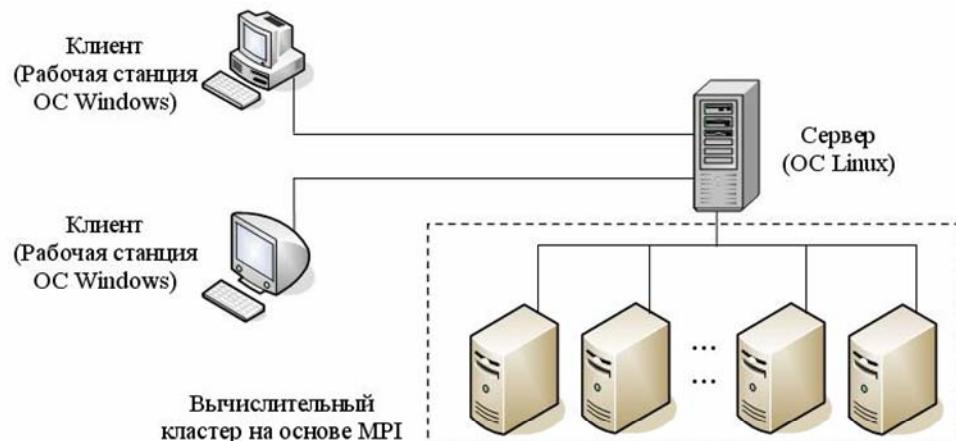


Рис. 1. Общая архитектура системы

Поиск глобального экстремума с помощью генетических алгоритмов процесс достаточно ресурсоёмкий. Для увеличения эффективности использования генетических алгоритмов предложено ряд вариантов: генерация начальной популяции на основе закона распределения [2], динамическая настройка параметров генетических алгоритмов, основанная на предварительном анализе [3]. Однако наиболее эффективным способом увеличения точности и быстродействия вычислений является параллельное выполнение генетического алгоритма.

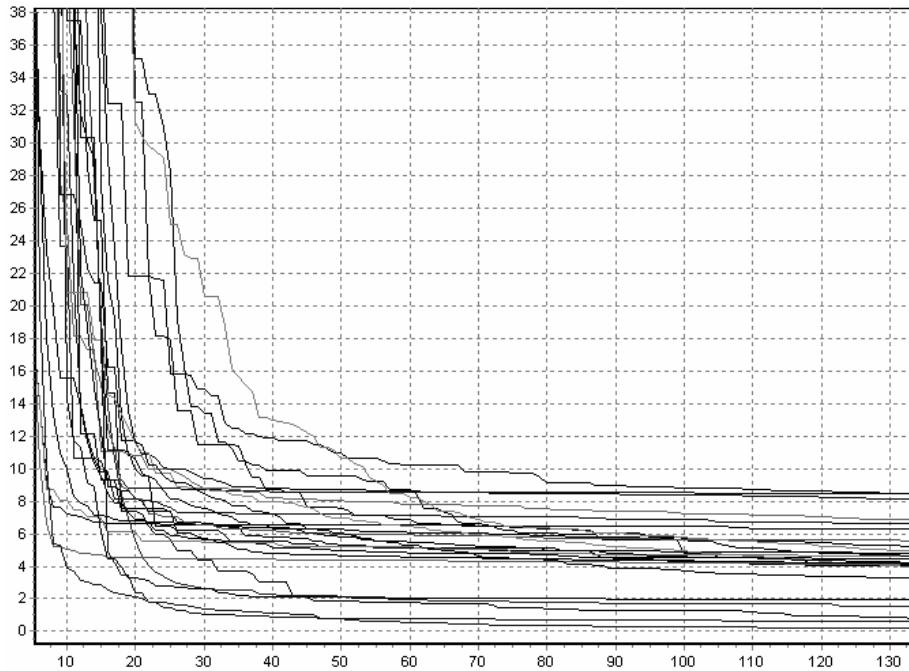


Рис. 2. Параллельное развитие нескольких популяций

На рис. 2 представлено независимое параллельное развитие 27-ми популяций. По оси абсцисс отложен номер поколения, по оси ординат – значение функции приспособленности лучшей хромосомы. В данном случае копии одной популяции развиваются с помощью различных комбинаций операторов кроссинговера и мутации. Каждая ветка в конечном итоге даёт некоторое приближённое решение. Следует выбрать наилучшее решение.

Различают следующие типы параллелизма генетических алгоритмов [4]:

- глобальный параллелизм;
- миграционная модель;
- диффузионная модель.

Использование глобального параллелизма подразумевает распараллеливание вычисления целевой функции и операторов генетического алгоритма. При использовании миграционной модели популяция разбивается на подпопуляции, которые развиваются (вычисляются) независимо в течении некоторого количества итераций. Далее происходит обмен генетическим материалом между подпопуляциями. При использовании данной модели необходимо определить топологию вычислительной системы, размер подпопуляций, правила и частоту миграций. В диффузионной модели популяция разделяется на большое количество немногочисленных подпопуляций. Генетические операторы применяются в ряде ограниченных областей. Достигается эффект «диффузии» (медленного распространения) решения, уменьшается вероятность попадания в локальный экстремум [4]. В среде вычислений генетических алгоритмов достаточно несложно распараллелить генетический алгоритм.

Большое количество задач сводятся к задаче поиска экстремума. Генетические алгоритмы являются альтернативным механизмом решения подобного класса задач. Можно сделать вывод, что разработанная среда вычислений позволяет более эффективно использовать генетические алгоритмы, получать удовлетворительное решение за приемлемое время.

Список литературы

1. Генералов К.А. Язык генетического программирования // Новые информационные технологии и системы: Труды 7 международной научно-технической конференции. Ч. 2. Пенза: Информационно-издательский центр ПГУ, 2006. С. 98.
2. Генералов К.А. Создание начальной популяции генетических алгоритмов на основе плотности распределения // Вычислительные системы и технологии обработки информации: Межвузовский сборник научных трудов. Вып. 6. Пенза: Информационно-издательский центр ПГУ, 2006.
3. Генералов К.А. Динамическая настройка параметров генетических алгоритмов, основанная на предварительном анализе // Научно-техническое творчество молодёжи – путь к обществу, основанному на знаниях: сборник научных докладов, Москва, 2006. С. 253.
4. Родзин С.И. Формы реализации и границы применения эволюционных алгоритмов // Перспективные информационные технологии и интеллектуальные системы. 2002. № 1.

АВТОМАТИЧЕСКАЯ КОРРЕКЦИЯ ДЕФЕКТА «КРАСНЫХ ГЛАЗ» НА ЦИФРОВЫХ ФОТОГРАФИЯХ

A.M. Герасимов, П.А. Колчин, П.П. Кудряшов, С.А. Фоменков

Волгоградский государственный технический университет

Известно, что на фотографиях, сделанных со вспышкой, глаза людей и животных часто имеют неестественный цвет. Так, у людей засвеченные глаза обычно красные (хотя встречаются и желтые, и белые), а у животных могут быть и зеленые, и желтые, и синие. Связано это с тем, что при малом угловом расстоянии от вспышки до объектива свет вспышки отражается от сетчатки и попадает обратно в объектив. Цвет сетчатки глаз человека – красный. У животных же он может быть различным. Фотографии с таким дефектом смотрятся неестественно и, фактически, являются испорченными. Возникновение подобного дефекта является принципиальной проблемой большинства современных фотоаппаратов.

Многие современные графические редакторы позволяют ретушировать указанный дефект в ручном режиме. Однако это неудобно при большом количестве испорченных фотографий и требует определенных навыков работы с графическими редакторами. В силу указанных причин обычно такие фотографии накапливаются в личных фотоальбомах, а их коррекция становится сложной проблемой.

В настоящее время существует несколько различных полуавтоматических и автоматических систем коррекции дефекта «красных глаз». Наиболее качественные из них – разработка лаборатории компании HP и технология компании FotoNation, используемая в некоторых фотоаппаратах компании Nikon.

Авторами разработана автоматическая система коррекции «эффекта красных глаз», которая позволяет обрабатывать фотографии в пакетном режиме без участия человека. Система выполняет комплексный анализ каждой фотографии, состоящий из нескольких стадий. В процессе анализа на изображении выявляются засвеченные и неестественно выглядящие глаза, которые затем корректируются.

Общая схема работы следующая. На первом этапе выявляются цветовые пятна, которые могут быть засвеченными зрачками. Затем с помощью целого ряда эвристик проверяется правдоподобие этой гипотезы. На основе выполненных проверок рассчитывается специальная мера правдоподобия, по которой затем происходит отсечение забракованных вариантов. На последней стадии для каждого цветового пятна рассчитывается вектор изменения цветов, который и будет применен к фотографии для исправления дефекта.

Авторами разработан набор специальных эвристик и алгоритмов, позволяющих с высокой степенью достоверности определять засвеченные зрачки на фотографиях. К таким эвристикам и алгоритмам относятся:

- раннее прогнозирование возможности появления дефекта на основе анализа условий фотосъемки;
- методы обнаружения ярких цветовых пятен;
- алгоритмы снижения «шума»;
- сегментация по цвету кожи;
- статистическая верификация распределения цветов в радужной оболочке и белке глаз;
- поиск и подтверждение характерных черт лица;
- верификация по соотношению геометрических параметров деталей глаз и лиц.

Кроме того, в системе используются и некоторые известные методы. В частности, алгоритм HAAR для распознавания образов лиц и глаз. Для повышения скорости и надежности распознавания авторами были использованы дополнительные методы анализа изображений, включающие в себя:

цветокоррекцию и сглаживание с сохранением контуров на изображении;

предварительное прогнозирование расположения детектируемых объектов на изображении;

исключение ложных срабатываний путем дополнительной верификации;

использование грубых и тонких каскадов в алгоритме HAAR.

На последней стадии (стадии исправления изображения) система рассчитывает, каким образом необходимо изменить цвета на фотографии. Для каждого испорченного глаза формируется наиболее естественное изображение-патч, которое затем бесшовно накладывается на фотографию. Какое именно изображение генерируется, зависит от типа, цвета и формы дефекта. Такой подход позволяет наилучшим образом учитывать и исправлять все возможные искажения цветов на зрачках глаз.

По завершении коррекции происходит сохранение изображения. Но, как известно, каждое пережатие фотографии форматом JPEG, являющимся стандартом де-факто в современных фотоаппаратах, ухудшает качество изображения в силу особенностей алгоритма сжатия. Как правило, «красные глаза» занимают по площади очень небольшую часть фотографии, поэтому полное пережатие всей исправленной фотографии алгоритмом JPEG избыточно. В системе реализована специальная поддержка изображений в формате JPEG, которая позволяет выполнять коррекцию без полного повторного сжатия.

Система реализована на языке C++ с использованием библиотеки Intel OpenCV. Для повышения производительности предусмотрена возможность использования мощности всех доступных на компьютере процессоров, а так же специально оптимизированных математических алгоритмов от компании Intel (Intel Performance Primitives Library).

Как уже было сказано выше, цвет человеческих глаз на испорченных фотографиях может различаться в широких пределах от белого или желтого до темно-красного. Относительные размеры и форма таких участков тоже варьируются. Такая высокая вариативность засветок является наиболее сложным препятствием для систем автоматической коррекции, т.к. не позволяет применять простейшие подходы (например, элементарный поиск круглых красных пятен). Ни одна из современных систем не справляется с задачей в полной мере. Кроме того, все существующие системы ориентированы на коррекцию единственного типа дефекта – глаза с красной засветкой. При этом совершенно не учитываются другие типы дефектов, например белые блики.

Разработанная система отличается более высокой точностью обнаружения, поскольку в ее основу положено не только распознавание образов или обработка лишь статистических характеристик цветовой информации. Система позволяет корректировать не только фотографии плохого качества со множеством различных красных объектов в кадре, но и такие случаи, которые никем до сих пор не обрабатываются (неестественные блики различных цветов в глазах людей на заднем плане).

ИСПОЛЬЗОВАНИЕ СЕМАНТИЧЕСКИХ ЗАВИСИМОСТЕЙ ПРИ ПОИСКЕ ФИЗИЧЕСКИХ ЭФФЕКТОВ

А.М. Герасимов, П.А. Колчин, С.А. Фоменков

Волгоградский государственный технический университет

Текущее представление баз данных физических эффектов (БД ФЭ) в виде последовательного списка удобно только с технической точки зрения простоты ведения и поддержки. Списковое представление не достаточно для восприятия человеком и резко теряет обозримость при количестве элементов списка больше 10-15.

Наиболее естественной формой представления информации для человека является ассоциативная сеть образов. Мышление позволяет держать в памяти тысячи понятий и ситуаций, перемещаясь от одного к другому по семантическим связям. Нашим предположением является следующий тезис: изменение «точки зрения» на БД ФЭ обнаруживает в БД новые интересные зависимости, визуализация которых позволит пользователям применять новые виды поиска и лучше ориентироваться во множестве ФЭ.

Построение семантических связей между ФЭ является сложной задачей, можно задавать явно, определяя специальные атрибуты в базе данных. Однако гораздо более интересной задачей является автоматизированное выявление зависимостей на основе анализа данных, доступных системе работы с ФЭ. Чем больше разных данных доступно

для анализа, тем выше вероятность обнаружения новых полезных зависимостей между ФЭ. По способу получения можно выделить два типа данных, доступных для анализа: статические (хранимые в БД ФЭ) и динамические (получаемые при работе человека с БД ФЭ).

К статическим данным относятся различные виды описания ФЭ, которые уже присутствуют в БД: тезаурусы, текстовые описания, ключевые слова, графики, формулы, изображения и другие. К динамическим относятся данные, как именно и в какой последовательности используются ФЭ, как они были найдены.

Анализ статической информации дает возможность найти общие закономерности в массивах данных о ФЭ. Например, оценивать степень схожести различных ФЭ или формировать кластеры ФЭ, подходящих для решения отдельных задач. Использование динамически получаемой информации о процессе работы пользователя с БД ФЭ позволит выявлять закономерности в обращении к ФЭ в определенных поисковых ситуациях. Например, появится возможность составлять список ФЭ, «наиболее часто используемых человеком при решении определенного класса технических задач».

В отдельных случаях для автоматизированного выполнения поиска зависимостей возможным представляется использование подходов и методов Data Mining. Однако широкое их использование затруднено, т.к. БД ФЭ содержит либо очень формализованную информацию, либо слабоструктурированный текст; кроме того, объем базы не очень велик. Выявление связей между ФЭ на уровне описаний потребует использование методов понимания и обработки текстовой информации.

Важным аспектом является визуализация сконструированных зависимостей и предоставление пользователю возможности навигации по связям между ФЭ. Нами предлагается использование при визуализации метафоры «фокальных объектов». Это означает, что просматриваемый на данный момент ФЭ находится в фокусе восприятия пользователя, а наиболее близкие ФЭ при этом должны быть доступны пользователю за один шаг.

Перечислим наиболее важные, по нашему мнению, виды информации для конструирования сетей семантических зависимостей между ФЭ:

- 1) ФЭ возможного продолжения цепочки ФПД относительно текущего в любом направлении;
- 2) группировка по областям физики;
- 3) группировка по зависимостям и характеру изменения входов и выходов;
- 4) ФЭ, сгруппированные по классификатору типовых задач;
- 5) наиболее близкие по формальному описанию ФЭ;
- 6) наиболее непохожие по формальному описанию ФЭ;
- 7) сходные по смыслу ФЭ (например, преобразуют воздействия только из разных областей физики – горизонтальный срез подобия в БД ФЭ);
- 8) визуализация сети всех возможных цепочек ФПД;
- 9) наиболее важные «узловые» ФЭ (например, много цепочек ФПД зависят от данного ФЭ);
- 10) «уникальные» ФЭ (по каким-то признакам единственные в БД);
- 11) ФЭ, являющихся «над-» или «под-эффектами» (подмножество признаков другого эффекта);
- 12) сходные словосочетания и предложения в текстовых описаниях ФЭ;
- 13) сходные применения;

- 14) сходные формулы зависимостей;
- 15) сходные изображения;
- 16) одинаковые фамилии ученых в описаниях;
- 17) одинаковые источники информации;
- 18) смысловые зависимости текстовых описаний;
- 19) наиболее популярные у пользователей ФЭ;
- 20) часто используемые совместно ФЭ;
- 21) наиболее полезные ФЭ (полезность с точки зрения возможного использования и популярности у пользователей);
- 22) кластеры ФЭ, часто используемых в определенных областях или для создания заданных классов объектов.

Указанный список в дальнейшем будет доработан и послужит базой для выявления и использования неявных зависимостей при поиске ФЭ.

СОЦИОНИКА И ВЗАИМОДЕЙСТВИЕ ЧЕЛОВЕКА И КОМПЬЮТЕРА

A.M. Герасимов, С.А. Фоменков

Волгоградский государственный технический университет

Соционика – это молодая наука, возникшая на стыке психологии, социологии и информатики в 80-х годах прошлого века, базируется на работах К.Г. Юнга и некоторых других ученых. Соционика изучает аспекты взаимодействия между людьми в информационно-психологическом плане. В этой науке вводится понятие «тип информационного метаболизма», или «психотип», описывающее, как именно человек воспринимает информацию разного типа и как на нее реагирует (т.е. как сам при этом продуктирует новую информацию, взаимодействуя с окружающим миром и людьми).

Выделяют 16 психотипов. Они различаются по ряду психологических признаков (базис Юнга, признаки Рейнина): логика-этика, интроверсия-экстраверсия, рациональность-иррациональность, интуиция-сенсорика, квестимность-деклатимность, позитивизм-негативизм и др. Поведение людей с разными психотипами существенно отличается. И, наоборот, люди с одинаковыми психотипами в одинаковых ситуациях ведут себя сходным образом.

Таким образом, можно прогнозировать, как человек воспримет ту или иную информацию или действие и как на них отреагирует. В основном соционика занимается вопросами взаимодействия между людьми, однако работа человека за компьютером тоже является видом информационного взаимодействия, а потому эта наука может оказаться полезной и в этом случае.

Наша гипотеза заключается в том, что пользовательский интерфейс можно рассматривать как набор информационных каналов, подающих разную по типу информацию. Зная, какую именно информацию необходимо подать человеку, и каким образом лучше это сделать, можно проектировать очень удобные и эффективные пользовательские интерфейсы для решения конкретных задач определенными группами пользователей.

Методика проектирования при этом будет заключаться в выявлении некоторых «интегративных» психотипов отдельных групп пользователей (а возможно даже и психотипов конкретных пользователей) и формировании наиболее адекватных каналов и способов подачи информации. Например, используя соционику, можно выявить наилучшие способы организации интерфейсов обучающих или развлекательных программ.

Известно, что стиль и оформление подаваемой информации сильно влияет на возможность ее восприятия. Так, например, многим людям сложно работать с абстрактным графическим элементом интерфейса «дерево», а для других – это наиболее адекватный и понятный способ выражения мыслей. Представляется возможным выделить наиболее удобные элементы графического пользовательского интерфейса для людей с различным типом информационного метаболизма. В дальнейшем такие знания можно использовать при построении узко специализированных пользовательских интерфейсов, рассчитанных на конкретные группы людей, или в процессе динамического формирования интерфейса в зависимости от типа работающего в конкретный момент человека.

Развитие этой гипотезы требует сбора и анализа обширного статистического материала, обобщение которого позволит выявить наиболее характерные паттерны организации информационного взаимодействия человека и компьютера.

СРАВНЕНИЕ ПОДХОДОВ К ВЫЧИСЛЕНИЮ ИНТЕГРАЛЬНОЙ ФУНКЦИИ РАСПРЕДЕЛЕНИЯ МНОГОМЕРНОЙ НОРМАЛЬНОЙ СЛУЧАЙНОЙ ВЕЛИЧИНЫ

А.С. Горбунова, Е.А. Козинов, И.Б. Мееров, А.В. Шишков

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В этой работе мы рассматриваем задачу в следующей постановке:

$$P(b) = \frac{1}{\sqrt{|\Sigma|(2\pi)^m}} \int_{-\infty}^{b_1} \int_{-\infty}^{b_2} \dots \int_{-\infty}^{b_m} e^{-\frac{1}{2}x^t \Sigma^{-1} x} dx, \quad (1)$$

где $x = (x_1, x_2, \dots, x_m)^t$, Σ – симметричная положительно определенная матрица ковариаций размерности $m \times m$; $b_i < \infty$, $i = \overline{1, m}$. Проблема вычисления функции $P(b)$ встает в задачах теории надежности, прогнозирования рисков в страховой сфере, подсчета справедливой цены опционов европейского типа и многих других.

Метод Генза (Genz)

Этот метод основан на приведении области интегрирования к единичному гиперкубу [1–3]. Для этого осуществим замену переменных $x = Cy$, где $\Sigma = CC^t$ – разложение Холецкого для матрицы Σ . При этом функция изменится следующим образом:

$$P(b) = (2\pi)^{-\frac{m}{2}} \int_{-\infty}^{b_1'(y)} e^{-\frac{y_1^2}{2}} \dots \int_{-\infty}^{b_m'(y)} e^{-\frac{y_m^2}{2}} dy, \quad (2)$$

где $b_i'(y) = (b_i - \sum_{j=1}^{i-1} c_{i,j} y_j) / c_{i,i}$. Положив $y_i = \Phi^{-1}(z_i)$ для $i = \overline{1, m}$, где

$$\Phi(y_i) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{y_i} e^{-\frac{t^2}{2}} dt, \quad \text{получим} \quad P(b) = \int_0^{e_1} \int_0^{e_2(z_1)} \dots \int_0^{e_m(z_1, z_2, \dots, z_{m-1})} dz, \quad \text{где}$$

$$e_i(z_1, z_2, \dots, z_{i-1}) = \Phi((b_i - \sum_{j=1}^{i-1} c_{i,j} \Phi^{-1}(z_j)) / c_{i,i}). \quad \text{Наконец, сделав замену } z_i = e_i \omega_i,$$

$i = \overline{1, m}$, приведем функцию к виду:

$$P(b) = e_1 \int_0^1 e_2(w) \int_0^1 \dots e_m(w) \int_0^1 dw, \quad (3)$$

где $e_i(\mathcal{W}) = \Phi((b_i - \sum_{j=1}^{i-1} c_{i,j} \Phi^{-1}(e_j(w) \omega_j)) / c_{i,i})$. После этих преобразований для вычисления значения функции может быть использовано квази-Монте-Карло интегрирование. Кроме того, для увеличения скорости сходимости метода необходимо перед модификацией функции осуществить сортировку переменных таким образом, чтобы наибольшие значения b_i оказались пределами внутренних интегралов.

Грубое Монте-Карло интегрирование

Мы можем модифицировать (1) следующим образом:

$$P(b) = \frac{1}{\sqrt{|\Sigma| (2\pi)^m}} \int_{-\infty}^{b_1} \int_{-\infty}^{b_2} \dots \int_{-\infty}^{b_m} e^{-\frac{1}{2} x^t \Sigma^{-1} x} dx = \int_X \varphi(x_1, x_2, \dots, x_m) dx, \quad (4)$$

где $X = (-\infty, b_1] \times (-\infty, b_2] \times \dots \times (-\infty, b_m]$. Тогда интеграл можно записать в виде:

$$P(b) = \int_X \varphi(x_1, x_2, \dots, x_m) dx = \int_{R^m} g(x) dF(x), \quad (5)$$

где $g(x)$ – индикаторная функция множества X :

$$g(x) = \begin{cases} 1, & \text{if } x \in X, \\ 0, & \text{иначе.} \end{cases}$$

Процедура интегрирования предполагает следующие шаги:

1) генерирование случайных чисел x_i , $i = 1, \dots, N$ с функцией плотности φ ;

2) вычисление несмешенной оценки $P(b)$ как $\bar{P} = \frac{1}{N} \sum_{i=1}^N g(x_i)$.

Можно доказать, что стандартная ошибка оценки выражается формулой $E = \sqrt{\bar{P}(1 - \bar{P}) / N}$ [2,4].

Оценки Буля-Бонферрони (Boole-Bonferroni)

Рассмотрим $P(b)$ как вероятность пересечения событий $A_1 = \{x_1 < b_1\}$, $A_2 = \{x_2 < b_2\}, \dots, A_m = \{x_m < b_m\}$, тогда

$$P(A_1 A_2 \dots A_m) = 1 - P(\overline{A_1} \cup \overline{A_2} \cup \dots \cup \overline{A_m}). \quad (6)$$

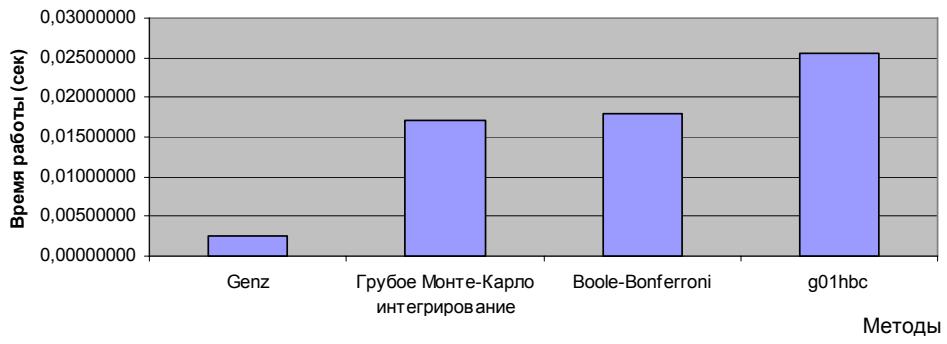
Эту величину можно оценить как $\frac{2}{k^* + 1} S_1 - \frac{2}{k^*(k^* + 1)} S_2 \leq P(\bigcup_{i=1}^m A_i) \leq \frac{2}{m} S_1 - \frac{2}{m} S_2$ [4], где $S_k = \sum_{1 \leq i_1 \leq \dots \leq i_k \leq m} P(A_{i_1} \dots A_{i_k})$, $k = 1, \dots, 2p$, $2p \leq m$ – это так называемые биномиальные моменты случайной величины μ , она определяется как количество событий A_1, A_2, \dots, A_m , которые осуществились при случайной попытке, т.е.

$$E[C_\mu^k] = S_k, \quad k = 1, \dots, m; \quad k^* = 1 + \left\lfloor \frac{2S_2}{S_1} \right\rfloor.$$

Экспериментальные данные

Результаты были получены с использованием следующего программного и аппаратного обеспечения: Intel® Pentium® D, 2,8GHz, 2 Gb RAM, L2 cache 1Mb; Intel® Math Kernel Library (Intel® MKL) 8.1; Intel® C/C++ Compiler 9.1; Microsoft ® Visual Studio 2003; NAG ® C Library, Mark 7.

Для наглядности сравним реализации этих алгоритмов с функцией g01hbc из библиотеки NAG.



Выводы

1. Описанные выше алгоритмы показывают относительные неплохие по качеству результаты и значительно опережают по времени классический алгоритм интегрирования, реализованный в g01hbc.
2. Наиболее хорошие результаты по времени работы и по качеству полученных результатов показывает метод Генза.

Наименование параметра	Значение параметра
Максимальное количество итераций	100 000
Требуемая точность	0.0001
Значения вектора b	(1,0, 4,0, 2,0)
Значения матрицы ковариаций	$\begin{bmatrix} 1 & 3,0/5,0 & 1,0/3,0 \\ 3,0/5,0 & 1 & 11,0/15,0 \\ 1,0/3,0 & 11,0/15,0 & 1 \end{bmatrix}$
Размерность задачи	3
Оцениваемое значение функции	0,82798
Полученное значение	
Метод Генза	0,82799000
Грубое Монте-Карло интегрирование	0,82796700
Нижняя оценка Буля-Бонферрони	0,82482600
Верхняя оценка Буля-Бонферрони	0,82795700
g01hbc	0,82798200

Список литературы

1. Genz A. Numerical computation of the multivariate normal probabilities // J. of Computational and Graphical Statistics 1 (1992) 141–150.
2. Genz A. Comparison of methods for the computation of multivariate normal probabilities // Computing Science and Statistics 25 (1993) 400–405.
3. Bernsten J., Espelid T.O., Genz A. An adaptive algorithm for the approximate calculation of multiple integrals // ACM Transactions on Mathematical Software 17 (1991) 437–451.
4. Szántai T. Improved bounds and simulation procedures on the value of the multivariate normal probability distribution function // Annals of Operation Research 100 (2000) 85–101.
5. Genz A. Methods for generating random orthogonal matrices // revised version published in Monte Carlo and Quasi-Monte Carlo Methods 1998, H. Niederreiter and J. Spanier (Eds.). Berlin, Springer-Verlag, (1999) 199–213.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЧИСЛЕННЫХ МЕТОДОВ РЕШЕНИЯ СИСТЕМ СТОХАСТИЧЕСКИХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

A.C. Горбунова, Е.А. Козинов, И.Б. Мееров, А.В. Шишкин

Нижегородский государственный университет им. Н.И. Лобачевского

Введение

Стохастическое дифференциальное уравнение (СДУ) – разновидность дифференциального уравнения, в котором один или несколько членов представляют собой случайный процесс. Системы стохастических дифференциальных уравнений (ССДУ) находят свое применение при математическом моделировании физических, химических, биологических, экономических и других процессов. Теория вероятностей, математическая статистика и их области приложения, такие как, например финансовая математика широко используют ССДУ.

В данной работе приводится обзор методов численного интегрирования ССДУ и их сравнительный анализ по вычислительной сложности и порядкам сходимости.

Постановка задачи

Рассмотрим систему стохастических дифференциальных уравнений в форме Ито:

$$\begin{aligned} dX_t^j &= a^j(X_t)dt + \sum_{k=1}^m b_k^j(X_t)dW_t^k \\ X_0^j &= X_0, \quad j=1,\dots,n, \\ 0 \leq t &\leq T, \end{aligned} \tag{1}$$

где $W^k = (W_t^k)_{t \geq 0}$ – независимые винеровские процессы, $k=1,\dots,m$; a^j, b_k^j – некоторые функции. Мы решаем задачу нахождения численного решения $X_t^j = X_j(t, W_t^1, \dots, W_t^m)$ системы (1) на интервале $[0, T]$. В одномерном случае существование и единственность решения гарантируется условием на коэффициенты $a(t, X_t)$ и $b(t, X_t)$:

$$|a(t, x)| + |b(t, x)| \leq K |x|. \tag{2}$$

Качество численных методов нахождения решения ССДУ характеризуется порядком локальной, глобальной и сильной сходимости.

Определение 1. Численная схема решения ССДУ X_n имеет локальный γ и глобальный β порядки сходимости, если

$$E(|X_{t_n} - \bar{X}_n|^2 | X_{t_{n-1}} = \bar{X}_{n-1}) = O(\Delta t^{\gamma+1}) \tag{3}$$

и

$$E(|X_T - \bar{X}_N|^2 | X_0 = \bar{X}_0) = O(\Delta t^\beta) \tag{4}$$

соответственно, где $\Delta t = T / N$ – шаг метода, X и \bar{X} – аналитическое и численное решения.

Определение 2. Численная схема решения ССДУ X_n имеет сильную сходимость порядка α , если при фиксированном шаге $\Delta t = T / N$

$$E(\max_{t_n < T} \|X_n - \bar{X}_n\|) \leq C\Delta t^\alpha. \quad (6)$$

Описание методов

В результате анализа литературы были выявлены около 20 методов решения СДУ и несколько их модификаций для случая ССДУ. Для решения прикладных задач наиболее часто используется метод Эйлера, также распространены методы Рунге-Кутта различных порядков и их модификации.

Предлагаются схемы с использованием коэффициентов специального вида, которые повышают порядок сходимости для частных классов задач. Существуют алгоритмы, позволяющие использовать адаптивный шаг интегрирования, но по сравнению с методами для решения ОДУ, в случае СДУ возникает сложность корректного пересчёта стохастической составляющей. Разработан также метод со случайным шагом интергрирования – для него не доказаны теоремы сходимости, но численные результаты показывают уменьшение ошибки. Все методы отличаются вычислительной сложностью и ограничениями, налагаемыми на коэффициенты.

Для реализации были выбраны 3 метода решения СДУ в одномерном случае.

- Метод Эйлер-Маруйама (Euler-Maruyama)

$$\begin{aligned} X_j &= X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})\Delta W_j, \\ j &= 1, \dots, N. \end{aligned} \quad (7)$$

Порядок сходимости: $\gamma = 1$, $\beta = 1$, $\alpha = 1/2$.

Это наиболее распространённый метод в силу небольшой вычислительной сложности.

- Схема, свободная от производных (Derivative-free)

$$\begin{aligned} X_j &= X_{j-1} + f(X_{j-1})\Delta t + g(X_{j-1})\Delta W_j + [g(X_{j-1} + \sqrt{\Delta t}g(X_{j-1})) - g(X_{j-1})] \frac{(\Delta W_j^2 - \Delta t)}{2\sqrt{\Delta t}}, \\ j &= 1, \dots, N. \end{aligned} \quad (8)$$

Порядок сходимости: $\gamma = 2$, $\beta = 2$.

- Эффективный метод Рунге-Кутта, использующий коэффициенты Ито

$$\begin{aligned}
X_j &= X_{j-1} + \frac{1}{2}[F_1 + F_2]\Delta t + \frac{1}{40}[37G_1 + 30G_3 - 27G_4]\Delta W_j + \frac{1}{16}[8G_1 + G_2 - 9G_3]\sqrt{3\Delta t}\Delta, j = 1, \dots, N, \\
F_1 &= f(X_{j-1}), \\
F_2 &= f(X_{j-1} + F_1\Delta t + G_1\Delta W_j), \\
G_1 &= g(X_{j-1}), \\
G_2 &= g(X_{j-1} - \frac{2}{3}G_1(\sqrt{3\Delta t} + \Delta W_j)), \\
G_3 &= g(X_{j-1} + \frac{2}{9}G_1(\sqrt{3\Delta t} + 3\Delta W_j)), \\
G_4 &= g(X_{j-1} - \frac{20}{27}F_1\Delta t + \frac{20}{27}(G_2 - G_1)\Delta W_j - \frac{10}{27}G_2(\sqrt{3\Delta t})). \tag{9}
\end{aligned}$$

Порядок сходимости: $\gamma = 2$, $\beta = 2$ для линейных и $\gamma = 3$, $\beta = 3$ – в случае нелинейных уравнений.

Выбор данных методов обусловлен их активным использованием для широкого круга задач, т.к. они не имеют дополнительных ограничений на коэффициенты уравнений и не требуют вычисления производных. Это подтверждает анализ литературы в различных областях применимости СДУ.

По схожим причинам для ССДУ чаще всего используется метод Эйлер-Маруйама, который и был реализован

$$\begin{aligned}
X^k(t_{i+1}) &= X^k(t_i) + f(X^k(t_i))\Delta t + g(X^k(t_i))\Delta W^k(t_i), \\
\Delta W^k(t_i) &\sim N(0, \Delta t). \tag{10}
\end{aligned}$$

Порядок сходимости: $\alpha = 1/2$.

Численные эксперименты

Результаты были получены с использованием следующего программного и аппаратного обеспечения:

- Intel® Pentium® M, 1,7GHz, 1 Gb RAM
- Intel® Math Kernel Library (Intel® MKL) 8.1
- Intel® C/C++ Compiler 9.1(опции /O2 и /QxP)
- Microsoft ® Visual Studio 2003

Мы применили выбранные методы для решения различного типа задач – линейных, нелинейных и с различной степенью влияния стохастической составляющей.

Численные эксперименты показали, что методы демонстрируют хорошую сходимость на разных классах задач.

Рассмотрим линейное уравнение

$$dX_t = \lambda X_t dt + \mu X_t dW_t, X(0) = 1 \tag{11}$$

с аналитическим решением

$$X(t) = \exp\left\{\left(\lambda - \frac{1}{2}\mu^2\right)t + \mu W(t)\right\}.$$

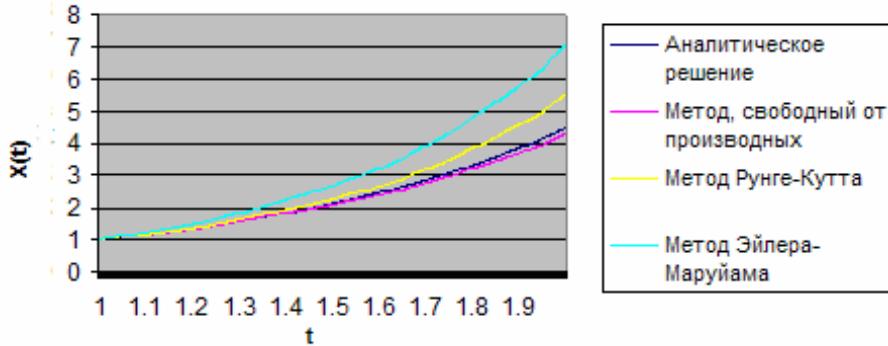


Рис. 1. Аналитическое и численное решение линейной задачи

Рис. 1 показывает, что наилучшую сходимость для данной задачи обеспечивает метод, свободный от производных, наихудшую – метод Эйлера.

Решение нелинейной задачи со слабой ($\beta = 0.1$) и сильной ($\beta = 1.5$) стохастической составляющей показало, что в первом случае все методы довольно стабильны, во втором – расходится метод Рунге-Кутта, а метод, свободный от производных, дает наилучший порядок сходимости.

Выбранные методы были протестированы на нелинейной задаче (рис. 2):

$$dX_t = -(1 + \beta^2 X_t)(1 - X_t^2)dt + \beta(1 - X_t^2)dW, X(0) = 0. \quad (12)$$

с аналитическим решением

$$X(t) = \frac{e^{-2t+2\beta W(t)} - 1}{e^{-2t+2\beta W(t)} + 1}.$$

Одной из важнейших характеристик метода является его вычислительная сложность. Результаты экспериментов показали (см. рис. 3), что наиболее быстрым является метод Эйлера-Маруйама.

Проведён анализ зависимости вычислительной сложности многомерного метода Эйлера от размерности задачи. В качестве тестовой использовалась следующая задача:

$$dX_t^i = \mu^i X_t^i dt + \sigma^i X_t^i \sum_{j=1}^N c_{ij} dW_j, X_0^i = 1, \\ i = 1, N, \quad (13)$$

где c_{ij} – разложение по Холецкому матрицы ковариаций, μ^i , σ^i – положительные константы. Численные эксперименты показали линейную зависимость (рис. 4).

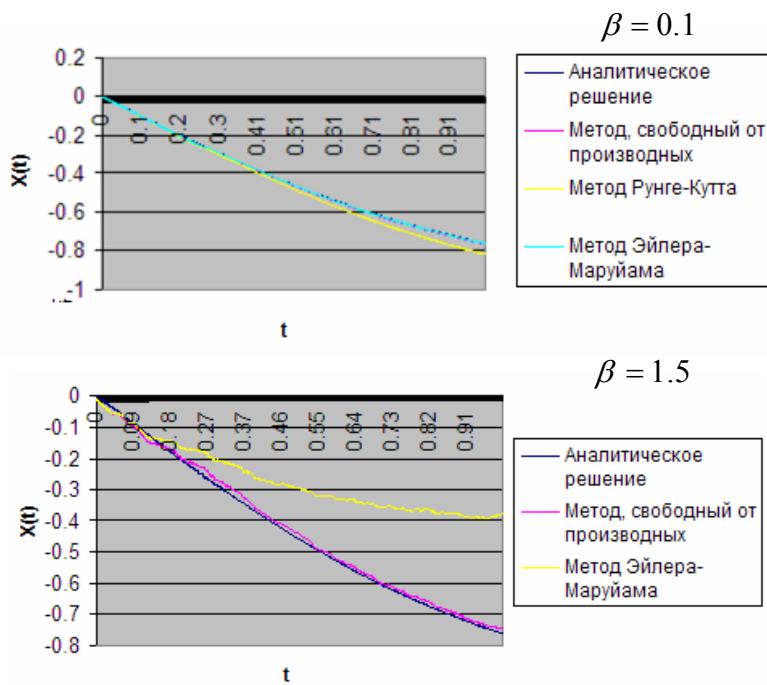


Рис. 2. Аналитическое и численное решение нелинейной задачи со слабым влиянием стохастической составляющей

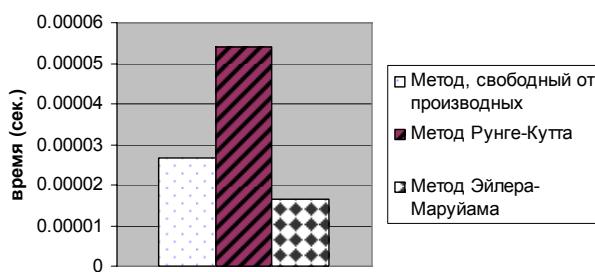


Рис. 3. Сравнение вычислительной сложности методов

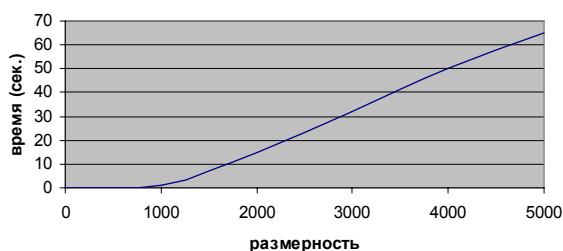


Рис. 4. Зависимость времени вычислений от размерности задачи при использовании много-dimensionalного метода Эйлера

Выводы

1. В прикладных задачах наиболее широкое распространение получил метод Эйлера для решения СДУ и ССДУ.
2. Среди рассмотренных методов лучшую сходимость в тестовых задачах демонстрирует метод, свободный от производных. При этом качество численного решения сильно зависит от класса решаемых задач.
3. Многомерный метод Эйлера имеет линейную зависимость времени вычислений от размерности задачи.

Список литературы

1. Булинский А.В., Ширяев А.Н. Теория случайных процессов. М.: Физматлит, 2003.
2. Ширяев А.Н. Основы стохастической финансовой математики. Т. 1,2. М.: Фазис, 1998.
3. Nigel J. Newton, Asymptotically efficient Runge-Kutta methods for a class of Ito and Stratonovich equations // SIAM J. Appl. Math. Vol. 51. № 2. Р. 542–567, April 1991.
4. Yoshihiro Saito, Taketomo Mitsui, Stability analysis of numerical schemes for stochastic differential equations // SIAM J. Numer. Anal. Vol. 33, No. 6. Р. 2254-2267, December 1996.
5. Tocino A., Ardanuy R. Runge-Kutta methods for numerical solution of stochastic differential equations // J. of Computational and Applied Mathematics. 2002. 138. Р. 219–241.

СИСТЕМА ПРОЕКТИРОВАНИЯ КИНЕМАТИКИ ПРОСТРАНСТВЕННЫХ МЕХАНИЗМОВ «MECHANICS STUDIO .NET» НА ОСНОВЕ MANAGED DIRECTX

E.C. Городецкий

Нижегородский госуниверситет им. Н.И. Лобачевского

Постановка задачи

Задача проекта состоит в разработке современной программной системы для проектирования пространственных механизмов. Программа должна иметь средства визуального редактирования механизмов, собираемых из некоторого набора параметризованных деталей. На втором этапе должна быть реализована возможность кинематического расчёта движения механизма. Программа ориентирована на специалистов, занимающихся исследованием пространственных механизмов.

Общее описание программы

Предметной областью разрабатываемой программы является теория механизмов и машин. Под механизмом [1] понимается совокупность взаимосвязанных твёрдых тел, предназначенная для преобразования движения входов на одном или нескольких твёрдых телах в движение на других твёрдых телах (выходах). Механизм может быть представлен своей структурной схемой, на которой выделяют звенья (твёрдые тела), соединяющиеся в подвижные кинематические пары с помощью различных типов геометрических элементов (вращательных, сферических, поступательных и др.).

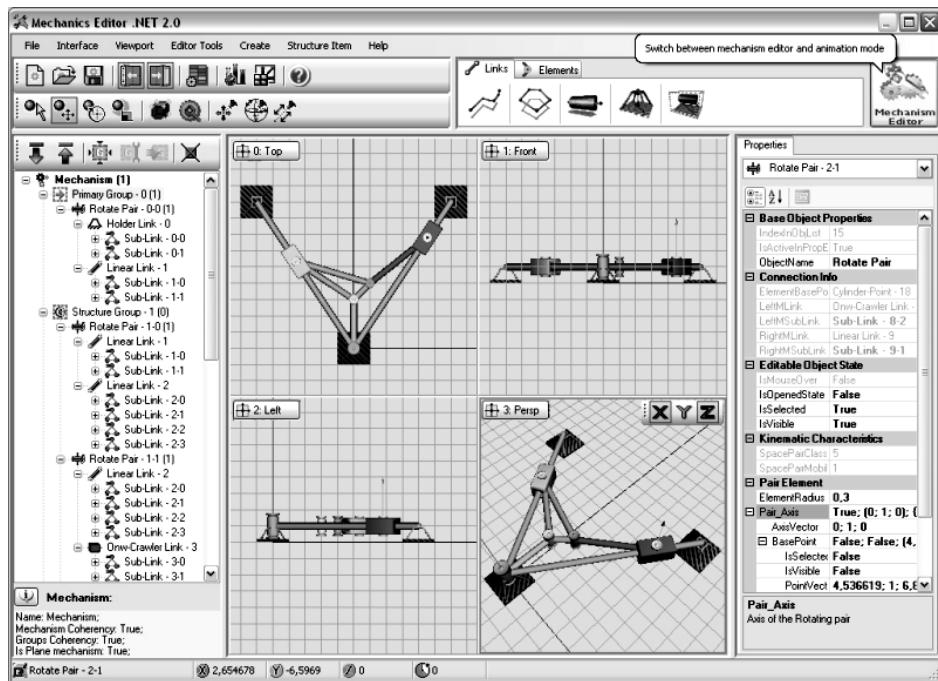


Рис. 1. Главное окно программы в режиме редактирования механизма

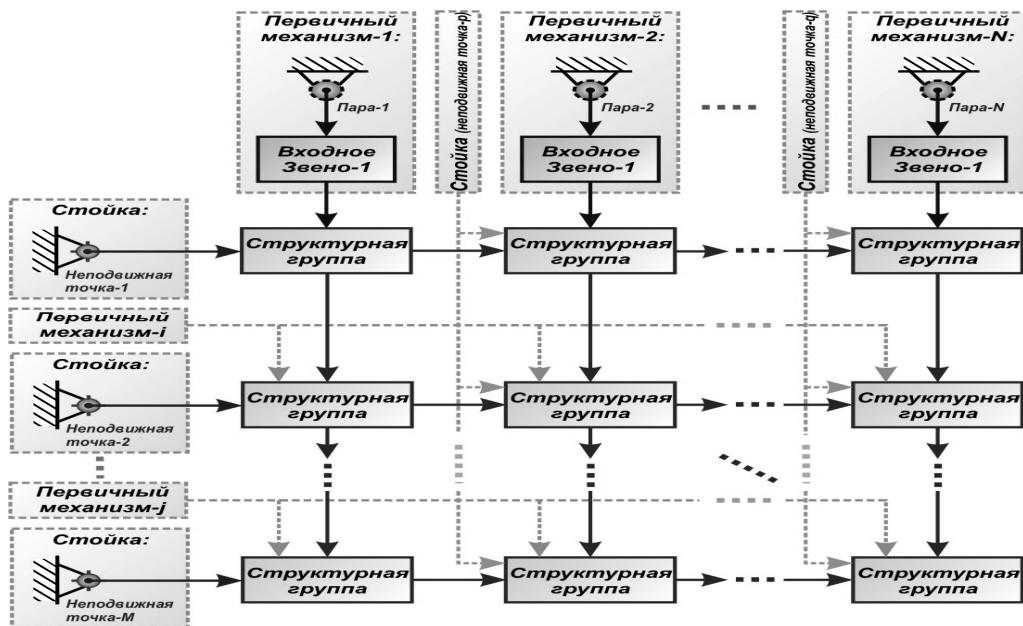


Рис. 2. Иллюстрация принципа Accura

Назначение программной системы заключается в создании моделей пространственных механизмов (ПРМ) средствами визуального 3D-моделирования, а так же в выполнении кинематических расчетов над построенными механизмами. Программа обеспечивает моделирование структурной схемы механизма и производит её рендеринг на нескольких проекциях. Программа имеет инструментарий, позволяющий легко оперировать с пространственными объектами (синтезировать и редактировать их), представлять внутреннюю структуру механизма и описывать конструкцию в терминах теории механизмов. Имеются методы сохранения и загрузки механизмов из файлов в формате XML.

При реализации проекта выполнены проектирование и разработка типов данных для внутреннего представления ПРМ, позволяющие эффективно оперировать с механизмом и решать расчетные задачи. Создан развитый графический интерфейс приложения (рис. 1), реализованы необходимые алгоритмы.

Описание решения прямой задачи кинематики механизмов

Согласно принципу Ассура [1], всякий механизм может быть представлен как совокупность первичных механизмов и наложение структурных групп, т.е. групп нулевой подвижности (рис. 2).

Прямая задача о положении для механизмов ставится следующим образом. Задано движение всех входных звеньев, удовлетворяющее связям, которые накладывают кинематические пары, соответствующих первичных механизмов. Необходимо найти движение всех остальных звеньев механизма, либо сообщить о невозможности реализации положения в тот или иной момент времени.

Положения всех структурных групп механизма могут быть рассчитаны при условии, что механизм имеет правильно построенную структуру (согласно принципу Ассура). Программа осуществляет проверку структуры механизма в процессе его конструирования и графически сообщает об ошибках.

Таким образом, прямая задача о положениях механизма сводится к решению ряда отдельных более простых задач расчета положений структурных групп механизма [3]. Эта задача имеет конечное число решений, поскольку структурная группа, по определению имеет нулевую подвижность – т.е. является статически определимой конструкцией. Существует два основных подхода к расчету структурной группы [3]:

- 1) аналитический подход основан на выводе точной формулы зависимости выходных параметров группы от её входов;
- 2) численный подход – выводится система нелинейных алгебраических уравнений относительно неизвестных параметров группы, которая затем решается численно.

Для окончательной формализации алгоритма решения прямой задачи кинематики механизма, необходимо установить последовательность расчета его структурных групп. Последовательность должна быть построена таким образом, чтобы положения поводков группы, необходимые для вычисления её положения были определены на предыдущих шагах. На рис. 3 приведён пример механизма выдвижения закрылков самолёта. Механизм состоит из одного первичного механизма и 4-х структурных групп ВВВ (из трех вращательных пар), связанных друг с другом. Каждая группа имеет два входа, необходимых для расчета её положения. Исходя из этих требований, строится последовательность расчета структурных групп, показанная на рис. 4.

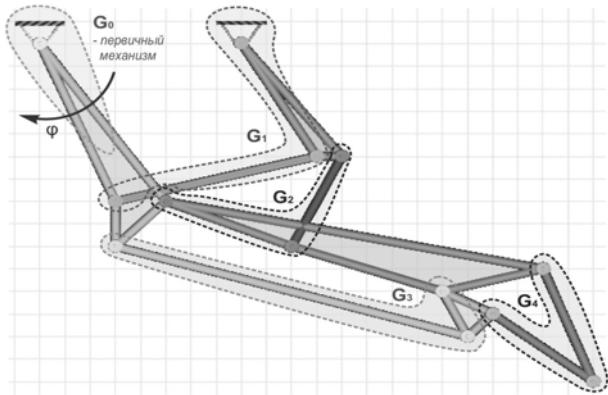


Рис. 3

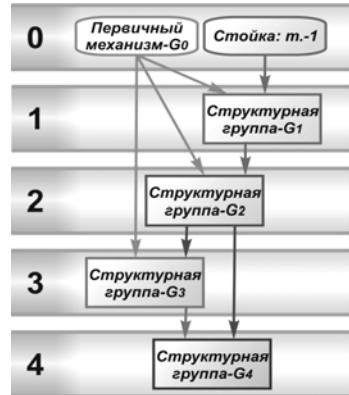


Рис. 4

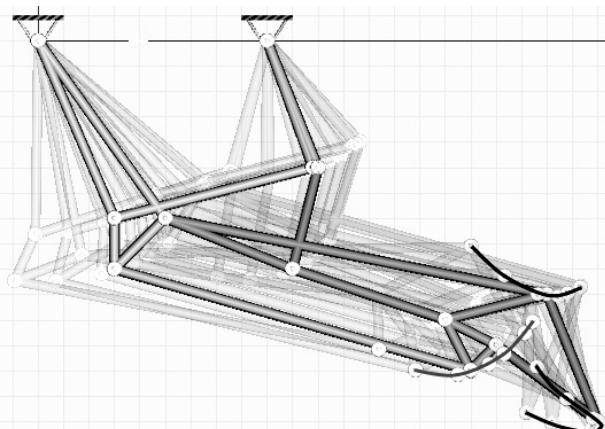


Рис. 5

Вычисления производятся по уровням 0, 1, 2 и т.д., сверху вниз (стрелками показаны соединения групп друг с другом). Программа реализует алгоритм, позволяющий построить последовательность расчета групп для любого корректного механизма.

Положения структурных групп ВВВ вычисляются аналитически [2]. Рассчитав положения механизма для разных углов поворота входного звена, мы получим движение механизма, показанное на рис. 5.

На данный момент про-

грамма поддерживает аналитическое вычисление групп ВВВ, ВСпС, ВВП и ВПВ [2]. Планируется расширение поддерживаемого набора групп.

Демонстрационная версия программы и технологии её реализации

Программа разрабатывалась для платформы Microsoft .NET 2.0, что позволило значительно сократить затраты на кодирование. Разработка выполняется в среде Microsoft Visual Studio 2005 на языках Visual C# и Visual C++/CLI. Причём C++/CLI используется для программирования ядра системы – классов представления и расчёта механизмов, а C# – для реализации программного интерфейса. Для программирования 3D графики используется библиотека Managed DirectX 9. Подробное описание этого проекта и демоверсия программы размещены на сайте <http://www.mechstd.narod.ru>.

Список литературы

1. Фролов К. В., Попов С. А., Мусатов А. К. и др. Теория механизмов и машин: учеб. для вузов / Под ред. К.В. Фролова – М.: Высш. шк., 1987. 496 с.
2. Турлапов В.Е. Задача о положениях и классификация одноконтурных структурных групп пространственных рычажных механизмов // Электронный журнал «Прикладная геометрия». МАИ, 2000. Вып. 2. № 2. С. 1–22.
3. Теория механизмов и машин: учеб. для вузов / К.И. Заблонский, И.М. Белоконев, Б.М. Щекин. – Киев: Выща шк. Головное изд-во, 1989. 376 с.

СЕРВИС WINDOWS COMMUNICATION FOUNDATION ДЛЯ ОРГАНИЗАЦИИ ВИДЕОКОНФЕРЕНЦИЙ «ГОВОРЯЩИХ ГОЛОВ»

Е.С. Городецкий, А.Н. Половинкин

Nижегородский госуниверситет им. Н.И. Лобачевского

Постановка задачи

Задача разработки сервиса возникла в результате работы над проектом Virtual Class Room (VCR) [1] по созданию системы видеоконференций виртуальных «говорящих голов». Конференция в системе VCR включает отображение «живой» головы удалённого пользователя, но в отличие от аналогичных систем, в VCR анимируется синтезированная 3D модель головы человека. Анимация головы осуществляется на основе параметров, поступающих от подсистемы распознавания лица пользователя по видеоизображению, получаемому с камеры. Таким образом, для осуществления анимации всех голов пользователей в каждом из удалённых клиентских приложений, необходимо реализовать обмен параметрами анимации голов в режиме реального времени. Такой подход к организации приложений видеоконференций позволяет значительно сократить нагрузку на сеть (отказавшись от передачи видеопотока) и в то же время увеличить мультимедийные возможности приложений, за счёт одновременного показа нескольких говорящих голов и даже целого класса.

Главным образом от сервиса требуется обеспечение быстрой и безопасной передачи данных между сервером и клиентом. Кроме передачи положений голов, сервис также должен осуществлять выполнение всех подготовительных операций к проведению онлайновой лекции, включая регистрацию пользователей и составление расписаний лекций.

Основные функциональные требования

Доступ к сервису должен осуществляться через локальную сеть или сеть Интернет по протоколу HTTP, обеспечивая достаточно быструю и безопасную передачу данных. Обеспечение безопасности означает:

- проверку личности пользователя;
- передачу только той информации, которая предназначена пользователю;

- передачу информации по защищённому каналу, исключающему возможность перехвата чужих данных.

Поскольку приложение предназначено для проведения конференций или лекций, то её пользователи делятся на две категории:

- учителя, отвечающие за организацию и проведение лекций;
- студенты, имеющие возможности регистрации на лекции и участие в них.

Далее приводится набор основных операций сервиса:

- 1) регистрация новых пользователей;
- 2) авторизация пользователей – проверка личности по имени и паролю;
- 3) получение информации о зарегистрированных пользователях;
- 4) загрузка виртуальных голов для личного или общего использования;
- 5) создание и изменение расписания лекций;
- 6) регистрация студентов на лекции и отмена регистрации;
- 7) загрузка презентаций для запланированных лекций;
- 8) получение собственного и общего расписания лекций;
- 9) получение подробной информации о текущей (уже начавшейся лекции), включающей презентацию, информацию об учителе и всех зарегистрировавшихся студентах (включая модели их виртуальных голов);
- 10) обмен анимационными параметрами положений голов всех участвующих в конференции пользователей;
- 11) изменение текущего слайда презентации, отображаемого студентам.

Технология Windows Communication Foundation

Для реализации сервиса VCR была выбрана новейшая технология создания сервис-ориентированных приложений Windows Communication Foundation (WCF) [5], входящая в последнюю версию платформы .NET Framework 3.0. Технология WCF имеет целый ряд преимуществ по сравнению со всеми существующими аналогами.

- Реализация сервиса не зависит от способа его использования.
- Хостом сервиса может являться любое .NET приложение или IIS.
- Декларативное конфигурирование через конфигурационный файл.
- Поддерживается ряд стандартных наиболее широко используемых транспортных протоколов, таких как: HTTP, TCP, Pipes, MSMQ.
- На выбор предлагается несколько стандартных кодировщиков сообщений: Text, Binary, MTOM (Message Transmission Optimization Mechanism – новый кодировщик, обеспечивающий наибольшую скорость).
- Так же возможно применение схем обеспечения безопасной и надёжной доставки сообщений за счёт использования протоколов WS-*.

Высокоуровневая архитектура сервиса

Все данные, с которыми работает сервис, хранятся в базе данных на SQL Server 2005. Основными задачами сервиса являются обработка, чтение-запись из БД, приём и отправка данных.

Для работы с данными базы создан отдельный логический уровень – так называемый уровень бизнес-логики. Этот уровень содержит набор классов, являющихся объектным представлением таблиц базы данных и «умеющих» работать с этими данными. Каждой из таблиц базы соответствует класс, позволяющий выполнять загрузку, сохранение или удаление данных и получать массивы связанных бизнес-объектов, соответственно связи по первичному ключу. Имеется механизм отслеживания созданных

бизнес-объектов в соответствии с записями в таблицах БД, позволяющий исключить создание дублирующихся бизнес-экземпляров. Уровень бизнес логики использует библиотеку ADO.NET для доступа к базе данных.

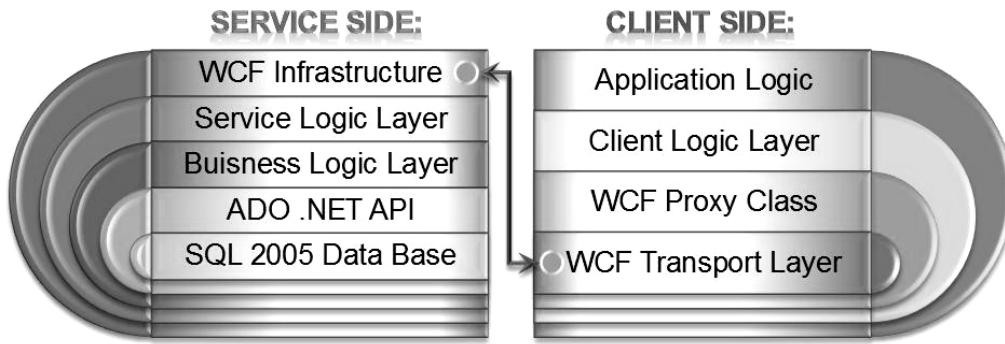


Рис. 1. Логические уровни сервиса и клиента

Уровень сервисной логики содержит так называемый сервисный контракт – интерфейс с описанием операций сервиса и класс самого сервиса, реализующий этот контракт. Реализация операций сервиса использует уровень бизнес логики, за счёт чего код становится простым и элегантным. Для корректной работы, сервис требует включение сессионного режима, чтобы обеспечить использование единожды открытого соединения с БД и помнить текущего авторизованного пользователя.

Вся инфраструктура для работы сервиса полностью обеспечивается платформой WCF. В качестве основного хоста сервиса был выбран IIS. Взаимодействие клиента с сервисом осуществляется через прокси-класс, автоматически генерируемый по метаданным сервиса. Для удобства, создан уровень клиентской логики, осуществляющий вызов методов прокси-класса, обработку возможных ошибок и некоторые дополнительные действия.

Структура базы данных показана на рис. 2.

Назначение её таблиц.

- User – данные о зарегистрированных пользователях.
- UserHead – модели общих и персональных голов пользователей.
- UserTrace – данные о сеансах присутствия пользователя на лекциях.
- UserTraceItem – последовательные данные изменения параметров головы пользователя, в течении сеанса присутствия на лекции.
- Lection – расписание лекций, созданное учителями.
- LectionVisitor – записи о регистрации студентов на лекции из расписания.
- LectionTrace – информация о переключении номера слайда презентации, а также рукописных и текстовых заметок к этим слайдам.

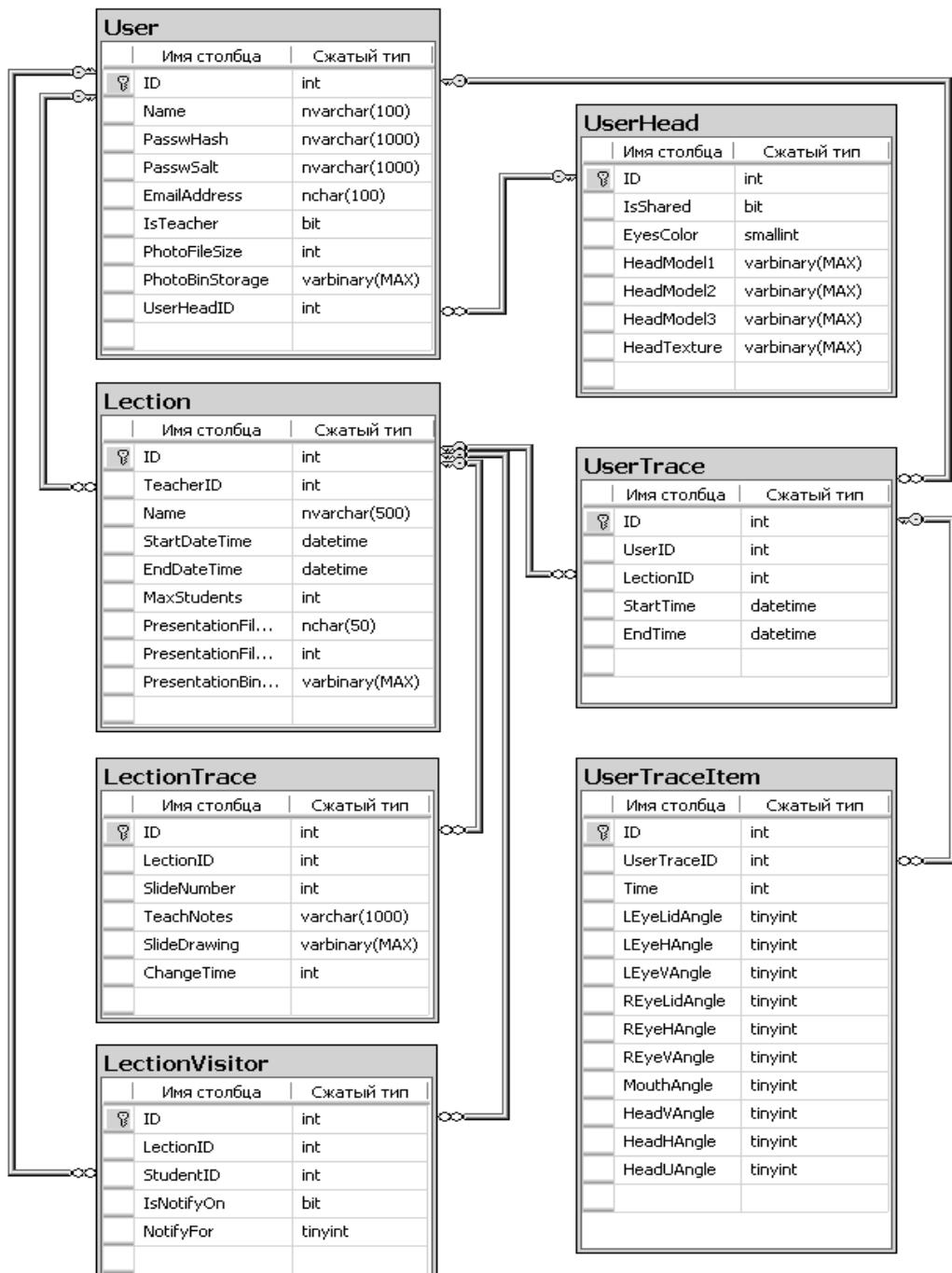


Рис. 2. Таблицы и связи базы данных

Ролевая безопасность методов сервиса

– Название Метода	– Требуется авторизация? *	– Разрешено Студенту?	– Разрешено Учителю?
– RegisterUser	– ✗ Нет	– –	– –
– AuthorizeUser	– ✗ Нет	– –	– –
– GetUserInformation	– ✓ Да	– ✓ Да	– ✓ Да
– ScheduleLecture	– ✓ Да	– ✗ Нет	– ✓ Да
– ReScheduleLecture	– ✓ Да	– ✗ Нет	– ✓ Да
– RegisterToLecture	– ✓ Да	– ✓ Да	– ✗ Нет
– UnRegisterFromLecture	– ✓ Да	– ✓ Да	– ✗ Нет
– GetFutureLectures	– ✗ Нет	– –	– –
– GetLecturesSchedule	– ✓ Да	– ✓ Да	– ✓ Да
– GetCurrentLecture	– ✓ Да	– ✓ Да	– ✓ Да
– ExchangeHeadPositions	– ✓ Да	– ✓ Да	– ✓ Да
– SetLectureTraceInfo	– ✓ Да	– ✗ Нет	– ✓ Да
– GetLectureTraceInfo	– ✓ Да	– ✓ Да	– ✓ Да

* Перед выполнением операции клиент должен выполнить аутентификацию, с помощью метода AuthorizeUser, предоставив правильные имя пользователя и пароль.

Выводы

За счёт использования технологии Windows Communication Foundation удалось выполнить все предъявленные требования. Результатом работы стали две библиотеки: с сервисной и клиентской логикой. Библиотека с сервисной логикой позволяет создавать различные сервисные хосты и настраивать их для работы в любом сетевом окружении. Клиентская библиотека содержит конечные методы работы с сервисом и может подключаться к различным клиентским приложениям.

Испытание сервиса в работе по локальной сети показало, что при обмене анимационными параметрами положений голов между 5 клиентами позволяет поддерживать скорость работы порядка 30-50 обменов в секунду, что более чем достаточно для анимации моделей говорящих голов.

Список литературы

1. Джувал Лоуи. Основы WCF. Что вам необходимо знать об односторонних вызовах, обратных вызовах и событиях // MSDN Magazine №11, (59) Ноябрь, 2006. С. 15-28.

**ИСПОЛЬЗОВАНИЕ НОВЕЙШИХ ТЕХНОЛОГИЙ MICROSOFT
В РЕАЛИЗАЦИИ КОММУНИКАЦИОННОЙ СИСТЕМЫ
ОРГАНИЗАЦИИ И ПРОВЕДЕНИЯ ВИДЕОКОНФЕРЕНЦИЙ
«VIRTUAL CLASS ROOM»**

Е.С. Городецкий, С.В. Сидоров

Нижегородский госуниверситет им. Н.И. Лобачевского

Описание проблемы и постановка задачи

Поиск эффективного способа обучения больших групп людей во все времена являлся актуальной проблемой. Эта проблема особенно обострилась с возникновением дистанционных форм обучения, в связи с потерей прямого взаимодействия учителя с учениками. Во время дистанционного обучения, учитель, как правило, контактирует со своими учениками, используя специализированный форум, средства электронной почты и, реже, средства удалённого показа презентаций. В результате, учитель не может отслеживать реакцию аудитории на рассказываемый им материал, а так же контролировать процесс обучения, из-за потери обратной связи с учениками. Ученики же теряют ощущение присутствия на лекции, не видя друг друга и самого учителя, в связи с чем, так же теряется эффективность обычного аудиторного обучения.

На сегодняшний день существует множество программ предоставляющих возможности интерактивного общения двух собеседников, с передачей их голоса и видеоизображения по локальной сети или сети Интернет. Проблема состоит в отсутствии программных средств, проведения интерактивных конференций для достаточно больших групп людей, обеспечивающих возможности живого общения всех её участников и контроля над процессом обучения со стороны лектора.

Задачей проекта Virtual Class Room (VCR) является разработка программной системы, позволяющей организовывать лекции, набирать учеников и проводить интерактивные конференции, видя лица всех её участников, слыша голос лектора (или одновременно нескольких собеседников) и просматривая презентацию.

Основные функциональные требования

К разрабатываемой системе предъявляются следующие требования.

1. Управление пользователями:
 - регистрация с разделением ролей на преподавателей и студентов;
 - система авторизации пользователей;
 - разграничение доступа пользователей по ролям.
2. Организация лекций:
 - создание расписания планируемых лекций учителями;
 - регистрация студентов на запланированные лекции.
3. Показ презентаций:
 - предварительная загрузка презентации учителем на сервер;
 - просмотр текущего слайда во время лекции всеми её участниками;
 - переключение текущего отображаемого слайда учителем.
4. Отображение «живых» лиц во время лекции:
 - учитель и все студенты должны видеть «живые» лица друг друга;
 - автоматический контроль присутствия пользователя на лекции.

5. Передача голоса во время лекции:

трансляция голоса лектора (с микрофона) всем присутствующим студентам; подключения к разговору нескольких студентов.

Технология виртуальных говорящих голов

После определения требований, предъявляемых к данной системе, возникает проблема их реализации, связанная, прежде всего, с ограниченными возможностями сети Интернет по передаче видео и аудиопотоков для реализации «живых» лиц и голосового общения в режиме on-line.

Одновременная передача одного исходящего и N входящих видеопотоков с камер, снимающих каждого из пользователей, совершенно невозможна по обычному подключению к сети Интернет, не говоря уже о параллельной передаче аудиопотока. Поэтому, для реализации просмотра «живых» лиц всех участников лекции было решено применить принципиально новую технологию – анимацию моделей говорящих голов, рабочающую по следующим принципам.

1. Пользователя снимает веб-камера, видеоизображение с которой обрабатывается программой.
2. На полученном видеоизображении программа распознаёт лицо пользователя, включая рот, глаза и зрачки [2].
3. Программа анализирует полученные о лице данные и вычисляет параметры головы: 3 угла её поворота, угол открытия рта, угол открытия век левого и правого глаза, по 2 угла ориентации зрачков обоих глаз.
4. Полученные параметры головы передаются по сети к каждой из программ участников лекции [1].
5. Конечная программа, получив положение и другие параметры головы, изменяет их на заранее подготовленной 3D модели головы соответствующего пользователя .

Таким образом, за счёт специальной программной обработки изображения можно получить положения самой головы и передавать их по сети, для этого предлагается использовать заранее подготовленную 3D модель. Для реализации этой функциональности применяются следующие библиотеки и технологии:

- Intel OpenCV (Computer Vision) и обёртка SharperCV (для использования под .NET) – библиотека используемая для распознавания лица в видеопотоке;
- Managed DirectX 9.0 – библиотека программирования 3D графики.



Другие технологии реализации

Для реализации остальной функциональности программы были использованы следующие библиотеки и технологии:

- Microsoft .NET 3.0 – управляемая платформа для исполнения приложения;
- Windows Communication Foundation (WCF) – новейшая технология построения распределённых систем, применённая для реализации сервиса;
- Windows Presentation Foundation (WPF) – технология создания интерфейсов нового поколения, применённая в клиентском приложении VCR;
- XML Paper Specification (XPS) – формат документов, используемый для хранения презентаций в системе VCR;

- SQL Server 2005 – используемая СУБД для базы данных VCR;
- ASP .NET 2.0 – технология создания сайтов;
- Microsoft Conference XP – библиотека, использованная для передачи и слияния аудиопотоков.

Высокоуровневая архитектура

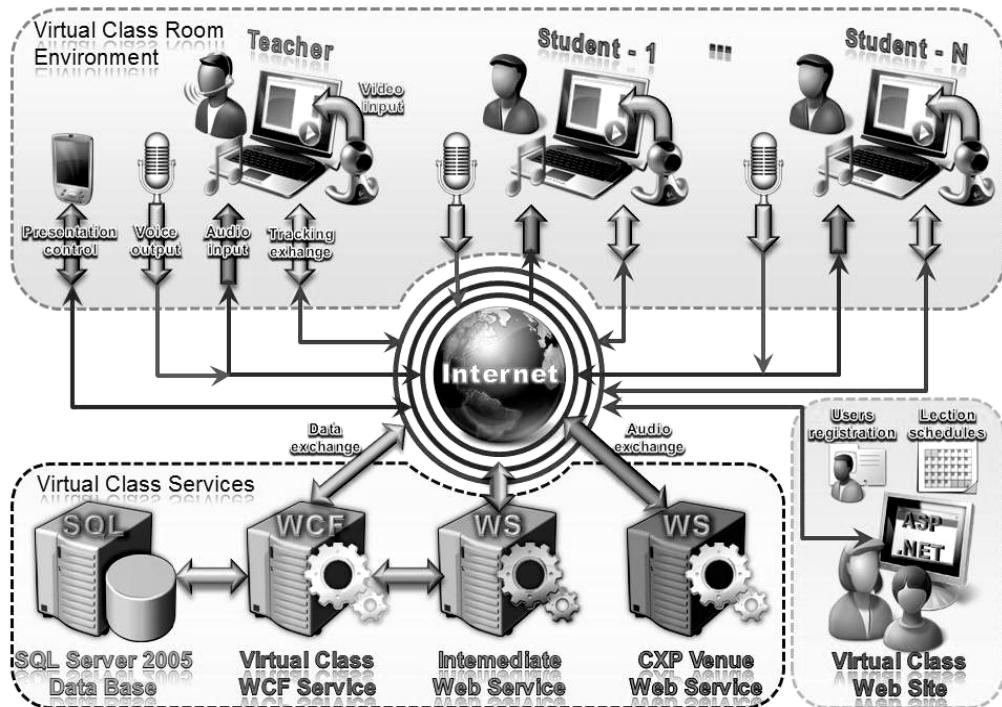
Система Virtual Class Room проектировалась как распределённая система, состоящая из ряда самостоятельных компонент, предназначенных для выполнения определённых групп задач и взаимодействующих по протоколу HTTP через центральный WCF сервис. В системе VCR были реализованы следующие компоненты.

1. VClassBase – база данных системы VCR [1] на SQL Server 2005.
2. VClassService – центральный сервис [1], осуществляющий операции с БД и передачу данных (сервису посвящена отдельная статья) на WCF.
3. VClassSite – веб-сайт на ASP.NET 2.0 осуществляющий регистрацию Пользователей и организацию лекций.
4. VClassVenue – специализированный веб-сервис (Venue Service) из Conference XP, используемый для передачи и слияния аудиопотоков.
5. VClassWinClient – клиентское Windows приложение на WPF, играющее роль виртуального пространства для проведения лекции. Функциями этого приложения являются:
 - а) аутентификация пользователя и ожидание начала ближайшей лекции из его расписания [1];
 - б) загрузка всей информации о начавшейся лекции из БД посредством сервиса, включая модели голов всех зарегистрировавшихся пользователей и презентацию лекции [1];
 - в) распознавание лица с камеры и вычисление параметров головы [2];
 - г) обмен положениями голов всех подключенных пользователей [1];
 - д) анимация моделей голов всех подключённых пользователей;
 - е) отображение презентации и управление переключением слайдов;
 - ж) трансляция аудиопотока говорящих пользователей.
6. VClassPpcClient – клиентское приложение для КПК, позволяющее управлять ходом презентации и добавлять рукописные заметки на слайды.

Выходы

Разработанная программная система Virtual Class Room демонстрирует применение новых подходов и технологий построения систем дистанционного общения и обучения. Важно отметить, что использование технологии виртуальных говорящих голов, совместно с распознаванием лица человека по видеоизображению, позволяет значительно повысить интерактивность коммуникационных приложений и создать эффект «присутствия», не увеличивая при этом сетевых расходов. Разработка этой функциональности значительно облегчилась за счёт применения библиотеки Intel OpenCV, имеющей методы работы с видеокамерами и каркас для распознавания изображений.

Использование открытой библиотеки Conference XP для передачи звука позволило задействовать самые современные технологии компрессии аудиопотоков и передачи их через Интернет, только за счёт предоставляемых сервисов и исходного кода CXR.



Применение .NET Framework 3.0 при создании приложений VCR позволило использовать такие современные технологии как Windows Communication Foundation (WCF) и Windows Presentation Foundation (WPF). Построение сервисной архитектуры на основе WCF позволило достичь необходимой скорости для обмена данными положений виртуальных голов по Интернету и обеспечить её безопасность. Использование новейшей технологии создания графических интерфейсов WPF позволило применить оригинальные подходы оформления, ранее практически не реализуемые в оконных приложениях Windows.

Список литературы

1. Джувал Лоуи. Основы WCF. Что вам необходимо знать об односторонних вызовах, обратных вызовах и событиях// MSDN Magazine №11, (59) ноябрь 2006. С. 15-28.
2. Вейнхардт Майкл. Основы создания приложений. Создайте удобную пользовательскую среду на основе Windows Presentation Foundation // MSDN Magazine №11, (59) ноябрь 2006. С. 36-47.

ОСОБЕННОСТИ ВНЕДРЕНИЯ СРЕДЫ ОБЕСПЕЧЕНИЯ УЧЕБНОГО ПРОЦЕССА В ВЫСШЕМ УЧЕБНОМ ЗАВЕДЕНИИ

О.П. Гущин

Удмуртский госуниверситет

Электронные обучающие средства применяются достаточно давно, задолго до широкого использования Интернета. В тот период, как средство распространения обучающих систем использовались дискеты, немного позже CD диски. Системы эти развивались и дожили до наших дней. Наиболее известные из них на сегодняшний день включают в себя 10 или 11 обновление: Trivantis Lectora Publisher, Macromedia Authorware, Outstart Trainersoft, Click2Learn ToolBook Instructor. С развитием вычислительных сетей ситуация изменилась – появились системы управления учебным содержимым (Blackboard, WebCT, e-College). В настоящее время по данным ведущих зарубежных аналитических агентств IDC и Gather Group рынок электронных средств обучения составляет более 2 триллионов долларов. Наблюдается устойчивый рост по отношению к 2006 году. Подобные тенденции наблюдаются с 2001 года. Системы электронного обучения используют как гиганты мировой индустрии, например (Airline Industry Computer Based Training Committee) и министерство обороны США (www.adl.com), многие государственные и коммерческие организации. Сегодня любой вуз в нашей стране в той или иной мере использует электронные средства обучения. В институте права социального управления и безопасности электронные средства обучения используются, начиная с 1987 года. За время использования данных средств отмечен ряд особенностей [1–4].

- Сложность овладения навыками работы с системой LMS, средствами autorware (авторские программные продукты) авторами, преподавателями, слушателями.
- Ограниченные возможности существующих систем для реализации авторского замысла в полной мере.
- Совместимость и переносимость электронных модулей в другие системы.
- Универсальность публикации и использования.

Рассмотрим вышеупомянутые проблемы и методы решения более подробно.

Сложность овладения навыками работы с системой LMS, средствами autorware (авторские программные продукты) авторами, преподавателями, слушателями. Обучение авторов, преподавателей позволяет создателям курсов, экспертам в предметной области, пользоваться ими без владения основами программирования. Данная проблема особенно актуальна для нетехнических специальностей вузов. Без специализированных курсов повышения квалификации проблема практически не решается. В нашем институте для решения этой задачи применяется комплексный подход. Используются обычные аудиторные занятия, чтобы научить пользоваться системой обучения и разработать необходимые курсы для авторов, преподавателей. Кроме того, организовано соответствующее подразделение сертифицированных инженеров службы технической поддержки и сертифицированных Microsoft преподавателей для чтения курсов повышения квалификации. Активно используются для подготовки авторов и преподавателей интерактивные компьютерные курсы, разработанные компанией «Мультимедиа технологии», которые можно использовать локально или удаленно, в том числе с использованием Internet в удобном для конечного пользователя месте и в удобное время.

Ограничения преподавательского замысла программных решений средств создания учебного контента существующих решений. Методы преподавания зависят от среды применения, специализации и аудитории слушателей, многих других факторов.

Для образовательных технологий, которым не требуются интерактивные возможности современных компьютерных технологий, подходит обычный гипертекст в формате HTML. Подобным образом, например организованы тексты нормативных актов для студентов юридических специальностей в информационно правовых поисковых системах. Для преподавателей знакомых с компьютерными обучающими технологиями вполне подходит Microsoft Power Point и Adobe Presenter. В таком виде представлены обучающие мультимедиа средства, используемые для обучения студентов иностранным языкам.

Для авторов не знакомых с компьютерными технологиями, но имеющими намерения создать курс по той или иной дисциплине, имеется специальная лаборатория, включающая программистов, дизайнеров, создающая авторские компьютерные учебники.

Последняя разработка данной лаборатории – курс по криминалистике, используемый для подготовки студентов юридических специальностей.

Применяемый в нашем случае подход заключается в том, чтобы не быть связанными рамками одной используемой системы и предоставить распределенную сетевую среду с возможностями добавления функциональных возможностей по мере необходимости. Необходимая функциональность может быть реализована различными платформами и сетевыми операционными системами. Наиболее эффективные возможности предоставляет группа серверов на базе операционной системы Microsoft Windows Server, к которым можно добавлять необходимые сервисы.

Структурная схема базовой серверной архитектуры представлена на рис. 1. Входящие соединения контролируются ISA сервером, который выполняет функции файрвола и маршрутизатора. Функции доставки мгновенных сообщений обеспечиваются Live Communication Server. Доставка e-mail и служба уведомлений при помощи Microsoft Exchange Server. Web порталы построены на базе IIS и службы Net Share Point. Данные хранятся на Microsoft SQL сервер. Данную связку можно выполнить как отказоустойчивый кластер. При этом уровень готовности, выраженный в SLA Service Level Agreement можно варьировать в широких пределах [5]. Аутентификация, хранение учетных записей пользователей управление всей информационной инфраструктурой пользователей и поддержка общей инфраструктуры обеспечивается за счет использования системы каталогов Active Directory.

Совместимость и переносимость электронных модулей в другие системы. Электронные учебники, разработанные по стандарту SCORM слабо совместимы с другими стандартами. И даже в рамках одного стандарта не гарантируется полная совместимость [2]. Ряд авторов отмечают сложную интеграцию в существующую информационную инфраструктуру вуза [1].

Проблема внедрения систем управления учебным содержимым осложняется тем, что в вузе уже имеется информационная инфраструктура. В частности это могут быть бухгалтерские программы и системы электронного документооборота. Для внедрения новых систем обучения приходится перестраивать информационную инфраструктуру, разрабатывать модули, устраняющие несоответствие между различными форматами и протоколами хранения и передачи данных. Например внедрить корпоративную обучающую систему подобную BlackBoard в учебном заведении достаточно сложно ввиду вышеупомянутых факторов. Вновь создаваемая система должна органично дополнять существующие решения. Для поэтапного внедрения подобных систем подходят уже существующие и используемые в вузе системы с аналогичными возможностями. В ча-

стности, для корпоративной межвузовской системы подходит иерархическая система Web узлов на базе Net Share Point корпорации Microsoft, которая в свою очередь позволяет создавать и использовать форумы, рассылки, опросы, события. WWW узлы могут содержать интерактивный контент на базе Dynamic HTML, JAVA , Visual Basic скрипты, флеш ролики, видео, презентации Power Point, ссылки позволяющие использовать уже созданные мультимедиа учебники доступные через систему Web. Для разработки учебного материала авторами, выдачи их, контроля работы с учебным материалом и его успешного освоения слушателями, используется переработанная и дополненная модулями собственной разработки система «Олимп» компании Термика. Общая структура информационной образовательной среды представлена на рис. 2.

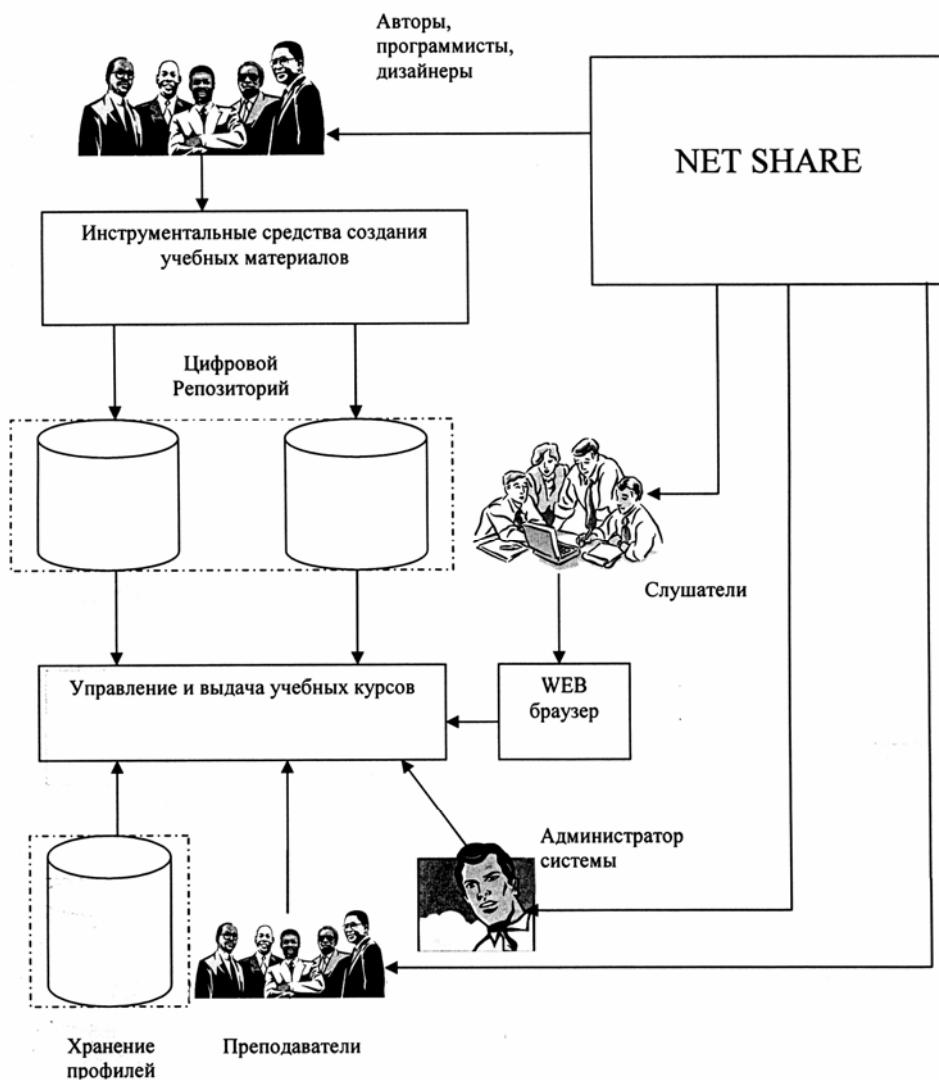


Рис. 1. Базовая серверная архитектура образовательной среды

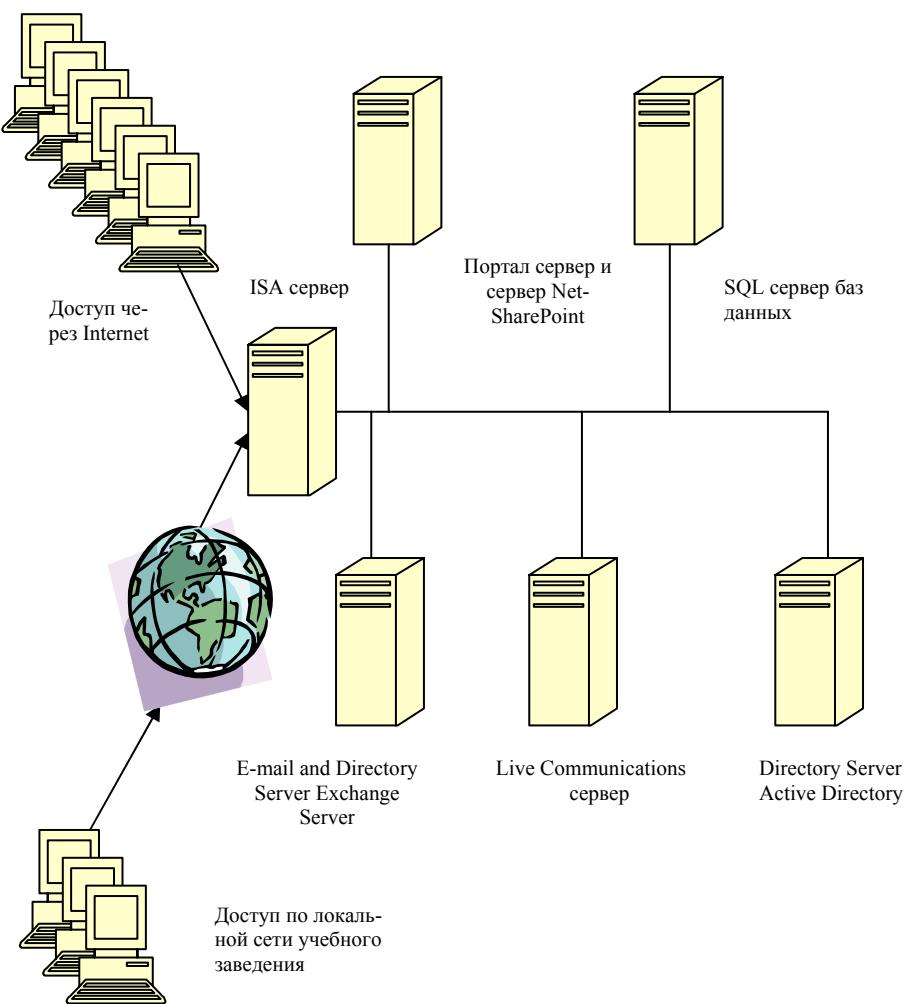


Рис. 2. Общая структура информационной образовательной среды

Авторы создают учебный материал в виде отдельных блоков, из которых получается электронный учебник, помещаемый в электронный репозиторий. Слушатели имеют возможность получать назначенный им учебный материал из цифрового репозитория, проводить с ним определенную работу, под контролем системы. Учитывается время, проведенное с учебным материалом, полнота освоения курса, путем выполнения промежуточных контрольных вопросов.

Если слушатель изучил материал, он может пройти контрольные тесты. Результаты выполнения работы отсылаются преподавателю с возможностью автоматизированной обработки ответов слушателей. Общее управление системой осуществляет администратор системы. Для обеспечения обратной связи преподаватель-слушатель и межсетевого общения между слушателями, вполне подходит комплекс программных средств предоставляемых компанией Microsoft. Net Share Point.

Данная система произведена корпорацией Microsoft, вследствие этого обеспечивается 100% совместимость с другими продуктами этой компании, минимальное обучение, для пользователей ранее работавших и знакомых с продуктами Microsoft.

Вывод. Предлагаемое решение помогает внедрить систему дистанционного образования в учебный процесс вуза и подготовить пользователей к активному использованию новых технологий в образовательном процессе. Такая информационная среда обладает гибкостью, минимальными затратами на обучение авторов и преподавателей для создания электронных обучающих средств; достаточно высокой степенью масштабируемости, надежности и готовности вследствие использования кластерных технологий.

Данная информационная среда может быть расширена любыми совместимыми с платформой Dot Net средствами и обеспечивает реализацию любых приемов и методик преподавания с использованием электронных обучающих средств. Это достигается в основном за счет продуктов компании Microsoft.

Список литературы

1. Виштак О.В., Ефремов Р.В. Новые информационные технологии при решении проблем учебного процесса в вузе / Сборник научных трудов «Образование и наука в 3-м тысячелетии». Алтайский ГУ, 2004. С 102–106.
2. Houghton J.W. «Crisis and Transition: The Economics of Scholarly Communication» Learned Publishing, 14 (July). P. 167–176, 2005.
3. Марченко Е.К. Электронная библиотека как системообразующий модуль системы дистанционного образования. Институт информатизации образования Минобрзования РФ // Открытое образование. 1998. № 2. С. 68 –72.
4. Гребнев А. Н. AtLeap – база для ИС научных коммуникаций в высшей школе. Материалы конференции «Свободное программное обеспечение в высшей школе». Университет г. Переславля им. А.К. Айламазяна, 2006. С. 106–112.
5. Гущин О.П. Использование кластерных технологий в обеспечении учебного процесса. Материалы конференции «XIII Всероссийская научно-методическая конференция». ЛИТМО, 2006. С. 154 –158
6. Создание Web-решений высокой доступности на основе Microsoft Windows 2003 Server Enterprise Server. MCSE. Издательство Microsoft Press, 2001. 540 с.
7. Разработка инфраструктуры сетевых служб Microsoft Windows 2003. MCSE. Издательство Microsoft Press, 2005. 542 с.
8. Microsoft Internet Security and Acceleration Server 2004. MCSE. Издательство Microsoft Press, 2005. 231 с.
9. Проектирование и реализация баз данных Microsoft SQL Server 2000. MCSE. Издательство Microsoft Press, 2001. 512 с.

**ПРОГРАММНОЕ ПРИЛОЖЕНИЕ «РАСПИСАНИЕ ЗАНЯТИЙ»
ДЛЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ
И СРЕДНИХ ПРОФЕССИОНАЛЬНЫХ УЧИЛИЩ**

В.М. Денисов, А.В. Шуляченко, А.В. Шилин

Нижегородский государственный архитектурно-строительный университет

Назначение и возможности

Приложение позволяет полностью автоматизировать труд по тарификации преподавателей и составлению расписания занятий на любой период от недели до целого учебного года для любого учебного заведения.

В настоящее время этой задачей занимается масса сотрудников, при этом считается, что существующие для этой цели программные продукты имеют недостатки:

- не терпят дефицита помещений;
- многие ситуации не учитывают;
- трудно настраиваютя на конкретные условия данного УЗ;
- требуют аппаратного и программного обеспечения, не доступного в обычной школе или ПТУ;
- при переходе на новый продукт приходится вручную вводить массивы уже «написанных» данных в базу данных;
- расписание составляют с перерывами (чаще всего, для преподавателей) в занятиях.

Эти недостатки учтены в данном приложении.

Поскольку большинство школ не могут позволить себе сложного оборудования, написано оно на VBA в среде приложения MS Excel, как наиболее доступного и простого в эксплуатации.

Имеет минимальное количество органов управления – всего 4 кнопки.

Для автоматического ввода существующей информации в базу данных используются специальная универсальная утилита – списки групп, количество часов и кто их проводит; «привязка» преподавателей к тем или иным урокам, – всё это – автоматически.

Еженедельную нагрузку вводить не требуется – достаточно указать при первом заполнении годовую (семестровую). В дальнейшем всё можно подкорректировать при необходимости.

Есть возможность вводить поэтапно: вначале теоретические занятия, и/или практику, экзамены и т.д.

Есть возможность составлять расписание не на две недели, а на каждую неделю – новое, хоть в течение всего года, сохраняя требуемую равномерность нагрузки преподавателей и групп. Это не требует больших трудозатрат, поскольку даже в полуавтоматическом режиме один работник средней квалификации легко справится с этой задачей.

Окончательная электронная таблица имеет удобную структуру; каждый урок имеет атрибуты: название, группа, комнаты для всех подгрупп (!) и все преподаватели (!) в этих подгруппах. Достаточно переслать этот файл по сети, чтобы получить полную информацию не только о расписании, но и тарификации преподавателей.

Очень легко вносить изменения в расписание: стереть те места, где надо изменить и нажать кнопку заполнения новыми данными. Это легко использовать при командировках или болезни отдельных преподавателей.

Приложение может использоваться для крупного вуза с:

числом групп – до 125;

численностью преподавателей – до 600;

численностью одноименных учебных предметов – до 600.

Требования к оборудованию:

Windows 2000/Me/XP с MS Office,

RAM 256 MB рекомендуется для комфортной работы,

CPU Pentium IV рекомендуется для комфортной работы.

Эксплуатация осуществляется обычным пользователем, от которого требуется умение выполнять элементарные операции на ПК.

Оптимизация расписания позволяет:

– переход с двухсменной работы на односменную,

– учет методических дней преподавателей,

– оптимизацию их загрузки в течение недели,

– учет трудности уроков (от теории в начале и к практике в конце дня).

Демонстрационный пример:

- время учёбы групп, преподавателей и другие настроочные предварительные данные могут быть указаны при помощи специальной встроенной утилиты «Настройка» (даже такие, например: некоторые уроки могут проводиться при разделе группы на три части, как в Нижегородском педагогическом колледже – сольфеджио, музыка и пение – в группах по 10 человек);
- некоторые преподаватели имеют свободные для методической работы дни или часы;
- преподаватели имеют свободные для организационной работы часы;
- по выбору можно построить расписание вначале для теории, затем последовательно добавить практические работы, стажировки, консультации, при необходимости курсовые работы и экзамены.

При этом к уже построеному расписанию добавляются новые уроки с учетом принятых требований.

Просмотр и распечатка страниц

Для удобства работы окна настроены оптимальным образом. На самом деле страницы при печати будут иметь вид стандартных документов. Их можно просмотреть и распечатать, выполнив следующую последовательность действий:

☞ ФАЙЛ ☞ Предварительный просмотр

☞ ФАЙЛ ☞ Печать

Файл является оперативным документом в электронном виде для «безбумажного» оборота. Возможна пересылка этого документа в сети.

Тарификация

На листе ТАРИФИКАЦИЯ измените ФИО преподавателей напротив каждого № преподавателя, начиная с первой строки.

Вы сможете изменить только нужные ячейки. Все вычисления сделает система. Листы учебных групп служат для классификации. При желании все группы можно раз-

местить на одном листе. Вверху листа измените название группы. В строках учебных предметов укажите число учебных часов за год в столбце для каждой группы. Укажите краткое название предмета и № преподавателя на листе ТАРИФИКАЦИЯ, например: «Литература53», где под №53 указан учитель, ведущий уроки. При первом заполнении введите только краткие названия предметов. Они будут фигурировать в расписании, их можно редактировать впоследствии на листе ТАРИФИКАЦИЯ.

Вернитесь на лист ТАРИФИКАЦИЯ и посмотрите, как изменилась нагрузка преподавателей. Приступайте к построению расписания

Составление расписания

На листе РАСПИСАНИЕ при необходимости добавьте дни слева в рамочках. По умолчанию расписание составляется на две недели.

Нажмите кнопку НАСТРОЙКА. Выполните выводимые на экран инструкции и ПРИ НЕОБХОДИМОСТИ заполните формы ГРУППЫ, ПРЕПОДАВАТЕЛИ и УРОКИ.

Нажмите кнопку РАСПИСАНИЕ. Возможно при этом составление расписания отдельно по теории, практике, семинарам и т.д.

ПРОГРАММНАЯ СИСТЕМА КОНСТРУИРОВАНИЯ И ИССЛЕДОВАНИЯ ХАРАКТЕРИСТИЧЕСКИХ МЕТОДОВ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ

Д.Н. Добряев, В.А. Гришагин

Нижегородский государственный университет им. Н.И. Лобачевского

Введение

Среди проблем принятия оптимальных решений существует класс задач многомерной многоэкстремальной оптимизации. Методам решения подобных задач свойственна высокая трудоёмкость, которая значительно увеличивается с ростом размерности задачи. Наряду с этим, практическое значение подобных задач очень велико. Поэтому вопрос построения программных комплексов, которые позволяют эффективно решать задачи оптимизации, достаточно актуален.

Данная работа посвящена описанию программной системы, которая позволяет решать многомерные многоэкстремальные задачи оптимизации на основе многошаговой схемы редукции размерности с использованием конструктора решающих правил характеристических методов.

Постановка задачи

Рассмотрим многомерную задачу оптимизации

$$\begin{aligned} f(x) &\rightarrow \min, x \in Q \subseteq R^N, \\ Q &= \{x \in D : g_j(x) \leq 0, 1 \leq j \leq m\}, \\ D &= \{x \in R^N : x_i \in [a_i, b_i], 1 \leq i \leq N\}, \end{aligned}$$

т.е. требуется найти минимальное значение функции в области, заданной как координатными, так и функциональными ограничениями.

Многошаговая схема редукции размерности

Одним из подходов к решению таких задач является использование многошаговой схемы редукции размерности. Данная схема позволяет свести исходную многомерную задачу к системе рекурсивно вложенных одномерных подзадач:

$$\min_{x \in Q} f(x) = \min_{x_1 \in \Pi_1} \min_{x_2 \in \Pi_2(u_1)} \dots \min_{x_N \in \Pi_N(u_{N-1})} f(x),$$

где $\Pi_{i+1}(u_i)$ – проекции сечений $S_{i+1}(u_i)$, которые определяются следующим образом:

$$\begin{aligned} u_i &= (x_1, \dots, x_i), v_i = (x_{i+1}, \dots, x_N), \\ S_1 &= Q, S_{i+1}(u_i) = \{(u_i, v_i) \in Q\}, 1 \leq i \leq N-1, \\ \Pi_{i+1}(u_i) &= \{x_{i+1} \in R^1 : \exists (x_{i+1}, v_{i+1}) \in S_{i+1}(u_i)\}. \end{aligned}$$

Для решения одномерных подзадач можно использовать характеристические методы поиска.

Характеристические методы поиска

Пусть поставлена одномерная задача оптимизации:

$$f(x) \rightarrow \min, x \in [a, b].$$

Алгоритм называется характеристическим, если, начиная с некоторого шага, выбор следующей точки испытания x^{k+1} осуществляется согласно нижеприведённой схеме.

1. Задать набор

$$\Lambda_k = \{x_0, \dots, x_\tau\}$$

точек области $Q = [a, b]$, полагая, что $a \in \Lambda_k, b \in \Lambda_k$, все координаты предшествующих испытаний расположены по возрастанию:

$$a = x_0 < x_1 < \dots < x_{\tau-1} < x_\tau = b.$$

2. Каждому интервалу (x_{i-1}, x_i) , $1 \leq i \leq \tau$, поставить в соответствие число $R(i)$, называемое характеристикой интервала.

3. Определить интервал (x_{i-1}, x_i) , которому соответствует максимальная характеристика:

$$R(t) = \max \{R(i) : 1 \leq i \leq \tau\}.$$

4. Провести очередное испытание в точке

$$x^{k+1} = D(t) \in (x_{t-1}, x_t).$$

Конструирование решающих правил характеристических алгоритмов

Структура характеристических алгоритмов представлена выше пунктами 1 – 4. Для проведения очередного испытания осуществляется разбиение отрезка $[a, b]$ на интервалы. Затем выбирается интервал с наибольшей характеристикой $R(i)$ и точка очередного испытания размещается внутри этого интервала в соответствие с правилом $D(t)$.

Таким образом, чтобы сконструировать решающее правило характеристического алгоритма, необходимо задать:

- 1) правило вычисления характеристики интервала $R(i)$;
- 2) правило выбора очередной точки испытания $D(t)$.

Решая многомерную задачу оптимизации с использованием многошаговой схемы редукции размерности, можно по каждой переменной сконструировать свой характеристический алгоритм поиска, задавая соответственно $R(i)$ и $D(t)$. Данный принцип лёг в основу разработки программной системы.

Описание программной системы

Программный комплекс, описываемый в данной работе, позволяет осуществить следующую завершённую последовательность действий, необходимую для решения задач многомерной оптимизации.

1. Постановка задачи и выбор методов поиска.

Целевая функция, координатные и функциональные ограничения можно задавать аналитически. Минимизируемая функция может быть выбрана либо из предложенного набора, либо введена пользователем. Также предусмотрена возможность задания оптимизируемых функций в виде подключаемых DLL библиотек. Далее по каждой переменной необходимо сконструировать характеристический метод. Алгоритм поиска может быть выбран как из списка, предлагаемого программой, так и определён самостоятельно путём задания правил вычисления характеристики и точки следующего испытания аналитически. Способность конструирования алгоритмов даёт широкие возможности как для эффективного решения задачи, так и для исследований.

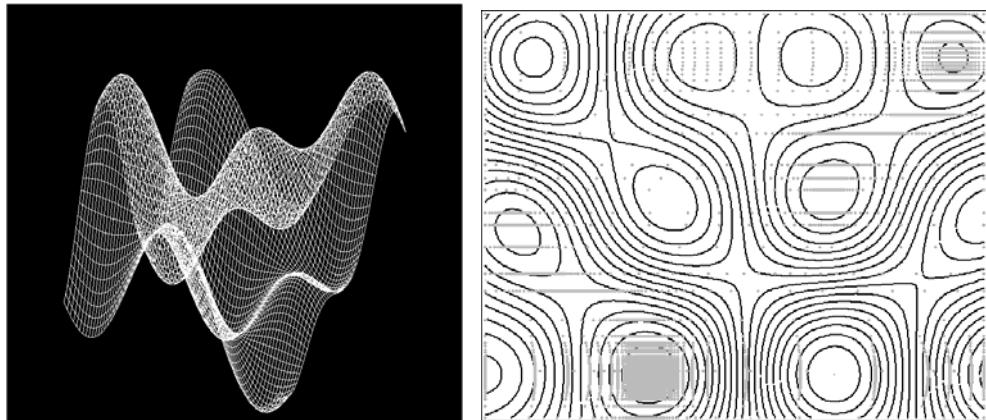
2. Решение поставленной задачи.

Процесс поиска оптимального значения осуществляется на основе многошаговой схемы редукции размерности с использованием сконструированных алгоритмов для решения одномерных подзадач. Для учёта ограничений используется метод штрафных функций.

3. Визуализация исходной задачи и процесса поиска.

Средства программной среды позволяют строить целевую функцию (сечения функции, если количество переменных больше двух), линии уровня и отображать точки испытаний. Визуализация делает возможным наглядно представить минимизируемую функцию и процесс поиска, а также качественно оценить правильность работы вычислительной схемы.

Пример отображения целевой функции и линий уровня с точками испытаний представлен на рисунке.



Заключение

Итогом проделанной работы является законченная программная система, обеспечивающая решение многомерных многоэкстремальных задач оптимизации и возможность исследования характеристических методов поиска. Разработанные программные средства позволили провести вычислительные эксперименты, подтвердившие работоспособность системы.

Список литературы

1. Стронгин Р.Г., Гергель В.П., Городецкий С.Ю., Гришагин В.А., Маркина М.В. Современные методы принятия оптимальных решений. 2002.
2. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. 1978.
3. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-Convex Constraints Sequential and Parallel Algorithms – Dordrecht. Kluwer Academic Publishers, the Netherlands, 2000. 728 p.

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ИНТЕГРИРОВАННОЙ СРЕДОЙ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ ННГУ

В.В. Домрачев, А.В. Сенин

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Для эффективного управления вычислительными ресурсами кластеров требуется специальное программное обеспечение (Система Управления Кластером), служащее для организации распределённых вычислений, поддержания равномерной загрузки узлов системы и предоставления удобного интерфейса взаимодействия с пользователем, обеспечивающего представление системы компьютеров как единого вычислительного ресурса.

Для проведения научно-исследовательских экспериментов в ННГУ имеется высокопроизводительный вычислительный кластер с пиковой производительностью 319.2 ГФlop/сек (48-ая позиция в 4-ой редакции списка Top50 самых мощных компьютеров СНГ от 04.04.06). Основной операционной системой, установленной на узлах кластера, является Windows: на рабочих станциях (Pentium D с EM64T 2.8 GHz 2Gb RAM) установлена Windows XP Professional, на серверах (2xXeon EM64T 3.6 GHz 2-8Gb RAM) установлена Windows 2003 64-bit. В качестве сетевого интерфейса используется Gigabit Ethernet. Кроме того, в университете есть несколько компьютерных лабораторий, использующихся в учебном и научно-исследовательском процессах. В целях повышения эффективности использования этих компьютерных ресурсов, обеспечения отказоустойчивости, улучшения контроля за использованием компьютерного времени и упрощения администрирования необходимо интегрировать вычислительный кластер и компьютерные лаборатории в инфраструктуру, управляемую единой системой. Однако это осложнено тем, что в части лабораторий уже используются некоторые системы управления кластерами (Microsoft Compute Cluster Server 2003, PBS). Кроме того, часть лабораторий работают под управлением операционной системы Windows, а часть – Linux. Указанные факторы делают чрезвычайно сложным или даже невозможным внедрение одной из существующих систем управления для объединения всех компьютерных ресурсов ННГУ, что обуславливает необходимость создания собственной системы.

В рамках учебно-исследовательской лаборатории «Информационных технологий» существует проект «Метакластер», целью которого является разработка системы управления кластерами, важнейшими отличительными особенностями которой являются: одновременное управление несколькими кластерами, поддержка различных операционных систем и интеграция с другими системами управления. Она должна предоставить широкие возможности для наиболее рационального использования вычислительных ресурсов.

Описание проекта «Метакластер»

Основными требованиями, выдвигаемыми к создаваемой системе управления кластером, являются следующие.

1. Возможность одновременного управления несколькими кластерами на базе различных операционных систем: Windows, Linux, FreeBSD.
2. Интеграция с системами управления третьих разработчиков (в случае использования таковых в компьютерных лабораториях и невозможности полного перехода на нашу систему): Microsoft Compute Cluster Server 2003, PBS и др.
3. Простой и удобный способ доступа, не требующий установки на рабочих станциях пользователей какого-либо специального программного обеспечения, позволяющий получить доступ к системе из любой точки (веб-интерфейс).
4. Разработка программ с командным и графическим интерфейсами для предоставления наиболее удобного и быстрого доступа к системе.
5. Административный интерфейс, управляющий ходом работы.
6. Программный интерфейс (веб-сервис, СОМ-интерфейс и др.).
7. Собственная система авторизации пользователей не связанная напрямую с системой авторизации операционных систем.
8. Реализация следующего набора операций над задачами.
 - 1.1. Загрузка задачи на сервер.
 - 1.2. Добавление задачи в очередь задач (под очередью задач мы понимаем множество всех задач, ожидающих выполнения на кластерах).
 - 1.3. Получение текущего статуса очереди задач и любой задачи.

1.4. Получение результатов вычислений (перехваченный поток вывода и файлы, созданные программой).

9. Сохранение задач пользователя после их выполнения и возможность их повторного запуска.

10. Автоматическое распределение задач по кластерам и далее – по узлам выбранного кластера.

11. Автоматическое сохранение результатов работы.

12. Устойчивость к отказам отдельных кластеров и отдельных узлов каждого кластера.

13. Ведение статистики использования компьютерных ресурсов, автоматическая генерация отчетов.

Архитектура системы «Метакластер 2.0», разработка которой идет в настоящее время, включает три основных компонента (см. рис. 1):

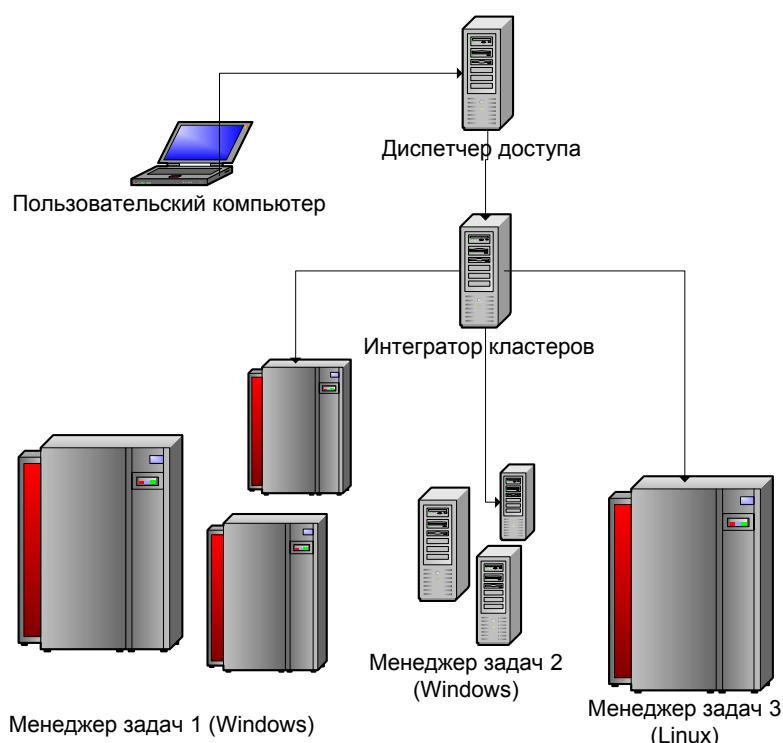


Рис. 1. Общая архитектура системы «Метакластер 2.0»

Диспетчер доступа

Диспетчер доступа предоставляет интерфейс для работы пользователя с системой с удалённого компьютера через сеть Интернет (в частности – веб-интерфейс). Компонент проводит регистрацию пользователей в системе, позволяет загружать задания, задавать параметры, отправлять задания на интегратор кластеров, отображать состояния запущенных задач, возвращать результаты работы пользователю.

Интегратор кластеров

Компонент, являющийся прослойкой между диспетчером доступа и менеджерами задач, над которыми он осуществляет централизованное управление. Именно за счет введения интегратора мы можем обеспечить поддержку многокластерности. Кроме того, получаем возможность автоматического распределения запуска задач по различным кластерам и обеспечения бесперебойной работы системы в случае временной недоступности одного из кластеров. Пользователь имеет возможность указать, на каком именно кластере он хочет запустить свою задачу. В случае отсутствия пользовательских предпочтений система принимает самостоятельное решение на основе имеющейся информации о загруженности кластеров, предыдущем опыте запуска и правах конкретного пользователя. Кроме того, интегратор кластеров по запросу диспетчера доступа предоставляет необходимую информацию (ход выполнения заданий, информация о загруженности кластера, результаты вычислений).

Менеджер задач

Компонент, фактически, являющийся локальной системой управления кластером. Менеджер задач осуществляет непосредственный запуск задач на кластере, мониторинг вычислительных узлов, сбор и сохранение результатов работы вычислительных задач (стандартный поток вывода и сгенерированные файлы). В том случае, если на кластере уже установлена система управления стороннего производителя, менеджер задач может выступать промежуточным звеном, транслирующим команды интегратора в команды установленной системы. Каждый менеджер задач разрабатывается с учетом особенностей кластера, на котором он будет работать (операционная система, установленное программное обеспечение, возможно, аппаратные характеристики). Благодаря реализации специфических менеджеров задач мы получаем возможность централизованно управлять кластерами с различными операционными системами и различными аппаратными характеристиками.

В системе «Метакластер 2.0» используется система мониторинга «Grid Performance Monitoring», созданная в рамках проекта «Методы и инструменты грид» (<http://www.software.unn.ac.ru/grid/>) в учебно-исследовательской лаборатории «Информационные технологии».

Текущее состояние разработки

В настоящее время в системе реализована следующая функциональность.

- 1) Обеспечение доступа к системе через web-интерфейс.
- 2) Поддержка базового набора операций: загрузка задачи на сервер, добавление задачи в очередь планирования запусков, остановка задачи, просмотр состояния задачи.
- 3) Поддержка произвольного числа кластеров на базе ОС Windows.
- 4) Возможность интеграции с Microsoft Compute Cluster Server 2003.
- 5) Возможность выбора в web-интерфейсе кластера для запуска из предложенного списка.
- 6) Возможность перехвата стандартного потока вывода запущенных задач с записью в указанный файл.
- 7) Сохранение списка задач между сессиями в базу данных.

Разрабатываемая система управления кластерами используется в Центре компьютерного моделирования при разработке и эксплуатации учебных и исследовательских программных комплексов. Среди них программная система для изучения и исследова-

ния параллельных методов решения сложных вычислительных задач («ПараЛаб»), система параллельной многоэкстремальной оптимизации «Абсолют Эксперт», широкий круг исследовательских проектов лаборатории «Информационные технологии».

Планы дальнейшего развития

Можно выделить следующие основные направления работы.

1. Расширение функциональности системы для операционной системы Windows, разработка версий для Linux, FreeBSD.
 2. Интеграция с другими системами управления кластерами (в ближайших планах – PBS).
 3. Доработка web-интерфейса с целью упрощения взаимодействия пользователя с системой.
 4. Разработка программ-клиентов с командным и графическим интерфейсами для предоставления наиболее удобного и быстрого доступа к системе.
 5. Административный интерфейс, управляющий ходом работы.
 6. Расширение сохраняемой статистики использования компьютерных ресурсов, автоматическая генерация отчетов.
- Большое внимание планируется уделить вопросам оптимального распределения вычислительной нагрузки по узлам кластера и обеспечению безопасной и отказоустойчивой работы кластеров, в том числе:
- 1) разработка безопасных протоколов взаимодействия с пользователем и между компонентами системы.
 - 2) вопросы шифрования при обмене информацией по сети.
 - 3) анализ системы с точки зрения предотвращения злонамеренных действий пользователя.
 - 4) вопросы поведения системы в случае отказа отдельных узлов, самовосстановление системы в случае аварийного завершения работы.

Список литературы

1. Baker M. et al. Cluster Computing White Paper. Final Release, Version 2.0.
2. Ed. By Thomas Sterling. Beowulf Cluster Computing with Windows. The MIT Press, Cambridge, Massachusetts, 2002 y.
3. Microsoft Windows Compute Cluster Server 2003 Beta 2 Reviewers Guide (<http://www.microsoft.com/windowsserver2003/ccs/reviewersguide.mspx>).
4. Боголепов Д., Алексин А. Архитектура системы мониторинга GPM. (<http://www.software.unn.ac.ru/grid/>).
5. Гергель В.П. Теория и практика параллельных вычислений. (<http://www.software.unn.ac.ru/ccam/kurs1.htm>).
6. Гергель В.П., Стронгин Р.Г. Высокопроизводительный вычислительный кластер Нижегородского университета. Материалы конференции Relam. Н. Новгород, 2002.

РАЗРАБОТКА МНОГОУРОВНЕВЫХ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ ИНТЕГРИРОВАННОГО СЕМАНТИЧЕСКОГО ПРЕДСТАВЛЕНИЯ

**Е.Н. Дубровина, Е.А. Казакова, А.В. Майоров,
И.А. Катыков, С.В. Шибанов**

Пензенский госуниверситет

Архитектура распределенных информационных систем (РИС) строится с учетом структурной или процессной организации предприятия, для которого создается система. Большое число предприятий, управленических структур и организаций обладает иерархической многоуровневой архитектурой. Как следствие, архитектура распределенной информационной системы для таких предприятий также будет являться и иерархической, и многоуровневой. Отдельные части информационной системы, которые развертываются в конкретном подразделении на конкретном уровне предприятия или организации, могут, в свою очередь, обладать достаточно сложной архитектурой. Тем не менее можно выделить некоторые общие элементы такой архитектуры:

- информационное хранилище оперативной и аналитической информации;
- классификаторы и справочники, архивные сведения;
- приложения, обеспечивающие управление информационным хранилищем и взаимодействие с клиентскими приложениями;
- клиентские приложения для решения конкретных задач оперативной и аналитической обработки данных, администрирования и настройки системы;
- интерфейсы и протоколы для взаимодействия с информационными системами других уровней.

Кроме того, части информационной системы, развертываемые в различных подразделениях и на различных уровнях, могут отличаться друг от друга аппаратно-программными платформами и функциональными возможностями. Это, несомненно, еще более усложняет информационную систему, тем самым, усложняется процесс проектирования, разработки, развертывания и сопровождения системы.

Для разработки современных систем подобного класса – многоуровневых распределенных информационных систем (МРИС) чаще всего используются те же методы, что и для любого программного обеспечения (ПО) информационных систем. Модель жизненного цикла ПО отражает этапы анализа предметной области, проектирования, разработки, внедрения и сопровождения программного продукта. При этом в любой информационной системе в различных формах накапливается и сохраняется семантика, характеризующая работу этой системы. В отдельных случаях она отражается явно, в других присутствует в виде алгоритмов, комментариев, структур данных. При этом вид, форма отображения этой семантики определяется средствами разработки, предпочтениями разработчика, сложившимися в команде разработчиков соглашениями по документированию системы. К таким формам хранения семантики можно отнести структуры баз данных, встроенные в них ограничения данных и комментарии, интерфейс и логику работы приложения, экраны помощи, документацию к системе и др.

Таким образом, семантика предметной области представляется в информационной системе в виде следующей совокупности:

- словарь данных, как правило, в нем хранится информация, необходимая для функционирования базы данных, а не для принятия решения об использовании этой информации в информационной системе;
- макроязыковое описание и унифицированное графическое отображение различных реализаций базы данных на физическом уровне в единую логическую модель с помощью различного рода Case-средств;
- приложения, например реализация интерфейсов приложений, экранов помощи, инструкций по работе с приложением;
- унифицированные объекты, с которыми работает информационная система, типовые алгоритмы, реализующие семантику предметной области, различные макросредства, созданные разработчиками ИС для реализации ее функциональных возможностей;
- документация, выполненная с учетом имеющихся нормативных документов: государственных стандартов, ведомственных и корпоративных нормативных документов. Документация фактически дублирует семантику, уже содержащуюся в компонентах АИС: коде приложения, структуре базы данных, типовых элементах данных. В связи с этим в любой ИС существует проблема синхронизации документации и содержания системы;
- классификаторы и справочники информационной системы, а также алгоритмы и ограничения, налагаемые на данные, которые также хранятся в виде данных (в частности, информация о разграничении доступа к данным средствами приложения, настраиваемые запросы и состояния модифицируемых алгоритмов и пр.).

Таким образом, семантика информационной системы рассредоточена в различных ее компонентах. Представляется необходимым, чтобы все перечисленные элементы семантического описания были увязаны для обеспечения эффективного проектирования и нормального функционирования системы.

Семантика предметной области может быть представлена в виде нагруженного псевдографа или семантической сети. В результате продукции семантической сети и набора компонент может быть получен набор модулей, обладающий заданной функциональностью (рис. 1). Важно отметить, что в модули информационной системы, таким образом, внедряется семантика предметной области.

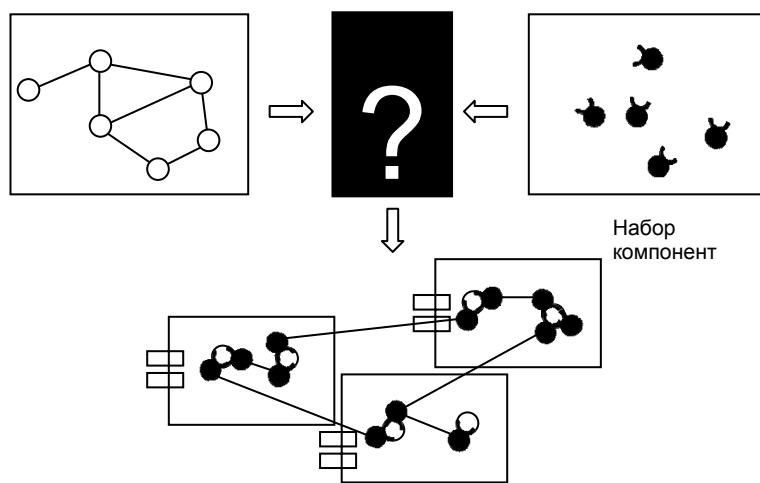


Рис. 1. Построение информационных систем на основе семантики предметной области и набора компонент



Рис. 2. Модель процесса разработки ИС на основе интегрированного представления семантики предметной области

Механизм совмещения семантического описания системы и набора интерфейсов позволяет предложить инновационную модель процесса разработки многоуровневых распределенных информационных систем, значительно упрощающую разработку, внедрение и сопровождение (рис. 2). В этой модели задача проектирования МРИС сводится к описанию семантической модели МРИС; задача реализации – к сборке проекта ядром системы анализа, к кодированию недостающих компонент и формированию семантического описания для них; задача сопровождения и модификации – к возможному внесению изменений в семантическое представление системы и синхронном обновлении псевдографа, хранящего семантику и изменяемых компонент.

Можно выделить основные направления исследований в этой области:

- подробное обоснование применения интегрированного представления семантики предметной области (ИПСПО) для построения информационных систем, в том числе распределенных и многоуровневых,
- исследование и разработка моделей ИПСПО,
- исследование и разработка методов интерпретации ИПСПО на этапах жизненного цикла МРИС,
- разработка программных средств управления ИПСПО,
- разработка программных средств автоматизации проектирования, реализации, документирования, развертывания и сопровождения МРИС на основе ИПСПО.

Достигнутые к настоящему моменту результаты позволяют говорить о том, что задача интеграции семантики предметной области и имеющихся программно-аппаратных средств, определенных набором ограничений и интерфейсов, является актуальной. Особое значение решение поставленной задачи приобретает для многоуровневых, распределенных, многокомпонентных и многоплатформенных информационных систем.

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТИ СОЗДАНИЯ САМООБУЧАЕМОЙ НЕЙРОННОЙ СЕТИ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПРОГНОЗИРОВАНИЯ ФИНАНСОВЫХ ВРЕМЕННЫХ РЯДОВ

A.B. Егошин

Марийский государственный технический университет

Постановка задачи

Разработка архитектуры нейронной сети, предназначеннай для прогнозирования финансовых временных рядов с известной вероятностью точности прогноза.

Искусственные нейронные сети (НС) получили широкое применение в задаче прогнозирования. В основном используются следующие архитектуры НС: многослойный персепtron, сети на основе радиальных базисных функций, рекуррентные сети, машина Больцмана, сеть Хопфилда (в случае прогнозирования как задача распознавания пространственно-временных образов). Применяются их варианты с элементами памяти: с памятью на линиях задержки, с ассоциативной памятью, гамма-памятью [1].

Однако проблема заключается в том, что в настоящее время не известна технология получения стабильного прогноза. Практически приемлемо точный прогноз можно построить, лишь опытным путем подбирая достаточно большое количество параметров: объем обучающей выборки, количество нейронов, архитектуру нейросети и ее специфические настройки (например, глубина памяти, коэффициенты скорости обучения и пр.). Экстенсивный путь создания механизма прогнозирования состоит в генерации так называемого комитета экспертов – наборов различных нейронных сетей, которые обучаются, и по результатам обучения отбираются в комиссию, которая и выдает прогноз. Соответственно, чем больше будет количество экспертов, тем выше вероятность того, что средний результат будет наиболее приближен к реальному. Хотя этот подход и дает возможность прогнозирования, он все же требует проведения огромного количества вычислений. Главный недостаток этого способа заключается в принципиальной ограниченности каждого эксперта (т.е. нейросети) внутри комитета, что в итоге приводит к неспособности получения информации о достоверности прогноза. Дело в том, что существующий подход нейросетевого прогнозирования предполагает аппроксимацию исторических ценовых данных финансового инструмента (акции, валюты и пр.) на их будущие значения. Обучаясь по методу с учителем, НС определяет функциональную зависимость, присутствующую во входных данных. При этом исторические данные рассматриваются как временная последовательность значений, однако не учитываются значения времени между поступающими данными. Т.к. зависимость во входных данных динамическая, то в каждом конкретном случае нужно опытным путем подбирать объем обучающей выборки, параметры сети и метода обучения.

Предлагается способ, отличающийся тем, что нейронная сеть, используемая для прогнозирования финансовых временных рядов, обучается методом без учителя. В настоящее время метод обучения без учителя используется в решении задачи нейропрогнозирования, но в качестве инструмента для выделения главных компонент во входных данных. Фактически это лишь метод предобработки данных, которые подаются уже на вход НС, которая обучается методом с учителем. В предлагаемом же способе не строится аппроксимация, а ведется поиск реально существующей зависимости в финансовых данных. Такой способ должен обладать такими преимуществами, как возможность получения прогноза с известной вероятностью его достоверности. Не исключено использование метода обучения с учителем, но для решения вспомогательных задач.

Разработка такого способа предполагает комплексный подход, а не только разработку архитектуры НС. Можно выделить следующие основные задачи:

1. Определение задачи прогноза.
2. Представление исходных данных для нейросети.
3. Архитектура нейросети, используемая для построения прогноза.

Определение задачи прогноза

Конечной целью прогнозирования финансовых временных рядов является получение прибыли, а не сам прогноз как таковой. Понимание этого факта приводит к тому, что прогнозировать можно следующие величины: достижение абсолютной цены, относительное изменение цены от последнего известного значения (рост/падение в процентах, изменение в число раз), достижение ситуации, позволяющей утверждать с высокой вероятностью о характере изменения цены в будущем (например, рост/падение не менее 1% в течение определенного периода), так называемая характеризующая ситуация.

Наиболее предпочтительным является прогнозирование относительного изменения цены, т.к. фактически именно эта информация и является основой для совершения при-

быльной сделки. Также она обладает преимуществами при использовании нейросетевого подхода, т.к. диапазон возможных значений гораздо уже и соответственно их легче нормализовать. Прогнозирование достижения характеризующей ситуации является более трудной задачей, требующей отдельного исследования.

Представление исходных данных

Представление исходных данных, их выбор, объем, обработка, а также особенности играют ключевую роль в задаче нейросетевого прогнозирования. Финансовые временные ряды обладают существенными характеристиками, которые не позволяют их рассматривать как просто последовательность чисел.

Прогнозирование временных рядов основывается на истории, поэтому, чем больше будет использоваться различной исторической информации, тем более точным должен быть прогноз. Большинство способов подачи данных при нейропрогнозировании используют только значения цены. Реже используются значения объемов. Практически никогда не используется время между получаемыми входными данными. Для прогнозирования примем априори, что нужно учитывать всю финансовую информацию: цены, объем сделок, время между сделками. Но при этом необходимо определиться, как подавать эти данные.

Разделение данных при подаче их на вход НС позволит в некотором роде внести априорную информацию о прогнозируемой величине, а также разделять принципиально различные типы данных, например цену и объем. Фактически на входе имеется 3 типа данных: время, цена и объем. Цена – основная входная величина. Именно для нее строится прогноз. Объем – характеризует «силу» цены. Отличие объема от цены состоит в том, что, рассматривая историю на более крупных временных периодах, значения объемов суммируются, тогда как значения цены нет. Поэтому этот тип информации должен подаваться отдельно от цены.

Существенной характеристикой финансовых временных рядов является их фрактальность. Фрактал – геометрическая форма, которая может быть разделена на части, каждая из которых – уменьшенная версия целого. В финансах это означает, что движения акции или валюты внешне похожи, независимо от масштаба времени и цены. Наблюдатель не может сказать по внешнему виду графика, относятся ли данные к недельным, дневным или же часовым изменениям. Это качество определяет графики цен как фрактальные [2]. Существующие способы подачи данных в нейропрогнозировании практически не используют эту интересную особенность.

Время между каждыми входными значениями стоит рассмотреть отдельно.

Время как характеристика финансового временного ряда

Рассмотрим, что значит время в контексте анализа ценовой истории финансового инструмента, например акции.

Торговля на бирже идет всю рабочую неделю. Торговый день идет без перерыва (как правило, 8 часов). Однако внутри торгового дня могут существовать периоды, в которые сделки не совершались. Большинство программ нейросетевого прогнозирования, а также весь технический анализ, работают с историческими данными как с последовательностью отсчетов. Это было бы справедливым, если бы значения цены поступали с одинаковым времененным интервалом. Однако это не так. Когда на вход нейросети подается последовательность значений, предполагается, что они все равнозначны. Но отличия между двумя последовательными значениями в середине торгового дня (с разницей в 1 час) и между последним значением в пятницу и первым значением в поне-

дельник все же, очевидно, есть. Тем более, если еще были праздничные дни, тогда между двумя последовательными отсчетами цены может быть интервал в несколько дней, и он совсем не равнозначен интервалу в один час. Также можно сказать, что даже с позиции человеческой психологии пятничная торговля отличается от торговли в понедельник. Торговля в середине дня отличается от торговли перед закрытием биржи. Но вся эта информация не учитывается, если рассматривать историю лишь как последовательность отсчетов, без привязки к значению времени, в которые они произошли.

Интересует даже не количество времени между двумя сделками, а относительная величина, зависящая от периодов между другими сделками. Если рассматривать ценовую историю с периодом в 1 час, то ряд значений интервалов будет приблизительно следующий (предполагая, что в каждый период совершалась хотя бы одна сделка): 1, 1, 1, 1, 1, 1, 16 (вечер–ночь–утро), 1, 1, 1, 1, 1, 1, 64 (вечер–ночь–утро+2 выходных), ... и т.д. Если предполагать, что периодов без сделок не существует (справедливо для высоколиквидных акций), то можно утверждать, что в году будет конечное число значений периодов между сделками, в случае, если не будет форс-мажорных ситуаций (рассматриваем для периода 1 ч): внутри дня, внутри рабочей недели (выходные), между установленными праздниками. Т. к. все эти периоды будут постоянны, их просто можно обозначать кодом, например: 1 ч – 1, 16 ч – 2, 64 ч – 3 и т.д. Это позволит, с одной стороны, упростить входные данные, а с другой, – не потерять информативности.

Нейросетевая архитектура

Предложена нейросетевая архитектура, предназначенная для прогнозирования финансовых временных рядов, основывается на следующих постулатах: сеть обучается без учителя (по крайне мере основная ее часть), учитывается время между поступающими входными данными, различные типы данных подаются на различные блоки НС, используется фрактальность входных данных, используются обратные связи.

Обучение без учителя можно рассматривать как возможность нахождения «истинной» зависимости во входных данных, позволяющей строить прогноз с известной вероятностью его точности. Сеть может и не «найти» зависимость, но об этом будет известно. Это позволит принимать решения со значительно меньшим риском. Также самоорганизующаяся НС может решить проблему выбора размера «окна» – интервала входных данных, на котором происходит обучение. Если данные подавать в обратном порядке, то стабилизация сети может означать нахождение достаточного объема входных значений. Вполне возможно, что это потребует выполнения значительного объема итеративных вычислений. Одним из принципов самообучения может быть конкуренция нейронов, дающих наиболее приближенные значения прогнозируемой величины.

Время между поступающими данными можно учитывать, специальным образом преобразовывая значения цены и объема, так что в них уже содержится информация о времени. Эксперименты с такого рода способом нормализации дали положительный результат. Принципиально другой подход заключается в разработке архитектуры НС, позволяющей структурно учитывать время между входными данными. Это связано с тем, что время не является числовым описанием прогнозируемой величины, как например цена и объем.

Фрактальность входных данных стоит выделить особо. Благодаря ей возможно рекурсивное использование одного и того же механизма прогнозирования на различные временные горизонты. Если исходить из того, что краткосрочное прогнозирование, легче чем долгосрочное, то, используя фрактальность финансовых данных, можно рассматривать долгосрочное прогнозирование как краткосрочное, но на больших времен-

ных периодах. Также, одновременно подавая данные с разных уровней фрактала (с разных временных периодов), можно охватить более глубокую ценовую историю при ограниченном входном объеме данных.

Программный инструментарий

В качестве удобного инструмента для исследования возможности создания такой самообучаемой НС предлагается использовать связку программных систем NeuroSolutions компании NeuroDimension, и табличный процессор Excel компании Microsoft. NeuroSolutions имеет в своем составе мощную графическую среду, для компонентного создания нейросетей практически всех известных архитектур, что положительно выделяет ее от программных систем данного класса. Более того, используя среду Visual C++, возможно создание собственных нейрокомпонентов и методов обучения. Стоит выделить возможность экспорттировать любую созданную нейросеть в виде проекта Visual C++. NeuroSolutions имеет в своем составе модуль NeuroSolutions for Excel, который позволяет в значительной степени автоматизировать работу по подготовке входных данных для нейросети, а также получать отчеты о работе в сети в виде графиков и таблиц, что позволяет не только удобно получать информацию, но и быстро подготавливать результаты для использования в документах. Другой модуль, DataManager, использующий платформу .NET, позволяет управлять входными данными, проводить предобработку и графический анализ. Таким образом, данные программные средства в совокупности представляют удобную среду для исследования нейросетей.

Список литературы

1. Хайкин С. Нейронные сети. Полный курс. – М.: Вильямс, 2006.
2. <http://www.xaoc.ru> (Мультифрактальная прогулка вдоль Уолл Стрит, Мандельброт Б.).

ОДНОРОДНЫЕ АЛГОРИТМЫ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ И МОДЕЛИ ЦЕЛЕВЫХ ФУНКЦИЙ

С.М. Елсаков, В.И. Ширяев

Южно-Уральский государственный университет

Постановка задачи

Задачи глобальной оптимизации, характерные для множества приложений [1], формулируются следующим образом:

$$\underset{x \in D}{\text{abs min}} f(x), \quad (1)$$

где D – допустимое множество, а $f(x)$ – целевая функция. Причем, как правило, единственным источником информации о целевой функции служат только вычисления значе-

ний целевой функции в заданных точках. Однократное такое вычисление будем называть *испытанием*.

Основные теоремы

Будем рассматривать алгоритмы глобальной оптимизации как алгоритмы, конструирующие последовательность испытаний:

$$x_{k+1} = \arg \max_{x \in D} P_k(x, \{x_i, f(x_i)\}_{i=1}^k), \quad (2)$$

где $P_k(x, \{x_i, f(x_i)\}_{i=1}^k)$ – функция перспективности проведения очередного испытания в каждой точке. Эта функция зависит от проведенных ранее испытаний, будем рассматривать эту зависимость не как зависимость от множества точек испытаний и множества значений целевой функции, а как зависимость $P(m_k(x), s_k(x))$: от двух функций $m_k(x)$ и $s_k(x)$. К этим функциям будем предъявлять следующие требования:

- A1) $m_k(x_i) = f(x_i), i = \overline{1, k};$
- A2) $s_k(x_i) = 0, i = \overline{1, k};$
- A3) $s_k(x) > 0, x \neq x_i, i = \overline{1, k};$
- A4) $m_k(x)$ и $s_k(x)$ – липшицевы.

Эти требования означают, что функция $m_k(x)$ будет совпадать с функцией $f(x)$ в точках проведенных испытаний, а функция $s_k(x)$ будет показывать, в каких точках испытания уже были проведены. Конкретный вид этих функций выбирается исходя из конкретных моделей многоэкстремальных функций.

Кроме того, обратимся еще раз к последовательности испытаний, и потребуем, чтобы алгоритм был *однородным*, т. е. чтобы для любых двух функций ($f(x)$ и $g(x)$), различающихся на константу ($f(x) = g(x) + c$), последовательности $\{x_k\}$ совпадали. Это требование порождает еще два условия на функции $m_k(x)$ и $s_k(x)$:

- A5) $m_k^f(x) = m_k^g(x) + c;$
- A6) $s_k^f(x) = s_k^g(x).$

Теорема 1. Если функция $P(m_k(x), s_k(x))$ дважды непрерывно дифференцируема, то для однородного алгоритма она может быть представлена как $P(m_k(x), s_k(x)) = q(s_k(x))m_k(x) + p(s_k(x))$, где $p(\cdot)$ и $q(\cdot)$ – некоторые функции.

Теорема 2. (Достаточное условие сходимости). Для того, чтобы множество предельных точек последовательности $\{x_k\}$ совпадало с множеством глобальных минимумов в задаче (1), достаточно, чтобы выполнялось соотношение

$$-\max_{x \in D} (P(s_k(x), m_k(x))) \leq \min_{x \in D} \max_{i=1, k} (f(x_i) - L \|x - x_i\|). \quad (3)$$

Преобразуя выражение (3), можно показать, что справедлива теорема.

Теорема 3 (Достаточное условие сходимости). Для того, чтобы множество предельных точек последовательности $\{x_k\}$ совпадало с множеством глобальных минимумов в задаче (1), достаточно, чтобы выполнялось соотношение

$$Ts_k(x) \geq \min_{i=1,k} (\|x - x_i\|).$$

Будем называть алгоритм *линейным*, если функция перспективности представима в виде $P(s_k(x), m_k(x)) = K * s_k(x) - m_k(x)$.

Примеры однородных алгоритмов

В качестве примеров однородных алгоритмов рассмотрим сначала одномерные алгоритмы. Для того чтобы перейти к рассмотрению алгоритмов, необходимо определиться с моделью целевой функции. Эта модель будет определять вид функций $m_k(x)$ и $s_k(x)$.

Рассмотрим в качестве модели винеровский процесс. В качестве функции $m_k(x)$ выберем условное математическое ожидание, а в качестве $s_k(x)$ – условную дисперсию. Линейный однородный алгоритм, построенный по такой модели, известен как информационно-статистический алгоритм Р.Г. Стронгина [2, 5].

Рассмотрим в качестве модели липшицевые функции. В качестве функции $m_k(x)$ выберем аналогично предыдущей модели кусочно-линейную функцию, а в качестве функции $s_k(x)$ выберем разность между $m_k(x)$ и минорантой липшицевой функции. Линейный однородный алгоритм, построенный по такой модели, известен как алгоритм ломаных С.А. Пиявского [4].

Рассмотрим в качестве модели выпуклые функции. В качестве функции $m_k(x)$ выберем кусочно-линейную функцию, а в качестве $s_k(x)$ разность между функцией

$m_k(x)$ и выражением $\max_{i,x \notin [x_{i-1}, x_i]} \left(f(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} [f(x_i) - f(x_{i-1})] \right)$. Полученный алгоритм будет алгоритмом локальной оптимизации в связи с использованием унимодальной модели целевой функции.

Предлагается новый многомерный однородный алгоритм глобальной оптимизации. Все допустимое множество разбивается на симплексы, вершины которых размещаются в точках проведенных испытаний. Внутри каждого симплекса функции $m_k(x)$ и $s_k(x)$ выбираются независимо. Среди всевозможных триангуляций выбирается триангуляция Делоне [6, 7]. Функция $m_k(x)$ выбирается кусочно-линейной, а функция $s_k(x)$ – кусочно-квадратичной с максимумом в центре описанной вокруг симплекса сферы.

С увеличением размерности вспомогательные вычисления для построения триангуляции Делоне становятся обременительными. Поэтому для пространств более высокой размерности предлагается воспользоваться реберно-симплексными развертками.

Построение этой развертки начинается с произвольной триангуляции, вершины которой совпадают с уже проведенными точками испытаний. К ребрам триангуляции применяется одномерный алгоритм глобальной оптимизации. Следующая точка испытания объявляется новой вершиной и соединяется со всеми необходимыми вершинами.

Теорема 4. Для того чтобы множество предельных точек последовательности, порождаемой одномерным алгоритмом глобальной оптимизации липшицевой функции с константой Липшица L , совпадало с множеством глобальных минимумов многомерной задачи оптимизации, функция $f(x)$ должна быть липшицевой с константой Липшица не более чем $L/2$.

Результаты вычислительных экспериментов

Для тестирования использовались тестовые функции вида:

$$f(x) = - \left\{ \left(\sum_{i=1}^7 \sum_{j=1}^7 A_{ij} a_{ij}(x) + B_{ij} b_{ij}(x) \right)^2 + \left(\sum_{i=1}^7 \sum_{j=1}^7 C_{ij} a_{ij}(x) + D_{ij} b_{ij}(x) \right)^2 \right\}^{0,5},$$

где $a_{ij}(x) = \sin(i\pi x_1) \sin(j\pi x_2)$, $b_{ij}(x) = \cos(i\pi x_1) \cos(j\pi x_2)$, коэффициенты $A_{ij}, B_{ij}, C_{ij}, D_{ij}$ – равномерно распределенные величины на отрезке $[-1; 1]$. Сравнение производилось на выборке из ста функций. Выбирался минимальный коэффициент K , при котором алгоритм во всех случаях находил минимум. Затем вычислялось среднее количество итераций, затрачиваемое на нахождение глобального минимума с найденным коэффициентом K . Точность полагалась одинаковой $\varepsilon = 0,01$.

Результаты тестирования приведены в таблице:

Таблица

Алгоритм	Минимальное значение K	Среднее количество итераций	Средняя точность по аргументу
Алгоритм на основе адаптивных диагональных криевых [3]	2,2	369,95	0,0046
Алгоритм на основе минимизации липшицевой монотонанты	1,4	328,99	0,0027
Алгоритм с использованием триангуляции Делоне	1,6	237,73	0,0019
Алгоритм на основе реберно-симплексной развертки	2,2	352,62	0,0019

Заключение

Как следует из проведенных экспериментов, построенные алгоритмы глобальной оптимизации потребовали наименьшее количество испытаний и обеспечили наивысшую точность из рассмотренных выше алгоритмов.

Список литературы

1. Антонов М.О., Елсаков С.М., Ширяев В.И. Нахождение оптимального расположения радиомаяков в разностно- дальномерной системе посадки летательного аппарата // Авиакосмическое приборостроение. 2005. № 11. – С. 41–45
2. Гергель В.П., Городецкий С.Ю., Гришагин В.П., Стронгин Р.Г. Современные методы принятия оптимальных решений. – Н.Новгород: Изд-во ННГУ, 2002.
3. Квасов Д.Е., Сергеев Я.Д. Многомерный алгоритм глобальной оптимизации на основе адаптивных диагональных кривых // ЖВМиМФ. 2003. Т. 43. № 1. – С. 42–59.
4. Пиявский С.А. Один алгоритм отыскания абсолютного экстремума функции // ЖВМиМФ. 1972. Т. 12. № 4. – С. 888–896.
5. Стронгин Р.Г. Численные методы в многоэкстремальных задачах. – М.: Наука, 1978.
6. Скворцов А.В. Триангуляция Делоне и ее применение. – Томск: Изд-во Томского ун-та, 2002.
7. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. – М.: Мир, 1989.

ОБ ОДНОМ ПОДХОДЕ К ИЗВЛЕЧЕНИЮ НЕЧЕТКИХ ЗНАНИЙ ИЗ СТАТИСТИЧЕСКИХ ДАННЫХ

А.С. Ефимов

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время основанные на знаниях экспертные системы (ЭС) активно применяются при решении сложных практических задач классификации и прогнозирования. Однако традиционный механизм приобретения знаний для ЭС зачастую основан на собеседовании инженера по знаниям и эксперта в предметной области, что существенно увеличивает время формирования базы знаний (БЗ) ЭС. Поэтому одним из наиболее перспективных направлений исследований в данной области, называемой Knowledge discovery in databases, является автоматизированное извлечение знаний для БЗ ЭС на базе имеющихся априорных знаний о проблеме и эмпирических данных, полученных из наблюдений (или статистических данных), обычно хранящихся в базах данных (БД).

Постановка задачи

Целью данного исследования является разработка и реализация в программной системе FKNOD (Fuzzy KNOWledge Discovery) эффективного подхода к генерации нечетких БЗ ЭС на основе имеющихся статистических данных и способа оптимизации полученной БЗ для повышения точности решения задач при сохранении возможностей для интерпретации извлеченных знаний человеком. Разработку подходов к генерации нечетких БЗ перспективно вести в классе гибридных нейро-фаззи-систем искусственного интеллекта, сочетающих преимущества работы с неполными и разнородными данными в нечетких системах с возможностями для обучения нейронных сетей на основе доступных статистических данных.

Архитектура предлагаемой реализации гибридной системы

В основе предлагаемой реализации гибридной нейро-фаззи системы лежит система нечеткого логического вывода Такаги-Сугено 0-порядка, основанная на базе правил вида:

$$IF (x_1 \text{ is } A_1^k) AND \dots AND (x_n \text{ is } A_n^k), THEN (y_1 \text{ is } b_{1k}) AND \dots AND (y_m \text{ is } b_{mk})$$

для $k = 1, \dots, K$, где K – количество нечетких правил в системе, A_i^k ($i = 1, \dots, n$) – термы входных лингвистических переменных (ЛП) x_i , b_{jk} ($j = 1, \dots, m$) – нечеткие синглетоны выходных переменных y_j . В данном исследовании в качестве функций принадлежности термов в антецеденте правил предлагается использовать всюду дифференцируемые гауссовые функции, что является особенно важным для организации процесса обучения на основе градиентных методов. Для реализации этапа фазификации в системе нечеткого вывода Такаги-Сугено 0-порядка выгодно использовать оператор Ларсена [1].

Для генерации нечетких продукционных правил на основе имеющейся обучающей выборки $D_N = \{x(t), y(t)\}_{t=1}^N$, сформированной из статистических данных, предлагается описанную выше систему нечеткого вывода реализовать в виде нечеткой сети, которая отражает структуру набора нечетких продукционных правил в своей топологии и реализует алгоритм нечеткого вывода.

Предлагаемая нечеткая сеть является сетью прямого распространения сигнала и имеет 4 слоя $U = L_1 \cup L_2 \cup L_3 \cup L_4$. Входной слой L_1 состоит из n нейронов, на вход которых подаются четкие значения входных переменных для очередного элемента обучающей выборки. Нейроны второго слоя L_2 организованы в K групп по количеству продукционных правил. Каждый нейрон $u_{ik} \in L_2$ принимает на вход четкое значение соответствующей входной переменной и вычисляет степень уверенности $\mu_{ik}(x_i)$ в том, что ЛП, соответствующая входной переменной x_i , имеет в качестве значения терм A_i^k (параметры гауссовых функций активации данных нейронов c_{ik} и a_{ik} составляют первую группу подстраиваемых в процессе обучения параметров сети). Функциональные элементы третьего слоя L_3 вычисляют степени уверенности в антецеденте каждого правила. Для каждого элемента этого слоя существует по n связей с нейронами предыдущего слоя по количеству простейших высказываний в антецеденте правила. Элементы последнего слоя L_4 вычисляют значения выходных переменных на основании алгоритма нечеткого вывода в системе Такаги-Сугено 0-порядка. Связи между элементами третьего и четвертого слоев имеют веса b_{jk} , соответствующие значениям синглетонов в заключениях правил, представляют из себя вторую группу подстраиваемых в процессе обучения параметров сети.

Этапы процесса извлечения нечетких знаний

Первый этап заключается в проведении структурной и параметрической идентификации предложенной реализации гибридной системы, в результате которой формируется начальная структура нечеткой сети. Основным преимуществом предлагаемой реализации этого этапа является тот факт, что количества нечетких производственных правил и графиков функций принадлежности термов ЛП в антецедентах правил вместе с начальными значениями параметров нечеткой сети определяются одновременно в процессе кластеризации имеющихся статистических данных. При этом полагается, что каждому сформированному кластеру соответствует ровно одно правило. Проекции сформированных в пространстве входных переменных кластеров вдоль каждой из координатных осей порождают графики функций принадлежности термов соответствующих ЛП. В данной работе предлагается использовать алгоритм сферической классификации, предложенный в [2], на основе модифицированной схемы конкурентного обучения. Суть модификации состоит в определении на каждой итерации нейрона-победителя и его ближайшего конкурента, коррекции их весов, а также в том, что в качестве меры близости используется евклидово расстояние, масштабируемое количеством побед данного нейрона, что позволяет успешно решать так называемую проблему мертвых нейронов.

Для определения термов ЛП в простейших высказываниях антецедента каждого правила применяется ассоциация компонентов вектора центра $c_k = (c_{1k}, c_{2k}, \dots, c_{nk})$ каждого кластера с центрами гауссовых функций принадлежности μ_{ik} термов в простейших высказываниях антецедента соответствующего правила. Значения ширины a_{ik} гауссовых функций принадлежности термов предлагается определять на основе эвристики ближайшего соседа [3], что обеспечивает примерно равномерное покрытие всего нормализованного универсума $[0,1]$ ЛП графиками функций принадлежности ее термов. Для определения неизвестных параметров синглетонов b_{jk} в заключениях правил предлагается использовать всю обучающую выборку, определяя при этом отношение между степенью уверенности в принадлежности элементов обучающей выборки к каждому сформированному кластеру и значениями выходных переменных.

Второй этап извлечения нечетких знаний состоит в проведении параметрической оптимизации на основе обучения с учителем (предлагается использовать градиентный алгоритм, аналогичный предложенному в [3]) построенной нечеткой сети. В результате подстраиваются параметры графиков функции принадлежности термов в антецедентах правил и параметры синглетонов в консеквентах правил для повышения точности решения задачи с помощью формируемой БЗ.

Апробация предложенного подхода

Предложенный в данном исследовании подход к извлечению знаний был опробован при решении различных практических задач. В частности, при решении классификационной задачи классификации ирисов Фишера. Средняя точность решения задачи при проведении кросс-валидации составила 94–96%. При решении задачи диагностики заболеваний сердца (Heart Disease) средняя точность решения задачи классификации составила 81–83%. Данные результаты сопоставимы с результатами других исследователей [3]. При решении задачи прогнозирования летального исхода у пациентов с инфарктом миокарда на базе статистики, предоставленной специалистами МЛПУ № 5 г. Нижнего Новгорода, полученная точность прогноза составила 70–80%.

Программная реализация в системе FKNOD

Практическая значимость и ценность данного исследования заключаются в создании программной системы FKNOD, основанной на системе NFLIS [4], для реализации предложенного подхода к извлечению нечетких знаний из статистических данных. В основе данной программной системы лежит объектно-ориентированная библиотека нечеткого вывода FLIS, объектно-ориентированная библиотека GKNB для проведения процедуры извлечения нечетких знаний на базе предложенного в работе подхода, объектно-ориентированная библиотека для работы с нечеткими сетями NFN и проведения их обучения на основе градиентного алгоритма, а также функциональность для работы с БД статистики и проведения экспериментов. Все созданные библиотеки классов написаны на языке C# для платформы .NET Framework v.2.0. На базе разработанных библиотек в среде Visual C# 2005 была создана программа-оболочка для проведения процедуры извлечения нечетких знаний, отображения результатов извлечения, обеспечения ручного редактирования созданных БЗ и проведения экспериментов.

Список литературы

1. Brown M. and Harris C.J. Neurofuzzy Adaptive Modelling and Control. Prentice Hall, Hemel Hempstead, 1994
2. Xu L., Krizak A. and Oja E. Rival Penalized Competitive Learning for Clustering Analysis. IEEE Trans. on Neural Networks 4(4), 1993. P. 636-649.
3. Castellano Giovanna. A Neurofuzzy Methodology for Predictive Modeling. A thesis for the degree of Doctor of Philosophy in Computer Science, University of Bari, Faculty of Science, Department of Computer Science, 2000.
4. Ефимов А.С., Морёнов О.А. FLIS – система нечеткого вывода. // Технологии Microsoft в теории и практике программирования: Материалы конференции. – Н.Новгород: Изд. Нижегородского госуниверситета, 2006. – С. 104–105.

ПРОТОТИП РАСПРЕДЕЛЕННОЙ СУБД РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ МЕТОДОВ УПРАВЛЕНИЯ ТРАНЗАКЦИЯМИ

А.В. Жарков

Пензенский госуниверситет

Введение

Большинство исследований в области СУБД реального времени опираются на имитационное моделирование поведения СУБД на основе реальных прототипов для определения необходимых характеристик [2,3]. Мы пошли по тому же пути и разработали прототип СУБД реального времени, ориентированный на исследование характеристик различных методов управления транзакциями. Основным отличием данного исследования является ориентация на использование СУБД реального времени в областях смежных с промышленными системами АСУТП. Такая специализация открывает ряд специфических проблем, которые необходимо решить при управлении транзакциями СУБД РВ, например, обработка данных от источников с разным семантическим приоритетом.

Постановка задачи

При разработке прототипа СУБД реального времени и построении экспериментальной вычислительной системы решались следующие задачи:

- формализовать параметры транзакций, используемые при управлении приоритетами транзакций;
- разработать метод управления транзакциями, учитывающий наличие статического приоритета транзакций, отражающего семантику транзакции;
- разработать архитектуру распределенной СУБД;
- провести экспериментальное исследование временных характеристик предложенного метода управления транзакциями в сравнении с традиционными методами управления транзакциями.

Параметры транзакций

При рассмотрении транзакций будем учитывать параметры атомарных транзакций. Под атомарной транзакцией (микротранзакция) будем понимать элементарную совокупность действий, которая поддерживается СУБД на физическом уровне. Каждую микротранзакцию можно описать следующим образом:

$$w_{micro} = \langle x, d, p, m, t_a, t_{rvi}, t_{exec}, t_{end} \rangle, \quad (1)$$

Введем следующие обозначения: x – объект базы данных, с которым связана транзакция; d – данные, ассоциированные с транзакцией; p – приоритет микротранзакции, обусловленный семантикой данных; $m \in M$, $M = \{\text{Add}, \text{GroupAdd}, \text{Insert}, \text{GroupInsert}, \text{Modify}, \text{GroupModify}, \text{ScanModify}, \text{Delete}, \text{GroupDelete}, \text{ScanDelete}, \text{ScanSelect}, \text{ScanSelectID}, \text{ScanIndex}, \text{ScanSort}, \text{NestedLoopsJoin}, \text{NestedLoopsIndexJoin}, \text{HashJoin}\}$ — множество типов микротранзакций; t_a — метка времени(arrival time), соответствующая времени регистрации микротранзакции в системе; t_{rvi} — относительный интервал достоверности микротранзакции. t_{exec} — метка времени, соответствующая началу выполнения микротранзакции; t_{end} — метка времени, соответствующая моменту завершения или отката микротранзакции;

Динамические характеристики микротранзакций

Помимо описанных в формуле (1) параметров, микротранзакции имеют совокупность динамических параметров, которые также используются при управлении приоритетами. Рассмотрим следующие динамические характеристики:

$DL(T) = t_a + \min(RVI(d), t_{rvi})$ — конечный срок жизни микротранзакции (deadline).

$DDL(T)$ — конечный срок жизни данных (data deadline), зависит от темпоральных характеристик объекта базы данных, определяется как:

$$DDL(T) = AVI(x) = LUT(x) + RVI(x), \quad (2)$$

где X – объект базы данных, который связан с транзакцией T , $LUT(X)$ – время последнего обновления объекта X , $RVI(X)$ – относительный интервал достоверности объекта.

$EET(T)$ – расчетное время выполнения транзакции (estimated execution time). Расчетное время выполнения транзакции зависит от статистического аппарата СУБД.

$ETT(T) = now() - t_{exec}$ — общее время, затраченное на выполнение транзакции;
 $now()$ — специальная функция, возвращающая текущее время в системе.

$SF(T) = DL(T) - (now() + EET(T))$ — фактор резерва времени, фактор, который учитывает количество времени, оставшееся до наступления конечного срока жизни транзакции.

$PR(T)$ — предложенная функция оценки динамического приоритета, которая комбинирует традиционные методы управления приоритетами транзакций (EDF, EDDF, LSF). Данная функция определяется как:

$$PR(T) = PR_{EDF}(T) \times (1 - \mu) + \frac{\mu}{SF(T) - 1}, \quad (3)$$

где $PR_{EDF}(T)$ — функция оценки приоритета транзакции, которая комбинирует методы EDF и EDDF, и определяется как:

$$PR_{EDF}(T) = (\alpha \times DDL(T) + (1 - \alpha) \times DL(T)). \quad (4)$$

Параметры α и μ являются настроочными параметрами и определяют текущий метод управления транзакциями.

Для того, чтобы учесть статический приоритет транзакций, была предложена функция сравнения приоритетов PR_c , которая определяется как defined as:

$$PR_c(T_1, T_2) = \beta \times SIGN(p(T_1) - p(T_2)) + (1 - \beta) \times SIGN(PR(T_1) - PR(T_2)), \quad (5)$$

где $\beta \in [0,1]$ — настроочный параметр для комбинации $p(T)$ статического приоритета с динамическим приоритетом $PR(T)$ (3);

$$SIGN(v) = \begin{cases} 1, & \text{if } v \geq 0 \\ 0, & \text{if } v < 0 \end{cases}$$

— функция определения знака.

Таким образом, получен набор функций, изменяя настроочные параметры α , β , μ которых, можно выбирать метод управления приоритетами транзакций.

Управление приоритетами микротранзакций

Рассмотрим методы управления транзакциями, которые могут быть исследованы с помощью разработанного прототипа СУБД РВ:

EDF — Earliest Deadline First — первыми обрабатываются транзакции, имеющие минимальный срок жизни. Данная стратегия выбора приоритета является исторически первой, и учитывает только конечный срок жизни транзакции без учета темпоральной целостности целевого объекта. Параметры настройки: $\alpha = 0, \beta < 0.5, \mu = 0$.

EDDF — Earliest Data Deadline First — первыми обрабатываются транзакции, которые имеют самый маленький срок жизни данных. Данная стратегия учитывает темпоральную целостность объекта базы данных, но совершенно не берет во внимание конечный срок жизни транзакции. Параметры настройки: $\alpha = 1, \beta < 0.5, \mu = 0$.

LSF — Least Slack First — наивысший приоритет имеют транзакции с минимальным неотрицательным резервом времени выполнения. В данной стратегии параметры в

формуле оценки приоритета равны $\beta < 0.5$, $\mu = 1$, параметр α никакого влияния не оказывает.

SPF – Static Priority First — стратегия, которая учитывает помимо темпоральных характеристик статический приоритет транзакции. В данной стратегии статический приоритет транзакции имеет значительно больший вес, чем темпоральные характеристики транзакции. Для данной стратегии используется значение параметра $\beta > 0.5$, параметры α и μ , зависят от выбранной вторичной стратегии оценки приоритета на базе темпоральных характеристик транзакции.

Архитектура СУБД реального времени

Обобщенная архитектура СУБД РВ представлена на рис. 1.



Рис. 1. Обобщенная архитектура прототипа СУБД РВ

Макет СУБД реального времени использует многопоточную архитектуру, все потоки запускаются в адресном пространстве одного процесса. Поэтому межпроцессное взаимодействие используется только для общения с клиентами. Для обеспечения работы с данными в реальном времени СУБД загружает в оперативную память всю базу данных или наиболее часто используемые объекты базы данных (таблицы или индексы).

Рассмотрим основные модули системы.

Интерфейс сетевого взаимодействия. Данный модуль используется для выделения логики межпроцессного взаимодействия. Задачей данного модуля является прием и передача специализированных пакетов данных. При получении пакета происходит выделение транзакции, привязка данных и передача на следующий уровень обработки.

Модуль предварительной обработки транзакций. Данный компонент осуществляет распределение и сортировку транзакций в зависимости от типа транзакции (составная или простая) и от типа действия (добавление, модификация, удаление, выборка). Все полученные транзакции помещаются в общую очередь.

Менеджер запросов. Данный компонент осуществляет преобразование запросов во внутреннюю форму и формирование соответствующего плану выполнения запроса набора связанных микротранзакций.

Модуль оценки приоритетов. Данный модуль используется для расчета динамических параметров микротранзакции и динамического приоритета в зависимости от заданной стратегии управления приоритетами.

Планировщик транзакций. Данный компонент содержит логику управления транзакциями при распределении их по потокам в зависимости от приоритета микротранзакции.

Пул потоков обработки транзакций. Данный компонент управляет набором рабочих потоков, внутри которых и осуществляется выполнение транзакций.

Менеджер объектов БД. Данный компонент предоставляет интерфейс для доступа к таким объектам базы данных, как таблицы, индексы, хеши. Он отвечает за работу с данными, синхронизацию информации в оперативной и долговременной памяти.

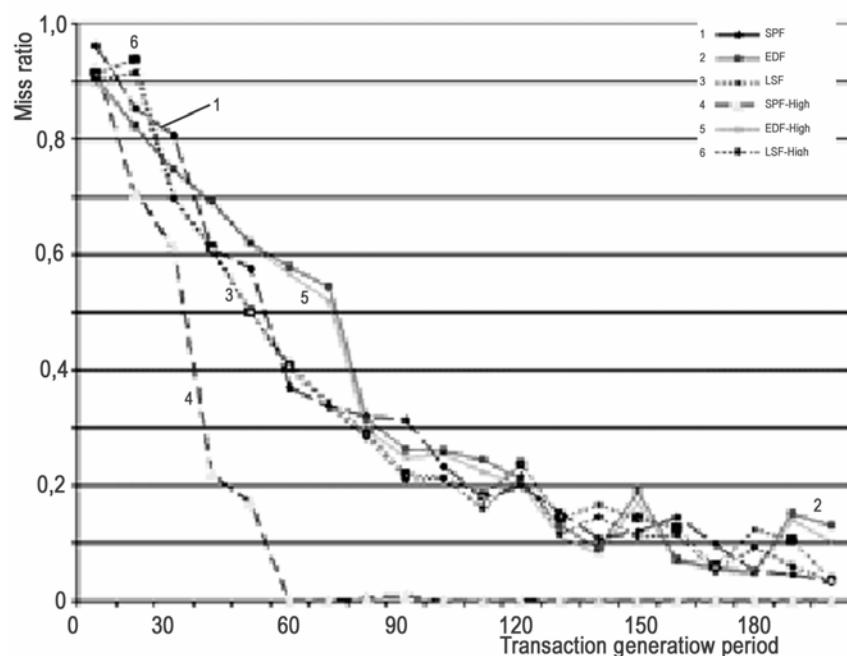


Рис. 2. Результаты экспериментальных исследований

Результаты экспериментов

Задачей экспериментов было оценить влияние метода управления транзакциями с учетом статических приоритетов на уменьшение количества неуспешных транзакций, которые имеют высокий статический приоритет. На рис. 2 представлены усредненные результаты экспериментов, которые показывают эффективность предложенного метода по сравнению с традиционными в плане уменьшения количества неуспешных высоко-приоритетных транзакций. При этом общий процент отказов был несколько выше, чем у традиционных методов.

Список литературы

1. Жарков А.В. Некоторые подходы к построению прототипа СУБД РВ с точки зрения оптимизации запросов. Труды VI Международной научно-технической конференции «Новые информационные технологии и системы» – Пенза, 2004. С. 233
2. Жарков А.В. Описание обобщенной архитектуры подсистемы управления материализованными представлениями СУБД реального времени и формальной модели внутренних структур пакетов материализованных представлений. Сборник статей Международной научно-технической конференции «Современные информационные технологии» – Пенза, 2004. С. 80.
3. Шашков Б. Д., Жарков А.В. Управление приоритетами микротранзакций в СУБД реального времени. Сборник статей Международной научно-технической конференции «Современные информационные технологии» – Пенза, 2005. С. 128.

АЛГОРИТМЫ РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ СМЕШАННЫХ РАЗМЕРОВ ПРИ ПРОЕКТИРОВАНИИ СБИС

А.В. Живодеров, К.В. Корняков, И.Б. Мееров

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Проектирование сверхбольших интегральных схем (СБИС) – чрезвычайно сложный процесс, включающий в себя несколько стадий. Одной из промежуточных стадий проектирования является размещение, в процессе которого определяется положение каждого логического элемента на плате. Задача размещения элементов интегральных схем (*circuit placement problem*) – известная задача дискретной оптимизации.

В настоящее время интегральные схемы содержат сотни тысяч элементов, основную часть которых составляют небольшие стандартные элементы, но также присутствуют и макроблоки. Наличие в одной схеме одновременно огромного числа небольших элементов и крупных блоков приводит к значительному усложнению алгоритмов размещения. Макроблоки требуют особого внимания в процессе размещения, поскольку качество их размещения во многом определяет будущую производительность проектируемой схемы.

Данная работа посвящена рассмотрению различных подходов к решению *задачи размещения элементов смешанных размеров (mixed-size placement problem)*.

Классы задач размещения

Наиболее простой из данного класса является *задача размещения стандартных элементов (standard cell placement problem)* – сотен тысяч или даже миллионов прямо-

угольных блоков одинаковой высоты (отсюда название стандартный элемент) и возможно различной ширины. В качестве основного минимизируемого критерия рассматривается суммарная длина соединений. Известными подходами к решению задачи размещения являются: метод моделирования отжига [1,8], аналитические методы [2] и методы, основанные на рекурсивной бисекции [3]. На практике, все они дают близкие по качеству размещения [7].

Качественно другой задачей является разработка *общей топологической структуры* (*floorplanning*). В этом случае схема содержит несколько сотен только «больших» прямоугольных блоков. Блоки должны быть размещены без перекрытий. Целевой функцией является, как правило, комбинация занимаемой площади и длины соединений. Малые изменения в размещении (например, смена ориентации блока) могут повлечь за собой частичное наложение блоков друг на друга, а также вызвать существенное изменение длины соединений. Для решения этой задачи используется в основном метод моделирования отжига [3], поскольку он позволяет получать наиболее качественные размещения. Существуют и другие методы [4, 5].

Между этими двумя классами задач располагается *задача размещения элементов смешанных размеров* (*mixed-size placement problem*), в которой на одной схеме присутствуют как крупные макроблоки, так и множество стандартных элементов. Далее будет рассмотрен ряд подходов для решения этой задачи «валунов и песчинок».

Размещение элементов смешанных размеров

Как уже было сказано выше, основная причина высокой сложности задачи размещения элементов смешанных размеров заключается в том, что необходимо одновременно размещать макроблоки и стандартные элементы. Положение макроблоков сильно влияет на суммарную длину соединений. Если макроблок передвинуть, он может образовать перекрытия с большим числом более мелких элементов, положение которых тоже нужно изменить, чтобы все перекрытия были устраниены. Разница в длине проводов при подобных манипуляциях приводит к тому, что поведение целевой функции становится непредсказуемым.

Далее рассмотрим основные подходы к решению задачи размещения элементов смешанных размеров.

Иерархический подход

Одним из первых подходов к решению этой задачи был иерархический подход [3]. Стандартные элементы сначала разбиваются на блоки, затем на множество макроблоков и блоков, полученных в результате разбиения, решается задача разработки общей топологической структуры, описанная выше. Целевой функцией при этом считается суммарная длина соединений. После размещения блоков стандартные элементы укладываются внутри блоков с помощью алгоритмов детального размещения.

Несмотря на то, что этот метод позволяет снизить размерность задачи, разбиение стандартных элементов на блоки может помешать достигнуть оптимума или близкого к нему значения, поскольку элемент, попавший в определенный блок, не сможет покинуть его в дальнейшем. В результате мы находим один из локальных минимумов целевой функции, значение которого может существенно отличаться от оптимального.

Подход, основанный на «измельчении» макроблоков

Метод, описанный в [6], производит размещение в несколько этапов. Авторы этой статьи используют *Carpo* – инструмент размещения стандартных элементов, и *Parquet* – инструмент для разработки общей топологической структуры.

На первом этапе все макроблоки измельчают на части, идентичные стандартным элементам, и связывают их в тесную сеть (путем добавления фиктивных соединений) для гарантии того, что эти элементы в итоге будут помещены рядом друг с другом. После этого производится глобальное размещение (*Caro*) с целью получения начальной конфигурации.

На втором этапе положение макроблоков определяется путем усреднения координат элементов, на которые он был разбит. Стандартные элементы сливаются в легкие блоки (*soft blocks*), и *Parquet* формирует легальное размещение макроблоков и легких блоков. На последнем этапе положение макроблоков полагается фиксированным, и такое размещение вновь подается на вход *Caro*, который размещает оставшиеся стандартные элементы.

Этот подход схож с иерархическим. Существенным отличием является то, что вместо алгоритма разбиения стандартных элементов на легкие блоки используется предварительное размещение, то есть при группировке в легкие блоки учитывается информация о предпочтительном размещении элементов, полученная во время предварительного размещения. Как результат, в легкие блоки группируются тесно связанные элементы, и финальное размещение получается более высокого качества, чем при иерархическом подходе.

Одновременное размещение макроблоков и стандартных элементов

Третий подход предполагает одновременное размещение стандартных элементов и макроблоков. В качестве примера рассмотрим процесс размещения инструмента *Dragon2005* [1]. Условно его можно разделить на две части: глобальное и детальное размещение.

На стадии глобального размещения множество элементов рекурсивно разбивается на кластеры. Каждый кластер помещается в бин – прямоугольный участок на плате. Элементам присваиваются координаты центров бинов. Число бинов удваивается после каждого разбиения за счет горизонтальных или вертикальных разрезов. Каждый элемент независимо от своих размеров должен целиком помещаться в своем бине. Поэтому бин, содержащий макроблок, должен быть поделен не пополам (иначе ни в один из подбинов макроблок не поместится), а в некоем отношении. Разбиение продолжается до тех пор, пока число элементов в каждом бине не окажется меньше некоего наперед заданного числа. Итогом глобального размещения является определение примерного местоположения для каждого из элементов.

После глобального размещения происходит легализация макроблоков (укладка в ряды и избавление от перекрытий), их положение фиксируется, и решается задача размещения стандартных элементов.

Данный подход, оперирующий одновременно с макроблоками и стандартными элементами, представляется наиболее перспективным в связи с постоянным увеличением числа макроблоков (другие подходы предназначены для сравнительно небольшого числа макроблоков). Данный подход эффективен и универсален, позволяет использовать общий алгоритм для размещения элементов схем с различным соотношением числа стандартных элементов и макроблоков.

Заключение

Авторы данной работы участвуют в разработке инструмента размещения *itlDragon*, который в настоящее время успешно решает задачу размещения стандартных элементов [8]. На текущий момент результаты его работы сравнимы с результатами инструментов, разрабатываемых в зарубежных университетах (*Dragon*, *Capo*, *FengShui*, *mPL*).

Дальнейшей целью нашей деятельности является, прежде всего, обобщение вычислительной схемы нашего инструмента для решения задачи размещения элементов смешанных размеров. Мы планируем использовать последний из описанных подходов, ввиду его удобства для внедрения в наш инструмент и наибольшей перспективности.

Список литературы

1. Taghavi T., Yang X., Choi BK., Wang M., and Sarrafzadeh M. Dragon 2005: Large-Scale Mixed-Size Placement Tool In Proc. of International Symposium on Physical Design(ISPD). 2005. P. 245-247.
2. Kleinhans J., Sigl G., Johannes F., and Antreich K.. GORDIAN: VLSI placement by quadratic programming and slicing optimization. // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 1991. 10(3). P. 356–365,
3. Agnihotri A., Yildiz M. C., Khatkhate A., Ono S. and Madden P. H. Recursive Bisection Based Mixed Block Placement.
4. Murata H., Fujiyoshi K., Nakatake S., and Kajitani Y. VLSI module placement based on rectangle-packing by the sequence pair // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 1996. 15(12). P. 1518–1524,
5. Yingxin Pang, Florin Balasa, Koen Lampaert, and Chung-Kuan Cheng. Block placement with symmetry constraints based on the o-tree non-slicing representation. In Proc. Design Automation Conf, 2000. P. 464–467,
6. S. N. Adya and I. L. Markov. Consistent placement of macroblock using floorplanning and standard-cell placement. In Proc. Int. Symp. on Physical Design, 2002. P. 12–17.
7. Adya S.N., Yildiz M., Markov I.L. et al. Benchmarking for large-scale placement and beyond, in Proc. Int. Symp. Physical Design, 2003. P. 95–103.
8. Гагарина С.А., Живодеров А.В., Корняков К.В., Курина Н.В., Мееров И.Б. Использование метода имитации отжига для решения задачи размещения элементов на схеме // Технологии Microsoft в теории и практике программирования. Материалы конференции / Под ред. проф. Р.Г. Стронгина. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2006. – С. 220–225.

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ MICROSOFT ПРИ АВТОМАТИЗАЦИИ ЗАДАЧ УПРАВЛЕНИЯ КОНТИНГЕНТОМ СТУДЕНТОВ В АСТРАХАНСКОМ ГОСУДАРСТВЕННОМ ТЕХНИЧЕСКОМ УНИВЕРСИТЕТЕ

Г.А. Ильин, А.В. Морозов

Астраханский государственный технический университет

Деканатами университета производятся сложные работы по учету контингента студентов и их успеваемости. Формируется значительное число документов и приказов, связанных с ведением личных дел студентов, учетом параметров их обучения (учебный план, итоговая успеваемость) Все это связано со значительными затратами ручного труда. В крупных вузах, в которых обучаются тысячи студентов, элементарные работы по учету студентов становятся очень трудоемкими и требуют обязательной автоматизации рутинных операций.

При анализе рынка программных продуктов был обнаружен ряд средств автоматизации задач деканата. Среди них АСУ «Деканат» МГТУ им. Баумана, АСУ «Деканат»

Московского государственного технического университета технологий и управления. Глубокая интеграция данных систем в глобальные АСУ институтов – разработчиков делает невозможным их использование без полной замены всех АСУ университета. Другим недостатком является неполное соответствие функций данных систем текущей образовательной политике АГТУ.

Из вышеизложенного можно сделать вывод, что необходима автоматизация процесса деятельности деканата. Это позволит понизить трудозатраты работников деканата за счет автоматизации рутинной работы и повысит качество работы деканатов.

Целью разработки системы «Деканат 2» является автоматизация ввода исходных данных, обработки информации и печати личных дел обучающихся, успеваемости обучающихся, приказов по движению контингента и изменению личной информации; формирования стандартной и нестандартной отчетности деканатов (дирекций).

Назначением системы является снижение трудоемкости работы деканатов (директорий) в повседневной работе, связанной с учетом контингента и успеваемости обучающихся, формированием стандартных приказов и нестандартной отчетности и ужесточение контроля делопроизводства деканатов.

Автоматизированная система «Деканат 2» включает в себя несколько подсистем.

1. Подсистема учета личных данных студентов.
2. Подсистема учета параметров обучения студентов.
3. Подсистема рейтинг-контроля успеваемости студентов.
4. Подсистема создания отчетности.

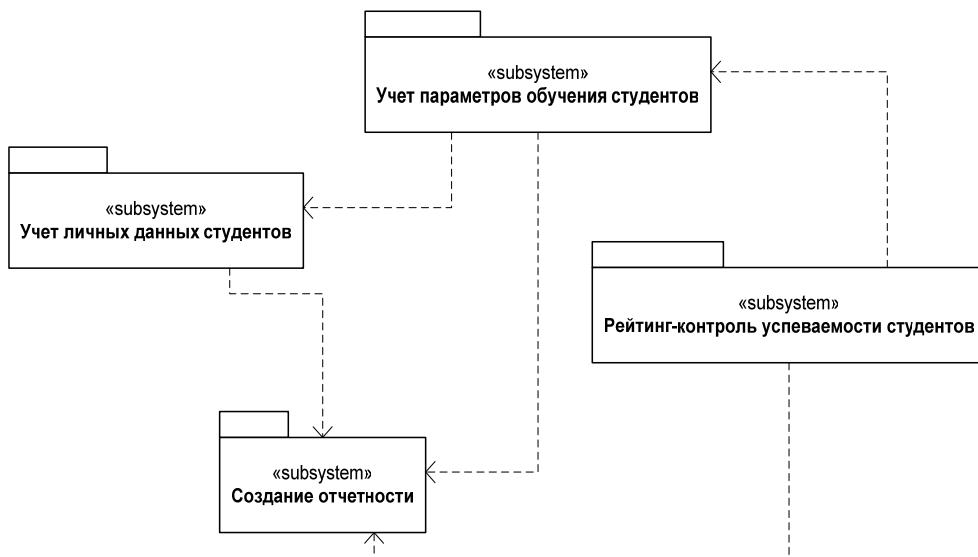


Рис. 1. Диаграмма взаимодействия подсистем

Подсистема учета личных данных студентов предназначена для быстрого получения и редактирования необходимой информации о студентах. Подсистема предоставляет различные средства выборки студента, в том числе:

- поиск по любому из четырех параметров (номер зачетки, фамилия, имя, отчество),

- фильтрации по: специальности, группе, курсу, форме обучения, стилю обучения, статусу (числится, отчислен, академический отпуск), форме оплаты.

Подсистема учета параметров обучения студентов. К основным параметрам обучения студентов относятся:

- Учебный план, на котором обучается студент.
- Статус студента.
- Группа, в которой обучается студент.

В свою очередь каждый из этих параметров регламентирует дополнительные параметры. Учебный план однозначно определяет:

- Факультет, на котором обучается студент.
- Специальность, на которой обучается студент.
- Специализацию, если таковая предусмотрена.
- Год набора – год, в котором студент был зачислен на первый год обучения.
- Форму обучения студента (очная, заочная).
- Стиль обучения студента (традиционный, ускоренный, индивидуальный, дистанционный).
- Степень, получаемая студентом по окончании обучения (специалист, бакалавр, магистр).
- Квалификация, получаемая студентом по окончании обучения.
- Период обучения.
- Дисциплины, изучаемые студентом в течение всего периода обучения.
- Количество аудиторных часов, отводимое на обучение студента по каждой дисциплине.
- Общее количество часов, отводимое на обучение студента по каждой дисциплине.
- Практики, которые необходимо пройти студенту во время обучения, и время, отведенное на них.
- Курсовые работы и проекты, которые необходимо выполнить студенту во время обучения.

Статус студента является показателем, по которому можно судить, обучается ли студент в данный момент времени или нет. Статус студента может принимать следующие значения:

- Числится (студент в настоящее время обучается).
- Отчислен (в настоящее время студент не обучается).
- Академический отпуск (студент временно приостановил свое обучение).

Основной причиной смены статуса студента, является текущее состояние его успеваемости.

Группа, в которой обучается студент, отражает специальность, на которой обучается студент, и однозначно определяет курс, на котором обучается студент.

Существует два основных типа документов, регламентирующих все основные операции по управлению параметрами обучения студента: приказ и распоряжение.

Подсистема рейтинг-контроля успеваемости студентов позволяет вести учет рейтинга студента. Рейтинг – это сумма баллов по 100 балльной шкале, определяющая оценку знаний студента по отдельной семестровой дисциплине. Для контроля текущей успеваемости (рейтинга) преподаватели на запланированных контрольных неделях заполняют экзаменационно-рейтинговую ведомость. Для реализации этой подсистемы была выбрана технология ASP .NET, применение которой позволило использовать ар-

хитектуру с тонким клиентом. Такая архитектура была выбрана по ряду причин, главными из которых являются:

относительно низкие требования к клиентским машинам,
возможность ввода рейтинга с любой машины, имеющей выход в Internet,
простота обновлений подсистемы, без изменения на клиентских машинах.

Подсистема создания отчетности позволяет формировать и печатать стандартную отчетность, необходимую деканатам. К такой отчетности относятся:

Экзаменационная и зачетная ведомости.

Список студентов.

Шаблон приложения к диплому.

Приложение к диплому.

Для реализации части отчетов (ведомости) применялся Crystal Report. Также применялись MS Word 2003 (шаблон приложения к диплому) и MS Excel 2003 (Список студентов).

Для реализации автоматизированной системы управления «Деканат 2» используются технологии компании Microsoft.

Архитектура системы включает в себя следующие элементы:

1. Сервер базы данных (ОС: Windows Server 2003 R2, СУБД: MS SQL Server 2000).
2. Толстый клиент (ОС: Windows XP Professional, подсистемы: учет личной информации, учет параметров обучения, создание отчетности).
3. Web-сервер (ОС: Windows Server 2003 R2, IIS, подсистема рейтинг-контроль).
4. Тонкий клиент (ОС: Windows 2000, Internet Explorer).

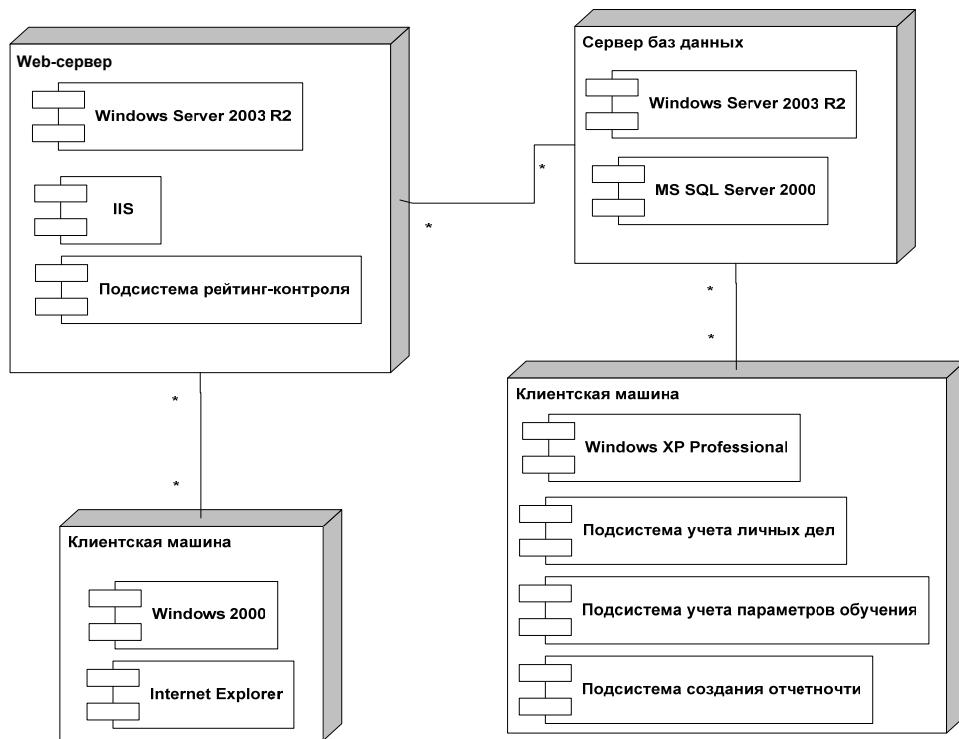


Рис. 2. Диаграмма развертывания

Взаимодействие с Web-сервером осуществляется по протоколу HTTP. В качестве средства разработки используется Microsoft Visual Studio 2005. При реализации системы использовался Gentle .Net, что сильно облегчило работу с базой данных.

Использование технологий Microsoft в данном проекте значительно облегчило реализацию и позволило значительно уменьшить экономические затраты на разработку проекта. В том числе использование технологий Microsoft позволило обеспечить полную совместимость всех компонентов разработки. Применение встроенных средств безопасности позволило значительно повысить защищенность информации. Одним из преимуществ системы VS.NET является легко применимое средство создания инсталляторов, которые позволяют быстро создать установочный файл, способный к полнофункциональному развертыванию системы на машине пользователя.

Список литературы

1. Мамаев Е. Microsoft SQL Server 2000. Наиболее полное руководство. – СПб.: «БХВ-Петербург», 2001.
2. Станек У.Р. Microsoft Windows Server 2003. Справочник администратора / Пер. с англ., 2-е изд. - М.: Издательско-торговый дом «Русская редакция», 2003. – 640 с.
3. Станек У.Р. Microsoft Windows XP Professional. Справочник администратора / Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2003. – 448 с.
4. Шилдт, Герберт Полный справочник по C# / Пер. с англ. – М.: Издательский дом «Вильямс», 2004. – 752 с.
5. Троялсон Э. C# и платформа .Net. Библиотека программиста. – СПб.: Питер, 2004. – 796 с.

ПРИЛОЖЕНИЕ ДЛЯ ПРОЕКТИРОВАНИЯ КИНЕМАТИКИ МЕХАНИЗМОВ В СРЕДЕ AUTOCAD С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ ATL

А.А. Ионов

Нижегородский госуниверситет им. Н.И. Лобачевского

AutoCAD предоставляет пользователю для разработки собственных приложений встроенные языки программирования: Visual LISP и Visual Basic. Используя эти языки программирования, программист получает лёгкий доступ к объектам AutoCAD. Но эти языки программирования являются интерпретируемыми языками, из чего следует низкая производительность программ, написанных на них, а при проектировании движения сложных механизмов производительность играет важнейшую роль, для обеспечения возможности оптимизации в реальном времени.

На помощь приходит технология COM. COM является платформо-независимой, объектно-ориентированной технологией, позволяющей создавать бинарные компоненты. Эти компоненты можно использовать как локально, так и в распределенном сетевом окружении. COM служит основой для: OLE (технология составных документов), ActiveX-объектов и элементов управления ActiveX, DCOM, COM+. COM-объект можно сравнить с объектом в понимании C++, VB или Java. Объект COM – это некоторая сущность, имеющая состояние и методы доступа, позволяющие изменять это состояние.

Предлагается создать некоторый COM-объект, который бы осуществлял сложные расчёты положения звеньев механизма и передавал бы результаты программе, написанной на Visual LISP или Visual Basic, которая отвечала бы только за взаимодействие с пользователем и изменение координат объектов в 3D модели в AutoCAD. В отличии от Visual LISP, Visual Basic AutoCAD имеет встроенные средства создания графического интерфейса (для Visual LISP приходится использовать дополнительные средства, такие как Object DCL). Также Visual Basic способен работать с типами данных, используемых в COM-технологии, что способствует более быстрому обмену данными между приложениями. Visual LISP работает в основном со списками, для данных, полученных от COM-объекта, необходимо выполнять операции преобразования их в списки, если это не простые типы: целое или с плавающей точкой. Из вышеуказанных причин программу, реализующую пользовательский интерфейс и взаимодействующую с COM-объектом, пишем на Visual Basic.

Реализовывать наш COM-объект будем в качестве ActiveX сервера. Серверы ActiveX бывают трёх типов: полные серверы (full server), мини-серверы (mini-server) и серверы автоматизации (automation server).

Полные серверы могут выполняться и как сервер, и как самостоятельное приложение. Например, Microsoft Word можно запустить в виде самостоятельного приложения, создать документ или брошюру и затем сохранить содержание в виде doc-файла. Или же можно запустить другое приложение, например WordPad, и включить в него содержание, созданное в Microsoft Word.

Мини-сервера, наоборот, могут использоваться только для включения их содержимого в другие приложения. Например, с Microsoft Word поставляется множество апплетов, помогающих создать документы профессионального вида. Одно из них называется WordArt. Если попытаться запустить эту программу, то выдается сообщение о том, что это приложение может выполняться только при запуске из другого приложения.

Серверы автоматизации стоят отдельно от полных серверов и мини серверов. Они не позволяют включать своё содержимое в приложение, которое выполняет такой сервер. Чаще всего сервер автоматизации предоставляет специальные объекты, методы и свойства, позволяющие управлять этим сервером. Предположим, например, что требуется объединить основной документ, созданный в Microsoft Word, с документом-источником, хранящимся в базе данных Microsoft Access. Можно запустить Microsoft Word и выбрать в меню команду Слияние. Но также можно написать приложение на Visual Basic (или любом другом языке, поддерживающем автоматизацию) и удаленно контролировать Word. Таким образом, можно настроить программу на автоматическое выполнение каждый день в 16:00 без вмешательства пользователя.

Т.к. от нашего ActiveX сервера нам требуются только результаты расчёта положений объектов 3D модели в AutoCAD, то нам подходит сервер автоматизации, так как от него требуются лишь методы, которые бы задавали серверу структуру механизма и координаты звеньев, а после этого сервер бы предоставлял методы, позволяющие получить параметры движения звеньев.

Серверы автоматизации, в свою очередь, бывают двух видов: серверы процесса (in-process servers) и локальные серверы (out-of-process server). Серверы процесса создаются на основании класса, хранящегося в файле DLL, загружаемого и выполняемого в том же адресном пространстве, что и само приложение. Все экземпляры класса имеют один и тот же код, но каждый имеет собственную область данных. Локальные серверы работают в своем адресном пространстве. Этот тип серверов выполняется в виде exe-файлов

(например, Word или Excel), которые могут либо управлять несколькими собственными экземплярами, либо запускать новую копию каждый раз при создании объекта сервера.

Т.к. нам сервер нужен только для осуществления расчёта положений звеньев механизма, а весь интерфейс будет реализовывать приложение на Visual Basic, к тому же он будет запускаться только при запуске AutoCAD, то логично реализовать наш сервер как сервер процесса.

В результате выбран следующий тип ActiveX сервера – ActiveX сервер автоматизации как сервер процесса.

Microsoft Visual Studio предоставляет две библиотеки для реализации ActiveX сервера и COM-объекта – это MFC и ATL. Обе этих библиотеки обладают преимуществами и недостатками. Библиотека MFC представляет прекрасную иерархию классов, но в некоторых случаях пользоваться ею неудобно, например, при использовании серверов ActiveX в Web приходится дополнитель но загружать несколько DLL, которые могут оказаться ненужными. Если статически связать ActiveX сервер с MFC, то получается довольно большой файл. Библиотека ATL является прекрасным решением для создания небольших компонентов, потому что она не связана с использованием MFC. К недостаткам можно отнести отсутствие возможности использовать Class Wizard, но создавать ActiveX сервера автоматизации с помощью ATL очень просто.

Для создания ActiveX сервера воспользуемся библиотекой ATL. Для этого создадим новый проект ATL COM AppWizard, укажем тип сервера Dynamic Link Library, а так же отметим опции Allow merging of proxy/stub coded и Support MFC. Мастер ATL Object Wizard генерирует код, необходимый для создания ActiveX сервера, после чего надо добавить сам COM-объект и его методы и параметры. Выбрав меню Insert->New ATL Object и указав необходимые параметры, получаем код, генерированный ATL COM AppWizard, реализующий COM-объект: интерфейс и реализующий его класс. После этого добавляем необходимые методы.

Теперь осталось скомпилировать код, зарегистрировать сервер в операционной системе, после чего можно будет обращаться к ActiveX серверу из других программ и работать с COM-объектом. Обращаться к COM-объекту можно через псевдоним, который строится как ИмяМодуля.ИмяОбъекта[.номер_версии].

Технология COM-поддерживает следующий набор типов, которые можно использовать при взаимодействии с сервером автоматизации:

VARIANT_BOOL – Логический тип. Может иметь значение True (-1) или False (0).

BYTE – 8-битное целое число без знака.

int – Целое число со знаком. На 32-х битных платформах имеет размер 32 бита.

long – 32-битное целое число со знаком.

short – 16-битное целое число со знаком.

double – 64-битное число с плавающей точкой.

float – 32-битное число с плавающей точкой.

CURRENCY – 64-битное число, с фиксированной точкой.

DECIMAL – 96-битный (12 байтов) тип, предназначенный для хранения чисел со знаком и переменной плавающей точкой (от 0 до 28 разрядов).

DATE – 64-битное число с плавающей точкой, используемое для хранения даты.

IDispatch * – указатель на disp- или dual-интерфейс.

IUnknown * – указатель на IUnknown. Используется для передачи обычновенных интерфейсов (не disp- или dual-интерфейсов).

BSTR – basic-строка или бинарная строка, на Win32-платформах указатель на строку Unicode-символов. Переменная этого типа хранит не только саму строку, но и ее длину.

Это позволяет помещать в тело строки любые символы, в том числе неограниченное количество NULL-символов.

VARIANT – тип, позволяющий создать переменную, тип которой определяется во время выполнения программы и может изменяться с течением времени. Поддерживаются все типы, доступные для Automation, за исключением самого VARIANT'а. По существу, VARIANT – это структура, содержащая поле vt, определяющее тип, и поле, созданное на базе объединения, содержащего доступные типы данных. VARIANT позволяет хранить в себе даже структуру (VT_RECORD), массив (VT_ARRAY) или указатель на интерфейс. Использование массивов VARIANT'ов и массивов структур позволяет создавать модели данных любой сложности. Единственная проблема типа VARIANT – размер. VARIANT занимает 16 байт. Впрочем, по сравнению с типом данных Any (из CORBA), он просто малютка.

SAFEARRAY – так называемый безопасный массив. С его помощью можно создавать динамические много- и одномерные массивы. Как и в случае с типом VARIANT, в качестве типа ячейки SAFEARRAY'я можно использовать любой Automation-тип, включая сам SAFEARRAY. В отличие от типа VARIANT, SAFEARRAY не приводит к перерасходу оперативной памяти, даже когда хранит в себе структуры (тип VT_RECORD). При объявлении массивов типа SAFEARRAY в MIDL используется синтаксис – SAFEARRAY(elementtype) *arrayname. Здесь elementtype – это тип элемента массива. SAFEARRAY может быть многомерным, но все его размерности должны быть одного типа.

Наибольшая скорость обмена данными требуется при передаче новых положений звеньев механизма от COM-объекта к приложению на Visual Basic, для этого используется трёхмерный массив SAFEARRAY, в который заносятся координаты осей вращения, угол вращения и вектор перемещения звеньев механизма. В других методах, которые передают структуру механизма COM-объекту, используются простые типы: int и double.

Сам COM-объект решает следующие ключевые задачи: моделирование кинематической схемы механизма, автоматическое разбиение механизма на структурные группы, решение задачи о положениях для основных структурных групп. Для решения задачи о положениях в группах 6Г и ЗВС используется метод, разработанный В.Е.Турлаповым и обобщающий метод замкнутого векторного контура В.А. Зиновьева.

Сейчас система продолжает совершенствоваться: расширяется класс структурных групп, для которых задача о положениях может быть решена, расчёт различных кинематических параметров, добавляется возможность работы с платформами Стюарта.

Разработанная структура приложения обеспечивает высокую скорость расчёта положений звеньев механизмов в пространстве, что является её преимуществом перед использованием только интерпретируемых языков программирования, таких как Visual LISP и Visual Basic. Также система не зависит от каких-либо внешних библиотек, например, ObjectARX, и свободно работает с любой версией AutoCAD, в отличие от приложений, использующих ObjectARX. ActiveX сервер может использоваться не только для AutoCAD, а и в любых других программах, поддерживающих COM-технологию и способных работать в качестве COM-клиентов.

ПРОБЛЕМЫ ИНТЕГРАЦИИ РАЗНОРОДНЫХ CASE СРЕДСТВ В СИСТЕМУ АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ ИС

Е.А. Казакова

Пензенский госуниверситет

На всем цикле разработки ИС разработчик использует разнородные CASE-средства на определенных этапах для фиксирования необходимых данных.

CASE-средства предоставляют много преимуществ. Но автоматизируя таким образом работу, можно столкнуться с проблемой преобразования результатов анализа в формат этого CASE (если для формализации результатов анализа использовался другой CASE-инструмент или не использовался никакой). Некоторые CASE-средства позволяют непосредственно перейти к проектированию, а к анализу можно вернуться путем обратного проектирования. Но при этом некоторая информация может быть потеряна.

Вся информация о системе, собранная на этапе определения стратегии, формализуется и уточняется на этапе анализа. Особое внимание следует уделить полноте переданной информации, анализу информации на предмет отсутствия противоречий, а также поиску неиспользуемой вообще или дублирующейся информации. Как правило, заказчик не сразу формирует требования к системе в целом, а формулирует требования к отдельным ее компонентам. К достаточным результатам следует отнести диаграммы потоков данных и диаграммы жизненных циклов сущностей. Довольно часто ошибки анализа возникают при попытке показать жизненный цикл сущности на диаграмме ER.

ER-диаграммы используются для разработки данных и представляют собой стандартный способ определения данных и отношений между ними. ER-диаграмма содержит информацию о сущностях системы и способах их взаимодействия, включает идентификацию объектов, важных для предметной области (сущностей), свойств этих объектов (атрибутов) и их отношений с другими объектами (связей). Во многих случаях информационная модель очень сложна и содержит множество объектов.

DFD показывает внешние по отношению к системе источники и стоки (адресаты) данных, идентифицирует логические функции (процессы) и группы элементов данных, связывающие одну функцию с другой (потоки), а также идентифицирует хранилища (накопители) данных, к которым осуществляется доступ. Структуры потоков данных и определения их компонентов хранятся и анализируются в словаре данных. Каждая логическая функция (процесс) может быть детализирована с помощью DFD нижнего уровня; когда дальнейшая детализация перестает быть полезной, переходят к выражению логики функции при помощи спецификации процесса (мини-спецификации). Содержимое каждого хранилища также сохраняют в словаре данных, модель данных хранилища раскрывается с помощью ER-диаграмм.

Жизненный цикл сущности относится к классу STD-диаграмм. Эта диаграмма отражает изменение состояния объекта с течением времени. При проектировании возникает необходимость регистрировать все обсуждаемые варианты и окончательные решения. Не секрет, что проектировщики порой меняют первоначальные решения. Это может происходить потому, что со временем участники проекта забывают аргументы в пользу принятого решения. Подобную информацию можно хранить в репозитарии используемого CASE-средства, в текстовых файлах, просто на бумаге.

В табл. 1 приведены основные показатели самых известных CASE-средств.

Таблица 1

Сравнительный анализ популярных CASE-средств

Название	Фирма	Нотация DFD	Поведение	BPR	Функции	Данные	События
CASE. Аналитик	Эйтекс	Гейн–Сарсон	Упр. потоки и процессы	–	+	+	+
CASE/4/0	MicroTOOL	Йодан (расш.)	Уорд–Меллор (с STD)	–	+	+	+
Design/IDEF	Meta Software			+	+	+	–
Design/200	Oracle	Гейн–Сарсон		+	+	+	–
EasyCASE	Evergreen CASE Tools	Гейн–Сарсон, Йодан	Уорд–Меллор (с STD)	–	+	+	+
ERWin	Logic Work		STD	–	–	+	–
BPWin	Logic Work			+	+	–	–
S-Designer	Sybase/ Powersoft	Гейн–Сарсон, Йодан		–	+	+	–
SILVERRUN	CSA	произвольная	Упр. потоки и процессы	–	+	+	+
Rational Rose 3.0,	Rational Software	UML	Г. Буч, ОМТ	+	+	+	+

При проектировании системы автоматизированной информационной системы «Сельское Муниципальное Образование» возникла проблема фиксации результатов анализа, проектирования и кодирования разработчиками в разных форматах документов.

Как пример решения подобной проблемы, можно предложить проект системы, в которой будет накапливаться информация различных форматах, преобразовываться во внутрисистемное представление (метаданные). Кроме анализа форматов, используемых при разработке CASE-средств, можно в метаданные внести и дополнительную информацию, часто не фиксируемую разработчиками (например, функциональные зависимости, ограничения на атрибуты в схеме отношений и базе данных). При этом система должна быть расширяема и открытой, чтобы также учитывать результат использования новых CASE-средств. В настоящее время разрабатывается система для автоматизации проектирования базы данных, включающая в себя обработку файлов, полученных из программных продуктов AllFusion Process Modeler, AllFusion ERwin Data Modeler 4, Microsoft SQL Server, Microsoft Access.

РАЗРАБОТКА ПРОГРАММНОЙ ФАЙЛОВО-ОРИЕНТИРОВАННОЙ СИСТЕМЫ РЕЗЕРВНОГО КОПИРОВАНИЯ ДАННЫХ

В.Г. Казаков, С.А. Федосин

Мордовский госуниверситет им. Н. П. Огарева

Введение

На современном этапе, когда люди все больше информации доверяют своим компьютерам, а средства хранения данных дешевеют, все большую популярность приобретают системы резервного копирования. Вместе с быстрым ростом объемов хранимых данных, которые использует стандартные традиционные алгоритмы, возрастает сложность их защиты [1].

Проблема утери и искажения информации неизбежно связана с использованием компьютеров. Причинами являются ошибки программного обеспечения, неумелая работа пользователей, сбои физических носителей информации и средств связи, злонамеренная порча данных. К сожалению, это происходит настолько часто, что практически любой пользователь и организация сталкивалась с необходимостью реанимировать операционную систему, восстановить данные или корректную работу программ. В то же время требования к сохранности информации иногда очень высоки.

Обзор существующих алгоритмов

Каждый алгоритм резервного копирования делает компромисс между характеристиками процессов создания копий и операций восстановления данных, когда данные извлекаются из носителей резервных копий. Важными характеристиками являются такие, как, например, скорость выполнения копирования/восстановления, объем требуемой памяти для хранения резервных копий [1].

С одной стороны, алгоритмы, оптимизированные для быстрого создания резервных копий, например, алгоритмы инкрементного (e.g., Pure Incremental, Incremental Only) резервного копирования, которые копируют файлы, измененные со времени последней резервной копии. Здесь недостаток очевиден – для восстановления информации потребуется доступ к большим объемам данных, что помимо прочих возможных неудобств занимает много времени [2].

Другая крайность – алгоритмы полного резервного копирования (Full Backup) требует значительно большего объема памяти, а фаза резервного копирования занимает существенно больше времени [1].

Между этими двумя крайностями есть другие алгоритмы, позволяющие найти нужный баланс между важными характеристиками. К примеру, уровневые (Level Scheme), где резервное копирование заданного уровня включает все файлы, изменившиеся с момента последнего резервного копирования в соответствии с данным уровнем. Другой алгоритм, разработанный в UC Berkeley, адаптирован для *online*-режима работы системы резервного копирования [3].

Алгоритм «Z scheme» осуществляет параллельные операции резервного копирования в несколько потоков. «Z scheme» основан на том, что позволяет задать пользователю параметр b , называемый базой. Так, файл пишется потоком i , если был изменен b^i периодов (e.g., дней) назад [4].

Исследование существующих программных решений

Нами был проведен анализ репрезентативного ряда программных решений, распространяемых через Интернет посредством электронных каталогов (download.com, soft.mail.ru). Выяснилось, что данные программы используют два традиционных самых простейших алгоритма – полного резервного копирования, инкрементного и их аналоги. Как показано выше, данные традиционные подходы представляют собой две крайности и зачастую неэффективны для пользователя, ибо не дают достаточной гибкости для выбора оптимального для пользователя соотношения скорости резервного копирования, скорости восстановления, требуемого объема памяти [5]. По нашему мнению, программа, реализующая различные алгоритмические подходы, обладала бы несомненным конкурентным преимуществом среди аналогов.

Текущее состояние проекта

На данный момент нами создан прототип программной системы резервного копирования. Система реализована на языке C#, в среде Microsoft Visual Studio 2005, является полностью функциональной, и оснащена базовыми функциями современных систем резервного копирования.

Модульная структура

Реализованная система состоит из трех модулей: системного сервиса, библиотеки классов и программы-менеджера.

Программа-менеджер представляет собой исполняемый PE-файл и служит для предоставления графического пользовательского интерфейса для управления работой системы резервного копирования.

Библиотека классов представляет собой динамически подключаемую библиотеку, содержит общий код, используемый как в системном сервисе, так и в программе-менеджере. Содержит классы, отвечающие за выполнение таких функций, как работа с базой данных конфигурации, протоколирования работы программы в журнале регистрации, работа с zip-архивами. Системный сервис представляет собой исполняемый PE-файл, служит для выполнения операций резервного копирования в многопотковом режиме по расписанию в соответствии с предоставляемой программой-менеджером управляющей информацией.

Основные принципы функционирования системы

Система проектно ориентирована. Каждый создаваемый проект резервного копирования имеет набор опций таких как, например, список файлов, папок и расписание.

Пользователю предоставлена гибкая система создания расписаний, позволяющая создавать любые необходимые временные правила резервного копирования. Все проектные настройки сохраняются в xml-формате.

Системный сервис получает проектные настройки, анализирует их и реализует операции резервного копирования в многопотковом режиме в соответствии с расписанием. Причем данные сохраняются в упакованном виде в формате ZIP. Сервис ведет журнал регистрации, где отмечает все предпринятые операции с указанием времени результата и прочей полезной для пользователя информацией.

Направления дальнейшей работы

Планируется внедрить в созданную систему различные алгоритмы резервного копирования, которые позволят максимально покрыть потребности пользователя в выборе различных соотношений скорости резервного копирования, скорости восстановления, требуемого объема памяти.

Список литературы

1. Kurmas Z., Chervenak A. Evaluating backup algorithms, Proc. of the Eighth Goddard Conference on Mass Storage Systems and Technologies, 2000.
2. Chervenak A.L., Vellanki V., Kurmas Z. and Gupta V. Protecting File Systems: A Survey of Backup Techniques. In Proceedings. Joint NASA and IEEE Mass Storage Conference, 1998.
3. Costello A., Umans C., and Wu F. Online backup and restore. Unpublished Class Project at UC Berkeley, 1998.
4. Kurmas Z. Reasoning Behind the Scheme Z 2002. <http://www.cis.gvsu.edu/~kurmasz/Research/BackupResearch/rbzs.html>
5. Simpson D. Reader survey reveals backup-and-recovery trends. InfoStor 2005. http://infostor.com/articles/article_display.cfm?Section=ARTCL&C=Feat&ARTICLE_ID=244384

ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ДЛЯ ОПРЕДЕЛЕНИЯ И АНАЛИЗА ФУНКЦИОНАЛЬНЫХ ЗАВИСИМОСТЕЙ НА ЭТАПЕ ЛОГИЧЕСКОГО ПРОЕКТИРОВАНИЯ РБД

Е.А. Казакова, Д.П. Дзюба

Пензенский госуниверситет

Одним из этапов логического проектирования реляционных баз данных является нормализация отношений, то есть приведение отношений к необходимой нормальной форме. Этот этап позволяет устраниить аномалии добавления, удаления, обновления, свойственные реляционной модели. Данный этап достаточно сложный для разработчиков, так как опирается на формальную теорию функциональных зависимостей и нормализации. Как правило, этот процесс либо игнорируется разработчиками, либо ограничивается интуитивным приведением схемы отношений к третьей нормальной форме (3НФ). Третья нормальная форма достаточно часто является оптимальной по уровню нормализации. Но для определенных информационных систем может быть рекомендована как повышение уровня нормализации, так и его понижение. Это может зависеть от многих факторов. Например, можно проследить зависимость оптимального уровня нормализации от размера базы данных на предполагаемом объеме хранимой информации, от необходимости обеспечить целостность информации. Автоматизировать этот процесс достаточно сложно, о чем свидетельствует анализ CASE-средств. При определении уровня нормализации важную роль играют функциональные зависимости (ФЗ). На основе определенного проектировщиком набора ФЗ можно сделать вывод об уровне нормализации и/или выработать рекомендации об изменении уровня нормализации.

Выделять и учитывать ФЗ необходимо не только для определения уровня нормализации отношений, но и для определения бизнес-правил и ограничений в базе данных. Поэтому считаем актуальным разработку соответствующих средств автоматизации для построения ФЗ и приведения схемы базы данных к необходимому уровню нормализации. При этом нецелесообразно отказываться от использования существующих CASE-средств, а напротив, предполагается тесная интеграция с ними. В частности, планируется создание инструментальных средств для доступа к метаданным о ФЗ из сред разработки. Схема проекта приведена на рисунке.

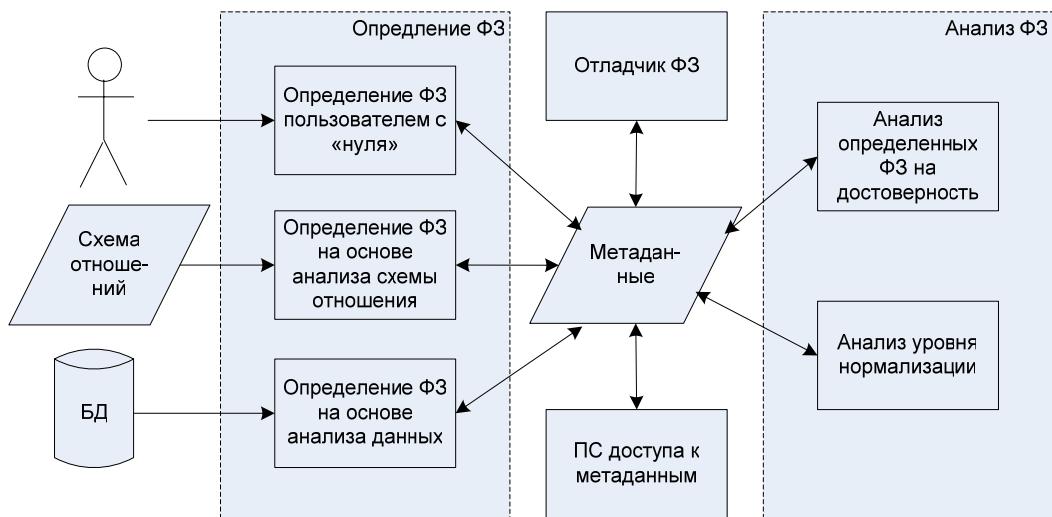


Рис. Схема проекта инструментальных средств

Определение ФЗ может осуществляться тремя способами:

- пользователем с «нуля»;
- на основе анализа схемы отношений;
- на основе анализа данных.

При определении ФЗ с «нуля» пользователь задает их на основе представления о предметной области, используя графический интерфейс и технологию Drag and Drop. В случае определения ФЗ на основе схемы отношения анализируется файл, сгенерированный средой ERWin или незаполненная база данных (например, SQL Server или MS Access). При анализе данных ФЗ выявляются на основе повторяющихся комбинаций значений атрибутов отношений. Определенные ФЗ хранятся внутри системы в качестве метаданных. Доступ к метаданным из сред разработки осуществляется проектировщиками через дополнительно разработанные программные средства. Кроме того, существуют приложения для проверки ФЗ и анализа уровня нормализации.

Для разработки программных средств используется среда Visual Studio .Net 2005, язык программирования C#, СУБД Microsoft SQL Server 2005. Данные продукты вместе предоставляют глубокие уровни интеграции между базой данных и средой разработки приложений. Для разработки и отладки объектов базы данных и сценариев применяются те же инструменты, которые используются для создания компонентов и служб .NET Framework промежуточного и клиентского уровня. Управляемые хранимые процедуры, триггеры, функции, пользовательские типы и пользовательские агрегаты кодируются на языке C#, используя платформу Framework 2.0. Язык C# предоставляет объектно-ориентированные конструкции, структурную обработку исключений, массивы, пространства имён и классы. Новые объекты базы данных могут развертываться непосредственно из среды Visual Studio без переключения в другие инструменты. Visual Studio 2005 непосредственно поддерживает все новые типы данных SQL Server.

Разрабатываемые инструментальные средства планируется применять для анализа баз данных, поддерживаемыми другими СУБД, в частности Microsoft Access.

ИНФОРМАЦИОННАЯ СИСТЕМА УЧЕТА ТЕКУЩЕЙ УСПЕВАЕМОСТИ СТУДЕНТОВ И ПОДГОТОВКИ ИТОГОВЫХ ДОКУМЕНТОВ

Е.А. Казакова, Ю.С. Барышева

Пензенский государственный университет

В связи с новыми требованиями к оформлению итоговых документов студентов, в частности листа-вкладыша в диплом с результатами обучения, появилась потребность в соответствующих средствах автоматизации процесса подготовки таких документов. Никаких коммерческих продуктов данного класса на сегодняшний день просто не существует. Поэтому каждое учебное заведение, факультет или кафедра должны обходиться своими силами. До недавнего времени на кафедре «Математическое обеспечение и применение ЭВМ» Пензенского государственного университета использовалась автоматизированная информационная система подготовки итоговых документов (листов-вкладышей в диплом), разработанная собственными силами с помощью MS Access 2000. Долгое время система успешно выполняла возложенные на нее функции. Однако она обладала серьезными ограничениями, которые, в конечном итоге, и определили необходимость разработки новой информационной системы. Объем хранимой информации очень большой, так как заносятся данные об оценках, полученных каждым студентом на протяжении всего курса обучения. Поэтому приходилось удалять сведения об итоговой успеваемости студентов каждый раз после того, как выходные документы для выпускного потока были подготовлены и вручены. Как следствие, повторная подготовка документов, например, при их утере или повреждении, была весьма хлопотным занятием. Постоянные изменения в формате выходных документов требовали перепроектирования отчетов MS Access, а часто изменение схемы базы данных и перепрограммирования модулей.

Таким образом, назрела необходимость разработки новой версии системы для сбора, обработки информации об успеваемости студентов и подготовки итоговых документов. Было решено сделать возможным применение разрабатываемой системы на других кафедрах и факультетах. Поэтому в базу данных системы добавлены сведения об институтах, факультетах, кафедрах и специальностях университета. Было решено использовать данную систему для текущего учета успеваемости студентов, анализа успеваемости, подготовки не только итоговых, но и промежуточных выходных документов (ведомостей, графиков, диаграмм и т.д.).

Обучение студентов, контроль успеваемости и подведение итого производится в соответствии с учебными планами. Поэтому в системе существует возможность создавать и редактировать учебные планы, перечни изучаемых дисциплин, а также виды обучения и отчетности.

В базе данных системы хранятся сведения о студентах: номер зачётной книжки, ФИО, пол, дата рождения, адрес, телефон, год поступления, предыдущий документ об образовании, номер группы. Причем хранятся сведения о студентах, которые в настоящий момент обучаются, а также сведения о бывших (уже закончивших обучение) студентах.

По мере сдачи студентом сессий, зачетов, экзаменов, курсовых работ и проектов в базе данных системы накапливаются сведения о результатах обучения студента. Во время обучения студент проходит три вида практики: учебная, производственная, преддип-

ломная. Каждая из практик имеет определенную продолжительность. В базе данных хранятся сведения о прохождении каждым студентом практик, срока прохождения и полученных оценках.

Кроме того, отдельно хранятся сведения о сданных государственных экзаменах и полученных на них оценках. Также хранится информация о выполненном дипломном проекте студента (тема, руководитель, дата защиты, дата выдачи, оценка) и выданном ему дипломе (регистрационный номер диплома, с отличием или без).

Основной задачей системы по-прежнему является формирование итоговых документов. Такой документ готовится по определенной форме и печатается на бланках государственного образца. Он содержит поля, содержащие сведения о студенте, специальности, дипломном проекте, дисциплинах, курсовых работах и проектах.

Кроме основного документа – диплома об окончании университета, система должна формировать отчеты разного назначения. Например, средний балл по группе и/или по потоку, по рейтинговой успеваемости, по студентам, не закрывшим сессию (задолжникам).

В системе выделены такие категории пользователей, как заведующий кафедрой/декан факультета, преподаватель, администратор, оператор, студент.

Оператор вносит информацию об успеваемости студентов по результатам сессии, после защиты отчета по практике, сдачи государственных экзаменов, защиты дипломов. Он также может производить изменения в учебном плане (список изучаемых дисциплин, виды занятий, количество часов по видам занятий), группах студентов (сведения о студентах). Кроме того, в задачи оператора входит создание, печать, импорт/экспорт промежуточных и итоговых документов.

Администратор системы имеет возможность создавать, редактировать, удалять учётные записи пользователей различных категорий. Каждая учётная запись имеет следующие атрибуты: имя пользователя, пароль входа в систему.

Преподаватель может просматривать успеваемость студентов, формировать темы курсовых работ и проектов по дисциплинам, которые он ведет. Для этого можно воспользоваться средствами непосредственного ввода данных, а также средствами импорта данных из документов MS Excel или MS Word, имеющих определенную структуру.

Студент имеет возможность просматривать внесённые о нём и его итогах данные. Это актуально перед заполнением дипломных листов, когда необходимо сверить содержащуюся информацию с оценками в зачетной книжке. Для студентов предполагается создание WEB-интерфейса. В локальной сети этот пользователь, просматривая свои данные, при нахождении ошибок помечает их. При достоверности отображаемой информации ставится отметка о просмотре и отсутствии несоответствий, после чего данные можно вносить в итоговые документы.

Помимо документов, определенных в системе, пользователю предоставляется возможность подготавливать документы свободной формы. Для этого разработан специальный редактор, который позволяет формировать документы из представленных компонентов, связанных с соответствующими полями базы данных информационной системы.

Информационная система учета текущей успеваемости студентов и подготовки итоговых документов является *сетевой и многопользовательской*. База данных системы располагается на сервере и доступ к ней можно получить с нескольких компьютеров. При больших объемах информации, заносимых при завершении обучения студентов, это несомненный плюс системы.

База данных разработана на языке SQL в среде MS SQL Server 2003. Пользовательское приложение закодировано на языке программирования C# в Visual Studio 2005.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ИССЛЕДОВАНИЯ УСЛОВИЙ ЖИЗНИ НАСЕЛЕНИЯ НА ПРИМЕРЕ ГОРОДА ЕЛАБУГА

И.С. Калганова, Е.С. Калганова

*Елабужский филиал Казанского государственного
технического университета им. А.Н. Туполева*

Объектом исследования является город Елабуга – город республиканского подчинения, центр Елабужского района. Площадь города – 41,1 кв.км. Население города насчитывает около 70000 человек. Город Елабуга – небольшой и тихий, имеющий богатую историю, известный в царской России и за ее пределами как торгово-купеческий – переживает свое новое рождение. В следующем году город отмечает свое тысячелетие.

Для исследования влияния экономических, социальных и экологических факторов на условия проживания жителей города использованы квартальные значения статистических данных за 1995–2005 гг. В качестве влияющих факторов были отобраны наиболее существенные, такие как расходные статьи бюджета города, численность населения, уровень безработицы, среднемесячная заработка, прожиточный минимум на члена семьи, стоимость продовольственной корзины, обеспеченность населения жильем, врачами, медицинским персоналом и больничными койками, общая раскрываемость преступлений, потребление чистой воды, выброс веществ в атмосферу, а также бытовые, промышленные и животноводческие отходы.

В качестве результативных показателей условий жизни населения выбраны: продолжительность жизни, рождаемость, смертность, естественный прирост, количество браков, число умерших детей в возрасте до 1 года, заболеваемость, преступность.

Оценка и анализ исходных статистических данных показали их пригодность для статистического исследования.

Было проведено прогнозирование результативных показателей условий жизни. По имеющимся квартальным данным за 11 лет был сделан прогноз на следующие 3,5 года. Прогноз показал основные тенденции в изменении результативных показателей условий жизни:

- малое увеличение продолжительности жизни и рождаемости;
- небольшое снижение смертности;
- снижение смертности детей в возрасте до 1 года и заболеваемости туберкулезом, онкологическими и сердечно-сосудистыми заболеваниями;
- стабилизация числа респираторных заболеваний;
- уровень преступности практически не изменяется.

Средняя продолжительность жизни – основной интегрированный показатель уровня жизни. По результатам прогноза, наблюдается очень малое его возрастание. С целью исследования тесноты связи между результативными показателями жизни и влияющими на них факторами был проведен корреляционный анализ. Если проанализировать результаты корреляционного анализа для средней продолжительности жизни, то можно сделать следующие выводы:

Большинство влияющих факторов имеют отрицательную корреляционную связь со средней продолжительностью жизни. Положительная связь определяется такими факторами, как обеспеченность населения врачами, медицинским персоналом, больничными койками, раскрываемостью преступлений, статьей бюджета «Национальная экономика».

В общем случае корреляционный анализ показал наличие связи между результативными показателями и влияющими бюджетными и экономико-социально-экологическими факторами, что дает возможность построения регрессионной модели условий жизни города.

Однако линейный регрессионный анализ не дал удовлетворительных результатов. Во-первых, по отношению стандартной ошибке к среднему лишь 3 уравнения из 27, что составляет 11%, удовлетворяют рекомендуемому значению 0,05. В-вторых, критерий Фишера достигает небольших значений. Для 18 уравнений значение критерия Фишера лежит в диапазоне от 0 до 4. Для 9 уравнений его значение превышает 10. Уровень значимости по критерию Фишера не превышает рекомендуемое значение 0,05 для 11 уравнений регрессии. В-третьих, очень малое количество факторов имеют удовлетворительное значение уровня значимости по критерию Стьюдента. Для 15 уравнений регрессии это количество равно нулю.

По большинству ограничений, наложенным на уравнения регрессии, не получены удовлетворительные значения для большинства уравнений.

Для достижения лучших результатов был проведен факторный анализ внешней среды и регрессионный анализ на основе факторного анализа. Основное предположение факторного анализа заключается в том, что корреляционные связи между большим числом наблюдаемых переменных факторов определяется существованием меньшего числа гипотетических ненаблюдаемых переменных или факторов. В результате было выделено 6 факторов:

F₁ – в данный фактор входят переменные: численность населения; численность населения трудоспособного возраста; объем промышленной продукции; среднемесячная заработка плата; прожиточный минимум на члена семьи; стоимость набора из 25 основных продуктов питания; обеспеченность населения общей площадью жилья на 1 жителя; ввод жилых домов; объем реализации платных услуг в расчете на 1 жителя; объем реализации бытовых услуг в расчете на 1 жителя; оборот розничной торговли на душу населения; оборот общественного питания на душу населения; общая раскрываемость преступлений. Назовем этот фактор «условия жизни населения».

F₂ – в данный фактор входят переменные: обеспеченность населения средним медицинским персоналом; количество автомобилей. Назовем этот фактор «обеспечение медицинским персоналом и автомобилями».

F₃ – в данный фактор входят переменные: численность работающих на крупных предприятиях; отходы промышленные. Назовем этот фактор «промышленность города».

F₄ – в данный фактор входят переменные: обеспеченность населения врачами. Назовем этот фактор «уменьшение обеспеченности населения врачами».

F₅ – в данный фактор входят переменные: отходы бытовые. Назовем этот фактор «уменьшение отходов бытовых».

F₆ – в данный фактор входят переменные: потребление чистой воды; выброс вредных веществ в атмосферу. Назовем этот фактор «экологическим».

На основе факторного анализа был проведен нелинейный регрессионный анализ. В результате получена регрессионная модель условий жизни города, представляющая собой совокупность из 27 уравнений регрессии

Проведена оценка степени влияния факторов на результативные показатели жизни населения по двум параметрам: по удельным весам и по коэффициентам эластичности.

Рассмотрим, как влияют на среднюю продолжительность жизни статьи бюджета:

- сильное влияние на повышение продолжительности жизни оказывают: статья бюджета «Жилищно-коммунальное хозяйство»; статья бюджета «Культура, кинематография и средства массовой информации»; статья бюджета «Здравоохранение. Спорт и физическая культура»; общий фактор «Обеспеченность населения медицинским персоналом и автомобилями»;
- сильное влияние на снижение продолжительности жизни оказывают: статья бюджета «Общегосударственные вопросы»; статья бюджета «Межбюджетные трансферты»; фактор «Промышленность города»; фактор «Снижение обеспеченности населения врачами».

На основе полученных зависимостей результативных показателей условий жизни от бюджетных факторов и выделенных общих скрытых факторов была проведена оптимизация.

В качестве целевой функции выбирается интегрированный показатель уровня жизни населения – средняя продолжительность жизни. Для остальных результативных показателей условий жизни ставятся ограничения на их значения, которые не должны выходить за рамки ранее достигнутых значений. Ограничения на бюджетные факторы также представляют собой минимальные и максимальные из ранее достигнутых значений. Кроме того, так как эти факторы представляют собой значения расходных статей бюджетов, то их сумма не должна превышать расходной части бюджета в заданном квартале. Общие факторы F отражают объективные не зависимые от человека экономико-социально-экологические показатели условий жизни города, поэтому при оптимизации их значения фиксированы и не изменяются.

При распределении статей бюджета, согласно результатам оптимизации, наблюдаются следующие тенденции:

- снижение расходов на общегосударственные вопросы;
- повышение затрат на национальную экономику и на здравоохранение, физкультуру, спорт;
- распределение остальных расходных статей бюджета существенно не изменилось.

По изменению результативных показателей получены следующие выводы:

- небольшое повышение рождаемости и естественного прироста;
- небольшое повышение заболеваемости системы кровообращения;
- небольшое снижение онкозаболеваемости;
- небольшое снижение числа особо тяжких, тяжких и преступлений средней тяжести, а также вымогательств;
- небольшое повышение числа причинений вреда здоровью.

Следует вывод, что при распределении расходных статей бюджета, согласно результатам оптимизации, наблюдается общее улучшение социальной ситуации в городе.

Таким образом, статистический анализ исходных данных, произведенный с использованием пакета прикладных программ Statistica, и оптимизация, произведенная с использованием табличного редактора Excel, позволили выработать эффективноеправленческое решение.

ПОСТРОЕНИЕ МОДЕЛЕЙ ПОЛУПРАВИЛЬНЫХ И СФЕРИЧЕСКИХ МНОГОГРАННИКОВ С ИСПОЛЬЗОВАНИЕМ САПР AUTOCAD

В.А. Кирьянов

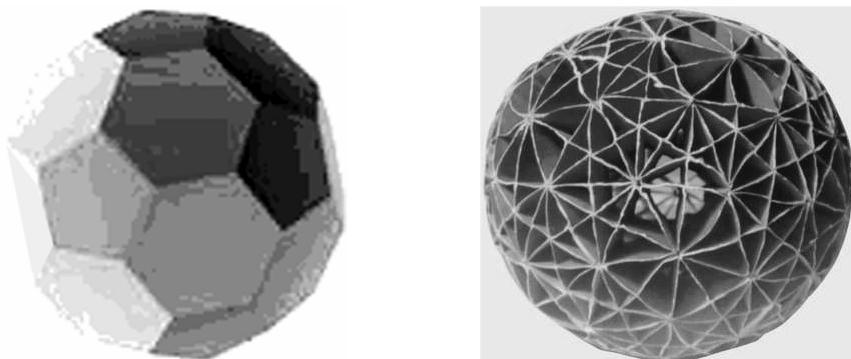
*Елабужский филиал Казанского государственного технического
университета им. А. Н. Туполева*

Целью данной работы является построение моделей многогранников в САПР AutoCAD, а также создание реальных бумажных и иных моделей, основывающихся на структуре многогранника.

Моделирование объектов правильной формы применяется во многих областях строительства, архитектуры, машиностроения. Правильные тела отличаются простотой, в их симметрии заключается красота, которая восхищала математиков еще в Древней Греции. Наиболее известны Платоновы и Архимедовы тела, или, иначе говоря, правильные и полуправильные многогранники. В одной из наиболее современных областей геометрии – сферической геометрии – существуют их аналоги: сферические многогранники.

Тем не менее довольно сложно представлять такие фигуры без предварительной теоретической подготовки. В качестве стандартных примитивов в программах трехмерного моделирования часто присутствуют пять Платоновых тел, но остальные многогранники необходимо создавать. Основной принцип создания сферических многогранников – поворот плоского цилиндра малой высоты относительно центра на рассчитываемый градус.

Приведем два примера данной работы: один в электронном варианте, другой – реальный, созданный из бумаги на базе компьютерной модели.



Для создания демонстраций использовались программы AutoCAD, 3DStudioMax. Были проведены теоретические и практические расчеты (в системе автоматизированного проектирования), в результате которых были получены распечатки шаблонов деталей для сборки реальной модели.

В перспективе планируется перенести модели в пакет Autodesk Inventor Series 11, что позволит упростить создание фотoreалистических изображений и анимационных роликов, оптимизировать работу с моделью, а также использовать трехмерный принтер для создания реальной модели.

РАЗРАБОТКА КОМПЛЕКСА ПРОГРАММ В СРЕДЕ VB.NET ДЛЯ СОГЛАСОВАНИЯ РАСПРЕДЕЛЕННОЙ БАЗЫ ДАННЫХ ИС ВУЗА

А.И. Кондакова, Е.А. Шустрова, Е.Ю. Малышева

Тольяттинский госуниверситет сервиса

Постановка задачи

На сегодняшний день Тольяттинский госуниверситет сервиса имеет несколько представительств в Самарской области: в Жигулевске, Кинели и других населенных пунктах. Наличие представительств и филиалов предполагает ввод данных в ИС вуза из территориально удаленных подразделений. Вся информация о студентах и их успеваемости из представительств должна быть внесена и обработана в ИС вуза.

Были рассмотрены два варианта решения данной проблемы. Первый – с использованием технологии ASP.Net. Данный вариант передачи данных сегодня успешно используется для подсистемы «Абитуриенты». Однако этот вариант для некоторых подразделений не может быть использован, т.к. удаленные подразделения вуза не всегда имеют устойчивый и высокоскоростной доступ в Internet.

Второй вариант – с использованием локальных баз данных в филиалах и представительствах и передачей и согласованием данных с использованием технологии VB.Net. Такой вариант больше подходит для повседневной работы, не требует устойчивого доступа в Internet, для этого варианта проще обеспечить безопасность данных.

Разработка комплекса программ для согласования распределенной базы данных ИС вуза также позволит осуществлять ввод данных за прошлые периоды или по отдельным тематикам, не затрагивая основную БД. Пользователи могут вносить данные в специальную рабочую БД, а затем администратор ИС осуществляет перенос данных в БД ИС вуза. Таким образом, данные в основной БД ИС вуза не пострадают даже при ошибках пользователя.

Краткое описание разработки комплекса программ для согласования распределенной базы данных ИС вуза

В качестве среды программирования использован программный продукт Microsoft Visual Studio. База данных разработана с помощью Microsoft Access.

Ниже на рис. 1 приведена архитектура комплекса программ для согласования распределенной БД ИС вуза (и его подразделений).

На основании БД ИС вуза формируется БД «эталон», которая содержит данные справочников БД ИС вуза и является основой рабочей БД. Формирование БД «Эталон» осуществляется администратором с помощью SQL – запросов без участия пользователя. В блоке «Работа с операционными таблицами» вводится текущая информация. В блоке «Перенос данных операционных таблиц БД ИС вуза» реализован обмен данными между рабочей БД и БД ИС вуза. Данная архитектура была успешно использована для ввода информации по успеваемости студентов за прошлые годы и данных по успеваемости студентов представительств вуза.

При создании данного комплекса программ основной проблемой является то, что необходимо учесть идентификаторы справочных и операционных таблиц. Эта проблема

решается параллельным учетом идентификаторов в модуле «Перенос данных операционных таблиц БД ИС вуза»

Основным инструментом при создании комплекса программ является применение параметрических SQL – запросов. С их помощью осуществляется сопоставление данных БД вуза и рабочей БД, ввод данных, а также непосредственно перенос (копирование) данных.

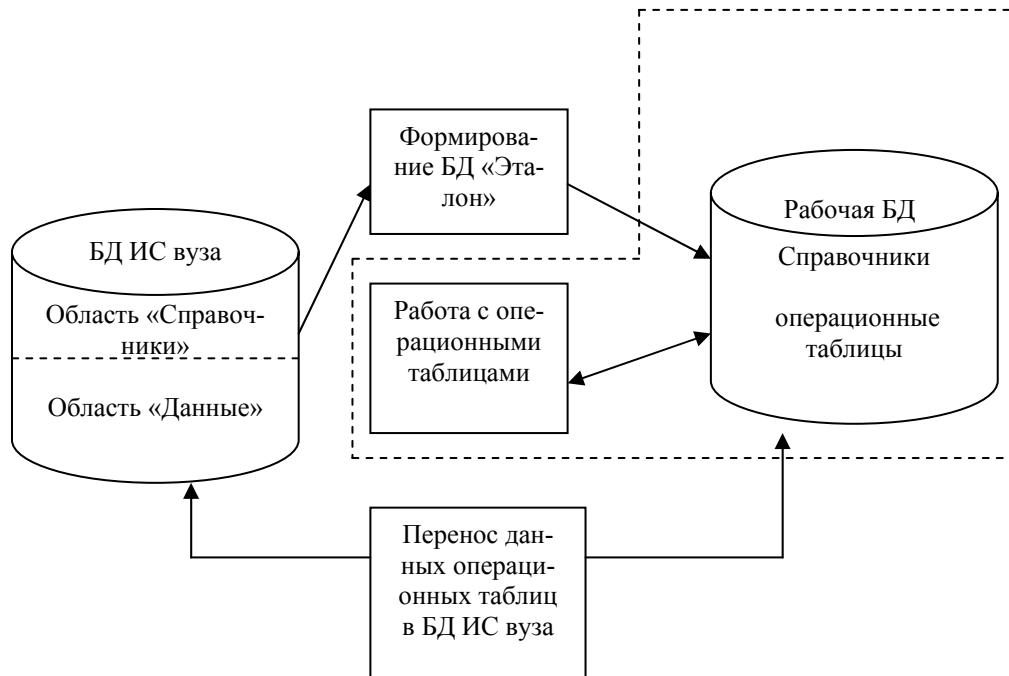


Рис. 1. Архитектура комплекса программ для согласования распределенной БД

На рис. 2 представлен фрагмент структуры БД ИС «Деканат» – части БД ИС вуза. Это область учет текущей успеваемости. Это реляционная БД, таблицы связаны между собой связями вида «один-ко-многим».

Комплекс программ состоит из двух модулей:

«Клиент деканат» – данный модуль предназначен для ввода и редактирования данных в рабочей БД. С модулем работают сотрудники филиалов, а также пользователи, занятые вводом данных об успеваемости студентов за прошлые годы. Поэтому данный программный модуль имеет удобный и понятный интерфейс.

«Копия с клиента» – данный модуль осуществляет передачу данных в БД ИС вуза. Пользователь выбирает нужную группу, рабочий план, дисциплину и ведомость в левой части окна, соответствующей рабочей БД, затем выбирает из списка в правой части окна (БД ИС вуза) ту же группу, рабочий план, дисциплину и ведомость.

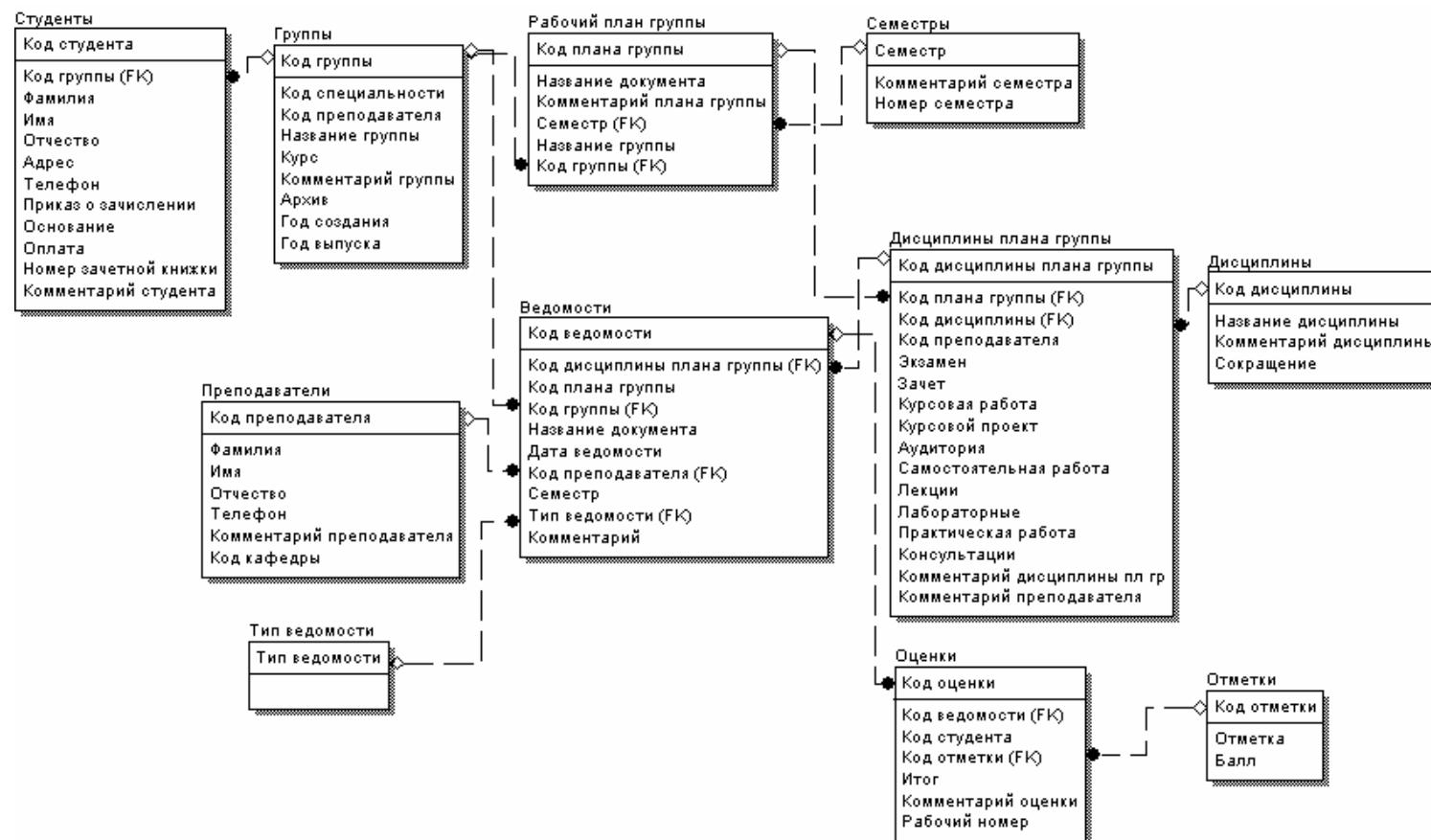


Рис. 2. Фрагмент структуры БД ИС «Деканат»

Перед переносом данных пользователю предлагается еще раз убедиться, совпадают ли группы (рабочие планы, дисциплины, ведомости), для которых будет произведен перенос данных. В данном модуле возможно осуществить как перенос данных отдельной ведомости, так и данных по всем дисциплинам и рабочим планам группы.

Заключение

Разработав комплекс программ для согласования распределенной базы данных ИС вуза, мы фактически решили две актуальные для нашего вуза проблемы:

- Предоставили сотрудникам представительств самостоятельно работать с данными о студентах и их успеваемости на местах, а также переносить эти данные в БД ИС вуза.
- Вводить данные за прошлые периоды и по конкретным тематикам, не затрагивая непосредственно при этом БД ИС вуза, что отвечает требованиям безопасности данных.

Таким образом, подход, используемый при разработке комплекса программ может найти применение в рамках любой задачи, требующей согласования распределенной БД.

Список литературы

1. Автоматизация управления предприятием / Баронов В.В. и др. – М.: ИНФРА-М, 2000. – 239 с.
2. Вендрев А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000.
3. Проектирование экономических информационных систем: Учебник / Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов; Под ред. Ю.Ф. Тельнова. – М.: Финансы и статистика, 2002.

АВТОМАТИЗАЦИЯ АДМИНИСТРИРОВАНИЯ ДАННЫХ ЭКОЛОГИЧЕСКОГО НАДЗОРА ПЕНЗЕНСКОЙ ОБЛАСТИ

Н.А. Коннова, Л.В. Гурьянов

Пензенский госуниверситет

Организации, занимающиеся экологическим надзором, часто сталкиваются с проблемой обработки данных разнотипных документов, полученных из налоговой инспекции, казначейства и других организаций. Большое число документов разного типа усложняет работу сотрудников и снижает эффективность системы управления экологическим надзором.

Для того чтобы упростить и повысить скорость обработки экологических данных, следует обобщить и структурировать имеющиеся документы. Решением этой задачи является консолидация экологических данных.

Результатом разработки является система автоматизации администрирования данных экологического надзора.

Разработка системы осуществлялась в два этапа. Первый – формирование единой базы данных. Второй – разработка программных средств ведения базы данных и предоставления данных и отчетов всем заинтересованным клиентам.

Формирование единой базы данных

Основной проблемой формирования базы данных является то, что многие документы являются уникальным в плане формата данных. Существуют документы, представленные только в бумажном виде, которые пришлось преобразовать к электронному виду с помощью сканера.

Для переноса информации были разработаны программы-конвертеры, которые занимаются разбором исходных документов, получением данных и занесением их в базу данных. Разбор максимально учитывает специфику каждого документа, такую, как тип документа, формат документа, возможные опечатки или некорректность данных.

Важнейшей характеристикой такой базы данных является целостность. Различают физическую и логическую целостность. Физическая целостность означает наличие физического доступа к данным и то, что данные не утрачены. Логическая целостность означает отсутствие логических ошибок в базе данных.

В дальнейшем речь будет вести о логической целостности. Поддержание целостности базы данных включает проверку (контроль) целостности и ее восстановление в случае обнаружения противоречий в базе данных. Целостное состояние базы данных задается с помощью ограничений целостности в виде условий, которым должны удовлетворять хранимые в базе данные. Если при разборе документов обрабатываемые данные нарушают эти условия, выдается сообщение об ошибке. Затем спорные и ошибочные данные уточняются в присутствии администратора базы данных и лиц, ответственных за создание ошибочных или противоречивых документов. Корректные данные заносятся в базу данных.

Разработка программных средств

Разработка программных средств велась на основе клиент-серверной технологии с использованием методов доступа ADO к базе данных.

Структура системы администрирования данных экологического надзора области приведена на рисунке.

Основными функциями серверной части программных средств являются: операции работы с базой данных (просмотр данных, добавление данных с помощью конвертеров, удаление и обновление данных) и формирование отчетных документов. Вид отчетов унифицирован. Программа максимально защищает отчеты от опечаток, ошибок. Для удобства работы поддерживается три формата отчетов rtf, xls, xml.

Более простой формат rtf используется, когда число записей в отчете заранее известно, фиксировано. После создания отчета в таком формате возможна его печать или сохранение в файле и в дальнейшем этот файл может использоваться для последующего редактирования в редакторе Microsoft Word.

Формат xls используется, когда число строк в отчете заранее неизвестно, в дальнейшем может использоваться для каких-либо дополнительных вычислений в редакторе Microsoft Excel.

Формат xml также используется, когда число записей в отчете неизвестно и позволяет создавать отчеты, вид которых может быть легко изменен с помощью таблицы стилей. Еще такие отчеты могут быть размещены на страницах Web-сайта данной организации.

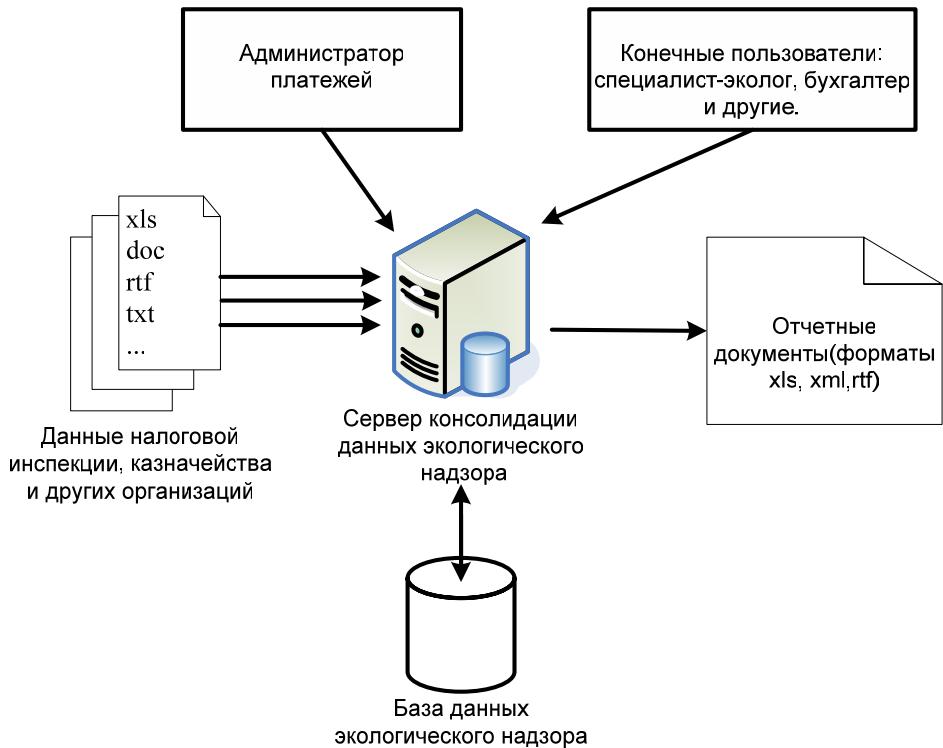


Рис. Система управления экологическим надзором

В разработанной системе осуществляется авторизация пользователей в зависимости от их категории. Выделены следующие категории пользователей:

1. Специалист-эколог – имеет возможность просматривать данные о плательщиках и их платежах за нанесенный ими экологический вред, формировать план/факт за текущий год.
2. Бухгалтер – имеет возможность просматривать данные о плательщиках и их платежах за нанесенный ими экологический вред, создавать отчеты, например, такие, как акт сверки, лицевой счет.
3. Администратор платежей – имеет возможность просматривать данные о плательщиках и их платежах за нанесенный ими экологический вред. Также может осуществлять корректировку данных, например добавление, удаление информации.
4. Администратор базы данных – осуществляет корректировку и обновление существующей базы данных.

В настоящее время завершаются работы по верификации созданной базы данных. Разработанные программные средства проходят комплексное тестирование в организации экологического надзора г. Пензы.

КОНСТРУКТОР ВЕБ-СТРАНИЦ ДОСТУПА К СТРУКТУРИРОВАННЫМ ДАННЫМ

В.М. Конюхова

*Елабужский филиал Казанского государственного
технического университета им. А.Н. Туполева*

Целью данной работы является создание приложения, позволяющего в полуавтоматическом режиме проектировать и получать программный код веб-интерфейса для доступа к структурированным данным в сети Интернет. В качестве структурированных данных могут выступать системы управления базами данных, структурированные бинарные файлы.

Большинство небольших сайтов в Российском сегменте сети Интернет на данный момент представляет собой более или менее хорошо оформленный интерфейс для доступа к базе данных, либо полностью статическую информацию, которая достаточно редко подвергается изменениям. Базы данных обычно используются для хранения новостей, досок объявлений, каталогов, гостевых книг, афиш мероприятий, прайс-листов, описаний товаров.

Рассмотрев ситуацию на рынке предложений создания сайтов, проектирования баз данных, систем управления содержимым сайта становится очевидным, что существует множество как бесплатных, так и коммерческих продуктов, основная цель которых – позволить пользователю, не имея никаких знаний в области программирования, создать сайт на основе шаблона.

Тем не менее для программистов и для опытных пользователей часто возникает задача создания нестандартных страниц доступа к данным, нестандартных форм добавления данных. Причем в качестве данных может выступать и содержимое одной таблицы, и содержащая множество связей сложная база данных.

Приложение написано на языке PHP и представляет собой веб-интерфейс, в котором пользователь пошагово указывает исходную информацию, свои предпочтения в дизайне страниц, параметры СУБД или другого источника данных, параметры и представление используемых для отображения на странице структурированных данных. Программа анализирует пожелания пользователя и на основе шаблонов составляет PHP-код результирующей страницы. Если это требуется, то создается архив с результирующими файлами, включающий в себя, кроме основной веб-страницы, дизайн (таблицу стилей, картинки), дополнительные файлы проверки правильности заполнения форм, java-скрипты.

Использование языка PHP предоставляет пользователю широкие возможности по переносимости на различные платформы, хотя не дает должного уровня надежности и защиты. В связи с этим в перспективе планируется перенести приложение на платформу .NET, а также добавить возможность использования языка разметки xml вместо html и совместимости с другими типами баз данных.

Стоит отметить, что Microsoft Access предоставляет возможность экспортieren данные из локальных баз данных предприятия на различные сайты, но не обеспечивает должной гибкости и свободы для программиста. Другая программа, решающая аналогичные задачи – это Macromedia Dreamweaver. Она включает в себя небольшой набор стандартных инструментов с использованием языка PHP для установления соединения

с базой данных и отображениями данных. Уникальность разрабатываемой программы состоит в доступности (веб-интерфейс может быть размещен в сети Интернет), сочетании большого количества настроек и применения шаблонов, в том числе и собственных шаблонов, которые может добавить программист.

Созданное приложение предоставляет пользователю достаточно настроек и возможностей для легкого и удобного конструирования структур доступа к произвольным структурированным данным и предоставляет широкие возможности по расширению своей функциональности.

АНАЛИЗ СЛОЖНОСТИ АЛГОРИТМА РАЗМЕЩЕНИЯ ЭЛЕМЕНТОВ СБИС ИНСТРУМЕНТА *ITLDRAGON*

К.В. Корняков, Н.В. Курина, И.Б. Мееров

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время задача размещения элементов сверхбольших интегральных схем (СБИС) весьма актуальна. Задача принадлежит к классу NP-трудных (NP-трудным является ее частный случай – задача определения оптимального порядка следования элементов на отрезке [1]). Огромные размерности современных задач (сотни тысяч элементов) приводят к тому, что использование точных алгоритмов становится неприемлемым с точки зрения времени работы алгоритма. Это обуславливает развитие эвристических методов решения, позволяющих за полиномиальное время получать размещения достаточно высокого качества.

К настоящему моменту сложилось несколько основных подходов к решению задачи размещения, и существует достаточно большое число инструментов размещения, реализующих эти подходы [2].

В данной работе приводится анализ сложности алгоритма размещения инструмента *itlDragon* (его структура совпадает со структурой алгоритма инструмента *Dragon* [3]), разрабатываемого в Нижегородском государственном университете. Исследуется зависимость времени работы инструмента от числа размещаемых элементов.

Постановка задачи

Инструмент *itlDragon* ориентирован на решение задачи размещения стандартных элементов СБИС, в качестве минимизируемого критерия рассматривается суммарная длина соединений между стандартными элементами (wirelength-driven standard cell placement).

Исходная информация о размещаемой схеме может быть представлена в виде четверки $C = (Area, Cells, Pins, Nets)$:

- Area – область для размещения. $Area = \{1, \dots, R\} \times \{1, \dots, S\}$, где R – число рядов, S – число сайтов в каждом ряду.

- Cells – множество элементов. Для каждого элемента известна его длина и высота. В настоящей постановке задачи все элементы являются стандартными, то есть имеют одинаковую высоту. Число элементов обозначим N.
- Pins – множество контактов. Для каждого контакта известно, на каком из элементов он находится, а также смещение контакта относительно центра данного элемента.
- Nets – список соединений. Для каждого соединения известен список соединяемых им контактов. Число соединений обозначим M.

Размещение можно записать в виде:

$$P = \left\{ (x_i, y_i) : i = 1, N, x_i \in S, y_i \in R \right\}.$$

Задача заключается в нахождении размещения P , обеспечивающего минимальное значение суммарной длины соединений. В качестве оценки длины каждого соединения принимается полупериметр прямоугольника, охватывающего все его контакты. Обозначим эту величину для i -го соединения $W_i(P)$. Тогда минимизируемый критерий записывается в виде:

$$\min_{P \in P} W(P) = \min_{P \in P} \sum_{i=1}^N W_i(P).$$

Алгоритм размещения инструмента *itlDragon*

Алгоритм размещения состоит из нескольких стадий. Мы не будем подробно останавливаться на каждой из них, а лишь охарактеризуем алгоритм в целом. Подробное описание используемого алгоритма может быть найдено в [3,4]. Блок-схема алгоритма представлена на рис. 1. Под каждой из стадий схематично представлено соответствующее этапу размещения расположение элементов на плате.

Условно алгоритм делится на два крупных этапа: глобальное и детальное размещение. Во время глобального размещения элементы располагаются в узлах прямоугольной сетки целями кластерами, перекрывая друг друга. Целью работы на данном этапе является определение предварительного местоположения для каждого из элементов. Детальное размещение включает процедуры легализации и финального улучшения размещения путем локальных преобразований.

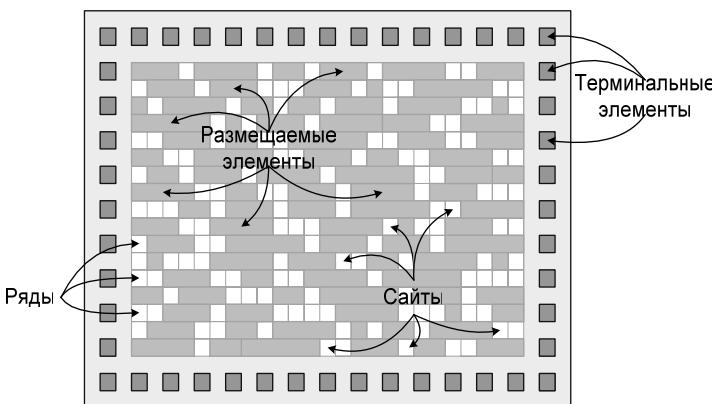


Рис. 1. Проектируемая интегральная схема

локализации для каждого из элементов. Детальное размещение включает процедуры легализации и финального улучшения размещения путем локальных преобразований.

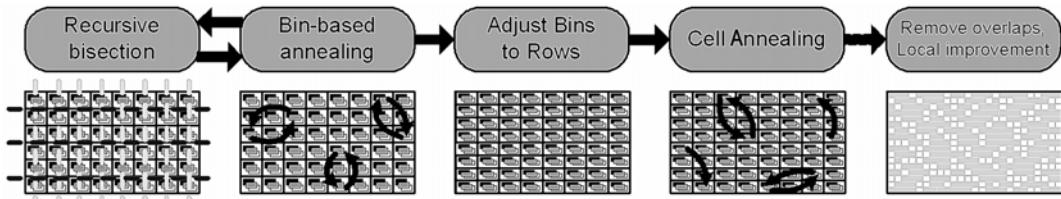


Рис. 2. Алгоритмическая схема инструмента *itlDragon*

Анализ сложности алгоритма

В результате проведения вычислительных экспериментов было установлено, что при анализе сложности алгоритма размещения особое внимание необходимо уделить процедуре *Cell Annealing*. Инструмент размещения показывает наилучшие результаты в случае, если *Cell Annealing* использует порядка 80% от общего времени работы. Таким образом, можно считать, что именно эта процедура определяет суммарное время работы инструмента, поэтому особенно важно изучить зависимость времени ее работы от числа элементов.

Далее мы сконцентрируем наше внимание на анализе сложности *Cell Annealing*. Сразу заметим, что мы опустим некоторые из деталей процедуры, несущественные с точки зрения анализа сложности.

Процедура *Cell Annealing* основана на методе моделирования отжига, и применяется для определения оптимального участка расположения для индивидуальных элементов. Это итеративная процедура, осуществляющая два типа перемещений:

Выбор произвольного элемента и перемещение его на выбранный случайным образом участок.

Обмен двух случайно выбранных элементов.

Заметим, что сложность перемещений второго типа вдвое выше, чем первого. Для простоты будем считать, что мы используем только преобразования второго типа и далее подсчитаем их общее количество *totalNumberOfMoves*. Это число мы примем в итоге в качестве оценки сложности алгоритма.

Исследуем сложность одного перемещения. После осуществления преобразования необходимо заново вычислить длину тех соединений, в которые входят перемещенные элементы. То есть трудоемкость операции зависит от числа соединений, в которые включены элементы, и не зависит от общего числа размещаемых элементов. Среднее число соединений, в которые входит каждый элемент, зависит от типа создаваемой схемы, и его можно считать константой.

Определим число перемещений на каждой итерации *Cell Annealing*. Это число зависит от числа участков, на которые поделена плоскость размещения. А именно, в *itlDragon* используется следующая формула:

$$\text{numberOfMoves} = \text{numTries} \times \text{numberOfBins},$$

где *numberOfBins* – число участков на плате, а *numTries* = 300. Его можно определить из следующих соображений: процедура рекурсивной бисекции завершается, когда среднее число элементов в каждом кластере становится меньше 10. Таким образом,

$$\text{numberOfBins} \leq N / 10.$$

Отсюда число перемещений на каждой итерации можно оценить следующим образом:

$$number\text{Of}Moves \leq 30 \times N.$$

Далее определим число итераций процедуры *Cell Annealing*. В методе моделирования отжига время работы алгоритма регулируется параметром, называемым температурой (идея метода пришла из физики). А именно, первоначально температура устанавливается на некотором уровне, затем на каждой итерации умножается на коэффициент, меньший единицы. Метод прекращает работу, когда температура снижается ниже определенного уровня. Обозначим начальный уровень температуры как INIT_TEMP, конечный как FINAL_TEMP, а коэффициент понижения – DUMP_COEFF. Общее число итераций алгоритма можно вычислить из следующего уравнения:

$$INIT_TEMP \times DUMP_COEFF^{number\text{Of}Iterations} = FINAL_TEMP.$$

Откуда, логарифмируя, получаем:

$$number\text{Of}Iterations = \log_{DUMP_COEFF} (FINAL_TEMP / INIT_TEMP).$$

Следует сказать, что в инструменте *itlDragon* используется адаптивный выбор начального и конечного уровня температуры. Сделано это прежде всего из соображений экономии времени работы. Дело в том, что если температура слишком высока, то *Cell Annealing* становится алгоритмом ненаправленного случайного поиска, который может резко ухудшить качество размещения. При низкой же температуре метод моделирования отжига вырождается в жадный алгоритм, который быстро попадает в локальный минимум и уже не выходит из него.

По этим причинам мы используем адаптивный подбор начального и конечного уровня температуры. При этом, однако, мы допускаем, чтобы эти значения находились в пределах некоторого отрезка [MIN_TEMP; MAX_TEMP]. Мы используем следующие значения: MIN_TEMP = 0,1 и MAX_TEMP = 1000. Следует сказать, что во время экспериментов на бенчмарках с числом элементов от 12 000 до 200 000, температура всегда оставалась в этих пределах. Таким образом, используя их в своих выкладках сейчас, мы дадим верхнюю оценку числа итераций *Cell Annealing*.

$$number\text{Of}Iterations = \log_{0,95} (0,1 / 1000) \approx 180.$$

Итак, мы можем оценить сверху величину *totalNumberOfMoves*:

$$totalNumberOfMoves = number\text{Of}Iterations \times number\text{Of}Moves = 180 \times 30 \times N = O(N).$$

Итак, мы получили линейную оценку времени работы алгоритма от числа размещаемых элементов. Это весьма оптимистичный прогноз, дающий основания думать, что время работы инструмента будет расти линейно с числом элементов. Однако следует упомянуть о некоторых возможных опасностях. Дело в том, что такие коэффициенты, как *numTries*, MAX_TEMP и MIN_TEMP, были подобраны опытным путем для конкретного набора бенчмарков. Они были выбраны с расчетом, чтобы алгоритм успевал сходиться, то есть процедура *Cell Annealing* продолжает работать до того момента, когда перестает давать ощутимый прирост качества.

Таким образом, выбранные нами значения для упомянутых параметров гарантируют линейное время работы, но могут не обеспечить сходимости алгоритма на крупных бенчмарках. Другими словами, они могут привести к завершению алгоритма раньше,

чем он сойдется к некоторому минимуму. Подобное прерывание может негативно отразиться на качестве финального размещения.

В связи с этим может потребоваться некоторое изменение рассматриваемых параметров, которое может повлиять на время размещения. Однако результаты наших экспериментов дают основания полагать, что слишком сильного изменения параметров происходит не будет, и что зависимость времени работы инструмента от числа элементов будет сохраняться линейной или близкой к ней.

Заключение

В данной работе была исследована зависимость времени работы инструмента *itlDragon* от общего числа размещаемых элементов. Было установлено, что время работы будет расти линейно с числом размещаемых элементов. Это весьма обнадеживающий результат, дающий основание полагать, что алгоритм инструмента *itlDragon* можно будет применять для размещения параметров с большим числом элементов.

Список литературы

1. Garey, M.R., Johnson, D.S., Stockmeyer, L. Some simplified NP-complete graph problems // Theor. Comp. Sci. 1 (1976), 237-267.
2. Adya S.N., Yildiz M., Markov I.L., Villarrubia P.G., Parakh P.N., and Madden P.H. Benchmarking for large-scale placement and beyond // Proc. Int. Symp. Physical Design, 2003. P. 95–103.
3. Wang M., Yang X., Sarrafzadeh M. Dragon2000: Standard-cell placement tool for large industry circuits // In International Conference on Computer-Aided Design. P. 260-263. IEEE 2000.
4. Гагаринова С.А., Живодеров А.В., Корняков К.В., Курина Н.В., Meerov И.Б. Об одном подходе к решению задачи о размещении // Технологии Microsoft в теории и практике программирования. Материалы конференции / Под ред. проф. Р.Г. Стронгина. – Нижний Новгород: Изд-во Нижегородского госуниверситета, 2006. – С. 215-219.

ИНТЕГРАЦИЯ MICROSOFT COMPUTE CLUSTER SERVER 2003 С СИСТЕМОЙ УПРАВЛЕНИЯ КЛАСТЕРАМИ «МЕТАКЛАСТЕР»

К.В. Корняков, А.В. Сенин, А.В. Шишков

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Для проведения широкого круга научно-исследовательских экспериментов и практических занятий студентов и научных сотрудников в распоряжении Нижегородского государственного университета (ННГУ) имеются мощные вычислительные ресурсы: несколько кластеров разной производительности, а также ряд компьютерных лабораторий. Для эффективного использования вычислительных ресурсов, в целях упрощения контроля и администрирования необходимо объединить все имеющиеся мощности под управлением единой системы управления кластерами.

В рамках проекта «Метакластер» лаборатории «Информационные технологии» идет разработка системы, которая позволит не только интегрировать вычислительные

ресурсы университета вместе, но и сохранить на локальных кластерах уже действующие системы управления сторонних разработчиков в том случае, если пользователи кластера не захотят менять привычную среду работы.

Важное направление работы проекта – интеграция с Microsoft Compute Cluster Server 2003 (CCS). Данная система управления используется на крупнейшем кластере ННГУ. В настоящей работе рассматривается вопрос использования специального программного интерфейса (API), предоставляемого CCS, для интеграции системы управления кластером от Microsoft с системой «Метакластер».

API взаимодействия с системой CCS

CCS предоставляет сразу несколько интерфейсов доступа: графический интерфейс, интерфейс командной строки и API. Пользовательское приложение может использовать API через динамическую библиотеку `srrapi.dll`, содержащуюся в поставляемом с CCS SDK. В данной библиотеке содержатся СОМ-интерфейсы, предоставляющие методы работы с CCS. Основными являются следующие интерфейсы: `ICluster`, `INode`, `IJob`, `ITask`. Подробная информация о каждом из них может быть найдена в документации по CCS, мы же рассмотрим основные возможности, предоставляемые данными интерфейсами. Их можно условно разделить на два класса: управление заданиями и мониторинг загрузки машин кластера.

Функциональность, относящаяся к первому классу, позволяет совершать над вычислительными задачами следующие операции: постановка в очередь планировщика, мониторинг состояния задачи, снятие задачи с выполнения, повторный запуск и некоторые другие. Среди возможностей, относящихся ко второму классу, можно упомянуть такие функции, как запрос числа запущенных задач, списка доступных машин, характеристик и загрузки каждой из них, количества ресурсов, потребленных конкретной задачей.

API удобно использовать в следующих ситуациях:

Посылка задач на удаленный кластер из собственных программ.

Создание систем мониторинга (генерация отчетов об использовании вычислительных ресурсов).

Реализация собственного интерфейса доступа к системе CCS.

Интеграция с системой управления «Метакластер»

Система управления «Метакластер» позволяет управлять сразу несколькими кластерами [1]. Система позволяет распределять задания как между кластерами, так и внутри кластера, по узлам. Однако, в том случае, когда на каком-то кластере уже установлена система управления, «Метакластер» может служить промежуточным звеном между локальной системой и пользователем. В последнем случае распределением заданий по узлам осуществляет система управления, с которой произведена интеграция. В текущей версии системы «Метакластер» возможна интеграция только с Microsoft Compute Cluster Server 2003, однако список поддерживаемых систем будет расширяться. Под интеграцией в данном случае мы понимаем предоставление возможности выполнения следующих операций: постановка задач в очередь CCS, мониторинг их состояния, принудительное снятие с выполнения.

Рассмотрим, каким образом осуществляется взаимодействие, и за счет чего появляется универсальность в управлении кластерами без сторонней системы управления и кластерами, работающими под управлением CCS. На рисунке представлена архитектура системы «Метакластер».

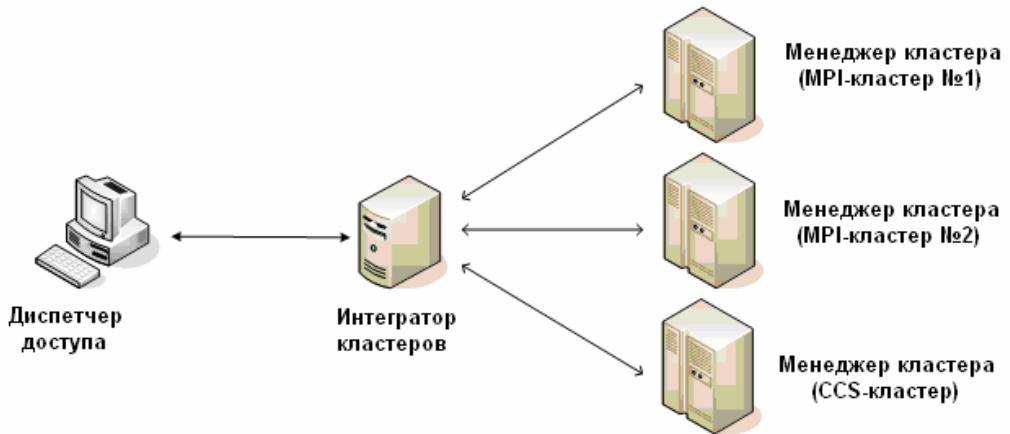


Рис. Архитектура системы «Метаклaster»

Можно выделить три основных компонента: диспетчер доступа, интегратор кластеров и менеджер кластера. Это классическая трехзвенная архитектура для систем управления, обеспечивающая гибкость управления несколькими разнородными кластерами [3,4,5]. Интегратор кластеров осуществляет единообразное управление несколькими кластерами, для каждого из которых создается специфический менеджер, учитывающий особенности архитектуры кластера, установленной операционной системы, конкретной версии библиотеки MPI. Менеджер ответственен за планирование и запуск задачий на машинах кластера, на котором он функционирует.

В нашем случае необходимо создать менеджер, который взаимодействовал бы с CCS. Его особенность состоит в том, что он сам не осуществляет планирование и запуск параллельных задач, как это происходит в случае с менеджерами MPI-кластеров. Он лишь передает задачи от интегратора к системе CCS, которая самостоятельно производит планирование и запуск.

Несколько слов о реализации. В компоненте менеджер кластера за планирование отвечает класс, называемый CScheduler, за управление задачами ответственен класс CLauncher. Их особенность в случае менеджера, взаимодействующего с CCS, состоит в следующем. Во-первых, CScheduler не имеет собственного алгоритма планирования, он лишь перенаправляет приходящие от интегратора задачи на CCS. Во-вторых, CLauncher использует вызовы CCS API вместо собственноручного запуска параллельных заданий и анализа их состояния.

Заключение

Предоставляемый CCS API позволяет решать достаточно широкий круг задач. В частности, авторы настоящей работы использовали его для интеграции с системой «Метаклaster». В настоящее время пользователи имеют возможность удаленного доступа к кластерам, в том числе и к кластеру, управляемому CCS.

Таким образом, был сделан серьезный шаг в сторону повышения эффективности и удобства использования вычислительного кластера ННГУ. Кроме того, использованный подход в дальнейшем может быть использован для интеграции с другими системами управления.

Список литературы

1. Корняков К.В., Сенин А.В., Шишков А.В. Организация единой вычислительной инфраструктуры на базе кластеров, работающих под управлением различных операционных систем. Материалы 6-ой Международной научно-практической конференции «Высоко-производительные параллельные вычисления на кластерных системах», Санкт Петербург, 12-17 декабря 2006 г.
2. Microsoft Windows Compute Cluster Server 2003 Beta 2 Reviewers Guide (<http://www.microsoft.com/windowsserver2003/ccs/reviewersguide.mspx>).
3. Baker M. et al. Cluster Computing White Paper. Final Release, Version 2.0.
4. Rajkumar Виууа. High Performance Cluster Computing. Volume 1: Architectures and Systems. Vol. 2: Programming and Applications. Prentice Hall PTR, Prentice-Hall Inc., 1999.
5. Ed. By Thomas Sterling. Beowulf Cluster Computing with Windows. The MIT Press, Cambridge, Massachusetts, 2002.

РЕАЛИЗАЦИЯ СРЕДЫ СОЗДАНИЯ ЭЛЕКТРОННОГО УЧЕБНИКА НА БАЗЕ ТЕХНОЛОГИИ .NET

А.А. Кудаков, Г.В. Кузенкова

Нижегородский госуниверситет им. Н.И. Лобачевского

Актуальность темы

Одним из современных направлений прикладной информатики является разработка электронных средств обучения (ЭСО) и создание удобных компьютерных средств реализации ЭСО с использованием современных информационных технологий. Программная реализация определенного электронного средства обучения обычно связана с областью учебной дисциплины. Все многообразие способов реализации ЭСО можно выделить в четыре группы: использование специализированных инструментальных систем, предназначенных для создания учебных программ (так называемых программ «оболочек»); использование универсальных профессиональных прикладных программных средств (профессиональные пакеты, как MathCad, MathLab, AutoCad и т.д.); использование языков программирования (преимущественно высокого уровня) создание ЭСО с «нуля»; комбинированное использование возможностей, предоставляемых первыми тремя способами.

Нам представляется интересной возможность создания ЭСО с применением новых, современных и перспективных информационных технологий, так как это позволяет расширить границы использования программных продуктов. В данной работе предлагается (или разработана) программа-оболочка для создания электронных учебников и электронных изданий для мобильных средств связи на базе технологии .NET (Microsoft). Выбор технологии связан со следующими факторами: скоростью и удобством программирования в Visual Studio 2005 в сочетании с указанной технологией, поддержкой мультимедийных технологий, возможностью переноса на мобильные платформы .NET Compact Framework набор библиотек для мобильных устройств на базе операционной системы Windows Mobile.

Реализация проекта

Программный комплекс состоит из двух основных частей: редактора проектов (рис. 1) и среды исполнения (рис. 2).

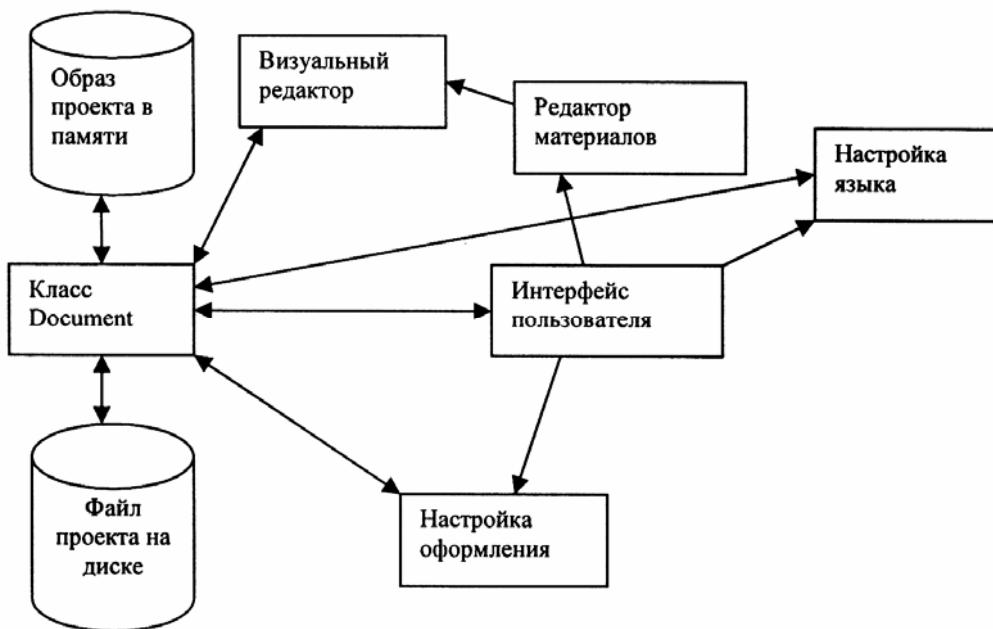


Рис. 1. Схема работы программы-редактора

Для создания редактора и оболочки электронного учебника была выбрана среда разработки Microsoft Visual Studio .NET 2005 Express Edition. Это связано с тем, что благодаря технологии Microsoft .NET Framework 2, значительно ускоряется процесс разработки приложений. Кроме этого, надо учитывать, что эта среда является самой перспективной для разработки Windows-приложений.

При разработке интерфейса программы-редактора за основу был взят принцип удобства для программиста и у добавлены привычные инструменты для обычного пользователя.

Правую часть окна занимает древовидная структура проекта. Под проектом мы понимаем создаваемый электронный учебник. В этой структуре, на данный момент, имеются три основных раздела: Оформление, Язык, Материалы.

В разделе «Оформление» находятся настройки электронного учебника, такие как фоновое изображение окна, изображения кнопок, их местоположение и настройки действия. Так же сюда включены определения стилей материалов.

В раздел «Язык» вошли языковые настройки оболочки. Учебник можно создавать на любом языке благодаря использованию кодировки Unicode.

В раздел «Материалы» включены материалы учебника. Они отсортированы по разделам (главам) учебника. Редактор визуально повторяет иерархию материалов в учебнике, что упрощает навигацию.

Иерархический список параметров проекта иллюстрирован красочными ассоциативными значками. Это улучшает восприятие и ускоряет обучение работе с программой.

В программу включен визуальный редактор материалов. Кнопки панели управления имеют такие же значки, как в Microsoft Office 2003.

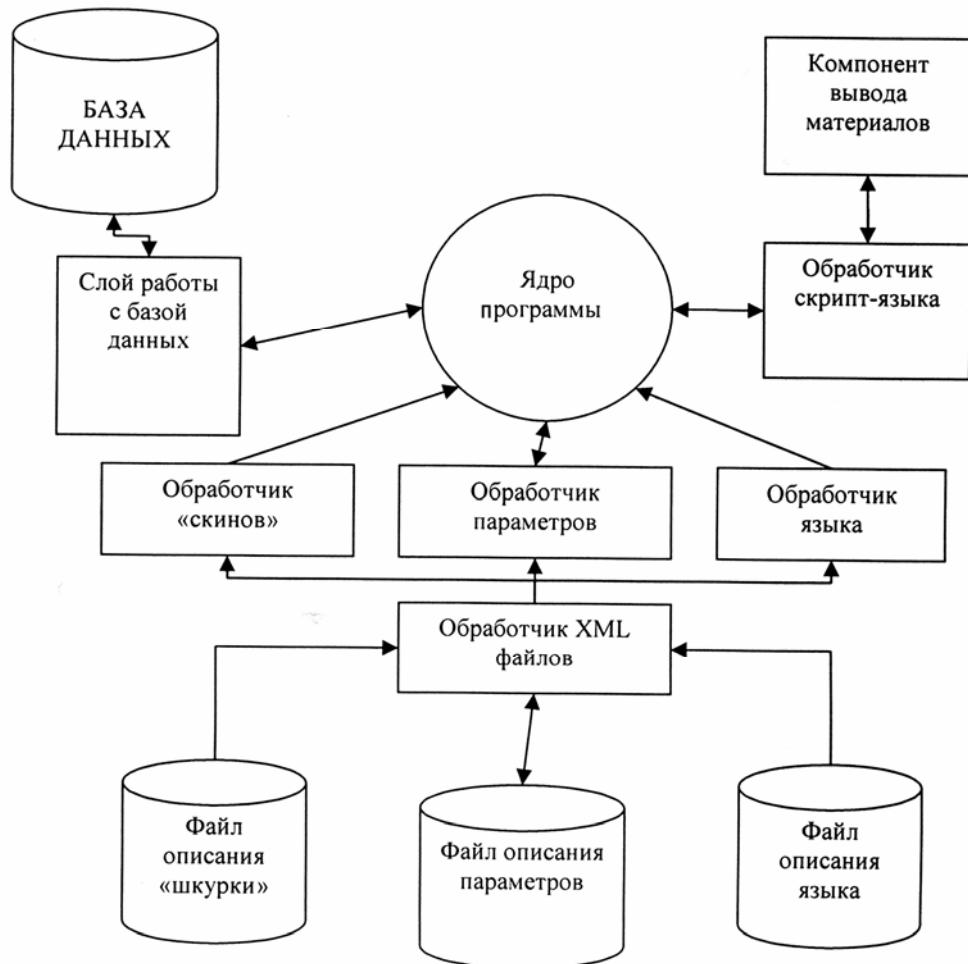


Рис. 2. Схема оболочки учебника

На данный момент визуальный редактор позволяет присваивать атрибуты абзацам и символам. Есть возможность работы с таблицами, изображениями. Можно добавлять гипертекстовые ссылки на другие материалы или документы в сети Интернет. Имеются функции работы с буфером обмена, возможность отмены и повтора изменений, поиска в тексте и вывода страницы на печать. Можно вставить линию разрыва, список (нумерованный и маркированный).

Также планируется добавить возможность быстрой вставки кода скриптового языка оболочки учебника и других объектов.

Программа-редактор работает с проектами, файлами формата керој. Эти файлы являются документами формата XML, в которых содержится описание свойств проекта, связи между объектами и т.п.

За представление файлов проекта в программе отвечает класс Document. Этот класс является сериализуемым, т.е. является интерфейсом ISerializable. Функции сохранения и чтения из файла реализованы на базе XML классов Microsoft .NET Framework.

При открытии или создании нового проекта создается новый образец класса Document, который заполняется данными из открытого файла (в случае, если файл открывается) или пустыми данными, если создается новый проект.

При работе с проектом через интерфейс пользователя, все изменения в проекте сохраняются в классе Document (в памяти).

После компиляции проекта его содержимое представляется в виде множества файлов. С ними работает программа-оболочка.

БЫСТРОЕ ОБНАРУЖЕНИЕ ЛИЦА ЧЕЛОВЕКА

П.П. Кудряшов, С.А. Фоменков

Волгоградский государственный технический университет

Целью данной работы является повышение скорости работы классификатора Haar за счет использования предварительного шага анализа цветовой информации изображения и алгоритма обрамляющих эллипсов. В настоящее время в связи со стремительным развитием цифровой фотографии и цифрового видео очень перспективной является задача распознавания объектов на цифровых изображениях.

Впервые представленный в 2001 году алгоритм Haar обладает не только высокой скоростью работы, но и хорошей точностью распознавания. Однако анализ работы этого алгоритма и использование априорных знаний о форме и цвете человеческого лица позволило нам выявить возможные пути улучшения характеристик работы алгоритма и создать быстрый алгоритм распознавания человеческих лиц.

Исследования проводились по двум направлениям: повышение скорости и точности распознавания. Быстрый алгоритм предлагается формировать из трех частей. Первая часть является грубой и позволяет быстро отсеять области, не содержащие лиц, используя статистические характеристики изображения. Вторая часть улучшает изображение для повышения качества работы следующей части. Третья часть использует непосредственно алгоритм Haar, модифицированный для повышения скорости и точности. При этом обрабатываются только "перспективные" области изображения, а также дополнительная информация, полученная первой частью алгоритма.

В первой части гибридного алгоритма для повышения скорости работы предлагается предварительная обработка изображения, направленная на исключение областей, в которых лицо человека заведомо не может быть найдено.

На первом этапе производится поиск областей, непригодных для обработки алгоритмом Haar для любых типов объектов. При этом исключаются монотонные области изображения, которые не содержат достаточного количества контуров и, следовательно, непригодны для дальнейшей обработки, а также области, содержащие слишком большое количество контуров.

Второй этап использует ограничения (априорные знания) задачи обнаружения человеческих лиц. Как известно, лица людей на цифровых изображениях имеют так называемый "цвет кожи", который достаточно компактно описывается в цветовых пространствах RGB, HSV и других. Сначала на изображении выделяются маски "кожи" – те пиксели изображения, которые могут быть кожей. После этого полученные пиксели обрабатываются алгоритмом, обнаруживающим контуры эллипсов вокруг границ компактного расположения таких пикселей. На этой стадии алгоритм выдает значительное количество эллипсов, большинство из которых не являются контурами человеческих лиц. Далее отсекаются слишком маленькие эллипсы, которые соответствуют цветовому шуму. Также удаляются эллипсы со слишком большим отношением большого и малого радиусов. Такие вытянутые эллипсы часто соответствуют конечностям человека, различным деревянным панелям мебели и др. По итогам получается маска "кожи", обведенная эллипсами, каждый из которых обводит предполагаемое лицо человека.



Рис. 1. Области цвета «кожи», обведенные эллипсами

На следующем этапе следует проверка на процент заполнения эллипса пикселями цвета «кожи». В случае, если процент ниже некоторого допустимого порога (такое происходит в случае, если два и более лиц людей соединяются на изображении), производятся морфологические операции, позволяющие уточнить истинные контуры лиц. Так, на рис. 1 в центре три человеческих лица образуют контур, который отмечается алгоритмом обнаружения эллипсов. Дальнейшее уточнение ввиду низкого процента заполненности эллипса пикселями маски «кожи» приводит к уточнению эллипсов, изображеному на рис. 2.

Необходимо отметить, что для еще большего повышения скорости работы первой части алгоритма на этапе обнаружения пикселов цвета «кожи» и обводящих их эллипсов возможно предварительное уменьшение разрешения картинки в 2, 4 или 8 раз. При этом происходит квадратичное снижение объема вычислений, связанных с анализом цветовых характеристик пикселов. Также не требуется удаление слишком маленьких ограничивающих эллипсов, т.к. подобные области цветового шума исчезают в результате операции понижения разрешения. В дальнейшей обработке обнаруженные эллипсы масштабируются для соответствия разрешению исходного изображения.

По результатам работы первой части алгоритма формируется набор прямоугольных под областей исходного изображения, которые содержат эллипсы, ограничивающие области маски «кожи». Данные прямоугольники являются потенциальными областями, содержащими лица людей.



Рис. 2. Уточненные регионы «кожи» после работы алгоритма обнаружения эллипсов



Рис. 3. Приоритетные направления обнаружения лица алгоритмом Haag

Во второй части работы алгоритма для повышения точности распознавания алгоритма Haag авторы предлагают использование препроцессинга найденных прямоугольников изображений перед их обработкой. Под препроцессингом понимается изменение изображения с целью повышения качества распознавания. Предлагается два основных направления препроцессинга: снижение шума изображения и цветокоррекция. С математической точки зрения цветокоррекция обуславливает улучшение распознавания за счет создания более естественного, "гладкого" цвета объекта, что позволяет получить значительно более плавные градиенты переходов интенсивности цвета. Такой прием дает возможность корректно обрабатывать участки изображения небольшого размера, для которых очень велико влияние даже единичных шумных пикселов.

В третьей части работы найденные прямоугольные подобласти исходного изображения проверяются алгоритмом Haag. Необходимо отметить, что алгоритм обнаружения эллипсов, выполненный на первом этапе, позволяет не только значительно сократить площадь изображения для последующей обработки алгоритмом Haag, выделив потенциальные подобласти. Он дает возможность использовать дополнительную информацию, а именно, расположение больших осей найденных эллипсов. Это объясняется тем, что

алгоритм Haar чувствителен к ориентации объекта на изображении и обладает определенной (около 20 градусов) устойчивостью к его поворотам. Следовательно, чтобы обнаружить лицо в произвольном положении, требуется $360 / 20 = 18$ поворотов изображения. Такого количества поворотов можно избежать, проанализировав направление большей оси эллипса. Проверка данного предположения на наборе из более чем 500 фотографий говорит о том, что в 90% случаев данная ось совпадает с осью головы человека.

Таким образом, для значительного (на порядок) повышения работы алгоритма Haar сначала проверяются два приоритетных направления, соосных большей оси эллипса и расположенных под углом в 180 градусов друг к другу, а потом все остальные (см. рис. 3). Кроме того, линейные размеры эллипса оказываются близкими к размерам лица человека, что позволяет запускать алгоритм Haar с наиболее вероятным размером поискового окна, исключая дополнительную фазу перебора всех его возможных положений и размеров.

Также авторами предлагается два способа улучшения качества работы алгоритма Haar. Первый основан на том факте, что распознавание объектов, на которые настроен алгоритм, достаточно устойчиво (около 20 градусов) к поворотам изображения. Случайные, ложные срабатывания, в отличие от истинных, не обладают такой устойчивостью. Следовательно, для обнаруженных предполагаемых объектов предлагается произвести повторную проверку обнаружения в соответствующих областях на изображении, повернутом на 10 градусов по и против часовой стрелки. В случае, если срабатывание ложное, с высокой степенью вероятности ни на одном из повернутых изображений оно не будет повторным.

Второй способ модифицирует процесс изменения размера поискового окна алгоритма. Так, базовый алгоритм использует постоянный шаг изменения размера и движения окна поиска вдоль изображения. Для сокращения числа ложных обнаружений предлагается значительно уменьшать шаг увеличения окна в окрестностях однократно найденного объекта и требовать, чтобы в большинстве полученных таким образом окон алгоритм также обнаруживал объект. При ложных срабатываниях в большинстве случаев анализируемые окна в окрестностях не дадут положительного результата.

Описанный алгоритм позволяет значительно сокращать объем расчетов, требуемых для обнаружения человеческих лиц на цифровых изображениях. При этом высвободившиеся вычислительные ресурсы можно направить на повышение качества работы алгоритма за счет гибкого изменения шага масштабирования поискового окна и механизма поворотов. Данный алгоритм реализован на языке C++ с использованием библиотеки компьютерного зрения Intel OpenCV. Экспериментальные результаты, полученные на наборе дневных иочных фотографий, дали следующие результаты: отсечение шумных и монотонных участков изображения позволяет обрабатывать их от 2 до 5 раз быстрее без снижения точности. Использование цветовой информации и обрамляющих эллипсов позволяет добиться ускорения от 5 до 100 раз за счет многократного снижения площади областей изображения, поступающих на обработку алгоритмом Haar, и проверки наиболее вероятных направлений расположения лица. Отметим также, что использование поворотов изображения и гибкого масштабирования размера окна в целях повышения точности, позволяет отсечь до 50% ложных срабатываний.

Подсистема распознавания лиц, основанная на данном алгоритме, используется в программном продукте полностью автоматической коррекции красных глаз StopRedEye с датой выпуска, намеченной на март 2007 г.

ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ ДЛЯ ПОВЫШЕНИЯ ТОЧНОСТИ РАСПОЗНАВАНИЯ ОБЪЕКТОВ

П.П. Кудряшов, С.А. Фоменков

Волгоградский государственный технический университет

Целью проведения данного исследования является повышение точности распознавания за счет использования предварительной обработки изображений.

В настоящее время в связи со стремительным развитием цифровой фотографии и цифрового видео очень перспективной становится задача распознавания объектов на цифровых изображениях.

Последние десять лет в этой области ведутся активные поиски и разработаны различные методы распознавания, использующие такие подходы, как метод главных компонент, гистограммы, нейросети, байесовские сети, алгоритмы Haag-а, статистические методы и т.д. Большинство этих методов осуществляют предварительную обработку изображений. Однако такие обработки зачастую включают в себя лишь геометрические операции, как-то: поворот, кадрирование, увеличение/уменьшение, центрирование, а также операции с цветовым пространством: перевод в градацию серого, снижение битности и др. Упомянутые действия обычно выполняются оператором и почти всегда – на этапе обучения классификатора объектов.

Значительно более перспективным, по мнению авторов, является препроцессинг изображений перед их обработкой классификатором объектов. Под препроцессингом понимается изменение классифицируемого изображения с целью повышения качества распознавания. В настоящее время предлагается два основных направления препроцессинга: снижение шума изображения и цветокоррекция. С математической точки зрения цветокоррекция обуславливает улучшение работы классификаторов за счет создания более естественного цвета объекта. Так, многие классификаторы человеческого лица базируются на цветовых порогах цвета кожи и могут исключать из рассмотрения изображения лиц со смещенным цветовым балансом, например, если кожа имеет зеленоватый оттенок. Снижение шумности изображения за счет пирамидального или гауссского, а также других, более совершенных методов, позволяют удалить цветовой шум, очень распространенный на изображениях, получаемых на современных цифровых фотоаппаратах. Снижение шума позволяет получить значительно более сглаженные градиенты переходов интенсивности цвета, что значительно улучшает результаты работы классификатора Haag.

Экспериментальные результаты, полученные на наборе дневных иочных фотографий с использованием классификатора лиц Haag из библиотеки Intel OpenCV, адаптивного шумоподавления и цветокоррекции из библиотеки CImg, показали, что наочных фотографиях среднее количество лиц, распознаваемых классификатором, составляло около 40%. При использовании цветокоррекции это количество поднялось до 50%. При использовании цветокоррекции и подавления шума удалось получить до 65% распознаваний. Наиболее сильно эффект от использования шумоподавления заметен наочных фотографиях, а также на объектах, находящихся в тени и на заднем темном фоне. Тестирование проводилось на выборке из 100 фотографий. Полученный результат подтверждает эффективность применения препроцессинга для повышения качества распознавания объектов на цифровых изображениях. Подсистема цветокоррекции и шумоподавления используется в программном продукте полностью автоматической коррекции красных глаз StopRedEye с датой выпуска, намеченной на март 2007 г.

УНИВЕРСАЛЬНЫЙ РАСТЕРИЗАТОР

А.И. Кузнецов

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Для сокращения объема передаваемых данных в современных плоттерах, как правило, используется векторный формат описания графических документов. В этом случае изображение представляется не в виде массива точек, как это принято в растровых форматах, а последовательностью графических команд, таких, как «нарисуй линию», «нарисуй окружность» и т.п. Векторное представление изображений имеет как плюсы, так и ряд недостатков.

К главному преимуществу векторного описания можно отнести простоту создания изображения из различных программ. Для создания растровой картинки программа (или человек) должна досконально знать растровый формат (а иногда – и несколько), и для рисования элементарных графических примитивов требуется написание отдельных достаточно громоздких подпрограмм. При формировании векторного файла имеется возможность комбинировать примитивы в более крупные фрагменты изображения (графические процедуры), что ускоряет и упрощает процесс создания графического образа.

Благодаря последнему факту написано немало программ, которые создают векторные графические описания. К их числу принадлежат такие известные программы, как *CorelDraw*, *Adobe Photoshop*, *AT&T Dot Utility* и КАРТ-ДОК. Нередки случаи, когда полученную картинку необходимо подкорректировать. Большинство графических редакторов позволяют редактировать файлы, ими созданные, но это не всегда так.

Некоторые графические форматы являются закрытыми (комерческими) и используются только для результирующей выдачи на устройство. Другие графические языки не позволяют выполнять определенные виды редактирования. Например, язык *HP-GL/2* не включает такую операцию, как вырезка прямоугольного участка изображения и манипуляции с выделенными фрагментами. Решение указанных проблем может существенно облегчить предварительное преобразование изображения из векторного формата в растровый. Существует много редакторов и утилит, которые позволяют редактировать и тем или иным способом обрабатывать растровые изображения.

Процесс преобразования векторного графического формата в растровый получил название растеризации, а соответствующие программы или более сложные технические комплексы называют растеризаторами.

Универсальный растеризатор (*UniRas.lib*), описываемый в данной работе, представлен библиотекой классов, позволяющей создавать индивидуальные конверторы векторных графических файлов, обладающие дополнительными возможностями. Эта библиотека включает классы, которые функционально разделяют все этапы преобразования графических форматов данных. Благодаря такому разделению появилась возможность вообще абстрагироваться от форматов входных и выходных данных, отсюда и название «Универсальный».

Большинство существующих на текущее время растеризаторов излишне жестко связаны со спецификой входного языка. Для устранения этой узкой специализации в универсальном растеризаторе предусмотрен набор операций над множеством векторных графических языков. Этот набор включает:

L_1 – список входных языков, которые должен уметь понимать универсальный растеризатор. Предполагается, что этот список можно расширять путем добавления новых входных форматов.

L_1 – язык векторных примитивов (MMF). Этот язык предусматривает всего десять слов-команд: *begin* (начало программы), *line* (нарисовать отрезок), *polygon* (нарисовать заливой полигон), *polygon_xp* (нарисовать заливой полиполигон), *arc/farc* (нарисовать эллипс/заливной сектор эллипса), *circle/fcircle* (нарисовать окружность/круг), *end* (конец графической программы), *por* (нет операции). Команды *begin*, *end* и *por* введены для системных целей.

L_2 – язык растровых примитивов: *lineh/linev* (нарисовать горизонтальный или вертикальный отрезок прямой толщиной в одну точку), *pix* (нарисовать точку), *eor* (конец графической программы). Команда *eor* введена для вспомогательных целей.

Введение языков L_1 и L_2 позволило абстрагировать большую часть процесса растеризации как от входных графических языков, так и от выходных графических форматов. Однако процесс формирования растрового изображения осложняется тем, что размер растрового файла может оказаться достаточно большим. Например, для его хранения может не хватить оперативной памяти компьютера. С этим можно бороться по-разному. Самым простым путём является растеризация по частям. Для этого создаётся движущееся окно, которое полностью убирается в оперативную память, и вся графическая программа выполняется для каждого нового положения окна. Причём команды, которые рисуют вне текущего окна, должны полностью или частично отсеиваться.

Такой подход приводит к повторению всех этапов растеризации для каждой части результирующего изображения, хотя часть ранее выполненной работы можно было бы и не повторять (например, можно не преобразовывать повторно числа из символьного формата в машинный, если входной формат был текстовым). Поэтому в растеризаторе целесообразно сохранять графическую программу на языке L_1 и результаты анализа каждой команды. Хранение изображения на языке L_2 неэффективно из-за размера растра, т.к. пришлось бы хранить каждую точку результирующего растра в виде команды, причём многие точки могли бы и повторяться (например, если один объект рисуется поверх другого).

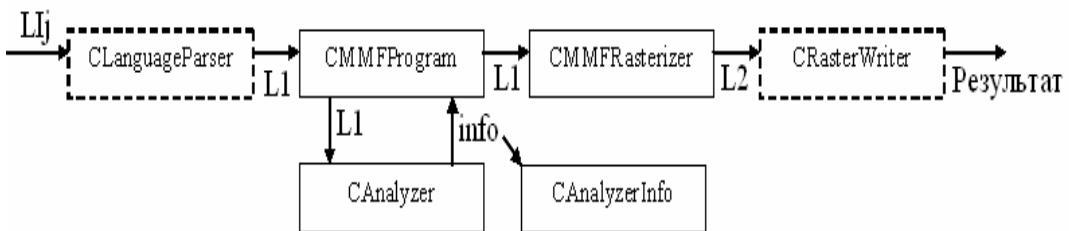


Рис. Принципиальная схема универсального растеризатора

На рисунке изображена принципиальная схема универсального растеризатора, где представлены основные классы библиотеки, названия которых даны в прямоугольниках. Схема показывает последовательность преобразований, которые претерпевают команды исходного векторного графического файла в процессе создания эквивалентного растрового файла. В пунктирных прямоугольниках указаны названия абстрактных классов, которые предлагаются унаследовать пользователям библиотеки с целью добавления своей функциональности.

Библиотека включает в себя шесть классов, задающих каркас любого процесса растеризации:

1. *CLanguageParser*. Данный класс задаёт шаблон парсера графического языка и является абстрактным. Основная цель парсера – потоковое чтение команд входного языка и преобразование каждой команды в последовательность команд языка L_1 . Преобразование выполняется с помощью вызова методов класса *CMMFProgram* в нужной последовательности. Для каждого нового векторного языка должен быть создан свой класс парсера, порождённый от *CLanguageParser*.

2. *CMMFProgram*. Данный класс отвечает за хранение промежуточной графической программы на языке L_1 , её первоначальный анализ с помощью класса *CAnalyzer* и пошаговое выполнение этой программы заданное количество раз. Каждый повтор сопровождается последовательностью вызовов соответствующих методов класса *CMMFRasterizer*. Промежуточная программа может запоминаться по выбору пользователя либо в файле на диске, либо в заранее выделенном буфере памяти. Последний вариант разумно выбирать при небольших объёмах работы (при растеризации изображений с маленьким количеством команд).

3. *CAnalyzer* и *CAnalyzerInfo*. Два класса. Первый класс – это анализатор графических команд. В версии по умолчанию (реализация библиотеки) данный класс определяет границы рисования для каждой команды и для всего изображения в целом. Также он производит индексирование цветов. Вся собранная информация в конце разбора входного файла собирается во втором классе *CAnalyzerInfo*, которая возвращается в класс *CMMFProgram* и запоминается вместе с программой. В случае необходимости обеспечения дополнительных возможностей анализа, эти классы можно расширить.

4. *CMMFRasterizer*. Ядро растеризатора. Для оптимизации некоторые методы класса написаны на языке Ассемблера с использованием команд процессоров *Pentium II* и выше. Именно этот класс выполняет преобразование графических примитивов языка L_1 в набор растровых команд языка L_2 . Далее полученные команды передаются объекту класса *CRasterWriter*.

5. *CRasterWriter*. Основное предназначение данного класса – запись растрового файла в конкретном формате. В задачи этого класса входят и заботы по управлению памятью – он старается не только запросить для себя память по максимуму, но оставить часть памяти для других объектов. Как и *CLanguageParser*, данный класс является абстрактным. В библиотеке реализован метод распределения памяти по умолчанию. Если памяти хватает на весь растр плюс пул безопасности, то она выделяется, и растеризация будет проведена в один проход. Если же памяти недостаточно, то растеризация выполняется окнами – горизонтальными полосами. Это не самая эффективная стратегия, и её можно переопределить (например, можно резать результирующее изображение квадратами, если выходной формат позволяет записывать файл таким образом).

Класс *CLanguageParser* задаёт структуру класса, который выполняет чтение и преобразование входного графического языка. Класс *CRasterWriter* представляет структуру класса, который используется для записи выходного растрового формата данных. На самом деле, библиотека уже содержит потомков данных классов: для чтения векторного языка *HP-GL/2* (класс *CHPGLImplementation*) и для записи растрового формата *wBMP 3.0* (класс *CBMPWriter*). Для добавления новых форматов пользователь должен произвести своих потомков классов *CLanguageParser* и *CRasterWriter*.

Если переопределять внутренние классы библиотеки (а она это позволяет), то имеется возможность создавать конверторы векторных файлов в векторные (для этого достаточно подменить *CMMFProgram*). Аналогичным образом можно заставить парсер

входного языка читать растровые файлы одного формата и преобразовывать их в другие растровые форматы. Однако производительность в этом случае может оказаться довольно низкой.

Для тестирования и отладки библиотеки выполнялось преобразование векторного графического файла, написанного на языке *HP-GL/2*, в растровый файл в формате *wBMP*. В качестве исходного документа использовалась морская навигационная карта размером 30 Мб, которая растеризовалась с качеством 600 *dpi*, что соответствует 213 Мб в результирующем файле. Результаты тестирования приведены в следующей таблице, где указано отдельно время разбора исходного векторного файла и генерации эквивалентного описания на языке *L₁*. По этому времени можно оценить производительность классов *CLanguageParser*, *CMMFProgram* (запись) и *CAnalyzer*. Под временем растеризации понимается время создания результирующего растрового файла, когда программа *L₁* уже создана. Это время показывает производительность классов *CMMFProgram* (чтение), *CMMFRasterizer* и *CBMPWriter*. В качестве хранилища промежуточной графической программы был выбран винчестер (тот же, на котором генерируется результат). Все времена приводятся в минутах.

Таблица 1

№	Процессор	RAM, Мб	Время L1	Время растеризации
1	<i>Celeron 2000</i>	240	1	6
2	<i>Celeron 2400</i>	490	0,8	2
3	<i>Pentium IV 2800 X2</i>	512	0,6	1

В следующей таблице приведены времена растеризации другими программными средствами.

Таблица 2

Средство	Время
<i>GNU hp2xx, Celeron 2000, 240 mb, 300dpi</i>	170
<i>HP DesignJet 800, 300dpi</i>	60
<i>Adobe Photoshop CS2, Celeron 2000, 240 mb, 300dpi (после hp2xx из EPS)</i>	60

Приведённые таблицы показывают, что универсальный растеризатор выигрывает у приведённых графических пакетов не только по простоте использования, но и по скорости выполнения аналогичных действий.

КОНТРОЛЬ КАЧЕСТВА ИЗОБРАЖЕНИЙ

А.И. Кузнецов

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Данная статья описывает возможность расширения библиотеки классов универсальный растеризатор (*UniRas.lib*), описанной в одной из предыдущих статей. Первая часть статьи посвящена возможности библиотеки к добавлению новых форматов выходных данных. Вторая часть описывает создание приложения, позволяющего контролировать качество получаемых растеризатором изображений.

Добавление выходного формата TIFF

Формат *BMP* хорош своей простотой создания. Однако простота это мнимая – он настолько сложен, что до сих пор не существует ни одной программы просмотра/печати/преобразования графических файлов, которая полностью бы поддерживала этот формат. Библиотека *UniRas* создаёт файлы версии 3.0 с возможностью сжатия алгоритмом *RLE*. Многие программы сторонних разработчиков (в том числе и *Adobe Photoshop 8.0*) почему-то не понимают в полной мере этого алгоритма, некоторые – не поддерживают *BMP*, имеющий один из размеров больше 32765, и, наконец, абсолютное большинство программ не понимает *BMP*, сжатый другими алгоритмами сжатия (*JPEG, ZIP, LZW, pack-bits*).

Таким образом, возникла необходимость выдачи в более распространённые растровые форматы, поддерживающие сжатие и большие объёмы информации. Идеальным кандидатом стал формат *TIFF*, который имеет расширяемую структуру хранения данных, а также указанные в спецификациях алгоритмы сжатия (*JPEG, ZIP, LZW, pack-bits*).

Для добавления в растеризатор возможности выдачи в новый формат данных, необходимо произвести свой класс из виртуального класса *CRasterWriter*. Новый класс называется *CTIFFWriter*:

```
class CTIFFWriter : public CRasterWriter
{
public:
    //! Вид сжатия.
    enum TCompression
    {
        none,      //!<< Нет сжатия.
        packbits,  //!<< Упаковка битов.
        JPEG,      //!<< CCITT JPEG
        LZW,       //!<< Lempel-Ziv-Weiss
        zip        //!<< deflate
    };
    CTIFFWriter(const TCompression cCompression=zip, const int par=3);
    virtual ~CTIFFWriter();
    virtual fctal::boolean init(const char* const filename);
    virtual void lineh(const TOutCoord x1, const TOutCoord x2, const TOutCoord y,
                      const TColor color);
    virtual void pix(const TOutCoord x, const TOutCoord y, const TColor color);
    virtual void eop();
    virtual DWORD set_data(CAnalyzerInfo* ai);
};
```

Из соображения экономии места здесь приведён только раздел *public* порождённого класса, который и содержит основные методы, которые требуется переопределить.

Опишем все методы:

CTIFFWriter. Конструкторы классов, порождённых от *CRasterWriter*, обязаны инициализировать все параметры объекта, которые должны быть известны на момент открытия файла и записи в него данных. В нашем случае такими параметрами являются алгоритм сжатия и коэффициент сжатия.

init. Данный метод создаёт и открывает файл с указанным именем, а также подготавливает его к записи растровых данных.

lineh, pix. Данные методы записывают на плоскость раstra горизонтальный штрих и точку заданного цвета соответственно.

eof. Этот метод вызывается при завершении работы с файлом, поэтому он долженбросить все буферизованные данные и закрыть файл.

set_data. Это интерфейсный метод, через который объект узнаёт информацию, полученную анализатором векторного файла. От метода только требуется сохранить её куда-нибудь, чтобы методы создания раstra могли потом ею воспользоваться.

Формат *TIFF* имеет слишком много возможностей, чтобы их реализовывать самому, поэтому было принято решение использовать свободно распространяемую библиотеку *libtiff* 3.0. Благодаря этому реализация класса *CTIFFWriter* приняла вид простого переходника к библиотеке *libtiff*. При появлении новых версий формата *TIFF* достаточно будет только взять новую библиотеку.

Также был проведён ряд тестов на производительность объектов нового класса. Тестирование показало потерю производительности при использовании класса *CTIFFWriter* всего лишь на единицы миллисекунд по сравнению с классом *CBMPWriter*, но это объясняется большим количеством служебной информации в формате *TIFF*.

Контроль качества

Стандартный подход к контролю качества, создаваемой растеризаторами графической информации, заключается в следующем. Сначала проводится растеризация с получением результирующего файла, затем запускается программа стороннего производителя, при помощи которой полученный файл просматривается. Если качество результата не удовлетворяет, то производится модификация исходного векторного файла и весь процесс создания результирующего раstra повторяется.

Библиотека *UniRas* позволяет существенно ускорить такой процесс построения растровых файлов. Идея проста – не надо каждый раз создавать результат – растр получается один раз и требуемого качества.

Как известно, библиотека *UniRas* использует в своей работе некоторый промежуточный векторный язык *MMF*. Причём данные в этом формате в любом случае создаются при растеризации (либо в памяти, либо во внешнем файле на диске). Процесс создания *MMF* имеет практически константную производительность, в отличие от процесса растеризации, который занимает всё основное время.

Для увеличения производительности разумно не выполнять растеризацию каждый раз, а просматривать промежуточный результат в формате *MMF*.

После того, как исходный векторный файл будет должным образом отложен, можно один единственный раз выполнить фактическую растеризацию с получением результирующего раstra. Причём при этой растеризации для экономии времени можно воспользоваться уже готовым *MMF* (который мы просматривали). Также разумно не сохранять *MMF* в файл на диске, а сразу просматривать полученную программу, сохраняя лишь удовлетворяющий результат.

Для реализации класса просмотра *CMMFViewer* нам понадобится часть цепи, указанной (см. рис. на стр. 141), вплоть до *CMMFRasterizer*. Для создания такого класса требуется произвести его из *CMMFRasterizer* и указывать объекты нового класса во всех предыдущих объектах, где раньше указывался класс растеризаторов:

```

class CMMFViewer : public CMMFRasterizer
{
public:
    CMMFViewer();
    virtual ~CMMFViewer();
    virtual fctal::boolean init(CMMFProgram* const prg);
    virtual fctal::boolean run(HDC ctx, const TOutRect& clp, const TInRect& sw);
    virtual fctal::boolean rerun(const TOutRect& sw, TInRect& outw);
    virtual fctal::boolean grid(const TInPoint& offset, const TInPoint& step,
        const COLORREF color, const TOutCoord width,
        const fctal::boolean lAttach);
    friend class UniRas::CMMFProgram;
protected:
    virtual void begin_vector();
    virtual void line(const TInPoint& p0, const TInPoint& p1,
        const TInCoord width, const TColor color, const TInRect& bRect,
        const TLineEnding le0=LE_ROUND, const TLineEnding le1=LE_ROUND,
        const TInAngle s0=0, const TInAngle s1=0);
    virtual void polygon(const TInPoint* const lpPts, const DWORD n,
        const TColor color, const TInRect& bRect);
    virtual void polygonXP(const TPolyPolygon* const lpPol,
        const fctal::dword n, const TColor color,
        const TInRect* const pbRect);
    virtual void arc(const TInPoint& p0, const TInCoord axeA,
        const TInCoord axeB, const TInAngle begAngle,
        const TInAngle endAngle, const TInCoord width,
        const TColor color, const TInRect& bRect,
        const TLineEnding le0=LE_ROUND, const TLineEnding le1=LE_ROUND,
        const TInAngle s0=0, const TInAngle s1=1, const TInAngle ellAngle=0);
    virtual void farc(const TInPoint& p0, const TInCoord axeA,
        const TInCoord axeB, const TInAngle begAngle,
        const TInAngle endAngle, const TColor color,
        const TInRect& bRect, const TInAngle ellAngle=0);
    virtual void circle(const TInPoint& p0, const TInCoord r,
        const TInCoord width, const TColor color, const TInRect& bRect);
    virtual void fcircle(const TInPoint& p0, const TInCoord r,
        const TColor color, const TInRect& bRect);
    virtual void end_vector();
};

}

```

Для экономии места вспомогательные элементы класса не приводятся. Перечислены базовые методы, которые класс обязан реализовывать. Можно заметить, что создан новый метод *run()* (он не перекрывает унаследованный виртуальный метод). Это сделано, потому что теперь он должен иметь другие параметры, а именно: контекст устройства вывода (можно передать и контекст принтера, если надо напечатать картинку без полей на бумаге), область рисования и участок изображения. Последние два параметра

добавлены для обеспечения возможности увеличения участков изображения для их более детального просмотра.

В реализации методов класса имеется одна хитрость. Все методы, запускающие выдачу на контекст, используют специальные индексы. Индексы разбивают всё изображение на прямоугольные участки. Благодаря этому достаточно сильно увеличенные фрагменты отображаются гораздо быстрее. Рассматриваются только те команды рисования, которые каким-либо образом влияют на изображение внутри области просмотра.

Для более эффективной работы с индексами введён дополнительный метод *rerun()*. Данный метод можно вызывать программно, если нужно увеличить какой-либо фрагмент уже отображённого на экране фрагмента. В этом случае не надо строить полный индекс – достаточно выбросить из уже существующего ненужные команды.

Следует также отметить, что вся работа, связанная с построением и работой с индексами, уже заложена в класс *CMMFProgram*, поэтому её не нужно реализовывать в *CMMFViewer*.

В результате процесс контроля качества изображения полностью вошёл в схему универсальной растеризации. Вместо растеризатора данные будут посыпаться классу *CMMFViewer*, который не будет тратить время на создание файла, а просто нарисует его на экране. Для получения результата полученный *MMF*-файл нужно передать объекту класса *CMMFRasterizer*.

Главное окно полученной программы приведено на рисунке. Программа контроля качества позволяет производить измерения элементов изображения, для чего существует специальный инструмент – линейка, которую можно приложить к любым двум точкам. Для целей измерения также поддерживается нанесение координатных сеток, как с привязкой к изображению, так и без. При перемещении мышки над областью рисования в правом углу отображаются координаты с точностью 1–2 мм.

Выдающейся возможностью программы является способность увеличивать изображение без ограничений на степень увеличения. При этом по увеличенному изображению можно легко перемещаться, протягивая окно просмотра по графическому документу. Другим аналогичным программам (например, *Splot32*) такое не под силу. Полученная в результате программы предварительного просмотра *MMF*

позволяет достаточно эффективно просмотреть содержимое довольно крупных векторных файлов (более 25Мб с почти миллионом команд). При этом окно просмотра изображения можно достаточно плавно продвигать по экрану.

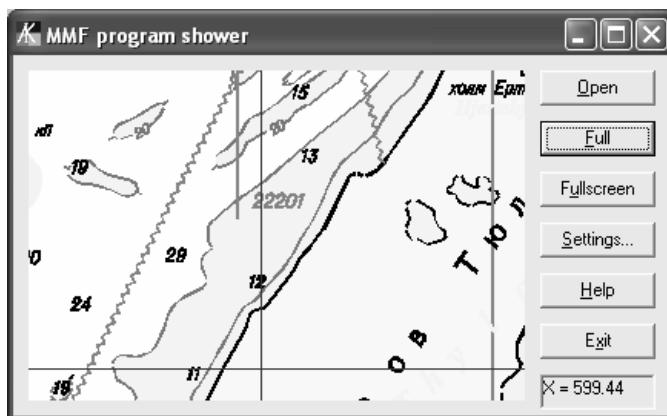


Рис. Главное окно программы

ОБЪЕМНОЕ ПРЕДСТАВЛЕНИЕ ТРЕХМЕРНЫХ ОБЪЕКТОВ

А.В. Кузьмин

Пензенский госуниверситет

Постановка задачи

Известно, что к наиболее ресурсоемким при обработке информации приложениям относится 3-Д графика. Описание, формирование, преобразование и представление сложных геометрических объектов являются основными операциями в компьютерной графике [1]. При этом во многих прикладных областях необходимым требованием является реалистичность получаемого изображения. Например, реалистичное представление трехмерного изображения сердца пациента поможет врачу-кардиологу повысить качество диагностики и эффективность лечения.

Как известно, центральным звеном любой системы визуализации является геометрическая модель, именно она определяет возможности системы визуализации. На данный момент наиболее распространенными являются поверхностные модели, построенные на основе опорных точек с помощью триангуляции Делоне. Такое представление объекта называют плоскогранным или, в более широком смысле, граничным. При этом для описания криволинейных поверхностей плоскогранное представление может аппроксимировать их некоторым количеством пластин треугольной или четырехугольной формы [2]. Отсутствие возможности визуализации внутренней структуры объекта, является фундаментальным отличием (и ограничением) графики поверхностной.

Поэтому, для реализации функций отображения внутреннего строения объекта и моделирования процессов, протекающих в нем, необходимо решить задачу получения объемного (или твердотельного) представления модели объекта на основе каркасной модели.

Разработка алгоритма вокселизации объекта

В первую очередь были проанализированы существующие подходы к объемному представлению трехмерных объектов. Анализ показал, что наиболее простым и универсальным способом получения объемного представления объекта является деление пространства. То есть сложный пространственный объект представляется перечислением простых объемных кусков или элементов. Здесь реализуется идея разделить пространство на куски и обозначить те ячейки, которые заполнены материалом [3]. Это могут быть наборы трехмерных кубов (называемых вокселями).

По определению, данному в [4], воксель – это кубический элемент пространства, центр которого располагается в точке дискретной решетки. Каждый воксель имеет численные параметры, ассоциированные с ним, которые представляют некоторые измеримые характеристики (например, цвет, прозрачность, плотность, материал, коэффициент преломления) объекта, занимающего часть объема, представляемого вокселием.

Объемные элементы можно организовать в более компактные и эффективные вычислительные структуры (деревья, иерархические графы и т.д.), поддерживающие эффективные алгоритмы обработки[5]. Одно из наиболее популярных представлений такого типа называется восьмеричное дерево: иерархический метод для представления группировок ортогональных трехмерных боксов (обычно кубов), где каждый бокс, пе-

ресекающий границу объекта, делиться на восемь боксов меньшего размера, и так далее, пока не будет достигнут требуемый уровень разрешения [6].

После выбора схемы объемного представления объекта встает проблема разработки алгоритма конвертирования каркасного представления объекта в воксельное. В нашем случае требуется получить воксельное представление объекта из поверхности этого объекта, заданного набором треугольников.

Воспользуемся двумя общеизвестными подходами: заполнение замкнутого объекта изнутри объемными элементами (получение воксельного представления) и рекурсивное деление пространства (получение восьмеричного дерева). Для их реализации разработаны алгоритмы, учитывающие фрактальный характер производимых операций (т.е. итеративное приращение элементов и рекурсивное деление пространства). У фракталов есть общее свойство - наличие рекурсивной процедуры их генерации, как правило, достаточно простой и максимально учитывающей свойство самоподобия. Использование системы итерированных функций [7] позволяет математически описать алгоритмы заполнения объекта вокселями и дробления пространства в виде аффинных преобразований. Более подробно разработка данных алгоритмов рассмотрена в [8].

После детального анализа каждого из описанных выше алгоритмов можно сделать вывод, что они имеют недостатки, препятствующие повышению скорости процесса вокселизации объекта.

В случае «заполнения» поверхности объекта вокселями узким местом является то, что при приращении каждого нового элемента необходимо проверять граничное условие. Проверка граничного условия предполагает громоздкие вычисления, независимо от того, как задана поверхность объекта. Также если учесть, что приращение происходит по одному элементу и эта процедура повторяется много раз (для достижения требуемого качества изображения число вокселей может достигать нескольких миллионов) данный алгоритм нельзя назвать оптимальным.

В случае восьмеричного дерева сама иерархическая структура генерируется достаточно быстро, но встает другая проблема: определение статуса «внутри – снаружи» для каждого узла восьмеричного дерева. Здесь встает та же проблема, что и в предыдущем алгоритме – многократная проверка граничного условия.

Для оптимизации работы алгоритмов и повышения скорости вокселизации объекта предлагается использовать комбинированный алгоритм. В первую очередь необходимо получить воксельное представление границы объекта, чтобы в дальнейших вычислениях не использовать математическое описание его поверхности. На следующем этапе необходимо заполнить эту замкнутую поверхность, состоящую из вокселей.

Для решения задачи вокселизации поверхности разработано множество алгоритмов. Например, вокселизация поверхности, состоящей из треугольников, рассматривается в [9], [10], обзор методов вокселизации различных типов поверхностей приводится в [11]. Мы для решения этой задачи воспользуемся уже описанным выше алгоритмом восьмеричного дерева. Каждый узел дерева имеет статус («внутренний», «наружный», «частичный»). Набор узлов, имеющих статус «частичный» (пересекающий поверхность объекта), можно считать воксельным представлением поверхности. Также при использовании восьмеричного дерева можно повышать или понижать разрешение представления поверхности.

Таким образом можно получить 6-разделяющую поверхность трехмерного объекта, т.е. через эту поверхность не может проникнуть 6-связная линия. Эта характеристика будет учтена при заполнении полученной поверхности.

Теперь, когда поверхность объекта представлена дискретно, т.е. набором элементов, можно воспользоваться алгоритмом заполнения. Он напоминает рекурсивный алгоритм закрашивания замкнутого контура, описанный в [12]. Разница состоит в том, что мы его реализуем в трехмерном пространстве, а не в двухмерном. Но наиболее используемым является алгоритм закрашивания линиями. «От приведенного ранее простейшего рекурсивного алгоритма он отличается тем, что на каждом шаге закрашивания рисуется горизонтальная линия, которая размещается между пикселями контура» [12]. Такой же подход можно применить и в нашем случае, только вместо одной линии будем строить сразу шесть линий, состоящих из вокселей (в соответствующем направлении каждой грани куба).

По мнению автора, данный гибридный алгоритм позволяет наиболее эффективно получать воксельное представление объектов, при этом решая задачу вокселизации поверхности и предоставляя возможность смены разрешения (с помощью перехода между уровнями восьмеричного дерева).

Экспериментальные результаты

После проведения вокселизации каркасный объект представляет собой набор объемных элементов одинакового (и малого относительно размеров объекта) размера в случае массива вокселей или разного размера в случае представления объекта восьмеричным деревом. При этом задача вычисления объема представленного объекта становится тривиальной и сводится к сумме объемов элементарных элементов (умножение объема одного элемента на количество таких элементов). Это удобно для таких объектов, вычисление объема которых с помощью аналитических методов затруднено (фигуры со сложной поверхностью, там, где применяют интегрирование и т.п.). Но такое вычисление дает определенную погрешность, т.к. объект представляется с помощью набора вокселей только приближенно, т.е. аппроксимируется [13].

В связи с этим, появляются следующие возможность для численной оценки точности приближенного представления. Так как чем ближе объем, полученный приведенным методом, к аналитически вычисленному объему, тем точнее представлен объект (наиболее наглядно это отношение представить в процентах).

Для вычислений выберем объекты, которые могут быть точно представлены триангуляцией (чтобы она не вносила дополнительную погрешность в представление объекта). Это могут быть пирамида и усеченная пирамида. Далее кратко приведем полученные экспериментальные данные в следующем виде (уровень восьмеричного дерева, количество аппроксимирующих объект вокселей, погрешность в процентах):

Пирамида: (4, 112, 56,08%), (5, 1632, 20,01%), (6, 14280, 12,51%), (7, 119280, 8,65%), (8, 1016160, 2,72%).

Усеченная пирамида: (4, 256, 42,64%), (5, 2912, 18,44%), (6, 25296, 11,44%), (7, 213888, 6,40%), (8, 1783392, 2,44%).

Таким образом, полученные данные показывают, что точность представления объекта зависит от геометрии самого объекта и размера объемных элементов (при переходе на следующий уровень длина ребра куба уменьшается в два раза). Но на восьмом уровне восьмеричного дерева погрешность представления различных объектов составляет менее 3%. При дальнейшем повышении уровня детализации резко возрастает количество элементов, что ведет к завышенным расходам памяти для хранения структуры и вычислительных ресурсов для ее обработки.

Заключение

В результате проведенного исследования разработан и реализован гибридный фрактальный алгоритм получения воксельного представления трехмерного объекта, сочетающий в себе алгоритмы рекурсивного деления пространства и заполнения пространства объемными элементами.

Полученная иерархическая структура восьмеричного дерева позволяет менять разрешение объемного объекта путем перехода между уровнями, что позволяет оптимизировать визуализацию объектов.

Разработанный алгоритм может использоваться как средство вычисления объема каркасного объекта с различной точностью.

Проведенные вычислительные эксперименты позволили оценить точность представления объекта и показали, что для адекватного (с погрешностью не более 3%) представления объекта необходимо 8 уровней восьмеричного дерева.

В дальнейшем планируется оптимизировать программную реализацию предложенного алгоритма для повышения скорости работы и точности получаемых результатов.

Список литературы

1. www.ixbt.com
2. Голованов Н.Н. Геометрическое моделирование. – М.: Издательство физико-математической литературы, 2002. – 472 с.
3. Requicha A.A.G. Representations of rigid solids: theory, methods and systems // ACM Computing Surveys, 1980. 12(4).
4. Kaufman A., Cohen D., Yagel R. Volume graphics / IEEE Computer. Vol. 26 No. 7, 1993.
5. Farin G., Hoschek J., Kim M.-S. Handbook of computer aided geometric design, Elsevier Science Publishers, 2002.
6. Meagher D. J. R. Geometric Modeling Using Octree Encoding, Computer Graphics and Image Processing, 1982. Vol. 19, No. 2.
7. Кроновер Р. М. Фракталы и хаос в динамических системах. Основы теории. – М.: Постмаркет, 2000. – 352 с.
8. Бодин О., Кузьмин А. Разработка фрактального алгоритма для построения трехмерной модели сердца// САПР и графика, 2005, № 3.
9. Dachille F., Kaufman A. Incremental Triangle Voxelization. Proc. Graphics Interface 2000, P. 205-212.
10. Huang J., Yagel R., Filippov V., Kurzion R. An Accurate Method For Voxelizing Polygon Meshes. Proceedings of the 1998 Symposium on Volume Visualization, pages 119-126. ACM SIGGRAPH, October 1998. P. 119–126.
11. Stolte N. Robust Voxelization of surfaces. Technical Report. TR.97.06.23, State University of New York at Stony Brook, 1997.
12. Порев В.Н. Компьютерная графика. – СПб.: БХВ-Петербург, 2004. – 432 с.
13. Игнатенко А. Геометрическое моделирование сплошных тел. CGM Journal 10/01/2003 (www.cgm.graphicon.ru).

РАЗВИТИЕ СИСТЕМЫ ПАРАЛАБ ДЛЯ ИЗУЧЕНИЯ И ИССЛЕДОВАНИЯ ПАРАЛЛЕЛЬНЫХ МЕТОДОВ РЕШЕНИЯ СЛОЖНЫХ ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ

Д.Ю. Лабутин, А.А. Лабутина, С.В. Ливерко

Нижегородский госуниверситет им. Н.И. Лобачевского

Характеристика системы

В работе представлен программный комплекс Паралаб для изучения и исследования параллельных методов решения сложных вычислительных задач, который отвечает следующим основным требованиям: позволяет проводить как реальные параллельные вычисления на многопроцессорной вычислительной системе, так и имитировать такие эксперименты на одном последовательном компьютере с визуализацией процесса решения сложной вычислительной задачи.

При проведении имитационных экспериментов предоставлена возможность для пользователя:

определить топологию параллельной вычислительной системы для проведения экспериментов, задать число процессоров в этой топологии, установить производительность процессоров, выбрать характеристики коммуникационной среды и способ коммуникации;

осуществить постановку вычислительной задачи, для которой в составе системы Паралаб имеются реализованные параллельные алгоритмы решения, выполнить задание параметров задачи;

выбрать параллельный метод для решения выбранной задачи;

установить параметры визуализации для выбора желаемого темпа демонстрации, способа отображения пересылаемых между процессорами данных, степени детальности визуализации выполняемых параллельных вычислений;

выполнить эксперимент для параллельного решения выбранной задачи. При этом в системе Паралаб предусмотрена возможность сформировать несколько различных заданий для проведения экспериментов с отличающимися типами многопроцессорных систем, задач или методов параллельных вычислений, для которых выполнение эксперимента может происходить одновременно (в режиме разделения времени). Одновременное выполнение эксперимента для нескольких заданий позволяет сравнивать динамику решения задачи различными методами, на разных топологиях, с разными параметрами исходной задачи. Для организации анализа полученных данных при проведении длительных вычислений, система обеспечивает возможность проведения серии экспериментов в автоматическом режиме с запоминанием результатов в журнале экспериментов;

накапливать и анализировать результаты выполненных экспериментов: по ним система позволяет построить графики зависимостей, характеризующих параллельные вычисления (времени решения, ускорения, эффективности) от параметров задачи и вычислительной системы.

Одной из важнейших характеристик системы является возможность выбора способов проведения экспериментов. Эксперимент можно выполнить в режиме имитации, т.е. на одном процессоре без использования каких-либо специальных программных средств типа библиотек передачи сообщений.

Кроме того, в рамках системы ПараЛаб обеспечивается возможность проведения вычислительного эксперимента в режиме удаленного доступа к вычислительному кластеру.

Для реальных экспериментов на многопроцессорных вычислительных системах зависимости времени и ускорения от параметров задачи и вычислительной системы строятся по набору результатов проведенных экспериментов.

При построении зависимостей для экспериментов в режиме имитации используются теоретические оценки в соответствии с имеющимися моделями параллельных вычислений. Используемые модели отображают реальные процессы, протекающие при решении вычислительной задачи при помощи параллельного алгоритма. Для каждого из проведенных экспериментов имеется возможность восстановить его для повторного проведения. Кроме того, обеспечено ведение журнала экспериментов с записью туда постановки задачи, параметров вычислительной системы и полученных результатов.

Реализованные таким образом процессы позволяют освоить теоретические положения и помогают формированию представлений о методах построения параллельных алгоритмов, ориентированных на решение конкретных прикладных задач.

Поддержка дополнительных модулей (плагинов) в системе ПараЛаб

Текущая версия системы ПараЛаб допускает расширение функциональности путем добавления модулей для решения новых задач, не входящих в стандартную комплектацию системы. Для этого предусмотрена возможность написания дополнительных библиотек (плагинов) сторонними разработчиками.

Система ПараЛаб разработана на платформе Microsoft .NET Framework с использованием языка C# и объектно-ориентированного подхода. Каждая задача (например задача умножения матриц) в системе ПараЛаб представлена абстрактным классом-наследником класса Task (MatrixMultTask: Task). Все поддерживаемые методы решения задачи представляются как классы, наследуемые от задачи (например FoxMatrixMultTask: MatrixMultTask – алгоритм Фокса для умножения матриц).

Чтобы добавить в систему новый тип задач, необходимо добавить в рабочий каталог системы ПараЛаб файлы динамических библиотек (DLL), представляющих собой библиотеки классов Microsoft .NET Framework. В одном из этих файлов должен содержаться класс, производный от абстрактного класса Task, который описывает новую задачу. В остальных добавленных файлах должны быть реализованы классы-наследники от этого промежуточного класса, которые будут соответствовать конкретным методам решения задач этого типа.

Все необходимые средства для разработки дополнительных модулей (абстрактный класс Task, классы топологий и описание их отдельных элементов, класс для работы с графами и многое другое) находятся в сборке ParaLabSDK.dll, которая входит в комплект поставки системы ParaLab и необходима для ее работы. Этую сборку необходимо подключить к проекту Visual Studio, в котором создается новый плагин.

Также можно подключить одну из сборок, соответствующих задачам из стандартной поставки (ArraySortTask.dll, MatrixMultTask.dll, GraphProcessingTask.dll), чтобы разработать реализацию новых методов решения этих задач. Следует отметить, что все задачи и методы из стандартной поставки системы ПараЛаб реализованы в полном соответствии с приведенной схемой в виде плагинов.

Класс Task содержит набор абстрактных методов и свойств, через которые система ПараЛаб взаимодействуют с объектами-задачами и которые должны быть реализованы в дочерних классах. Часть из них обычно реализуется в классе, соответствующем самой задаче, а остальные – в классе, соответствующем конкретному методу решения задачи. Как именно распределить методы по этим классам, определяется особенностями решения задачи реализуемыми методами.

Итак, создание описанных выше сборок и помещение их в рабочий каталог системы – это все необходимо для добавления поддержки в системе ПараЛаб новой задачи. После запуска система ПараЛаб произведет сканирование файлов с расширением DLL в своем рабочем каталоге и, используя механизм рефлексии типов .NET, определит те из них, которые содержат описания абстрактных классов, наследуемых от ParaLab.Task – они будут соответствовать добавляемым типам задач. Неабстрактные же классы, наследуемые от них, соответственно, определяют конкретные методы решения и в главное меню включены не будут. Также в меню не будут включены задачи, для которых не найдено ни одного метода и методы, которым ни соответствует ни одна задача.

Поддержка многопроцессорных и многоядерных архитектур в системе Пара-Лаб

Последняя версия системы ПараЛаб позволяет эмулировать параллельные эксперименты не только на системе, состоящей из «простых» однопроцессорных и одноядерных компьютеров. Каждый компьютер может содержать несколько процессоров, а каждый процессор – несколько ядер. Ядро системы не ограничивает количество ядер и процессоров, но для удобства визуализации были введены следующие ограничения: число процессоров в рамках одного вычислительного узла (сервера) может равняться 1 или 2, а число ядер в процессоре – 1, 2 или 4.

При этом процесс обмена данными между ядрами разделяется на 3 этапа:

1. В рамках одного вычислительного узла (в том числе между ядрами разных процессоров), а также отправка ядрами данных во внешнюю сеть (через сетевой адаптер).

2. Передача данных между отдельными серверами по локальной сети. При этом, если все данные в пакете были отправлены одним ядром, то этот пакет визуализируется отдельным конвертом, если же в нем находятся данные сразу от нескольких ядер – то пачкой конвертов.

3. Прием данных в рамках одного вычислительного узла (от сетевого адаптера).

В связи с тем, что архитектура вычислительной системы существенно усложнилась (по сравнению с более ранними версиями системы ПараЛаб), использование заранее определенных формул для подсчета времени обмена данными невозможно. С этой целью используются результаты имитации. Разработчик модуля визуализации задачи должен указать следующее.

1. Время выполнения базовой вычислительной операции алгоритма (например, для алгоритмов сортировки – это операция сравнения); время выполнения операции должно быть выражено в числе стандартных операций с вещественными числами.

2. Количество базовых операций, которые выполняются каждым ядром вычислительной системы на каждой итерации алгоритма.

3. Размер единицы обмена данными, выраженный в байтах.

4. При формировании списка пар ядер, которые будут обмениваться данными на текущей итерации, необходимо указать не только номера ядра-отправителя и ядра-получателя, но и объем передаваемых в этой паре данных в условных единицах.

По этой информации можно произвести автоматизированный расчет времени, потраченного на вычисления и на передачу данных.

Как и ранее, для упрощения модели предполагается, что процессы передачи данных и выполнения операций ядрами разделены во времени: на время выполнения обмена данными вычисления на всех ядрах приостанавливаются.

РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ РАСПРЕДЕЛЕННОЙ ТРАССИРОВКИ ЛУЧЕЙ НА БАЗЕ ИНСТРУМЕНТАРИЯ ALCHEMI

Д.Ю. Лабутин, Д.К. Боголепов, А.А. Алехин

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

К настоящему времени разработано множество алгоритмов расчета освещенности искусственных трехмерных сцен. В данной работе рассматривается реализация одного из таких алгоритмов – *метод трассировки лучей*, который моделирует процесс распространения световых лучей от момента их испускания источниками до взаимодействия с объектами сцены и попадания в приемник. Данный метод является основным для синтеза высококачественных компьютерных изображений. Следует заметить, что расчет компьютерного изображения с помощью метода трассировки лучей является трудной вычислительной задачей. Он требует проведения огромного числа векторных операций, поэтому обработка сложного изображения на одном компьютере может занимать значительное время. Естественным способом преодоления этой трудности является использование *систем распределенных вычислений*. В работе рассмотрены основы метода трассировки лучей, а также реализация данного метода для проведения расчетов на распределенной вычислительной системе.

В качестве такой системы был выбран инструментарий Alchemi (<http://www.alchemi.net>), разрабатываемый в университете Мельбурна и реализующий ряд концепций грид-компьютинга. Данный инструментарий интересен тем, что предназначен для операционной системы Microsoft Windows, тогда как большинство аналогичных решений рассчитаны на использование в операционных системах класса Unix. Реализация инструментария Alchemi использует передовые возможности платформы Microsoft .NET Framework, наглядно демонстрируя ее перспективы в области разработки распределенных систем (в том числе систем грид-компьютинга), ориентированных на работу в сети Интернет. Следует также заметить, что данный инструментарий отличается весьма изящной моделью, близкой к традиционному многопоточному программированию, которая позволяет с легкостью разработать новое приложение для грид.

Результатом проделанной работы является распределенное приложение, реализующее классический алгоритм трассировки лучей. (Реализация данного приложения была проделана в достаточно короткие сроки.) При использовании инструментария Alchemi программист может полностью сосредоточиться на основной логике приложения, не отвлекаясь на вопросы функционирования вычислительной грид. Приложение может работать одновременно на нескольких рабочих станциях, которые средствами инструментария быстро и довольно просто объединяются в производительную вычислительную сеть, даже если они произвольным образом распределены по всему миру. Следует также отметить, что новые вычислительные ресурсы могут добавляться во время выполнения приложения.

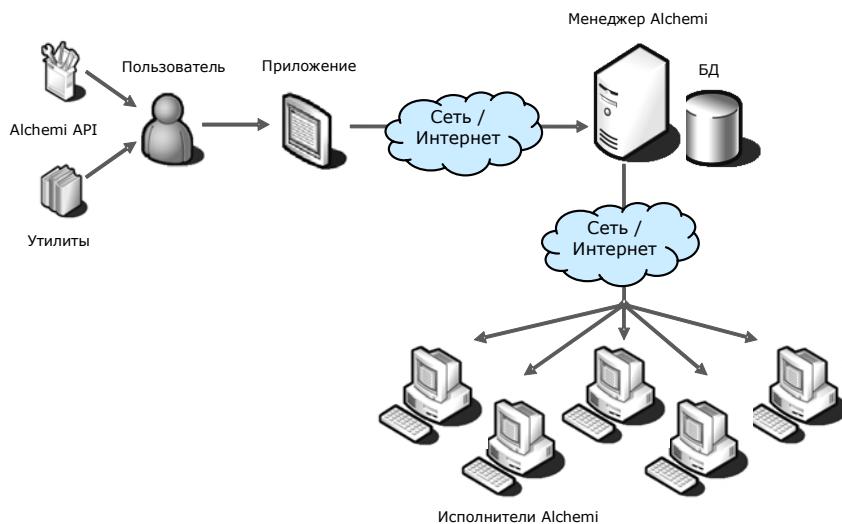
Вначале будут рассмотрены основы метода трассировки лучей. Затем будет дана краткая характеристика инструментария Alchemi для грид-компьютинга. В заключении приводятся результаты тестирования производительности и некоторые выводы относительно использования распределенных вычислений.

Инструментарий Alchemi

В вычислительной грид, построенной на базе инструментария Alchemi, участвуют следующие *четыре компонента*: Менеджер (Manager), Исполнитель (Executor), Пользователь (User) и Межплатформенный Менеджер (Cross-Platform Manager). Взаимосвязи этих сущностей показаны на рисунке.

Как видно из представленной схемы, инструментарий Alchemi построен по клиент-серверной схеме. Для объединения некоторой совокупности компьютеров в вычислительную сеть, нужно выбрать один узел, который будет играть роль сервера, и установить на него Менеджер. На один или несколько компьютеров сети устанавливаются клиенты – Исполнители, которые настраиваются на работу с Менеджером для выполнения вычислений. В дистрибутив Alchemi входят удобные программы инсталляции, требующие минимальной настройки, так что процесс создания Alchemi-грид является достаточно простым.

Пользователи могут разрабатывать, выполнять и осуществлять мониторинг грид-приложений, используя специальный API и инstrumentальные средства, включенные в Alchemi SDK.



Благодаря характеру инфраструктуры грид (слабосвязанные, гетерогенные ресурсы, взаимодействующие по ненадежному, с высокой латентностью каналу), грид-приложения должны обладать следующими особенностями:

- распараллеливаться на множество независимых вычислительных частей;
- параллельные части в основном работают над вычислениями, а не над установкой соединения.

Разработка грид-приложения может рассматриваться как написание распределенной программы для абстрактной «виртуальной» машины с множеством центральных процессоров. Основная задача различного промежуточного программного обеспечения состоит в том, чтобы предоставить программисту эту абстракцию. В различном программном обеспечении промежуточного уровня это делается по-разному. В Alchemi выбран способ, упрощающий процесс разработки программного обеспечения для грид насколько это возможно.

Достигается это во многом благодаря объектно-ориентированной модели программирования, близкой к традиционному многопоточному программированию. Элементарной единицей параллельного исполнения в этом случае является *грид-поток*; множество грид-потоков образуют *грид-приложение*.

Разработчики, у которых нет необходимости досконально знать тонкости функционирования грид, могут полностью сконцентрироваться на основной логике и манипулировать такими абстракциями, как грид-приложение и грид-поток. Кроме того, данные абстракции позволяют использовать такие конструкции языка программирования, как события, применительно как к локальному, так и к удаленному коду. Вся эта функциональность доступна через Alchemi .NET API.

Для разработки нового приложения необходимо выполнить следующие основные шаги.

Произвести анализ имеющейся вычислительной схемы и осуществить ее разделение на части, которые могут быть выполнены независимо друг от друга. В инструментарии Alchemi этим частям соответствуют грид-потоки.

Реализовать с использованием Alchemi API класс грид-потока. В процессе работы программы грид-потоки будут запускаться на удаленных узлах для выполнения части вычислений.

Организовать обработку потоков в грид с использованием класса грид-приложения. Для этого необходимо выполнить подключение к грид-сети, создать необходимое число грид-потоков и добавить их к грид-приложению и, наконец, запустить его на выполнение.

Реализации приложения распределенной трассировки лучей

За основу был взят классический метод трассировки лучей, реализованный для платформы Microsoft .NET на языке программирования Visual C#. Данная реализация позволяет рассчитывать независимо цвет каждого пикселя изображения, имея на входе только описание трехмерной сцены. На основе этой реализации была построена первая версия приложения, не поддерживающая распределение вычислений. Целью первого этапа являлась реализация отлаженного кода алгоритма трассировки лучей.

Целью второго этапа являлось распараллеливание вычислений для проведения расчетов на основе инструментария Alchemi, при этом предложено три способа распределения вычислений.

Очевидный вариант распределения вычислений состоит в разбиении изображения на отдельные пиксели и проведение расчетов отдельно для каждого пикселя. Однако данный подход является крайне неэффективным, т.к. для каждого пикселя необходимо пересыпать на вычислительные узлы сцену, что является источником больших наклад-

ных расходов по сравнению с затратами на обработку одного пикселя. К тому же количество доступных вычислительных узлов обычно меньше на несколько порядков по сравнению с числом пикселей. Выход из этой ситуации вполне очевиден: отдельные пиксели нужно объединять в группы.

Вторым предложением было разделение изображения на прямоугольные блоки, реализация его оказалась несложной. Тестирование показало, что данная реализация хорошо работает при количестве потоков в несколько раз (от 3 до 8) превышающих количество вычислительных узлов. Однако, при количестве потоков сопоставимом с числом узлов (преследовались цели минимизации накладных расходов) было замечено, что некоторые блоки изображения рассчитываются существеннее других, в результате чего загрузка вычислительных мощностей была не полной. Причинами данного явления было то, что вычислительные затраты на обработку каждого пикселя не одинаковы, это приводило к неравномерной нагрузке вычислительных узлов. Однако мы обрабатывали не отдельные пиксели, а несколько тысяч пикселей, что должно было как-то сгладить неравномерность вычислительных затрат. Но блоки изображения содержали целые области пикселей с очень низкими или высокими затратами на обработку.

Данный факт заставил задуматься о способах борьбы с данным явлением. Необходимо было модифицировать реализацию таким образом, чтобы нагрузка на вычислительные узлы была одинаковой. В параллельном программировании данную проблему решают с помощью механизмов динамической балансировки нагрузки, но простых реализаций таких механизмов для инструментария Alchemi не существует. Тем не менее, решение проблемы было найдено: был использован так называемый *принцип когерентности*, который говорит о том, что вероятность совпадения затрат на обработку соседних пикселей близка к единице. В соответствии с этим принципом и был предложен другой способ объединения пикселей в блоки. Он состоял в том, что блоки надо было создавать разреженными. Это приводило к тому, что соседние пиксели попадали в различные блоки и в результате выравнивало затраты на расчет блоков в соответствии с принципом когерентности.

Приведем конфигурацию рабочих станций, использованных для тестирования приложения.

Название компоненты	Характеристики
Операционная система	Microsoft Windows XP SP2
Процессор	Pentium Dual-Core 2.8 ГГц
Оперативная память	2048 Мб
Видеокарта	Radeon X300 Series
Сетевое соединение	Gigabit Ethernet

Приведем сводную таблицу.

Номер сцены	Время работы (секунд)			
	1 Исполнитель	2 Исполнителя	3 Исполнителя	4 Исполнителя
Сцена № 1	35	18	14	12
Сцена № 2	105	60	40	35
Сцена № 3	610	344	272	185

Выводы

Результаты тестирования свидетельствуют о получении трехкратного ускорения на четырех Исполнителях, что является весьма приличным результатом при небольших затратах. Платформа Alchemi .NET предоставляет множество удобств при использовании грид-сети. Вычислительная инфраструктура может формироваться динамически и меняться во время работы приложения. Новые Исполнители могут добавляться и отключаться, и скорость работы приложения будет изменяться в зависимости от количества активных исполнителей, при достаточном количестве потоков. Таким образом, платформу Microsoft .NET Remoting и основанную на ней грид-систему Alchemi можно считать подходящими инструментами для быстрого решения несложных задач.

Список литературы

1. Никулин Е. А. Компьютерная геометрия и алгоритмы машинной графики. СПб.: БХВ-Петербург, 2003. 560 с.
2. Alchemi User Guide.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ПОИСКОВОГО КОНСТРУИРОВАНИЯ И ИЗОБРЕТЕНИЯ НОВЫХ ВИБРАЦИОННЫХ УСТРОЙСТВ, ЭФФЕКТОВ И ТЕХНОЛОГИЙ «AIPS VIBRO»

Д.О. Ладыгин, В.А. Камаев

Волгоградский государственный технический университет

Успешное решение задачи ускорения научно-технического прогресса требует разработки и создания в новых вибрационных устройствах на уровне лучших мировых образцов для различных отраслей народного хозяйства. Разрабатываемая методика поискового конструирования вибрационных устройств (ВУ) включает в себя поиск, модернизацию известных и синтез принципиально новых конструкторских решений ВУ. В основе лежит банк данных по виброэффектам (ВЭ) и виброустройствам.

Цели разработки системы.

1. Повышение производительности труда инженеров-конструкторов и разработчиков новой техники (механиков и изобретателей) за счет автоматизации начальных стадий проектирования, связанных с синтезом технических решений ВУ на уровне изобретения.
2. Улучшение работы механизмов (модернизация).
3. Образовательная – помочь в обучении заинтересованных лиц и повышение креативности обучения.
4. Вибрационный эффект (его введение является изюминкой) определяется как взаимосвязь между двумя или более явлениями, происходящими при непосредственном участии вибраций. Для реализации необходимы определенные условия: причина эффекта и его следствие (например, эффекты вибрационного резания, перемещения, кристаллизации, очистки и т.д.).

Наглядной и полезной является схема представления ВЭ 3-хкомпонентной структурой:

$$A \rightarrow B \rightarrow C , \quad (1)$$

где A – вход, B – объект, C – выход.

Объект – тело или совокупность тел, представляющих собой определенную целостность независимо от их степени сложности (лед и ледокол, винт и гайка, газ и твердое тело). В качестве объекта чаще всего выступают макроскопические материальные тела, а также свойства материи, процессы и взаимодействия. Объект подвергается воздействиям со стороны окружающей среды, которые являются причинами возникновения тех или иных эффектов и называются входными воздействиями (входами). Реакция системы на входное воздействие – выход.

Такая трехкомпонентная система не всегда подходит для описания ВЭ, в которых происходит качественное изменение состояния объекта. Например, при описании эффекта вибрационного разрыхления (дробления, разрушения) какое состояние тела выбрать: твердое монолитное или сыпучее гранулированное; спекание поликристаллических порошков (смесь фаз или однофазное тело?) и т.д.

Если выбрать в качестве объекта его начальное состояние, это приведет к существенной потере информации при поиске и к принципиальным ошибкам при синтезе физических принципов действия.

Поэтому предлагается структурированное описание ВЭ по признакам входа, выхода и объекта по 4-хкомпонентной схеме:

$$A \rightarrow B_1 \rightarrow B_2 \rightarrow C , \quad (2)$$

где B_1 и B_2 – начальное и конечное состояние объекта. Одновременно расширяется и круг возможных запросов к системе, касающихся деталей перехода от состояния B_1 к состоянию B_2 : как вызвать переход $B_1 \rightarrow B_2$ и какие он может вызвать последствия, в какие B_2 может переходить B_1 , из каких B_1 может возникнуть B_2 и т.п. Отметим особо, что ранее описанные эффекты, в которых не происходит структурных изменений объекта, рассматриваются как частный случай новой схемы, когда $B_1 = B_2$.

Из анализа массива изобретений составляется таблица применений ВЭ при решении изобретательских задач, которая содержит два поля: требуемое действие или свойство (например, изменение положения или перемещение объекта, разделение смесей, создание больших давлений, разрушение) и название ВЭ (его номер).

Процесс ПОИСКА необходимого ВЭ или ВУ.

1. Выявление потребности: «Что нужно сделать?»: перемещение груза, дробление, измельчение, конденсация, очищение, преобразование, резание; совмещение перекачки жидкости и дегазации. Существует словарь ТФ. Какие эффекты могут это реализовать? Допустим, вышло 3 ВЭ: а что если совместить их и получить новое ВУ?!

2. Формирование технического задания (ТЗ – список требований и ограничений, которым ВУ должно удовлетворять) на основе базы данных. ТЗ включает в себя определение множества возможных входов A и ожидаемых выходов C разрабатываемой технической системы или технологии, выделение множества допустимых и запрещенных воздействий, ВЭ и объектов B): с какой скоростью перемещать, в каком направлении, максимальная масса перемещаемого груза.

Дальше идет запрос к системе и она выдает все ВЭ, удовлетворяющие ТЗ. Поиск нужного ВЭ в базе данных может производиться по любому компоненту (входному или выходному воздействию, объекту) или по любой их комбинации. (Кроме того, в модель добавляется кодировка с помощью свободно выбираемых дескрипторов (ключевые сло-

ва и словосочетания). Например, необходимо выяснить, как можно провести дегазацию расплава металла. Анализ запроса показывает, что поиску подлежат ВЭ с известным объектом (смесь расплава металла и газа), с известным выходом (уменьшение концентрации газа), а вход при этом предусматривается произвольным, т.к. ответы АИПС в данном случае и есть искомый результат.

Или «как преобразовать оптическое излучение в звук?» Поиску подлежат эффекты с известным входом (электромагнитное излучение) и с известным выходом (упругие (акустические) волны).

Комбинируя различные признаки входа, выхода, технической функции и объекта ВЭ, система может обрабатывать достаточно широкий круг запросов.

- 1) Каким образом можно разрушить горную породу (пласт земли и т.д.)?
- 2) Возможно ли уменьшение объёма, занимаемого телом? (виброуплотнение).
- 3) Как можно разделить смесь по фракциям (например бетон), используя вибрацию?
- 4) Как вибрацией можно рассеять туман, очистить воздух от пыли?
- 5) Можно ли получить смесь из 2-х или более несмешивающихся (трудно смешивающихся) жидкостей?
- 6) Как применить вибрацию для создания антифрикционных материалов?
- 7) Возможно ли передать звук на большое расстояние?
- 8) Как усилить радиосигнал?

Режим МОДЕРНИЗАЦИИ. Используется база данных по эвристическим приемам, которые представляют собой предписание или указание, как преобразовать имеющиеся ВУ с целью устранения различных технических противоречий. Пользователь формулирует противоречие и с помощью программы предоставляет возможные способы его разрешения. Устранение технического противоречия осуществляется с постановкой достигаемой цели. Для упрощения все цели классифицируются и объединяются в группы по признакам: количественные изменения, преобразования формы, преобразования в пространстве или во времени, уменьшить вредное влияние вибраций, увеличить плавность запуска и останова, уменьшить потери, увеличить КПД. Группы целей содержат непосредственно описание режима модернизации, т.е. цель с указанием объекта действия (уменьшить потери на трение, магнитные потери, электрические потери, потери жидкой рабочей среды). А советы АИПС выглядят следующим образом: заменить источник энергии, концентратор выполнить в форме пирамиды, концентратор выполнить в форме конической пружины.

Технологическая схема процесса СИНТЕЗА новых ВУ предполагает выявление потребности (перемещение груза, дробление, измельчение), формирование технического задания (список требований, которым ВУ должно удовлетворять), синтез потоковой функциональной структуры (способ преобразования одного вида энергии в другой: преобразование энергии вращательного движения в энергию поступательного движения), синтез физической структуры (описание на уровне конкретных ВЭ, где возможно данное преобразование энергии: виброрезания, виброперемещения), синтез конструктивной функциональной структуры (модель проектируемого ВУ в виде некоторой структуры элементов), анализ полученного конструкторского решения.

Работа в системе будет «провоцировать» пользователя на создание новых изделий и технологий (синтез вибровозбудителей, вибропреобразователей). Используется метод индукции: возьмем 2 (или более) ВЭ и получим новое техническое решение. Например, соединим ветроустановку с синхронизацией; или процесс резания пластмассы – пусть смазывающая охлаждающая жидкость добавляет автоколебания (это приведет к лучше-

му дроблению стружки); перекачку жидкости совместим с одновременной ее дегазацией; загрузка/разгрузка с одновременной сегрегацией; виброкаток (сообщим колебания колесам обычного катка, что позволит повысить производительность асфальтоукладки) и т.д.

Разработанная система генерации подсказок для разработчика («Интеллектуальный советчик») написана на C++, работает под операционными системами Microsoft Windows XP/2003, при помощи MS Office готовятся оригинальные изобретения, а Microsoft SQL Server позволяет обрабатывать запросы к базам данных программы.

Уже сейчас пилотные варианты внедрены в учебный процесс для проведения лабораторных работ по курсу «Методы инженерного творчества», где студенты реально изобретают! Основное ядро системы инвариантно и позволяет оперативно перестроить ее с генерации ВУ на любой тип технических решений без ограничения относительно типов и классов.

Список литературы

1. Фоменков С.А., Давыдов Д.А., Камаев В.А. Моделирование и автоматизированное использование структурированных физических знаний. – М.: Машиностроение, 2004. 297 с.

СОВЕРШЕНСТВОВАНИЕ УЧЕТА ПЕРЕГОВОРОВ С ПОТЕНЦИАЛЬНЫМИ КЛИЕНТАМИ С ЦЕЛЬЮ УВЕЛИЧЕНИЯ ОБЪЕМА ПРОДАЖ ФИРМЫ

В.Г. Маврин, И.В. Макарова

Камская государственная инженерно-экономическая академия

В условиях рыночной экономики и конкуренции фирма сталкивается с проблемой реализации продукции. Для её решения необходимо не только снижать стоимость продукции, улучшать ее качество, но и привлекать потребителей. В связи с этим особую роль приобретает бизнес-процесс «Маркетинг», с помощью которого предприятие при условии применения информационных технологий в состоянии занять лидирующие позиции на рынке производителей товаров и услуг посредством грамотно поставленной работы со своей клиентурой. Так, одним из способов решения проблемы сбыта может являться привлечение потенциальных клиентов. Российский опыт показывает, что при успешной реализации этой задачи прирост объема продаж фирмы может составить до 13%. По нашему мнению, этого достичь невозможно без применения информационных технологий, особенно для крупных организаций, у которых список потенциальных клиентов может доходить до десятков тысяч. Возникает проблема хранения данных, поиска информации, планирования, учета переговоров, предоставление отчетности. Во избежание этих проблем фирма должна использовать интерактивную систему учета переговоров с потенциальными потребителями продукции фирмы. С её помощью у пользователя должна быть возможность внесения данных по предприятиям (потенциальным клиен-

там), их изменения и поиска, ведения учета и планирования переговоров с каждым предприятием.

В результате, перед нами была поставлена задача создания такой системы, с возможностью работы в сети. Интерфейс приложения разработан в Borland Delphi 7.0. База данных – Microsoft Access. Доступ к базе данных осуществляется с помощью компонент ADO. Выбор данных средств разработки обусловлен тем, что многие организации имеют устаревшую производственно-техническую базу. Созданное таким образом приложение не требовательно к аппаратной части ПК.

База данных состоит из 8-ми таблиц для хранения соответствующих списков:

Country – стран;

Okrug – федеральных округов по каждой стране;

Subject – субъектов федерации по каждому округу;

Town – городов по каждому субъекту федерации;

PP – предприятий по каждому городу;

K_Litso – контактных лиц по каждому предприятию;

Manager – сотрудников, ведущих переговоры с потенциальными клиентами.

Zvonok – звонков с потенциальными клиентами.

В данную базу можно вносить данные предприятий (потенциальных клиентов): название предприятия, город, должность, ФИО, телефон и e-mail руководителя и контактных лиц, уставной фонд (для определения масштаба предприятия), приоритет предприятия для звонящей фирмы (например, может быть список значений: заинтересован, не нуждается в продукции фирмы, не определен (не дозвонились) и т.д.). Учет переговоров осуществляется путем внесения данных звонков: менеджер, ведущий переговоры с потенциальным клиентом; предприятие и контактное лицо, с которым ведутся переговоры; дата и время текущего и следующего звонков; содержание разговора.

Достоинствами данной базы является то, что в ней исключается дублирование информации, что снижает время выполнения операций и объем базы, она отличается информативностью и содержит значимые для фирмы, ведущей переговоры с потенциальными клиентами, поля. Кроме того, введение других полей не приведет к заметному увеличению времени обработки данных.

Для расчета эффективности решения задачи проведения переговоров с потенциальными клиентами с помощью данной программы может использоваться формула:

$$\mathcal{E}_n = \frac{\Pi'}{\Pi},$$

где \mathcal{E}_n – эффективность переговоров с потенциальными клиентами, Π' – количество предприятий, с которыми ведутся преддоговорные работы, Π – общее число предприятий, с которыми велись переговоры. Программный продукт предоставляет возможность выведения отчета, необходимого для расчета эффективности.

Внедрение программы в работу предприятия может дать следующие виды эффективов:

- 1) управленческий эффект заключается в уменьшении затрачиваемого времени на решение задачи, изменении качественных характеристик труда;
- 2) экономический эффект – затраты на внедрение программного продукта должны окупиться в результате экономии трудозатрат;
- 3) синергетический эффект – результаты решения задачи учета переговоров с потенциальными клиентами используются другими задачами деловых процессов, при их более эффективном учете может качественнее выполняться проведение переговоров. Это может привести к увеличению продаж фирмы, её выручки и прибыли.

ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ РЕКЛАМАЦИЙ ПО ОТКАЗАМ ГАРАНТИЙНОЙ АВТОМОБИЛЬНОЙ ТЕХНИКИ КАМАЗ В СОСТАВЕ СИСТЕМЫ ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА ПРЕДПРИЯТИЯ ПОСРЕДСТВОМ СУБД MS SQL SERVER 2000

И.В. Макарова, А.И. Беляев

Камская государственная инженерно-экономическая академия

Постановка задачи

Формирование единого информационного пространства предприятия автомобильного сервиса на основе электронного документооборота – одна из задач создания информационной системы любой организации. Современные системы электронного документооборота позволяют решать большой круг задач, направленный на уменьшение потерь времени при передаче информации между подразделениями предприятия, а также на разграничение прав доступа организационных единиц к важной информации.

Одним из направлений деятельности современного автосервисного предприятия является выполнение гарантийных обязательств по качеству автомобильной техники. В ходе выполнения гарантийного послепродажного обслуживания персонал технического отдела получает, оформляет и пересыпает множество различных документов, имея корреспонденцию как внутри предприятия, так и за его пределами. При этом сведения таких документов постоянно дублируются, поэтому их ручное заполнение требует значительных затрат времени инженеров технического отдела. Введение системы электронного документооборота позволяет минимизировать потери времени при оформлении и прохождении документов между подразделениями и службами и, как следствие, повысить эффективность работы технического отдела.

Наиболее важным этапом проектирования системы электронного документооборота автосервисного предприятия является проектирование подсистемы учета гарантийного обслуживания. Упрощенная схема движения документов в процессе гарантийного обслуживания представлена на рис. 1, где символами конвертов обозначены электронные документы, передаваемые с помощью системы корпоративной электронной почты.

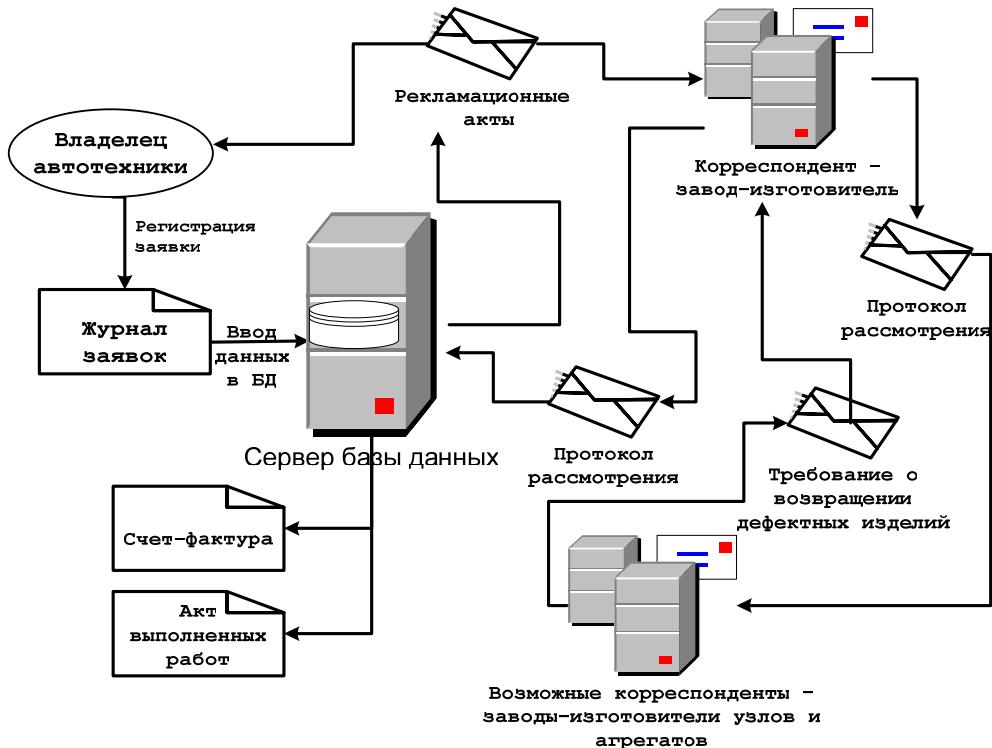


Рис. 1. Схема электронного документооборота технического отдела

Первой и самой сложной задачей при проектировании подсистемы учета гарантийного обслуживания является формирование базы данных обращений владельцев зарекламированной автомобильной техники для постановки на гарантийное обслуживание. Моделирование и ведение информационной базы позволяют на основании точно сформулированных предпосылок с учетом практических результатов выводить логические следствия и рекомендации, необходимые для обоснования и принятия управлеченческих решений. (Исследования такого рода позволяют выполнять логический и количественный анализ сложных процессов гарантийного обслуживания, зависящих от множества факторов.) Статистическая обработка опытных данных и анализ закономерностей, установленных в процессе такой обработки, позволяют прогнозировать техническое состояние и вырабатывать рекомендации по повышению надежности узлов и агрегатов автомобильной техники. Выявление количественных закономерностей, свойственных процессам, происходящим при эксплуатации технических систем, на основе данных по периодичности отказов узлов автомобильной техники позволяет определить перспективное состояние, а также оптимизировать объем и периодичность гарантийного обслуживания.

Таким образом, направлением нашей разработки является проектирование и создание программного продукта, содержащего классификаторы отказов автомобильной техники, а также основных узлов и агрегатов автомобилей КАМАЗ. Это позволяет осуществлять ввод первичной информации по обращениям владельцев автомобильной техники, составлять текущую и сводную отчетность на заданные периоды времени, а также

проводить статистический анализ данных, хранящихся в информационной базе. В качестве объекта исследования было выбрано предприятие ОАО «Набережночелнинский автоцентр КАМАЗ», осуществляющее все виды сервисного обслуживания автомобильной техники.

Краткое описание проведенного исследования

В результате исследования предметной области была получена информационно-логическая модель базы данных. Анализ основных документов и технологических процессов управления (ТПУ) позволяет выявить направление и содержание входных и выходных информационных потоков в рамках технической службы предприятия, занимающейся послепродажным гарантийным обслуживанием автомобильной техники.



Рис. 2. Технология ввода и накопления информации

К входной информации относится условно-постоянная информация (справочники моделей автомобилей, типов отказов, владельцев и автомобилей, составных частей и деталей автомобилей) и оперативно-учетная информация (журнал заявок владельцев, прейскурант цен на услуги, запасные части, протоколы рассмотрения рекламаций). Выходными формами являются рекламационный акт, счет-фактура и акт выполненных работ, а также периодические отчеты о выполнении работ по гарантийному обслуживанию и составленным рекламационным актам. Технология организации ввода и накопления данных представлена на рис. 2.

Программная реализация задачи

Программная реализация задачи произведена в среде разработки приложений Borland Delphi 7 с использованием технологии доступа к данным ADO (ActiveX Data Object). Системой управления базами данных, содержащей таблицы и представления программного продукта, является СУБД Microsoft SQL Server 2000, основанная на языке запросов Transact-SQL.

Программный продукт представляет собой совокупность форм, отображающих табличную и запросную информацию базы данных. Пользовательский интерфейс исполнен в стандартном оформлении Windows. При запуске программы автоматически настраиваются элементы управления, находящиеся на главном окне, а также при запуске дочерних окон. Открывающееся окно дает непосредственный доступ к списку заявок владельцев автомобилей на гарантийный ремонт. Окно работы с приложением имеет вид, представленный на рис. 3.

№	Экспл. организация	Дата обращ.	Дата восстан.	Модель авто	VIN	Номер шасси	Номер двиг-ля	Пробег
5	ИП "ИВАНОВ ВЛАДИМИР ГРИГОРЬЕВИЧ"	13.06.2006	01.07.2006	КамАЗ - 65115	543643224321	6243231	3424541	13050
8	ООО "НИНЖНЕКАМСКГЭССТРОЙ", г. Ни	13.06.2006	01.07.2006	КамАЗ - 54115	293482359721	4235436	3244611	4500
7	ООО "НИНЖНЕКАМСКГЭССТРОЙ", г. Ни	12.06.2006	01.07.2006	КамАЗ - 54115	293482359721	4235436	3244611	4500
4	ОАО "Набережночелнинский элеватор", г. Н	12.06.2006	01.07.2006	КамАЗ - 5511	632423121352	1634243	3423562	11500
10	ЗАО "Форт-Диалог", г. Наб. Челны, Моск	12.06.2006	01.07.2006	КамАЗ - 4310	739832891019	1241312	1242394	32400
3	ОАО "Набережночелнинский элеватор", г. Н	11.06.2006	01.07.2006	КамАЗ - 454120	324745623891	5574923	2303266	10560
6	ИП "ИВАНОВ ВЛАДИМИР ГРИГОРЬЕВИЧ"	03.06.2006	01.07.2006	КамАЗ - 5410	735643213610	3443114	2355373	34010
12	ООО "АвтодорстройИИ", г. Наб. Челны, Мос	03.06.2006	01.07.2006	КамАЗ - 6520	243837248103	3294123	2314342	3245
11	ЗАО "Форт-Диалог", г. Наб. Челны, Моск	02.06.2006	01.07.2006	КамАЗ - 43114	432392874011	1234631	2133214	235
9	ООО "НИНЖНЕКАМСКГЭССТРОЙ", г. Ни	01.06.2006	01.07.2006	КамАЗ - 5320	932947351901	3243249	1233241	34000
1	ЗАО "Авторемдизель", г. Наб. Челны, пр. М	14.05.2006	01.06.2006	КамАЗ - 65228	127364189411	1244521	6452118	2456

Рис. 3. Общий вид программного продукта

Рабочее окно приложения состоит из заголовка окна, панели инструментов и таблицы. Элементы панели инструментов позволяют осуществить доступ к базе данных, работу с заявками владельцев, изменение представления записей, редактирование справочников и реестров.

Таблица представляет собой электронный журнал заявок владельцев, где представлена информация о заявке, владельце и автомобиле, находящемся на гарантийном обслуживании. При двойном щелчке на таблице открывается форма с описанием текущей заявки, которая позволяет редактировать номер, дату поступления заявки, информацию об автомобиле, а также отображает информацию обо всех дефектных узлах. Для редактирования полей предусмотрены необходимые элементы управления: для ввода номера заявки можно воспользоваться спаренной кнопкой последовательного ввода чисел, для выбора даты предусмотрен выпадающий календарь.

Для добавления нового дефекта в заявку необходимо щелкнуть по кнопке со знаком «+» в рамке «Перечень дефектных узлов», после чего произойдет составление нового рекламационного акта (номер присваивается автоматически, однако при желании можно его изменить). Для вывода печатной формы документов необходимо заполнить информационные поля на вкладках «Описание дефектного узла», «Используемые запасные части», «Калькуляция работ» и «Результат исследования дефектного узла», после чего следует нажать кнопку с именем соответствующего документа «Рекламационный акт», «Счет-фактура», «Акт выполненных работ». Для ввода дефектного узла его необходимо выбрать из формы классификатора узлов и составных частей автомобилей, который появляется при нажатии кнопки «...» около описания составной части (рис. 4). Для выбора дефектного изделия выбирают нижний уровень в дереве объектов и нажимают кнопку «Выбрать». Описание и номер составной части, группы деталей и подгруппы появятся вместе с описанием дефектного узла.

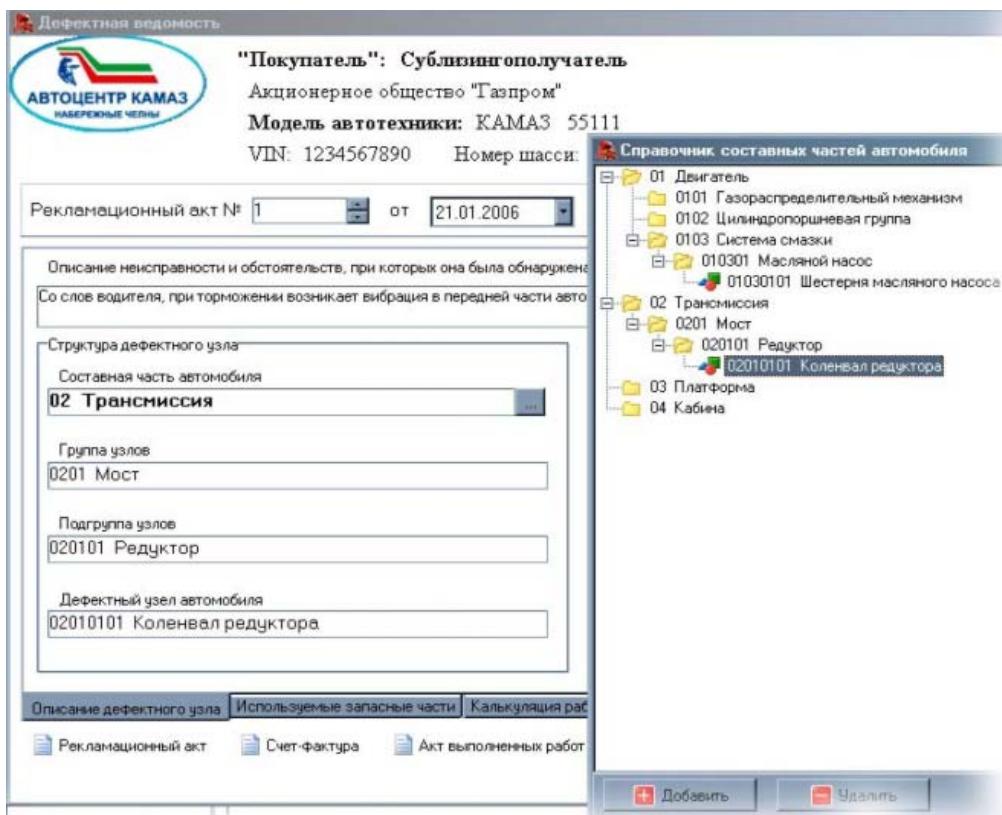


Рис. 4. Пример заполнения формы рекламационного акта

Заключение

Реализация данной задачи является первым шагом на пути к проектированию общей системы электронного документооборота. Заполнение информационной базы и анализ результатов деятельности предприятия на основе статистической обработки данных позволяет принимать обоснованные управленические решения по оптимизации работы постов гарантийного обслуживания, а также вырабатывать рекомендации по повышению качества выпускаемой автомобильной техники КамАЗ. Кроме того, данный программный продукт позволяет унифицировать все вторичные документы технической службы автоцентра, занимающейся послепродажным гарантийным обслуживанием автотехники, что сокращает время составления документов.

Список литературы

1. Решетник М. Информационная система предприятия: логистические принципы построения // Конъюнктура товарных рынков, I-II 2005. С. 121–126.
2. Процедура ТПУ 19.08.4-2002 Выполнение гарантийных обязательств по качеству автомобильной техники КамАЗ.
3. Техническая эксплуатация автомобилей: Учебник для вузов. 4-е изд., перераб. и дополн. / Е.С. Кузнецов, А.П. Болдин, В.М. Власов и др. М.: Наука, 2001. 535 с.

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ «БИБЛИОТЕКА» ПОСРЕДСТВОМ СУБД MS SQL SERVER 2000

И.В. Макарова, П.А. Буйвол

Камская государственная инженерно-экономическая академия

Введение

На пороге XXI века мир вступил в новую эпоху – эпоху информационного общества. Информационная революция открыла новую эпоху в прогрессе человечества. Индустриальное общество, существовавшее в XX веке, быстро сменяется информационным обществом XXI века. Человечество находится в процессе революции, причем возможно, самой значительной из всех пережитых человечеством. Это ставит новые задачи перед исследователями и, в первую очередь, задачи связаны с ростом информационных потоков. Только вдумайтесь: если во времена Маркса удвоение информации происходило через пятьдесят лет, то в наше время этот процесс занимает в среднем полтора года. Если раньше новости доходили до считанных единиц людей и с большим опозданием, то ныне информация буквально заливает земной шар, проникая в его самые отдаленные уголки. Это приводит к качественным сдвигам в экономике, науке, общественной жизни и культуре, вносит дух сотрудничества в человеческие отношения. Информация становится новым ресурсом человечества. Пророческими становятся слова Уинстона Черчилля «Кто владеет информацией – владеет миром». «Знание» действительно и в полной мере становится «силой», материально подтверждая крылатое выражение английского мыслителя Ф. Бэкона спустя более 150 лет.

Еще в 1986 г. Петрович Н.Т. [1] писал: «Мировой информационный поток катастрофически нарастает. Этот рост подчинен экспоненциальному закону $у=e^x$: пропорциональность между величиной потока и скоростью его нарастания. Чем больше поток, тем быстрее он нарастает, а чем быстрее он нарастает, тем еще стремительнее увеличивается и сам поток, и скорость его нарастания. Беда состоит еще и в том, что и сам поток, и его скорость нарастания уже сегодня настолько велики, что зачастую с ними уже не совладать. Если наблюдаемый сегодня экспоненциальный рост не замедлится, а сохранится таким же до 2000 года, то, по прогнозам, общий выброс информационного вулкана должен возрасти не в два-три раза, а в 30 раз!». Эти прогнозы оправдались в полной мере.

Возможность доступа к мировым информационным ресурсам, несомненно, имеет положительные стороны, но в то же время создает проблему поиска в этом огромном массиве нужной информации. Без применения электронных баз знаний и баз данных, способных ускорить поиск нужной информации, человечество рискует «захлебнуться» в этом потоке. Большинство фирм ведет базы данных клиентов, товаров, счетов, финансовых операций, однако внедрение подобных технологий в сферу услуг и обслуживания (в частности социальную сферу) позволит интенсифицировать процесс обслуживания и свести время, затраченное на поиск нужной информации до минимума.

Таким образом, проблема «ЧЕЛОВЕК – ИНФОРМАЦИЯ», несмотря на стремительный рост потока информации, может быть решена при некоторых условиях. Главными из них являются:

- содружество человека и быстродействующих ЭВМ по обработке, хранению и поиску информации;
- широкое использование быстродействующих каналов связи с выходными экранами, подключаемых к информационным ЭВМ, для запроса и просмотра интересующих сведений;
- достойное отношение к ценнейшему информационному ресурсу, новая тактика работы с ним, бережное отношение человека как к своему, так и чужому времени.

Каждый год объем производимой человечеством информации увеличивался в среднем на 30%, а среднестатистический житель США посвящает ее изучению 46% своего времени. К таким выводам пришли ученые Университета Беркли [2].

В наше время успешность бизнеса (и доходы), а также остальные области притязаний человека не в последнюю очередь зависят от своевременности и бесперебойности доступа к огромным объемам информации. Однако оперативность обработки такого объема имеет больший приоритет.

По прогнозу IDC расходы на приобретение систем хранения данных к 2007 году будут составлять от 67% до 75% корпоративных ИТ-бюджетов (несмотря на резкое уменьшение стоимости 1 МБ памяти, особенно за последние 3 года). Рынок систем хранения данных растет в 4 раза быстрее рынка ПК [3].

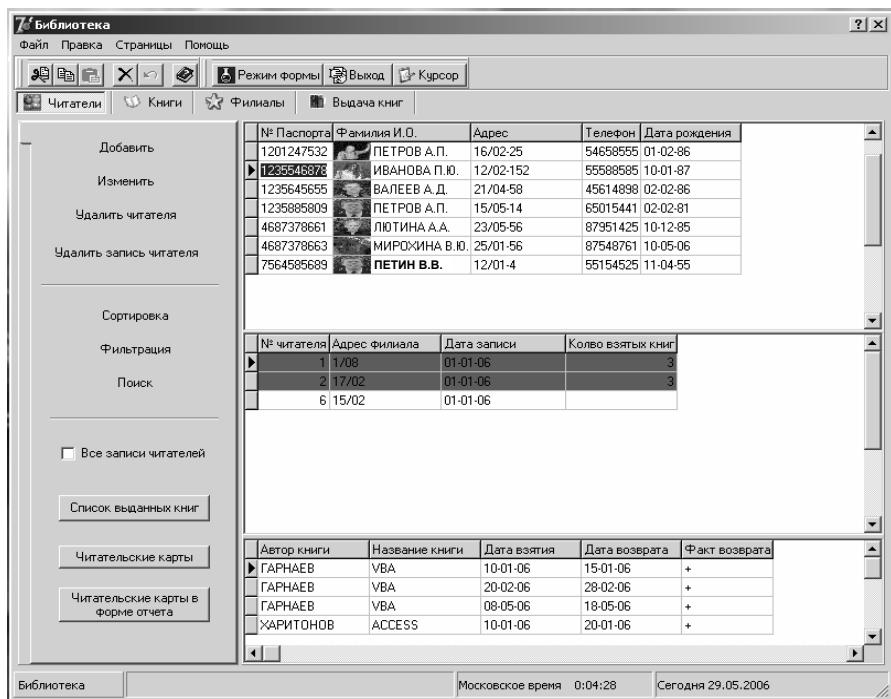
Краткое описание проведенного исследования

В этой связи представляется важной задачей организация работы электронных библиотек. Это позволит значительно быстрее узнать, где находится нужная книга, осуществить поиск нужных изданий по заданным параметрам. Кроме того, ведение учета посещений библиотеки повысит эффективность управления ее функционированием путем анализа интенсивности потока читателей и степени популярности отдельных экземпляров книг, чтобы удовлетворить потребности читателей в информационных ресурсах.

На базе таблиц MS SQL Server с языка программирования Delphi было разработано приложение, которое может быть использовано в библиотеке, состоящей из нескольких филиалов. Так как приложение рассчитано на корпоративное использование, то необходимо использовать серверную СУБД.

В связи с тем, что средой служит ОС MS Windows и имеется достаточно хороший провайдер Microsoft OLE DB Provider for SQL Server 4.0, то это будет СУБД MS SQL Server, поддерживаемая механизмом ADO. Интерфейс ADO реализован в рамках технологии Microsoft OLE DB, которая обеспечивает доступ к реляционным и нереляционным БД, а также к электронной почте, файловым системам и разного рода специализированным бизнес-объектам. Установка не должна создавать особых проблем, т.к. этот механизм входит в состав последних версий Windows.

Целью данного программного продукта явилось создание автоматизированной системы обработки информации «Информационно-аналитическая система «Библиотека» для хранения информации о каталогах библиотеки, автоматизации основного документооборота и оценки деятельности библиотеки по различным аспектам.



Данный проект содержит каталоги книг, читателей, филиалов, работников. Редактирование данных в этих таблицах осуществляется с помощью созданных пользовательских форм, что позволяет значительно сэкономить время работы.

Кроме того, получена таблица «Выдача книг» с результирующими данными для последующего анализа. Эта таблица предназначена для работы в качестве единого читательского абонемента, в него заносится информация о номере книги, которую берут, номере читателя, дате выдачи, дате возврата и факте возврата книги.

Для удобства пользователя реализованы поисковая система, фильтрация, сортировка. Построены диаграммы и ряд отчетов, позволяющих проводить анализ данных. В частности, составлены ведомость всех выданных на данный момент книг, список имеющихся наименований книг, читательские карты, статистика выдачи за все дни работы библиотеки, подсчитано количество читателей в каждом филиале, рейтинг книг. Также включена справочная система.

Таким образом, если оценивать данный проект в целом, можно сказать, что он в состоянии удовлетворить запросы среднестатистической библиотеки, обладающей достаточно большой базой книг и читателей.

Данная программа реализована для учета книг и пособий, содержащихся в библиотеке автомеханического колледжа ГОУ ВПО «Камская государственная инженерно-экономическая академия».

После первого запуска программы необходимо осуществить ввод первоначальных данных. Для этого вводится информация в справочники, на данном этапе заполняются таблицы КНИГИ, ЧИТАТЕЛИ, ФИЛИАЛЫ, затем РАСПРЕДЕЛЕНИЕ КНИГ, ЗАПИСЬ ЧИТАТЕЛЕЙ, РАБОТНИКИ, и в самом конце ВЫДАЧА КНИГ. В таблицы Книги и

Распределение книг поступают данные о каталоге книг и их распределении по филиалам библиотеки. Таблицы Читатели и Запись читателей содержит персональные данные о каждом читателе и их записи по филиалам библиотеки. В таблицах Филиалы и Работники хранится информация о месторасположении библиотеки, телефоне, заведующем, а также о работниках и их персональных данных. Таблица Выдача книг содержит информацию о том, когда, кому и какая книга была выдана и в какой срок должна быть возвращена с отметкой о факте возврата. Таблица является результирующей и хранит в себе соответствующую информацию.

В каждой таблице предусмотрена возможность выполнения корректировки данных. После ввода информации можно производить ее корректировку и обработку (добавлять, изменять и удалять данные; производить сортировку, фильтрацию и поиск данных по выбранному полю в режиме таблицы) и осуществлять анализ данных (составление ведомостей).

На основе этих семи таблиц могут быть получены результирующие данные, отвечающие запросам пользователей программного продукта и помогающие провести анализ деятельности библиотеки.

Список литературы

1. Петрович Н.Т. Люди и биты. Информационный взрыв: что он несет. 1986 / <http://www.iu.ru/biblio/archive/nikolay%5Fludi/>
2. Хроника информационного взрыва. 03.11.2003 / Cnews.ru / <http://www.ione.ru/scripts/events.asp?id=143152>.
3. Управление бизнесом в условиях информационного взрыва. 21 ноября 2005 / URL: <http://itc.ua/22491>

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА СЕЛЬСКОГО МУНИЦИПАЛЬНОГО ОБРАЗОВАНИЯ НА МОБИЛЬНОЙ ПЛАТФОРМЕ

А.В. Майоров, Е.Н. Дубровина, Е.А. Казакова

Пензенский госуниверситет

Постановка задачи

Наблюдавшееся в течение нескольких последних лет в России резкое повышение уровня информатизации общества привело к очень интересным последствиям.

Задачи управления, планирования, организация взаимодействия между участниками бизнес-процессов теперь решаются на качественно ином уровне, в том числе и с помощью привлечения мобильных технологий. Мобильные устройства (сотовые телефоны, смартфоны, карманные компьютеры, планшеты, ноутбуки) вместе со встроенным комплексом программных средств превращаются в полноценные рабочие места управления производством, оперативные центры технической помощи, удаленные площадки

для работы с клиентами. Бизнес получил новые возможности: прямая связь с клиентом, оперативная реакция на его запросы, предоставление актуальной информации, что приводит к улучшению качества предоставляемых услуг и повышает эффективность бизнеса.

Современные мобильные бизнес-приложения ориентированы, прежде всего, на поддержку системы электронной коммерции, CRM- и ERP-системы для среднего и малого бизнеса. Однако, такие неоспоримые преимущества мобильных рабочих мест как возможность дистанционного выполнения работ, оперативное реагирование на ситуацию в любое время, удаленный доступ к информационным ресурсам, управление персоналом и обмен информацией между сотрудниками в режиме реального времени могут быть эффективно применены и в сфере государственного управления.

Разрабатываемая на кафедре МОиПЭВМ Пензенского госуниверситета Автоматизированная информационная система сельского муниципального образования (АИС СМО) предназначена для ведения и анализа сведений похозяйственного учета (информации о людях, хозяйствах, землях и материально-технических средствах), находящихся на территории соответствующего муниципального образования.

Учитывая последние тенденции развития мобильной техники и технологии, принято решение начать работу по проектированию и реализации «АИС СМО на мобильной платформе».

Цели проекта

Конечными целями при работе над «АИС СМО для мобильной платформы» являются:

- оценка применимости мобильных решений для государственных организаций;
- оценка существующих мобильных технологий, выбор платформы, программно-аппаратных средств и средств разработки;
- разработка общей концепции и функциональности мобильной АИС СМО, учитывющей необходимость полной интеграции с существующей многоуровневой распределенной системой АИС СМО;
- реализация демонстрационной версии;
- тестирование и оценка эффективности работы мобильной АИС СМО, после внедрения на рабочие места государственных служащих.

Реализация

Следуя разделению стационарной АИС СМО по уровням, комплекс программных средств мобильной АИС СМО обладает настраиваемой функциональностью в зависимости от того звена управления, в котором он используется.

Пользователями мобильной системы могут быть сотрудники правительства области, руководители, главы районных администраций, их заместители, управляющие делами и операторы на уровне сельской администрации. В соответствии с занимаемой должностью и положением, каждый пользователь имеет доступ к определенным функциям системы, может выполнять операции в соответствии с занимаемой должностью.

В мобильной АИС СМО выделяется 2 типа рабочих мест: руководителя и сотрудника.

Мобильное рабочее место сотрудника предназначено для работы с жителями, непосредственно при выезде в населенный пункт для проведения опроса или переписи. Мобильные устройства позволяют сотрудникам сельской администрации вносить сведения похозяйственного учета в АИС СМО при помощи технологии интерактивного

опроса и мастеров, минуя бумажную технологию. Возможность ввода в карманные компьютеры и смартфоны по технологии «One click, one choice» и механизм отложенного ввода данных увеличивает скорость получения информации из первоисточников, ускоряет проведение опросов и переписей. Ведение клиентской базы данных в карманном компьютере решает проблему запоминания временной информации, с последующим переносом ее на сервер в базу данных. Хранение вспомогательных классификаторов и справочников в мобильных устройствах упрощает использование информационной системы. Наличие обратной связи и режима синхронизации баз данных упрощает проверку и передачу данных в стационарную АИС СМО. Поддерживается два режима синхронизации: непосредственная и удаленная. Первый режим подразумевает выгрузку хранящейся в локальной БД записей и проводится при непосредственном подключении мобильного устройства к рабочему месту сотрудника сельской администрации через USB порт. Второй режим заключается в установлении удаленного защищенного соединения с веб-порталом АИС СМО и синхронизацией данных через него.

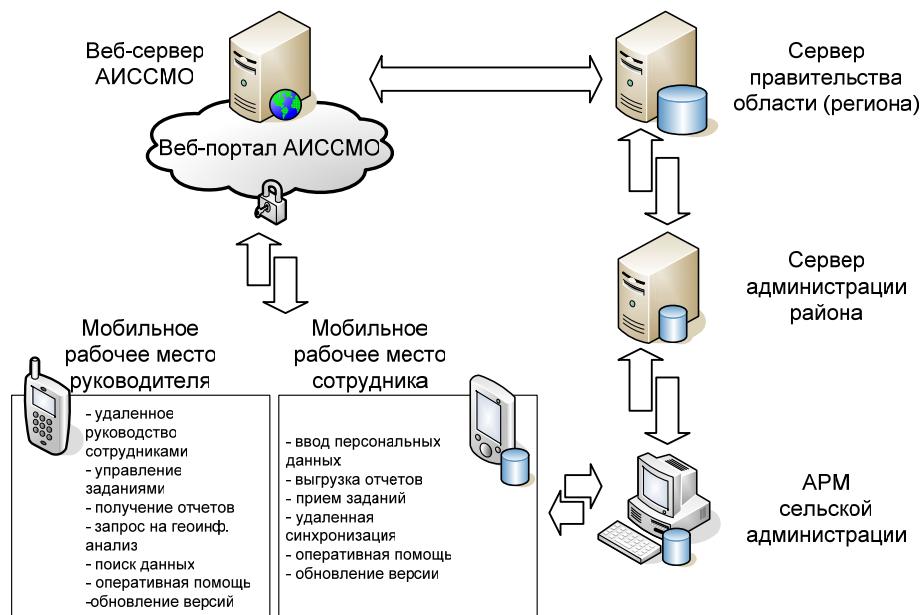


Рис. 1. Схема взаимодействия АИС СМО на мобильной платформе с серверами АИС СМО

Мобильное рабочее место руководителя предназначено, прежде всего, для получения информации в режиме реального времени. При помощи Интернет-браузера руководитель подключается к веб-серверу АИС СМО и получает доступ к оперативной, аналитической, справочной, статистической и прочей информации похозяйственного учета. Наличие активного подключения позволяет получать актуальную информацию в любое время, в любом месте и всегда находиться в курсе событий. Основными функциональными возможностями рабочего места руководителя являются:

возможность удаленного руководства служащими (при помощи sms и e-mail рассылок с Интернет-портала АИС СМО);

управлениями заданиями (формирование, отправление, получение отчета о выполненной работе на мобильное устройство);
получение статистических отчетов (например, количественных характеристик головья крупного рогатого скота у населения, или качественных характеристик состояния жилого фонда);
получение результатов геоинформационного анализа (диаграмм и графиков с привязкой к местности);
поиск данных;
оперативная помощь.

В зависимости от уровня доступа к данным руководители разных уровней (правительства, района) имеют доступ только к «своей» информации.

В качестве средств доступа к информации пользователи системы могут применять смартфоны, коммуникаторы и карманные компьютеры с установленной операционной системой Windows Mobile 2003 и выше. Кроме того, возможно использование сотовых телефонов, поддерживающих загрузку информации из Интернет.

Платформа и средства разработки

Выбор средств разработки и аппаратно-программной платформы определялся следующими условиями.

Необходимо интегрировать разрабатываемую систему для мобильной платформы с существующим комплексом АИС СМО.

Мобильные устройства и установленная на них операционная система должны быть широко распространены и пользоваться популярностью.

Должны иметься развитые средства разработки для платформы.

Предпочтение отдано продуктам фирмы Microsoft как наиболее функционально развитым и позволяющим с наименьшими затратами реализовать проект.

Разработка проводится для смартфонов, коммуникаторов и карманных компьютеров с операционной системой Window Mobile 2003 и выше, с установленной версией .Net Compact Framework.

Для доступа к информационному порталу используется встроенный веб-браузер, поддерживающий обмен информацией по защищенному HTTPS соединению (например, Internet Explorer). Водимая информация, классификаторы и справочники на мобильных устройствах хранятся в локальной базе данных Microsoft SQL Server 2005 Mobile Edition.

Интернет-портал организован на базе Information Services 6.0 для ОС Microsoft Server 2003. Используемая технология – ASP.Net.

В качестве среды разработки веб-портала и пользовательских приложений для мобильной платформы выбрана Microsoft Visual Studio .NET 2005. Язык разработки C#.

ПРОЕКТИРОВАНИЕ И ПРИНЦИПЫ ПОСТРОЕНИЯ ПРИЛОЖЕНИЙ, ПРЕДНАЗНАЧЕННЫХ ДЛЯ ОБРАБОТКИ ЭЛЕКТРОННЫХ КАРТ

З.А. Матвеев

Нижегородский госуниверситет им. Н.И. Лобачевского

Для описания электронных карт на практике нередко используются векторные графические форматы. Их применение обусловлено в частности большим объёмом входных данных и высокими требованиями к точности получаемого изображения. Программные продукты, разрабатываемые для просмотра и редактирования таких большеформатных векторных изображений, должны отвечать некоторым специфическим требованиям: минимизация времени растеризации, предоставление возможности удобной навигации в рамках полученного изображения, возможность модификации отображаемой графической информации.

Такие требования к программному обеспечению налагаются отпечаток на особенности программной архитектуры. На основе анализа, проведившегося автором при разработке подобных программных продуктов, можно сформулировать следующие рекомендации по проектированию программного обеспечения.

Обработку графических данных следует осуществлять в оперативной памяти РС (без дополнительных обращений к дисковому накопителю). Это означает, что в рамках программного продукта необходимо хранить образ обрабатываемого документа (целесообразно использовать не копию исходного документа, а некоторый внутренний формат хранения графической информации, оптимизированный с учётом выдвинутых основных требований). В большинстве случаев целесообразно пожертвовать производительностью на этапе построения этого образа с тем, чтобы добиться большего быстродействия на этапе навигации. Для достижения этой цели не следует экономить на дополнительных накладных расходах, связанных с хранением вспомогательной оптимизационной информации (кэш, индексы и т.д.).

Алгоритм построения такого «образа» документа в памяти (преобразование во внутренний формат) должен быть максимально *абстрагирован от конкретной структуры хранения и от характера входных данных* (этого можно добиться, например с применением шаблона проектирования «Строитель»). Такое абстрагирование позволяет использовать один и тот же программный модуль обработки графической информации для различных входных векторных форматов; кроме того, становится возможным менять характеристики внутреннего формата «на лету», в зависимости от структуры входного документа, что позволяет повысить производительность.

При разработке необходимо грамотно разделить программный код поддержки пользовательского интерфейса и алгоритмическую часть приложения. При этом возникает некоторое противоречие между необходимостью создания развитого графического интерфейса и необходимостью применения всевозможных низкоуровневых алгоритмов оптимизации. Дело в том, что, работая с библиотеками, позволяющими создавать полнофункциональный GUI, приходится использовать «низкопроизводительные» платформы разработки. В первую очередь это касается продуктов линейки Borland (библиотека VCL, среда разработки BDS). Это также справедливо и по отношению к WindowsForms (платформа .NET).

Есть несколько подходов к разрешению данного противоречия. Во-первых, можно оформить алгоритмическую часть в виде динамической библиотеки (DLL). При этом графи-

ческий интерфейс может быть реализован с использованием не столь эффективных высоковерхневых средств разработки. Модуль поддержки пользовательского интерфейса в этом случае должен осуществлять *единичные* обращения к алгоритмической DLL.

При работе в рамках IDE Microsoft Visual Studio.NET, предоставляется другая возможность естественного совмещения GUI и алгоритмического ядра путём совмещения managed и un-managed программного кода.

Третий подход заключается в программировании пользовательского интерфейса вообще *без* применения графических библиотек или с использованием тонких обёрток вокруг Window API (например, WTL – Windows Templates Library). Именно этот подход используется автором. Отмечу, что библиотека WTL также была разработана сотрудником корпорации Microsoft, а в настоящий момент является свободно распространяемым программным продуктом.

При проектировании программного обеспечения следует предусмотреть возможность «распараллеливания» алгоритмов построения изображения, для чего в общем случае достаточно разнести выполнение задачи по нескольким потокам (threads). Это особенно актуально в связи с массовым выпуском многоядерных процессоров в последние годы, а также в связи с появлением компиляторов, оптимизированных для работы с этими процессорами (компиляторы фирм Microsoft, Intel).

В завершение отметим, что в случае, если приложение спроектировано с применением одного из описанных подходов, можно выделить алгоритмическую часть (для векторизации и возможного разбора входного файла электронной карты) в виде серверного приложения. Становится возможным удовлетворить поставленному требованию минимизации времени построения в рамках *распределённой* геоинформационной системы. Это вдвое актуально в связи с развитием Web-интерфейсов современных ГИС и систем GPS-навигации.

Список литературы

1. Кетков Ю. Л., Кирьянов С. К. Оптимизация времени отображения векторных графических изображений большого размера. Труды конференции «11-я Международная конференция Графикон'2001. Н. Новгород, Нижегородский университет, 10-15 сентября».
2. Ласло М. Вычислительная геометрия и компьютерная графика на C++. Бином, 1997.
3. Вельтмандер П.В. Машинная графика (Учебное пособие в 3-х книгах). Новосибирский государственный технический университет. Факультет автоматики и вычислительной техники. Кафедра вычислительной техники. Интерактивная компьютерная графика. Вводный курс на базе OpenGL, 2-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2002.

АВТОМАТИЗАЦИЯ ДЕЛОПРОИЗВОДСТВА АРБИТРАЖНОГО УПРАВЛЯЮЩЕГО

Л.Н. Махмутов

Казанский государственный технический университет им. А.Н. Туполева

Целью данной работы является разработка программы, позволяющей автоматизировать делопроизводство арбитражного управляющего. В России существует несколько систем автоматизированного документооборота и учета документов. Это, на-

пример, Система учета документов «Канцелярия», Система электронного документооборота и управления взаимодействием DIRECTUM, Система «Дело». Это объемные и достаточно мощные пакеты, ориентированные на крупные предприятия, в которых основной документооборот составляют документы внутреннего пользования. В этих программах важным является создание полностью электронного документооборота. Основные функции аналогичных систем: работа над проектами документов, регистрация документов, контроль исполнения.

Специфика работы арбитражного управляющего и некоторых других организаций, специализирующихся на канцелярской поддержке судебного делопроизводства, неизбежно является бумажной и имеет ряд других особенностей: создание и распечатка пакета из нескольких документов на основе одной информационной карточки, учет и сведение в общие отчеты большого объема информации.

Остановлюсь на основных качествах системы.

Гибкость системы. Используются документы формата MS Word, MS Excel. Добавление происходит путем замены меняющихся частей документа специально определенными последовательностями символов. После обучения это доступно любому пользователю компьютера, следовательно, не требует вмешательства специалиста.

Эффективность системы. Создание полного пакета документов может осуществляться на основе одной информационной карточки (наборе основной информации о предприятии, например). Регистрация и учет исходящей документации происходит без участия пользователя.

Преимуществами применения разрабатываемого приложения являются:

сокращение количества человека-часов для создания и подтверждения запросов на информацию;

уменьшение количества логических и семантических ошибок, которые возникают при рутинной работе с повторяющейся информацией;

обеспечение планирования и мониторинга деятельности предприятия, составление расписания занятости.

Программное обеспечение создано на базе платформы Visual Studio.NET. Возможно расширение системы для многопользовательского использования с применением Microsoft SQL Server.

ИССЛЕДОВАНИЕ МОДЕЛЕЙ МАКРОЭКОНОМИЧЕСКОГО РОСТА С ПОМОЩЬЮ ПАКЕТА MATLAB

О. В. Мичасова

Нижегородский госуниверситет им. Н.И.Лобачевского

Введение

Изучение макроэкономических задач, позволяющих проанализировать взаимосвязи между экономическими субъектами и влияние различных факторов на экономическое положение в целом, является одной из важных задач, стоящих перед учеными в настоящее время. Такой анализ позволяет не только найти причины возникающих про-

блем, но и выявить решения и стратегии, которые позволяют повысить эффективность и социальную направленность функционирования рыночной экономики.

Создание и исследование математических моделей макроэкономической динамики значительно упрощается при использовании специальных программных средств. Одним из таких программных пакетов является пакет MatLab, разработанный фирмой Math Works, Inc. Название программы появилось в 1980 году и расшифровывается как матричная лаборатория (matrix laboratory). Основным элементом данных в MatLab является матрица (массив), что позволяет значительно уменьшить время решения задач для матриц и массивов, по сравнению с такими «скалярными» языками программирования как Си, например.

В MatLab важная роль отводится специализированным наборам инструментов Toolboxes, которые позволяют изучать и применять специализированные методы: обработка сигналов, системы управления, идентификация систем, построение и анализ нейронных сетей, поиск решений на основе нечеткой логики, решение нелинейных дифференциальных уравнений, финансовый и статистический анализ и т.д.

Модель Солоу-Свена

Многие математические модели экономического роста являются непосредственным обобщением известной неоклассической модели Солоу-Свена (см. [1, 2]), созданной в середине XX века. В этой модели в общем случае используется неоклассическая двухфакторная производственная функция $Y = F(K, L)$, где Y – агрегированный выпуск, K – физический капитал, L – трудовые ресурсы. В практических задачах широко применяется производственная функция Кобба – Дугласа

$$Y_t = A_t K_t^\alpha L_t^{1-\alpha}, \quad 0 < \alpha < 1. \quad (1)$$

Здесь параметр A_t описывает уровень технологий. Ниже при построении примеров используется функция вида (1).

Основное уравнение модели Солоу-Свена, описывающее динамику роста физического капитала, может быть записано в виде

$$\dot{K}_t = I_t - \delta K_t = sF(K_t, L_t, A_t) - \delta K_t. \quad (2)$$

В уравнении (2) использованы следующие обозначения: I_t – инвестиции, δ – уровень амортизации капитала, s – норма накопления. Предполагается, что трудовые ресурсы составляют постоянную долю населения, темп роста которого постоянен. Это предположение приводит к зависимости вида

$$L_t = L_0 e^{mt}, \quad (3)$$

где m – темп роста населения. Кроме того, считается, что $A_t = \text{const}$ (так что отдача от масштаба является постоянной). Для преобразования модели вводится понятие фондо-вооруженности k ($k = K / L$) и удельного выпуска ($y = Y / L$). Тогда

$$y_t = f(k_t), \quad (4)$$

где $f(k) = F(k, 1) = F(K/L, 1)$ – неоклассическая однофакторная производственная функция. В результате преобразования уравнения (2) с учетом (3), (4) получаем дифференциальное уравнение:

$$\frac{dk}{dt} = sf(k) - (\delta + m)k \equiv sf(k) - nk, n \equiv \delta + m. \quad (5)$$

Состояния равновесия уравнения (5) и их зависимость от параметров модели легко определяются на основе несложного графического анализа (см. рис. 1).

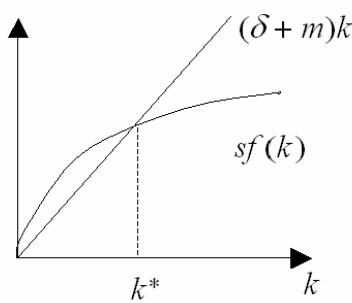


Рис. 1. Графический анализ состояний равновесия уравнения (5)

Дифференциальное уравнение имеет два решения, причем O – неустойчивое состояние равновесия, а k^* – устойчивое. Условие существования устойчивого решения имеет вид:

$$sf'(0) > \delta + m. \quad (6)$$

Исследование модели Солоу с помощью пакета MatLab

Пакет MatLab позволяет значительно упростить процесс анализа моделей экономического роста. Для того чтобы решить дифференциальное уравнение с помощью MatLab, уравнение надо записать в виде функции и сохранить в M-файл (это текстовый файл, который содержит код MatLab). Для этого в главном меню следует вы-

брать File→New→M-file, и в окне редактора M-файлов объявить переменные и определить функцию:

```
function dkdt = solou(t,k)
global s alfa A n
dkdt=s*A*(k^alfa)-n*k;
```

Для использования функции необходимо ее сохранить. Команда global делает переменные, перечисленные за ней, глобальными (то есть их значения можно изменить как внутри функции, так и в командном окне).

Библиотека MatLab включает несколько функций, которые реализуют различные методы решения дифференциальных уравнений (ode, ordinary differential equations) [3, 4]. Синтаксически эти функции различаются лишь именами, способ обращения к ним одинаков. Эти функции используют численные методы разного порядка. Рассмотрим здесь функции, наиболее часто используемые на практике.

В функции ode45 используется явный метод Рунге-Кутты 4-го и 5-го порядков в модификации Дормана и Принца. Если характеристики задачи неизвестны, рекомендуется первую попытку решения сделать с помощью этой функции.

В функции ode23 используется явный метод Рунге-Кутты 2-го и 3-го порядков в модификации Богацки и Шампина. Если не требуется большая точность и задача не очень жесткая, эта функция может оказаться более эффективной, чем ode45.

В функции ode113 используется метод Адамса, Башфорта и Моултона типа предиктор-корректор переменного порядка. Если требуется большая точность, а каждое

вычисление интегрируемой функции является «дорогостоящим», эта функция может оказаться более эффективной, чем `ode45`.

В нижеследующем описании имя функции дано обобщенно в виде `ode***`, где `***` – любой из приведенных ранее алфавитно-цифровых суффиксов. Простейшее обращение к любой функции `ode***` имеет следующий вид:

```
[tout,yout]=ode*** (fun,tspan,y0)
```

Здесь:

`fun` – указатель на функцию вычисления правых частей дифференциального уравнения;

`tspan` – вектор, содержащий «контрольные значения» независимой переменной (можно указать только начальное и конечное значение, а можно и промежуточные значения);

`y0` – начальное значение зависимой переменной (скаляр или вектор-столбец);

`tout` – вектор-столбец контрольных значений независимой переменной;

`yout` – решение, представленное массивом, в котором каждая строка соответствует одному элементу в столбце `tout`.

Для того, чтобы увидеть результаты вычислений для конкретных числовых параметров, в командном окне необходимо написать следующий программный код:

```
>> global s alfa A n
>> A=0.9;
>> alfa=0.5;
>> s=0.8;
>> n=0.05;
>> [t,k]=ode45('solou',[0 500],[1]);
>> plot(t,k)
```

Функция `plot` позволяет построить график зависимости параметра k от параметра t (рис. 2). В простейшем случае при вызове процедуры `plot` MatLab автоматически создает окно, в котором размещены стандартное меню и линейка инструментов, выделено прямоугольное поле с графиком функции, причем масштабирование и разметка по координатам производится автоматически.

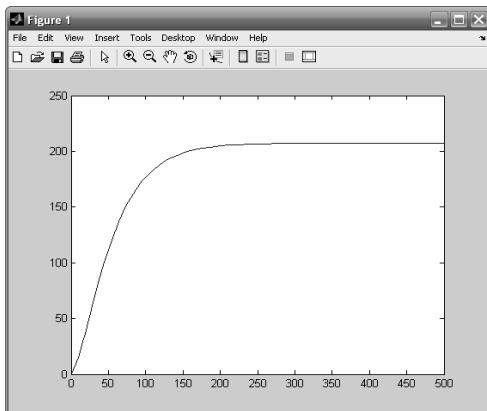


Рис. 2. Зависимость фондооруженности от времени

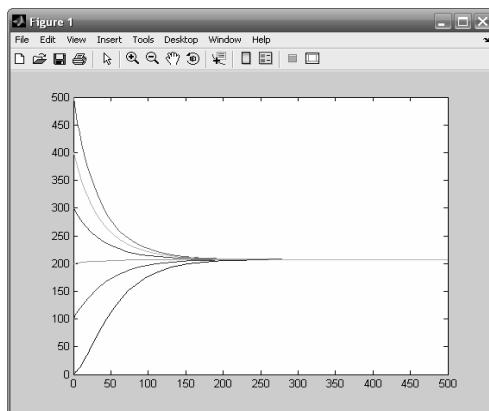


Рис. 3. Устойчивость состояния равновесия k^*

Из рис. 3 видно, что k стабилизируется на уровне 207 единиц. Убедимся в том, что это значение (k^*) является устойчивым: для этого следует рассчитать траектории для различных начальных условий (рис. 3).

Заключение

Рассмотренный пример позволяет сделать вывод о том, что пакет MatLab может эффективно использоваться при анализе задач теории экономического роста и позволяет значительно ускорить процесс решения систем дифференциальных уравнений и исследования их зависимости от параметров.

Список литературы

1. Ашманов С.А. Введение в математическую экономику. М.: Наука, 1984.
2. Иванилов Ю.П., Лотов А.В. Математические модели в экономике. М.: Наука, 1979.
3. Кетков Ю.Л., Кетков А.Ю., Шульц М.М. MatLab 6.x.: программирование численных методов. СПб.: БХВ-Петербург, 2004.
4. Getting started with MatLab. (www.mathlab.ru).

ПРИМЕНЕНИЕ ПАКЕТОВ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДЛЯ АНАЛИЗА МОДЕЛЕЙ ЭКОНОМИЧЕСКОГО РОСТА

O.B. Мичасова

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время все большее распространение получает имитационное моделирование различных систем. Практике возникает необходимость исследования сложных систем, для которых не всегда возможно использование аналитических методов.

Имитационное моделирование – это разработка и выполнение на компьютере программной системы, отражающей поведение и структуру моделируемого объекта [3, 4]. Эта область моделирования является достаточно широкой и использует различные подходы к решению практических задач. Одним из таких подходов является системная динамика.

Системная динамика – это метод изучения сложных систем, начало которому было положено в 40-х годах прошлого века при изучении систем с обратными связями. Кайл (1996) дал следующее определение методологии системной динамики:

«Системная динамика имеет дело с зависящим от времени поведением управляемых систем с целью описать систему и понять (с помощью качественных и количественных моделей), как информационные обратные связи обуславливают ее поведение, построить четкую структуру информационных обратных связей и управлять стратегией через имитацию и оптимизацию» [9].

Принцип системной динамики реализуется во многих системах имитационного моделирования, которые могут использоваться для анализа моделей экономического роста (Ithink, Powersim, Vensim, Stella) [6, 7].

Модель Солоу экономического роста

Под экономическим ростом в экономической теории понимаются не кратковременные взлеты и падения реального объема производства, а долговременные изменения его естественного уровня, связанные с развитием производительных сил на долгосрочном временном интервале [8]. В самом общем виде экономический рост означает количественное и качественное изменение результатов производства и его факторов (их производительности). Многие математические модели являются непосредственным обобщением широко известной неоклассической модели Солоу–Свена (см. [1]), созданной в середине XX века. В этой модели в общем случае используется неоклассическая двухфакторная производственная функция

$$Y = F(K, L),$$

где Y – агрегированный выпуск, K – физический капитал, L – трудовые ресурсы. В практических задачах широко применяется производственная функция Кобба – Дугласа

$$Y_t = A_t K_t^\alpha L_t^{1-\alpha}, \quad 0 < \alpha < 1. \quad (1)$$

Здесь параметр A_t описывает уровень технологий, при построении примеров используется функция вида (1).

Основное уравнение модели Солоу–Свена, описывающее динамику роста физического капитала, может быть записано в виде

$$\dot{K}_t = s Y_t - \delta K_t. \quad (2)$$

В уравнении (2) использованы следующие обозначения: δ – уровень амортизации капитала, s – норма накопления. Предполагается, далее, что трудовые ресурсы составляют постоянную долю населения, темп роста которого постоянен. Это предположение приводит к зависимости вида

$$L_t = L_0 e^{mt}. \quad (3)$$

где m – это темп роста населения. Кроме того, ниже считается, что $A_t = const$ (так что отдача от масштаба является постоянной). Для преобразования модели вводится понятие фондоооруженности k ($k=K/L$) и удельного выпуска ($y=Y/L$). Тогда

$$y_t = f(k_t), \quad (4)$$

где $f(k) = F(k, 1) = F\left(\frac{K}{L}, 1\right)$ – неоклассическая однофакторная производственная функция.

В результате преобразования уравнения (2) с учетом (3), (4) получаем дифференциальное уравнение:

$$\frac{dk}{dt} = sf(k) - (\delta + m)k \equiv sf(k) - nk, \quad n \equiv \delta + m. \quad (5)$$

Состояния равновесия уравнения (5) и их зависимость от параметров модели легко определяются на основе несложного графического анализа (см. рис. 1).

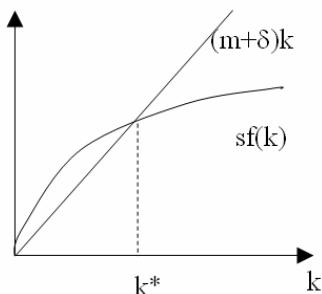


Рис. 1. Графический анализ модели Солоу

Дифференциальное уравнение имеет два решения, причем O – неустойчивое состояние равновесия, а k^* – устойчивое. Условие существования устойчивого решения имеет вид:

$$sf'(0) > \delta + m. \quad (6)$$

Очевидно, что эта модель является сильным упрощением реальной экономической системы, но она отражает основные тенденции экономического роста.

Построение модели Солоу с помощью пакета Ithink

Построим потоковую диаграмму (традиционный способ представления моделей для системной динамики) для модели Солоу, приведенной выше.

При этом используется пакет структурного моделирования Ithink, подходящий для создания как непрерывных, так и дискретных моделей [2, 5].

Структурная схема модели Солоу из встроенных блоков пакета имитационного моделирования представлена на рис. 2.

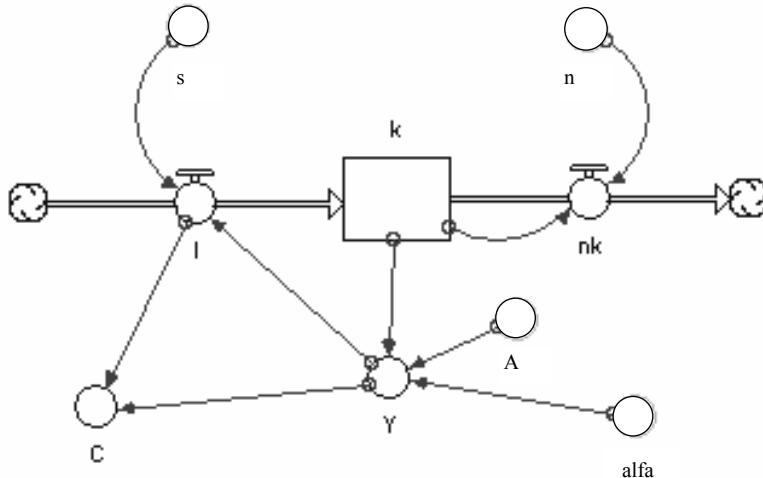


Рис. 2. Потоковая схема модели Солоу

Здесь использованы следующие встроенные блоки:
резервуар k представляет собой фондооруженность;

потоки I и m_k отражают значения инвестиций и амортизации в каждый момент времени t ;

конверторы C и Y представляют потребление и инвестиции, а s , m , A и α – константы, для определения соответствующих норм и значений. Конверторы обозначены разными цветами, чтобы отличить переменные от констант, такое разделение облегчает восприятие и понимание модели.

Между собой блоки соединены розовыми стрелками – коннекторами, которые показывают информационные связи в модели.

Следующим шагом в создании модели является задание формул и параметров, которое осуществляется при помощи стандартных диалоговых окон пакета Ithink. Программный код генерируется автоматически.

В результате работы модели зависимость фондооруженности от времени будет иметь вид (рис. 3).

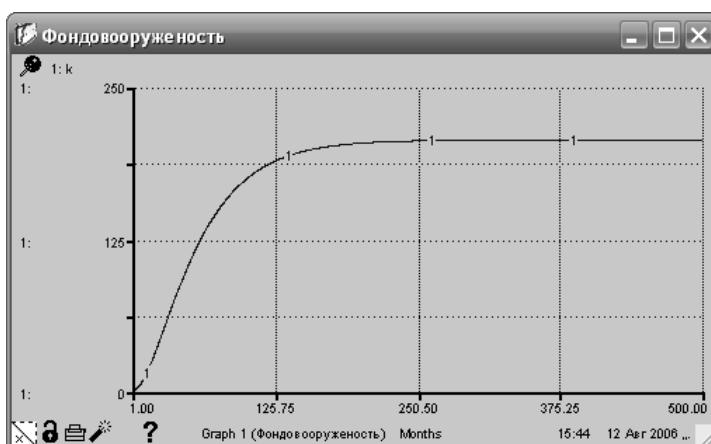


Рис. 3. Зависимость фондооруженности от времени

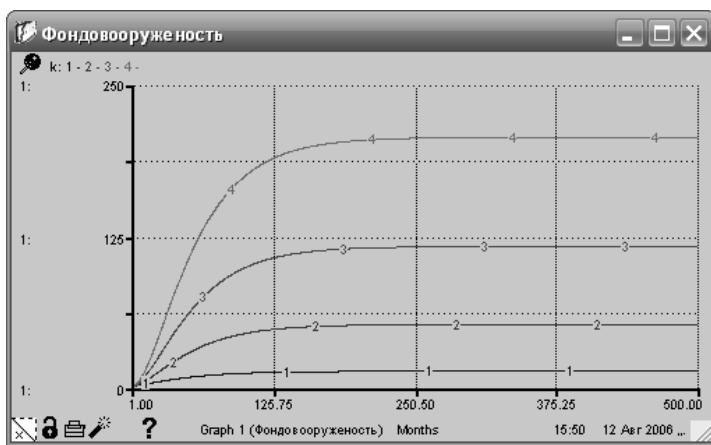


Рис. 4. Анализ чувствительности для параметра s

Используя пакет Ithink, можно провести анализ чувствительности [10], изучить, как влияет изменение значения параметра s на фондооруженность k (рис.4):

Таким образом, чем больше значение параметра s , тем больше k^* .

Также можно провести анализ чувствительности, исходя из начального значения k : (рассмотрим 0, 100, 200, 300, 400, 500). В результате получим график, представленный на рис. 5.

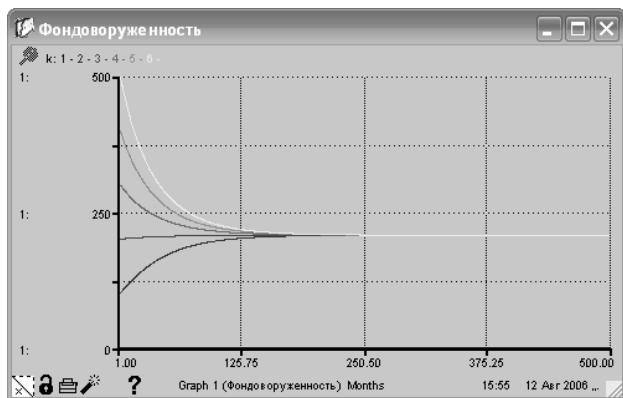


Рис. 5. Устойчивость состояния равновесия k^*

стями для качественного исследования систем, но, несмотря на это обладают широкими возможностями для визуализации моделей и проведения численных экспериментов. Системно-динамическое представление модели позволяет лучше понять и оценить структуру причинно-следственных связей в реальной системе, что значительно повышает эффективность моделирования.

Список литературы

1. Ашманов С.А. Введение в математическую экономику. М.: Наука, 1984. 296 с.
2. Горбунов А.Р. Пакет структурного моделирования Ithink: инвестиционные проекты, ренжиниринг, стратегия. М.: Тора-центр, 1997. 24 с.
3. Емельянов А.А., Власова Е.А., Дума Р.В. Имитационное моделирование экономических процессов. М.: Финансы и статистика, 2004. 368 с.
4. Карпов Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5. СПб.: БХВ-Петербург, 2005. 400 с.
5. Кузнецов Ю.А., Перова В.И., Мичасова О.В. Работа с программным пакетом Ithink: Учебно-методическое пособие. Н.Новгород: Изд-во ННГУ, 2005. 72 с.
6. Сидоренко В.Н. Системная динамика. М.: ТЕИС, 1998. 205 с.
7. Сидоренко В.Н. Системно-динамическое моделирование экономического роста. // Моделирование и прогнозирование социально-экономических процессов, 1999. С. 33-44.
8. Экономическая теория / Под ред. А.И. Добринина, Л.С. Тараксевича. СПб.: Питер, 1999. 544 с.
9. Sterman J.D. System Dynamics Models for Project Management. (www.rub.ruc.dk)
10. Technical Documentation for the ITHINK & STELLA Software // High Performance Systems. Inc., 2003.

Таким образом, с помощью имитационных экспериментов мы подтвердили теоретический вывод о том, что состояние равновесия k^* является устойчивым.

Заключение

Построение модели Союзу с помощью пакета имитационного моделирования Ithink не требует от исследователей знания специальных языков программирования. Конечно, такие средства моделирования обладают ограниченными возможностями для качественного исследования систем, но, несмотря на это обладают широкими возможностями для визуализации моделей и проведения численных экспериментов.

ПОИСК ОПТИМАЛЬНЫХ РЕШЕНИЙ ДЛЯ ИСЛЕДОВАНИЯ И УПРАВЛЕНИЯ ПРЕДПРИЯТИЕМ

В.В. Мокшин, И.М. Якимов

Казанский государственный технический университет им. А.Н. Туполева

Значительную роль в управлении предприятием имеет «информация», которой в современной промышленности нет недостатка. Значения производственно-экономических факторов говорят нам о таких вещах как основные средства, запасы, коммерческие расходы, фонд заработной платы и т.д. В результате анализа получаются количественные данные о выручке (нетто) от продажи товаров, продукции, работ, услуг; прибыли от продаж; прибыли на 1 работника и других результативных показателях эффективности. На многих предприятиях находятся залежи подобных данных. И часто цифры просто коллекционируются без всякого понимания цели или смысла или же во имя целей, ставившихся в прежние годы.

Данные только что указанного типа представляются в виде таблиц чисел. В этих числах могут быть завуалированы некоторые соотношения или же эти соотношения могут непосредственно следовать из данных. Для любых задач с изменяющимися количественными переменными представляет интерес исследование влияния (действительного или подозреваемого) некоторых переменных на остальные. Таким влиянием может быть функциональная связь между переменными. Даже тогда, когда по смыслу не существует физической связи между переменными, мы можем стремиться к тому, чтобы отразить ее с помощью математического уравнения данного вида.

Цель работы состоит в повышении эффективности работы предприятия за счет использования автоматизированной информационной системы статистических исследований, включающую математические методы для анализа состояния завода и выработки управленческих решений. В качестве вероятностного объекта исследования рассматривается ЕлАЗ (Елабужский автомобильный завод). Поставлены и решены задачи исследования функционирования предприятия; разработана модель и информационная система, повышающая эффективность решения статистических и управленческих задач.

Разработана методика анализа вероятностных объектов с непрерывными параметрами, в основу которой положен регрессионный анализ. Создана модель функционирования предприятия с учетом всех существенных переменных, функционально представляемых в следующем виде

$$y_j = f_j(x_1, x_2, \dots, x_M); \quad j = \overline{1, K},$$

где y_j – j -й показатель функционирования предприятия; K – результативные показатели эффективности функционирования предприятия; x_i – i -й фактор, влияющий на функционирование предприятия; M – общее число экономических, производственных факторов.

В ходе решения поставленных задач проведен корреляционный анализ, позволивший заключить о нелинейной связи между производственно-экономическими факторами. Тесноту связи между переменными принято характеризовать парными коэффициентами линейной корреляции r_{ij} , вычисляемыми по формуле

$$r_{ij} = \frac{\frac{1}{n} \sum_{g=1}^n v_{gi} v_{gj} - \frac{1}{n} \sum_{g=1}^n v_{gi} \cdot \frac{1}{n} \sum_{g=1}^n v_{gj}}{\sqrt{\left(\frac{1}{n} \sum_{g=1}^n v_{gi}^2 - \left(\frac{1}{n} \sum_{g=1}^n v_{gi} \right)^2 \right) \cdot \left(\frac{1}{n} \sum_{g=1}^n v_{gj}^2 - \left(\frac{1}{n} \sum_{g=1}^n v_{gj} \right)^2 \right)}} ;$$

$$i = \overline{1, M + K}; \quad j = \overline{1, M + K}.$$

где n –учитываемые интервалы времени; M –производственно-экономические факторы $x_i(x_j)$; K –результативные показатели эффективности $y_i(y_j)$; $v_{ig}(v_{gj})$ –значение i -ой (j -ой) переменной на g -ом учитываемым интервале времени.

Система разработана на базе статистического пакета прикладных программ Statistica 6.0, стандартного пакета прикладных программ Excel 2003 и Visual Studio.Net Enterprise Architect 2003 for Windows 2000/XP.

Для обработки и хранения информации спроектирована база данных по результатам, полученным при использовании программного продукта Platinum BPwin.

Получив данную модель функционирования предприятия, есть возможность принимать эффективные управленческие решения, варьируя значениями производственно-экономических факторов (x_i). Кроме того, получив результаты временного прогнозирования, можно оценить основные тенденции развития предприятия при отсутствии влияния результатов анализа разработанной автоматизированной системы. Проведенное прогнозирование методом Бокса-Дженкинса позволяет учесть общую тенденцию изменения прогнозируемых параметров и их сезонную составляющую. Он реализован в виде соответствующей процедуры ARIMA в ППП STATISTICA 6.0. При прогнозировании удалось отобразить как общую тенденцию, так и сезонную составляющую.

Используя разработанную модель функционирования предприятия, можно получить оптимальные значения показателей эффективности, например, прибыли от продаж – y_j , из полученных уравнений регрессии. Причем существует возможность оценки каждого производственно-экономического фактора на результативные показатели эффективности, используя гистограммы коэффициентов эластичности и диаграммы удельных весов этих факторов. Коэффициенты эластичности показывают изменение результативного показателя эффективности при изменении конкретного фактора на 1 процент.

Разработанная модель необходима для использования руководству для принятия эффективных решений в управлении предприятием. Поэтому, создание информационной системы, позволяющей эффективно проводить исследование производственно-экономических факторов и результативных показателей эффективности, играет важную роль в управлении предприятием и при решении различных производственных задач. Более того, такая система позволит наилучшим образом оптимизировать работу предприятия. Отметим, что предложенная модель актуальна не только для конкретного предприятия. Существует возможность использования предложенной системы и модели на других производствах и предприятиях.

Данная работа является частью дипломного проекта под руководством И.М. Якимова.

Список литературы

1. Афанасьев В. Н., Юзбашев М. М. Анализ временных рядов и прогнозирование: Учебник. М.: Финансы и статистика, 2001. 228 с.

2. Боровиков В.П., Ивченко Г.И. Прогнозирование в системе Statistica в среде Windows. Основы теории и интенсивная практика на компьютере: Учебн. пособие. М.: Финансы и статистика, 1999. 384 с.
3. Балансовые отчеты, отчеты о прибылях и убытках, подаваемые в налоговые органы, и сведения о количестве работников, заработной плате, подаваемых в Государственный Комитет Статистики.
4. Дорф Р., Бишоп Р. Современные системы управления. М.: Москва, 2002. 831 с.
5. Мокшин В.В. Создание интеллектуальной системы, объединяющей разнородные автоматизированные системы для географически распределенного производства. Материалы всероссийской научной конференции «Робототехника, мехатроника и интеллектуальные системы». Таганрог, 2005.

ПРИМЕНЕНИЕ ГИБРИДНЫХ СИСТЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ РЕШЕНИЯ ЗАДАЧ МЕДИЦИНСКОЙ ДИАГНОСТИКИ

О.А. Морёнов, Е.С. Ефимов

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Принципиальные трудности при решении задач медицинской диагностики возникают не из-за дефицита необходимой информации, а из-за отсутствия объективных методов ее структуризации и анализа. Одним из эффективных подходов решения данной проблемы является использование обучаемых на основе данных экспертных систем (ЭС). При этом важнейшая роль отводится решению задачи формирования ее базы знаний (БЗ) как ключевого компонента в формализации процесса принятия диагностических решений.

Постановка задачи

Целью работы является апробация предложенных авторами реализаций гибридных систем искусственного интеллекта для формирования БЗ ЭС для задач медицинской диагностики. Основная задача заключается в исследовании эффективности решения и предоставлении практических рекомендаций в процессе диагностического поиска при решении задачи диагностики летального исхода у пациентов с острым инфарктом миокарда, а также в выявлении недостатков используемых реализаций гибридных систем.

Материал и методы исследования

В качестве методов исследования предлагается использовать две реализации гибридных нейро-фаззи систем искусственного интеллекта: реализация, предложенная О.А. Морёновым, основана на системе нечеткого логического вывода Мамдани и сети Ванга-Менделя [1]; реализация, предложенная А.С. Ефимовым, основана на использовании системы нечеткого логического вывода Такаги-Сугено 0-порядка и нечеткой сети специального вида. В качестве прикладных рассматриваются задачи медицинской диагностики диабета и заболеваний сердца [2], а также диагностики летального исхода у пациентов с острым инфарктом миокарда, предоставленная специалистами МЛПУ №5

г. Нижнего Новгорода. Последняя задача имеет 99 записей в базе данных (БД) о пациентах при 127 полях-атрибутах, значительное количество категориальных признаков, а также отличается недостаточной репрезентативностью имеющейся БД по цели прогнозирования (только 21 запись соответствует пациентам с летальным исходом).

Основные результаты

В задаче диагностики летального исхода у пациентов с острым инфарктом миокарда при учете только количественных признаков (всего их 35) в БД зависимость точности прогноза от задаваемых нечетких производственных правил в БЗ изображена на рис.1: с ростом количества правил точность прогноза растет и имеет тенденцию к стабилизации в районе 80%. При участии эксперта из исходного набора 127 признаков было выделено 36 значимых признаков. Однако в силу плохой репрезентативности выборки и преобладания категориальных признаков над непрерывными в большинстве случаев гибридные системы при своем функционировании «вырождались». Почти всегда в качестве прогноза выдавался более распространенный в выборке результат (отсутствие летального исхода).

При дальнейшем сокращении из 36 признаков эвристически было выделено 14, при использовании которых достигнута в среднем большая 80% точность прогноза. Среди выделенных признаков оказались как признаки, относящиеся к функциональному состоянию пациента (шкала оценки клинического состояния, тест 6-минутной ходьбы и др.), так и лабораторные данные (гемоглобин, холестерин, скорость клубочка фильтрации и др.). Выделение данного набора признаков, при котором достигается удовлетворительная точность решения задачи диагностики, служит основой для выдвижения практических рекомендаций специалистам: обращать больше внимания на признаки из указанной совокупности при оценке состояния и прогноза для каждого пациента.

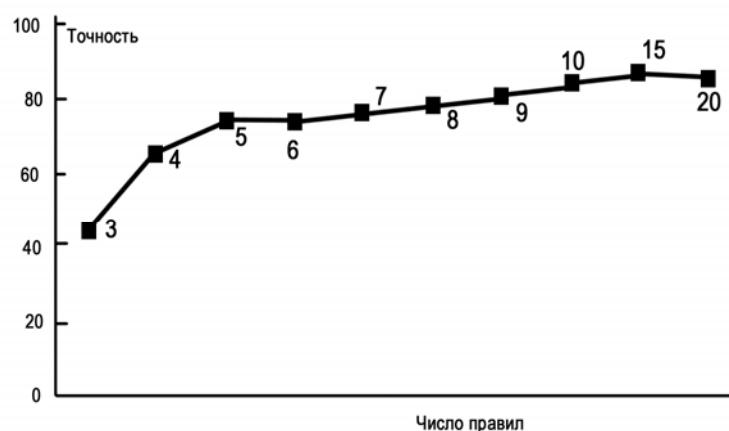


Рис. 1. Зависимость точности прогноза смертельного исхода от задаваемого количества правил в БЗ

Результаты использования имеющихся у авторов двух реализаций гибридных систем при решении задачи не показали существенных преимуществ друг перед другом: вариации точностей прогнозов укладывались в естественные статистические колебания как следствие случайности при формировании выборок. Вместе с тем был выявлен су-

щественный недостаток предложенных реализаций – неудовлетворительные результаты задачи прогнозирования в условиях значительного преобладания категориальных признаков над количественными и плохой репрезентативности исходной выборки по цели прогнозирования.

При решении задачи диагностики диабета средняя точность прогноза составила 75% при использовании гибридной системы на основе системы нечеткого вывода Мамдани и 70% – при использовании гибридной системы на основе системы нечеткого вывода Такаги-Сугено 0-порядка. Близкие результаты были получены и при решении задачи диагностики заболеваний сердца: 85% и 82% соответственно. Данные результаты сопоставимы с результатами, полученными другими исследователями при решении данных задач [3].

Заключение

Предложенные реализации гибридных систем искусственного интеллекта реализованы в программных системах NFLIS v2.0 (как развитие системы [4]) и FKNOD, созданными авторами для платформы .NET Framework v.2.0 в среде Visual C# 2005 и основанными на объектно-ориентированных библиотеках нечеткого вывода, извлечения знаний и работы с нечеткими сетями. Разработанные программные системы имеют общую оболочку для обеспечения дружественного интерфейса работы с указанными реализациями гибридных нейро-фаззи систем при решении различных практических задач классификации и прогнозирования.

Результаты апробации предложенных реализаций гибридных систем продемонстрировали их эффективность и указали направление их совершенствования: разработка эффективных подходов для работы с категориальными признаками и работа со статистическими базами данных, имеющими сопоставимое количество записей и полей-признаков.

Список литературы

1. Wang L., Mendel J.M. Fuzzy Basis Functions, Universal Approximation, and Orthogonal Least Squares // IEEE Trans. on Neural Networks, 1992a. 3(5). P. 807–814.
2. UCI Repository of Machine Learning databases. <http://ftp.ics.uci.edu/pub/machine-learning-databases/>
3. Castellano Giovanna. A Neurofuzzy Methodology for Predictive Modeling. A thesis for the degree of Doctor of Philosophy in Computer Science, University of Bari, Faculty of Science, Department of Computer Science, 2000.
4. Ефимов А.С., Морёнов О.А. FLIS-система нечеткого вывода // «Технологии Microsoft в теории и практике программирования»: материалы конференции. Н.Новгород: Изд. Нижегородского госуниверситета, 2006. С. 104-105.

ИНТЕЛЛЕКТУАЛЬНАЯ ИНФОРМАЦИОННАЯ СИСТЕМА ИНДИВИДУАЛЬНОГО КОМПЛЕКСНОГО МОНИТОРИНГА РАЗВИТИЯ СТУДЕНТОВ В ТЕЧЕНИЕ ВСЕГО ПЕРИОДА ОБУЧЕНИЯ

Л.С. Нетребский

Самарский государственный архитектурно-строительный университет

Общей характеристикой объектов деятельности специалистов с высшим образованием является, прежде всего, высокий уровень их системности, что требует от специалистов этой категории учета максимума связей как внутрисистемных, так и с другими объектами (системами). Для описания (изучения, преобразования) свойств объекта-системы деятельности специалисту с высшим образованием необходимо владение соответствующим научным языком. Он должен быть подготовлен к решению эвристических задач, требующих от него способности формулировать (видеть, выделять) проблему, определять возможные пути ее решения, принимать систему мер к реализации намеченной программы.

Резюмируя, мы приходим к следующим показателям, комплексно характеризующим студента: *учебная деятельность*, как ключевая составляющая, *личностные качества*, выраженные во *внеделательности*, а так же *творческая деятельность*.

Целью работы является разработка системы комплексного мониторинга профессионального роста студента в течении всего периода обучения, основываясь на показателях его учебной, внеучебной и творческой активностей.

Задачи:

проанализировать современные подходы к мониторингу деятельности студента в период обучения;

выработать систему показателей комплексно характеризующих обучаемого;

разработать методику расчета численных показателей и представления мониторинговой информации;

осуществить программную реализацию методики с целью накопления реальной информации для последующего анализа, а так же для обеспечения учебного и внеучебного процессов.

Оценка учебной активности студентов

С 2003 года на факультете информационных систем и технологий Самарского государственного архитектурно-строительного университета (далее ФИСТ) существует и успешно применяется в работе система мониторинга учебного процесса, разработанная аспирантом Д.А. Крутиковым под руководством С.А. Пиявского. Эта система представляет собой web-приложение, позволяющее вести учет учебной активности студентов по неделям. По каждому предмету перечисляется список контрольных точек с указанием недели сдачи. По каждой дисциплине существует технологическая карта, где по каждой неделе по каждой дисциплине выставляется итоговая оценка от 0 до 5. Оценки по всем дисциплинам собираются в одну общую оценку на неделю. Таким образом, формируется динамика учебной активности студента на каждой учебной неделе.

Оценка творческой активности студентов

Творческая деятельность студентов оценивается по достигнутой научной квалификации. На сайте ФИСТ подсистема оценки и анализа научной квалификации студентов разрабатывается В.Е. Кадочкиным под руководством С.А. Пиявского. Для оценки пользователю предлагается ответить на тестирующие вопросы, выбрав ответ из предложенных вариантов или вписав текст с ответом. Все ответы переводятся в числовые значения от 0 до 3. На основе заложенного в системе математического аппарата осуществляется оценка научной квалификации студента в виде 9 функций исследовательской деятельности.

Оценка внеучебной активности студентов

Мониторинг внеучебной активности студентов на ФИСТ ведется с весны 2006 года. Минимальной составляющей оценки внеучебной деятельности является активность или достижение. Каждый студент может описать свою внеучебную деятельность, например участие или победу в конкурсах, в виде обычного текста. После чего, доверенное лицо от кафедры имеет возможность классифицировать эту активность и оценить ее (от 0 до 3).

К настоящему времени были выделены 4 класса внеучебной деятельности студентов (они могут быть дополнены в будущем): *спорт, общественная деятельность, культура и наука* (под наукой, в отличие от научной квалификации, понимается участие и победа в различных конкурсах).

Эмпирическим путем, проанализировав достаточный объем описанных достижений, были получены критерии для оценки внеучебной активности студентов (см. табл. 1).

Расчет оценки внеучебной активности на дату происходит следующим образом.

На конкретную дату влияет каждая из оценок конкретного достижения, полученная ранее. При этом учитывается устаревание информации по следующей формуле:

$$M(m, d) = (1 - 0.0025d)m, \quad (1)$$

где d – количество дней, прошедших с отмеченной оцениваемой активности, m – оценка активности.

Таким образом получаем оценку этой активности на текущую дату с учетом устаревания. Причем оценки, проставленные более 399 дней назад, не учитываются.

Считаем, что значимости каждого класса достижений равны, потому оценка внеучебной активности студента на заданную дату:

$$f = \sum_{d=0}^{399} \sum_{j=1}^{c_d} (1 - 0.0025d)m_{d,j}, \quad (2)$$

где c_d – количество оценок, полученных d дней назад (относительно заданной даты),

$m_{d,j}$ – оценка № j , полученная d дней назад (относительно заданной даты).

На основе этой формулы можно рассчитать рейтинг студентов на заданную дату, а также получить динамику внеучебной активности студентов за период.

Для мониторинга внеучебной активности студентов было специально разработано web-приложение. Каждый зарегистрированный пользователь может ввести достижение не только свое, но и любого другого студента.

Таблица 1

Критерии оценки внеучебной активности студентов

Направление активности	Оценка	Критерии оценки
Наука	3	<ul style="list-style-type: none"> • За I,II,III место в конференциях, олимпиадах, выше городских (областные, региональные, российские, международные) • Публикация статьи в научном журнале, сборник трудов
	2	<ul style="list-style-type: none"> • За лауреатство в конференциях (олимпиадах) уровнем выше городских. • За I,II,III место в конференциях (олимпиадах) городского, районного, вузовского уровня • За лауреатство в конференциях (олимпиадах) городского, районного, вузовского, факультетского уровня
	1	<ul style="list-style-type: none"> • За доклад в научно-внеклассных компаниях (типа Net Group) • Публикация тезисов • За отправку статьи в научный журнал, сборник трудов • За участие в городских, областных конференциях с докладом, вне своего вуза
Культура	3	<ul style="list-style-type: none"> • За признание художественных работ (картина, скульптура, фото, дизайнерские модели и т.д.) на уровне выше городского (областные, региональные, всероссийские, международные выставки)
	2	<ul style="list-style-type: none"> • За признание художественной работы в выставках городского, районного, клубного уровней
	1	<ul style="list-style-type: none"> • За постоянное участие в "клубах по интересам" • За признание художественной работы в выставках на уровне университета
	0,2	<ul style="list-style-type: none"> • За разовое участие в культурно-массовых мероприятиях (театр, концерт, выставка и т.д.).
Спорт	3	<ul style="list-style-type: none"> • За I,II,III место в личном или командном зачете в соревнованиях уровнем выше городских (областных, региональных, всероссийских, международных)
	2	<ul style="list-style-type: none"> • За лауреатство в личном, командном зачете в соревнованиях уровнем выше городских (областных, региональных, всероссийских, международных) • За I,II,III место в личном или командном зачете в соревнованиях городских, районных, университетских, факультетских, дворовых и т.д.
	1	<ul style="list-style-type: none"> • За участие (личное, командное) в соревнованиях городских, районных, университетских, факультетских, дворовых и т.д. • За постоянные спортивные занятия в секциях в течение года
	0,2	<ul style="list-style-type: none"> • За разовое посещение спортивных мероприятий
Общественная работа	3	<ul style="list-style-type: none"> • За активную общественную работу во внеучебных общественных организациях (профком вуза, городские общественные партии или организации) • За активную помощь в организации вузовских и подобного уровня мероприятий • За организацию научно-внеклассных компаний, встречи в рамках города
	2	<ul style="list-style-type: none"> • За годовую работу старосты группы • За годовую работу профоргра группы • За активную помощь в организации факультетских и подобного уровня мероприятий • За помощь в хозяйственных, трудовых делах вуза
	1	<ul style="list-style-type: none"> • За помощь в хозяйственных, трудовых делах факультета

Новое достижение добавляется в список, но оно не будет отнесено к конкретной области достижений и оценено. Только ответственный от кафедры, перейдя в список достижений, может оценить конкретную запись.

Таким образом мы рассмотрели все три составляющих, формирующих комплексные оценки. Дальнейшая интеграция оценок, поддержка ввода и хранения которых осуществляется на портале, позволяет:

получить различные формы рейтингов;

проследить динамику как комплексную, так и по составляющим, активности студентов на факультете;

проводить исторический анализ с учетом накопленных оценок.

Формирование комплексного рейтинга

Для расчета комплексного рейтинга и свертки имеющихся показателей обратимся к средствам векторной оптимизации.

В качестве варианта решения будет выступать отрезок времени – неделя. Таким образом, для расчета динамики изменения деятельности студентов во времени, необходима оценка эффективности каждого варианта (отрезка времени):

$$f(y) = (f_1(y), f_2(y), \dots, f_N(y)), \quad (3)$$

где N – количество критериев (дисциплин), y – номер отрезка времени по порядку.

Необходимым условием решения поставленной задачи является минимальное участие лица принимающего решения (ЛПР) в расчете рейтинга. Для этих целей выбран метод ПРИНН.

Для оценки комплексной деятельности в качестве критериев выступают оценки отдельных составляющих. Каждая составляющая имеет свою группу важности, но эти критерии нам заранее известны. Поскольку учебная деятельность является наиболее важной, она отнесена ко второй группе важности, внеучебная и творческая активности отнесены к первой.

Таким образом от ЛПР или, в контексте разработанного web-приложения, пользователя не требуется никаких дополнительных действий для формирования комплексного рейтинга (описание метода ПРИНН см. С.А. Пиявский. Методы оптимизации и оптимального управления: учебное пособие. Самарский государственный архитектурно-строительный университет. Самара, 2004).

Все эти рейтинги используются для оперативного управления учебным процессом (учебный рейтинг). При постановке вопроса об отчислении обращается внимание на внеучебную деятельность студентов. Рейтинги регулярно вывешиваются на информационной доске факультета и тем самым используются как стимулирующий фактор.

Так же в тестовом режиме функционирует подсистема кластерного анализа контингента обучаемых на основе всех составляющих его активности.

Список литературы

1. Куприна А.И. Мониторинг как средство повышения качества управления образовательным процессом. Автореф: канд. пед. наук. Екатеринбург, 1999.
2. Савельев А.Я., Семушина Л.Г., Ка германьян В.С. Модель формирования специалиста с высшим образованием на современном этапе. – (Содержание, формы и методы обучения в высшей школе: анализ. Обзоры по основным направлениям развития высшего образования / НИИВО; Вып. 3). М., 2005. – 72 с.
3. Семушина Л.Г., Михалева Т.Г., Ярошенко Н.Г. и др. Исследование путей оптимизации формирования номенклатуры специальностей для подготовки кадров в средних специальных и высших учебных заведениях // Основные результаты исследований НИИ высшего образования в 1989 г. М., 1990.
4. Пиявский С. А. Управляемое развитие научных способностей молодёжи. М.: Высшее образование – XXI век, 2001. 109 с.
5. Пиявский С.А. Методы оптимизации и оптимального управления: учебное пособие. Самарский государственный архитектурно-строительный университет. Самара, 2004.
6. Камальдинова З.Ф., Нетребский Л.С. Разработка системы комплексного мониторинга формирования ИТ-специалиста в вузе // Проблемы развития одаренной молодежи в информационном обществе. Международная научно-техническая конференция / Самарский государственный архитектурно-строительный университет. Самара, 2006.
7. Кадочкин В.Е. Оценка научной квалификации и рейтинга творческих работ студентов технических вузов // Проблемы развития одаренной молодежи в информационном обществе. Международная научно-техническая конференция / Самарский государственный архитектурно-строительный университет. Самара, 2006.

ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ КОРПОРАТИВНОЙ РАБОТЫ НАД ПРОЕКТАМИ В СТРОИТЕЛЬСТВЕ

Д.Е. Одушев

Нижегородский государственный архитектурно-строительный университет

В последние годы сложился новый тип организации строительного производства, когда возникли строительные фирмы, объединяющие в себе как проектные, так и строительные функции. При этом этапы архитектурного (сметного) проектирования и проектирование подготовки строительного производства выполняются в одной и той же организации. Поэтому для повышения эффективности работы над проектами появляются возможности применения корпоративных методов работы, основанных на использовании современных информационных технологий.

Проблемы корпоративного проектирования в последнее время стали особенно актуальны в России и за рубежом.

Для того чтобы выполнить задачи, поставленные заказчиком строительства, с учетом установленных сроков, цен и качества, участники планирования должны обмениваться информацией и вести совместную работу. При этом участники распределенного планирования должны обрабатывать общие материалы (например, модели, знания, стандарты) и действовать согласованно с тем, чтобы достичь общей цели. В целях разработки новых методов с использованием информационных технологий в Германии была создана

программа исследований «Процессы совместного сетевого планирования в строительном проектировании». Эта группа применяет современную информационную технологию, которая служит для проведения процессов планирования с использованием мощностей компьютерной техники, сети и совместной деятельности различных групп планирования.

Рабочая группа «Сетевое моделирование процессов» германского университета в Ганновере совмещает научные проекты и направление изменения процессов планирования, проектирования зданий и сооружений с целью применения распределенных ресурсов, развития соответствующих моделей совместной работы технического планирования посредством обмена информацией между участниками проектов.

Научно-исследовательский проект «Моделирование реляционных процессов при совместном планировании зданий» [Кениг и другие, 2004] разрабатывается в Ганновере в Институте строительной информатики. В рамках этого проекта модель процесса описывается по трем блокам: организационная структура участников, структура здания с характеристиками и структура процесса по действиям. Проект включает концепцию математической модели реляционных процессов планирования. Такие подмодели описываются при помощи математического метода на основе реляционной теории и теории графов.

Университет Бахума, Германия ведет разработки методов агентных технологий, являющихся перспективным направлением для раздельного решения задач планирования, так как они помогают проектировщикам строительных конструкций. Мультиагентная система (МС) может быть представлена в качестве свободно связанной сети независимых и взаимодействующих решателей задач (агентов программного обеспечения), которые ведут совместную деятельность в целях решения определенных задач, которые они не могут решить самостоятельно. Таким образом, общие возможности взаимодействующих объектов в рамках МС превышают объединение возможностей отдельных элементов.

Данный подход к решению задач основывается на новаторском внедрении агентских технологий с учетом конкретных нужд и требований проектирования строительных конструкций.

Фактически общая агентная модель проектирования строительных конструкций разработана с выделением четырех подмоделей: 1) агентная модель сотрудничества, 2) агентная модель поддержки процессов, 3) агентная модель продукции и 4) агентная модель компоновки системы программного обеспечения.

В России набирают популярность корпоративные информационные системы по управлению проектами, которые могут использоваться для корпоративного управления подготовкой строительного производства. Следует заметить, что применение в управлении подготовкой строительным производством программного обеспечения по управлению проектами, такого как Microsoft Project Professional существенно изменит связи между функциональными подразделениями организации, потребует более детальной разработки регламентов о порядке решения задач и более четкого распределения задач. В условиях автоматизированной системы необходимо более детальное распределение прав и ответственности между ИТР, направленное на предотвращение дублирования функций и задач, рациональную организацию потоков информации в организации. Поэтому внедрение автоматизированной системы следует начать с проектирования ее организационной структуры, разработки соответствующей организационно-распределительной документации. Как показывает анализ, подготовкой строительного производства занимаются многие службы: СДО, ПТО, группа ППР и т.д. Такое положение возникло из-за сложности сооружаемых объектов, прогресса в строительной технике и технологии. Появились новые, весьма трудоемкие работы по подготовке строительного

производства, качественное выполнение которых не возможно без использования современных методов и инструментов.

Список литературы

1. Товб А.С., Ципес Г.Л. Управление проектами: стандарты, методы, опыт. М.: ЗАО «Олимп-Бизнес», 2003. 240с.: ил.
2. Фолькер Беркхан. Сетевое моделирование процессов. Материалы Международного Российско-Германского симпозиума «Применение информационных технологий в строительстве и учебном процессе».
3. Йохан Билек, Дитрих Хартманн. Проектирование строительных конструкций на основе мультиагентных систем и аддитивных ассоциативных сетей. Материалы Международного Российско-Германского симпозиума «Применение информационных технологий в строительстве и учебном процессе».
4. Уве Рюппель, Удо Ф.Майнер, Штеффен Греб, Мирко Тайсс. Процессы совместного сетевого планирования в строительном проектировании. Материалы Международного Российско-Германского симпозиума «Применение информационных технологий в строительстве и учебном процессе».

ПОДСИСТЕМА ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ТЕКСТА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Ю.А. Орлова, А.В. Заболеева-Зотова

Волгоградский государственный технический университет

Целью исследования явилась разработка системы автоматизированного анализа текста технического задания для повышения эффективности проектирования программного обеспечения. Такая система состоит из трех подсистем: предварительной обработки текста, синтаксического анализа и построения моделей программного обеспечения. В данной работе рассматривается первая подсистема.

Предварительная обработка текста необходима для разделения исходного текста технического задания на отдельные лексемы. Эта операция выполняется в три этапа: разделение на разделы, предложения и отдельные лексемы.

Уже после первого этапа производится работа не со всем текстом технического задания, а с его частями, представленными по разделам. По ходу работы текст технического задания дробится сначала на все более мелкие разделы, затем на отдельные предложения (с сохранением структуры разделов) и лексемы с указанием принадлежности к предложениям.

Входной информацией является текст технического задания на ограниченном естественном языке, выходная информация – таблицы разделов, предложений и лексем рассматриваемого технического задания.

Алгоритм работает следующим образом: сначала весь текст ТЗ делится на разделы первого уровня («1.», «2.», «3.» и т.д.), затем делится каждый раздел и так далее до третьего уровня.

Разделения на разделы, предложения и лексемы осуществляется с помощью конечных автоматов.

Разделения на разделы осуществляется с помощью конечного автомата, описанного на рис. 1. Входной символ конечного автомата: c_1 – пустое пространство, c_2 – пробел, c_3 – новая строка, c_4 – конец текста, c_5 – ‘1’..‘9’, c_6 – ‘П’, c_0 – любой другой символ.

Промежуточные состояния автомата: a_1 – начало разбора номера раздела, a_2 – последовательность символов – текст, a_3 – последовательность символов – нумерация, a_4 – начало разбора названия раздела, a_5 – последовательность символов – название раздела, a_6 – начало разбора текста раздела или приложения, a_7 – последовательность символов – продолжение текста раздела или приложения, a_8 – начало разбора названия приложения, a_9 – последовательность символов – название приложения, a_0 – конец ТЗ.

Анализ текста начинается с разбора разделов первого уровня и разбора приложений. После формирования таблицы первого уровня начинается анализ текста каждого из разделов первого уровня.

Список предложений по разделам в виде таблицы предложений формируется на основе описанной выше таблицы разделов. Таблица предложений связана по полю «код раздела» отношением типа «многие к одному» с таблицей разделов.

Предложением называется фрагмент текста, не содержащий точек и символов «точка с запятой». Как правило, в реальных документах список функций системы оформляется как маркированный список, где описание каждой функции начинается с маленькой буквы, а заканчивается точкой с запятой или точкой.

Таблица предложений связана по полю «код раздела» отношением типа «многие к одному» с таблицей разделов.

Конечный автомат разбора на предложения представлен на рис. 2. Входной символ конечного автомата: c_0 – c_4 – как в автомате разбора на разделы, c_7 – буква или цифра, c_8 – прописная буква, c_9 – ‘;’, c_{10} – ‘.’.

На вход автомата разделения на предложения подается таблица разделов, полученная на предыдущем этапе. Создаем новое предложение в таблице предложений и накапливаем символы до точки, точки с запятой или символа новой строки. При попадании в автомат точки производится проверка на наличие сокращения. Если сокращение, то продолжаем накапливать символы, если нет, то сохраняем накопленную последовательность в таблицу и начинаем разбор нового предложения. Производим разбор до тех пор, пока не встретим конец текста.

На основе таблицы предложений формируется таблица лексем, содержащей поля: код лексемы – уникальный числовой идентификатор; код предложения – число, равное коду предложения, в котором находится лексема; номер лексемы – число, равное номеру лексемы в предложении; текст – текст лексемы.

Рассмотрим конечный автомат для разбора на лексемы. Входной символ конечного автомата: c_0 – c_{10} – как в автомате разбора на предложения, c_{11} – ‘–’. На вход автомата подается каждое предложение из таблицы предложений. Производится накопление символов до точки, пробела или конца предложения и сохранение в таблицу лексем. Разбор производится до тех пор, пока не встретится конец предложения.

Для адекватной работы подсистемы необходимо анализировать текст технического задания, написанного в соответствии с ГОСТом.

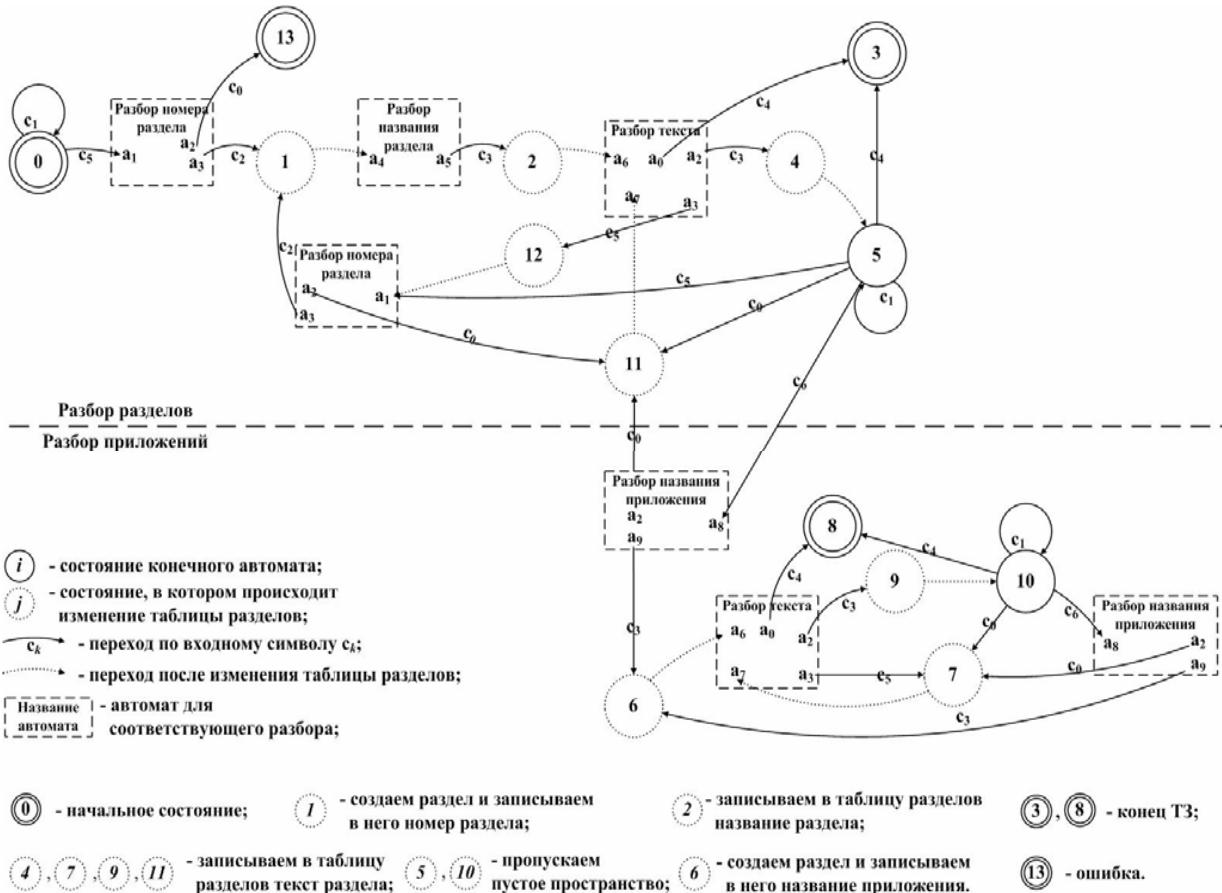


Рис. 1. Автомат разбора технического задания на разделы

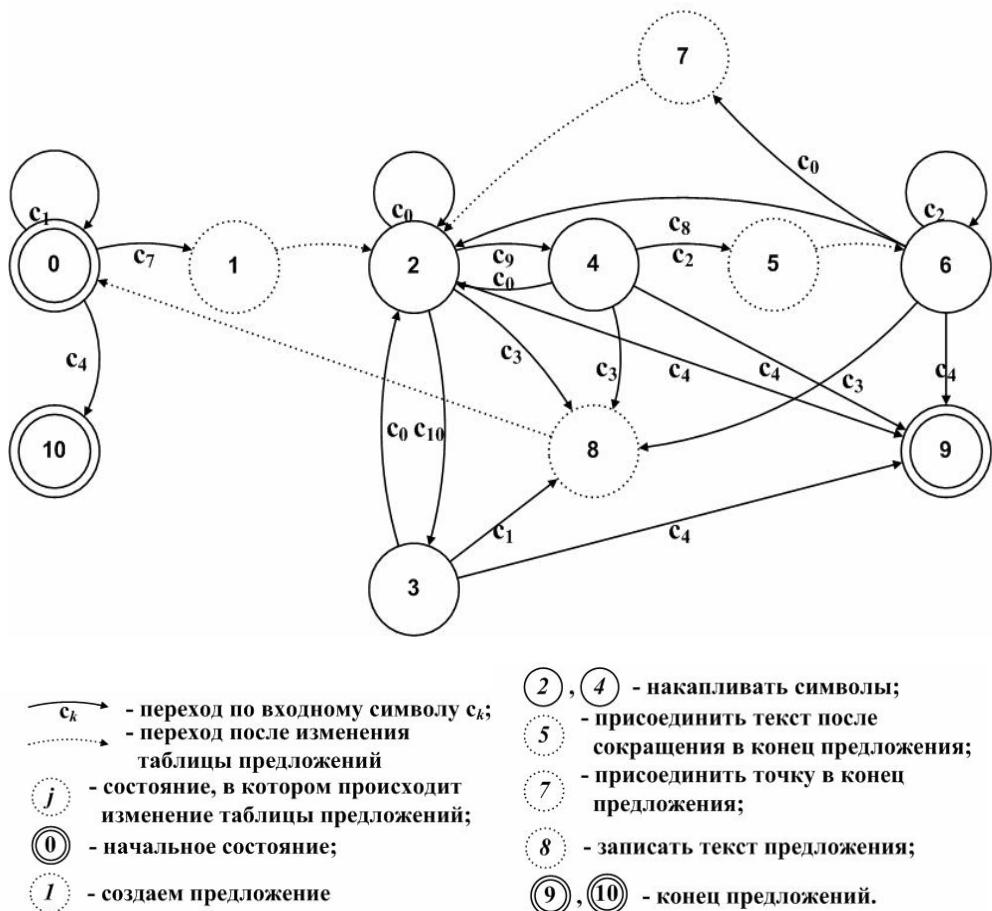


Рис. 2. Автомат разбора технического задания на предложения

Таким образом, разработан принципиально новый алгоритм предварительной обработки текста технического задания на ограниченном естественном языке, реализованный с помощью конечных автоматов в подсистеме. Данные, получаемые на выходе, в дальнейшем используются подсистемой проведения синтаксического анализа.

Проект разработан на платформе Microsoft .NET Framework 2.0 (язык разработки C#) с использованием визуальной среды программирования Visual Studio 2005. Таблицы разделов, предложений и лексем хранятся в отдельных XML-файлах. Визуальное представление данных таблиц формируется с использованием XSL-преобразования.

Список литературы

1. Заболеева-Зотова А. В. Естественный язык в автоматизированных системах. Семантический анализ текстов [Текст] – Волгоград: ВолгГТУ, 2002.
2. Заболеева-Зотова А.В. Лингвистические системы: модели, методы, приложения [Текст] Волгоград: ВолгГТУ, 2004. – С. 220.
3. Заболеева-Зотова А.В., Орлова Ю.А. Разработка средств автоматизированного проектирования программного обеспечения на основе анализа текста технического задания // «Интеллектуальные системы»(AIS'05) и «Интеллектуальные САПР» (CAD'2005): сб. трудов международных научных конференций. Т. 2. М.: Физматлит, 2005. С.41–43.

РЕШЕНИЕ ОБРАТНЫХ ЗАДАЧ ОПТИКИ С ПОМОЩЬЮ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

Р.А. Павлов

Чувашский госуниверситет им. И.Н. Ульянова

Введение и постановка задачи

Обратные задачи оптики [1] – класс задач, возникающих при применении оптических и неоптических методов диагностики в экспериментальном исследовании самых различных процессов. Среди оптических методов, для которых необходимо их решать можно отметить следующие: теневые методы, интерферометрию, методы малоуглового рассеяния, рентгенографию, спектр-интерферометрию, поляризационные, методы спектроскопии и другие [2–5]. Проблемой является то, что в реальном эксперименте часто трудно (или даже невозможно) получить полное распределение интенсивности света (сигнала) в плоскости изображения объекта (возможно измерение сигнала только в части плоскости изображения). Это, в принципе, не позволяет использовать классические методы решения обратных задач.

Целью работы является разработка технологии применения искусственных нейронных сетей (ИНС) (например, [6]) для решения обратных задач оптики в случаях, когда применение классических методов невозможно.

Методика решения

Описана методика обучения (тренинга) ИНС и полученные к настоящему времени результаты, показывающие возможности ИНС при решении обратных задач на основе неполных данных о функции распределения сигнала в плоскости изображения, в частности, при использовании всего одного значения сигнала для полного определения подынтегральной функции.

В качестве эмулятора ИНС была выбран Neural Networks Wizard (NNW) 1.7, созданный BaseGroupLabs (www.basegroup.ru). Возможности решения обратной задачи были изучены на примере вычисления подынтегральной функции для случая цилиндрической симметрии объекта.

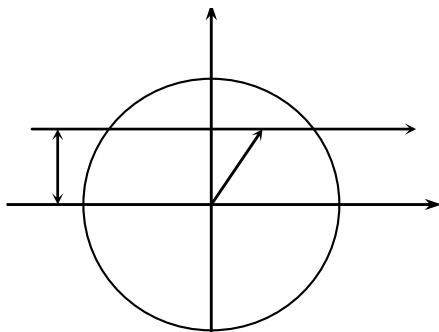


Рис. 1. Геометрия объекта и хода лучей

функций вида $n(r)=1 + a*r - b*r^2$, где a и b – различные коэффициенты. Всего было использовано 7 различных функций, отражающих реально существующие распределения локальных характеристик в объекте, например, распределения температуры и других термодинамических параметров в пламенах и газодинамических потоках:

$$\begin{aligned} n(r) &= 1-r^2; \quad n(r) = 1+0,1r-1,1r^2; \quad n(r) = 1+0,3r-1,3r^2; \\ n(r) &= 1+r-2r^2; \quad n(r) = 1+2r-3r^2; \quad n(r) = 1+3r-4r^2; \quad n(r) = 1+4r-5r^2. \end{aligned}$$

Обучающая выборка составлялась следующим образом. Вычислялись интегралы (1) от каждой из 7 функций.

Далее для различных r вычислялись значения $S(p)$.

На этапе обучения ИНС входными данными были значения S , p , и r (аргументы подынтегральной функции). Выходными данными были значения подынтегральной функции $n(r)$, соответствующие каждому значению радиуса. Всего было использовано около 600 комбинаций 4 вышеизложенных параметров.

Результаты тестирования ИНС для подынтегральной функции $n(r)=1+0.5r-1.5r^2$, не участвовавшей в обучении, представлены на рис. 2. Здесь приведены погрешности расчета для двух прицельных расстояний ($p=0.2$ и $p=0.9$), а также погрешность, усредненная по всем прицельным расстояниям в интервале от 0.1 до 0.9.

Заключение и выводы

Результаты показывают, что ИНС могут достаточно точно решать обратную задачу для случая цилиндрической симметрии. Дальнейшие перспективы работы связаны с реализацией возможностей ИНС в следующих направлениях.

1. Создание ИНС-моделей решения обратных задач для других видов симметрии объекта.
2. Создание ИНС-моделей распознавания однородного объекта (его формы и ориентации в пространстве) по неполным данным о его проекции на плоскость.
3. Применение ИНС при создании автоматических систем диагностики, тестирования и управления в различных областях науки и техники.

Методика решения была следующей. Безразмерное интегральное уравнение Абеля (случай цилиндрической симметрии объекта) для произвольного сечения объекта может быть представлено как:

$$S(p) = 2 \int_0^{\sqrt{1-p^2}} n(r) dz, \quad (1)$$

где $S(p)$ – распределение сигнала в плоскости регистрации, $n(r)$ – подынтегральная функция, $z^2+p^2=r^2$, z – путь луча в неоднородности, p – прицельное расстояние ($0 \leq p \leq 1$), r – переменный радиус ($p \leq r \leq 1$).

С помощью (1) были получены интегралы $S(p)$ от различных подынтегральных

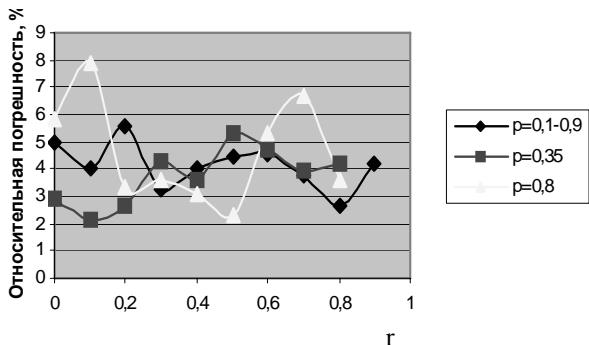


Рис. 2. Погрешность восстановления подынтегральной функции $n(r) = 1 + 0,5r - 1,5r^2$ для различных значений прицельного расстояния r

Список литературы

1. Преображенский Н.Г., Пикалов В.В. Неустойчивые задачи диагностики плазмы. Н.: Наука, 1989 г. 242 с.
2. Ладенбург Р. Физические измерения в газодинамике и при горении. М.: Иностранная литература. 1957.
3. Хауф В., Григуль У. Оптические методы в теплопередаче. М.: Мир, 1973.
4. Климкин В. Ф., Солоухин Р. И. Сверхскоростная оптическая регистрация нестационарных потоков. В кн.: Лазерная диагностика плазмы: материалы Междунар. школы-семинара, Минск, 1978.
5. Якоби Ю. А. Оптические методы исследования газовых потоков. В кн.: Лазерная диагностика плазмы: материалы Междунар. школы-семинара, Минск, 1978.
6. Neural Networks for Instrumentation, Measurement and Related Industrial Applications. Proceedings of the NATO Advanced Study Institute on Neural Networks for Instrumentation, Measurement, and Related Industrial Applications (9-20 October 2001, Crema, Italy)/ Ed. by Sergey Abameyko, Liviu Goras, Marco Gori and Vincenzo Piuri, IOS Press, Series 3: Computer and Systems Sciences. 2003. V. 185. 329 p.

БИБЛИОТЕКИ SHARPERCV.NET И INTEL OPENCV В РЕАЛИЗАЦИИ СЛЕЖЕНИЯ ЗА ЛИЦОМ ЧЕЛОВЕКА В ВИДЕОПОТОКЕ

А.Н. Половинкин, С.В. Сидоров, А.П. Суслов

Нижегородский государственный университет им. Н.И. Лобачевского

В настоящей работе рассматривается подход к слежению за параметрами лица человека в видео-потоке. Реализованные алгоритмы используют библиотеки SharperCV .NET[1] и Intel OpenCV[2].

Постановка задачи

Задача определения параметров человеческого лица является актуальной в ряде областей науки и техники. Применение алгоритмов определения параметров лица можно встретить в самых разных современных системах: от телекоммуникаций, систем слеже-

ния и до цифрового дома. При этом в качестве исходных данных для такого рода алгоритмов являются видеопоследовательности, полученные с web-камеры или из файлов. Алгоритмы же осуществляют обнаружение человеческих лиц, параметров лица (угол открытия рта, поворотов зрачков и головы).

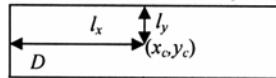
Использование возможностей открытых библиотек

Для реализации всех излагаемых алгоритмов использованы методы библиотеки Intel OpenCV, обеспечивающие получение изображения с web-камеры, а также нахождение на отдельном кадре прямоугольников, содержащих лица. Библиотека SharperCV.NET представляет собой "обёртку" над библиотекой OpenCV, реализованную в среде .NET.

Слежение за ртом (открытие и закрытие)

Метод слежения за ртом состоит из двух основных этапов: это нахождение прямоугольной области, содержащей рот, и слежение от кадра к кадру за перемещением точек из данной области.

Для нахождения прямоугольника был использован градиентный метод локальной оптимизации. При этом в качестве критерия оптимизации F была взята функция суммарной интенсивности цвета в прямоугольной области D



где x_c , и y_c – абсцисса и ордината центра прямоугольника, l_x , l_y – половины длин горизонтальной и вертикальной сторон.

$$F = F(x_c, y_c, l_x, l_y) = \sum_{(x,y) \in D} (r_{x,y} + g_{x,y} + b_{x,y}), \quad (1)$$

где $r_{x,y}$, $g_{x,y}$, $b_{x,y}$ – соответственно красная, зеленая и синяя составляющая цвета пикселя (x, y) .

Необходимо отметить, что параметры начального приближения выбирались на основе вычисленных параметров прямоугольника лица. При этом для l_x и l_y были заданы фиксированные значения l_0 , l_1 , полученные на основе параметров прямоугольника лица и статистических данных.

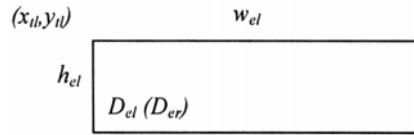
Таким образом, задача поиска рта сводилась к поиску локального максимума функции $\max_{x_c, y_c} F(x_c, y_c, l_0, l_1)$.

Алгоритм определения угла открытия рта основан на использовании фильтра оптического потока Lucas & Kanade, реализованного в библиотеке OpenCV. С помощью данной техники возможно получение горизонтальных и вертикальных перемещений отдельных пикселей на соседних кадрах. Таким образом, применяя данный метод к найденной области рта, возможно получить смещение точек по оси y ; при этом учитывая изменение положения головы между кадрами возможно вычисление относительного открытия рта от кадра к кадру.

Следение за зрачками

Реализованный алгоритм слежения за зрачками содержит 2 этапа. На первом этапе осуществляется поиск прямоугольных областей D_{el} и D_{er} , содержащих глаза, но без какой-либо части брови.

Для обеспечения этих критериев используется алгоритм, который на начальном этапе находит прямоугольную область D_{el} (D_{er}), содержащую глаз. Для этого используется переборный алгоритм по переменной y_{tl} на интервале (y_{min}, y_{max}) , минимизирующий функцию F , представляющую собой суммарную интенсивность цвета в прямоугольной области D_{el} (D_{er}).



где x_{tl} и y_{tl} – абсцисса и ордината верхнего левого угла прямоугольника, h_{el} и w_{el} – длина и ширина прямоугольника.

$$F = F(x_{tl}, y_{tl}, h_{el}, w_{el}) = \sum_{(x,y) \in D} (r_{x,y} + g_{x,y} + b_{x,y}), \quad (2)$$

где $r_{x,y}, g_{x,y}, b_{x,y}$ – соответственно красная, зеленая и синяя составляющая цвета пикселя (x, y) .

Значение интервала перебора (y_{min}, y_{max}) , параметров x_{tl} , h_{el} , w_{el} вычисляются на основе параметров прямоугольника лица, используя статистические данные.

Таким образом, задача поиска прямоугольной области D_{el} (D_{er}), содержащей глаз, сводится к задаче поиска локального минимума функции

$$\min_{y \in (y_{min}, y_{max})_c} F(x_{tl}, y_{tl}, h_{el}, w_{el}).$$

После отыскания прямоугольной области D_{el} (D_{er}) алгоритм переходит к этапу коррекции местоположения прямоугольника, добиваясь того, чтобы прямоугольная область не содержала какой-либо части брови. Для этого используется алгоритм, основанный на смещении прямоугольной области D_{el} (D_{er}) по оси ординат на Δy от текущего местоположения. Смещение Δy вычисляется как среднее арифметическое смещений при разных x_i .

$$\Delta y = \sum_{i=1}^N \Delta Y_i / N,$$

где N – количество экспериментов, ΔY_i – значение смещения, вычисленного при $x_i \in (x_{tl}, x_{tl} + w_{el})$.

Вычисление смещения ΔY_i при x_i осуществляется путем последовательного сравнения (начиная с $y_i = y_{tl}$) интенсивности пикселя с координатами (x_i, y_i) , где $y_i \in (y_{tl}, y_{tl} + h_{el})$, со средним значением интенсивности пикселя в прямоугольной области D_{el} (D_{er}), вычисляемое по следующей формуле

$$F_{sr} = F(x_{tl}, y_{tl}, h_{el}, w_{el}) / (h_{el} * w_{el}).$$

Критерием остановки является нахождение первого пикселя (x_i, y_k) с интенсивностью, большей среднего значения. Таким образом смещение при x_i вычисляется по формуле

$$\Delta Y_i = y_k - y_{tl}.$$

На втором этапе происходит уточнение положения зрачков в найденных областях. Для этого используется переборный алгоритм, максимизирующий функцию F (см. (1)), представляющую собой суммарную интенсивность цвета в области, изображенной серым цветом на рис. 1.

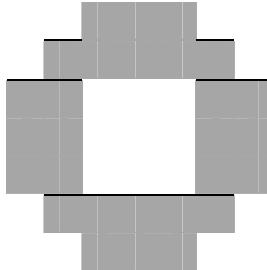


Рис. 1

Таким образом, координаты центра зрачка определяются следующим образом:

$$(x_{ir}, y_{ir}) = \arg \max_{(x,y) \in D_{el}(D_{er})} F(x, y).$$

Выбор данного вида функции для максимизации обуславливается тем, что на большинстве изображений в центре зрачков присутствует светлая область. Проведенные эксперименты показали эффективность предложенного подхода по сравнению с максимизацией суммарной интенсивности в сплошной области. Использование переборного алгоритма объясняется тем,

что характер функции F в большой степени зависит от освещенности: во многих случаях F имеет несколько локальных максимумов, что делает невозможным использование методов локальной оптимизации.

На рис. 2 приведен результат работы всех алгоритмов. Необходимо отметить, что для улучшения качества распознавания требуется высокая освещенность помещения.

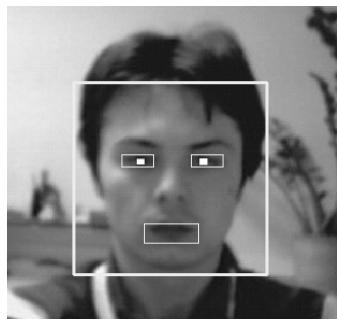


Рис. 2

Список литературы

1. Библиотека SharperCV: <http://www.cs.ru.ac.za/research/groups/SharperCV/>
2. Библиотека Intel OpenCV: <http://sourceforge.net/projects/opencvlibrary>
3. C# meets OpenCV, EP Wentworth Department of Computer Science ,Rhodes University, Grahamstown 6140, South Africa, 2003
4. Vezhnevets V. Method For Localization Of Human Faces In Color-Based Face Detectors And Trackers. In Proc. Third International Conference on Digital Information Processing And Control In Extreme Situations. May 2002. Minsk, Belarus. P. 51-56.

АЛГОРИТМ НАХОЖДЕНИЯ ОПТИМАЛЬНОГО РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА С ПОМОЩЬЮ НЕСКОЛЬКИХ БЛИЗКИХ К ОПТИМАЛЬНОМУ РЕШЕНИЙ

А.М. Перелюбский, Д.И. Батищев

Нижегородский госуниверситет им. Н.И. Лобачевского

Декомпозиционный подход к решению задачи коммивояжера с помощью генетических алгоритмов [1, 2] позволяет получать решения задач большой размерности за короткое время. В то же время нельзя говорить с уверенностью, что полученное решение действительно оптимальное, а не близкое к оптимальному. Данный подход предлагает из нескольких уже полученных решений задачи коммивояжера получить новое решение, которое, возможно, будет лучше всех уже полученных.

Пусть генетический алгоритм, реализующий декомпозиционный подход к решению задачи коммивояжера размерности N , отработал M раз. Таким образом, получаем M решений задачи коммивояжера в виде перестановок P_i ($i = 1, \dots, M$):

$$P_1 = (p_1^1, p_2^1, \dots, p_N^1)$$

...

$$P_M = (p_1^M, p_2^M, \dots, p_N^M).$$

Данный подход опирается на следующее наблюдение: в решении, близком к оптимальному (близком в смысле значения критерия), присутствует достаточно много ребер, которые есть и в оптимальном решении. Будем предполагать, что ребро присутствует в оптимальном решении, если оно содержится в достаточно большом числе решений, близких к оптимальному. Будем считать, что ребро (a, b) присутствует в оптимальном решении, если это ребро присутствует в решениях P_1, \dots, P_M не меньше, чем L раз. Введем в рассмотрение функцию $\text{func}((a, b), P_i)$, где (a, b) – ребро графа $K_n(V, d)$

$$\text{func}((a, b), P_i) = \begin{cases} 1, & \text{если } (a, b) \in P_i \\ 0, & \text{если } (a, b) \notin P_i \end{cases} \quad i \in \overline{1, M}. \quad (1)$$

Тогда считаем, что $(a, b) \in \Omega$ (Ω – оптимальное решение задачи коммивояжера), если

$$\sum_{i=1}^M \text{func}(a, b, P_i) \geq L, \quad (2)$$

где L – параметр алгоритма.

Допустим, мы сделали предположение о том, что ребра $(a_1, a_2), (a_2, a_3), \dots, (a_{q-1}, a_q)$, где $q > 1$, принадлежат оптимальному решению $((a_1, a_2) \in \Omega, \dots, (a_{q-1}, a_q) \in \Omega)$. Тогда естественным будет предположить, что вся последовательность ребер $(a_1, a_2, \dots, a_{q-1}, a_q)$ принадлежит оптимальному решению задачи коммивояжера $((a_1, a_2, \dots, a_{q-1}, a_q) \in \Omega)$.

Суть предлагаемого алгоритма состоит именно в нахождении таких последовательностей ребер. Затем представляем каждую из таких последовательностей как отрезок, и переходим от решения задачи коммивояжера к решению задачи соединения отрезков (или трубопроводной задачи) [3] гораздо меньшей размерности. Решив получен-

ную задачу соединения отрезков, можно (если все наши предположения о включаемых отрезках были верны) получить оптимальное решение задачи коммивояжера. Если полученное решение не оптимально, оно может оказаться лучше, чем решения P_1, \dots, P_M .

Таким образом, наш подход позволяет использовать имеющуюся информацию о попытках решения задачи, и, с помощью этой информации, перейти к решению задачи соединения отрезков существенно меньшей размерности.

Встает вопрос о подборе параметров алгоритма L и M таким образом, чтобы отношение L/M ($0 \leq L/M \leq 1$) не было слишком велико или мало.

При близких к нулю значениях отношения L/M в задачу соединения отрезков попадает слишком много отрезков, которых может и не быть в оптимальном решении. Таким образом, как бы хорошо мы не решили задачу соединения отрезков, оптимальное решение задачи коммивояжера не будет получено, поскольку за фиксированные были приняты «лишние» отрезки.

При близких к единице значениях отношения L/M в задачу соединения отрезков попадает слишком небольшое число отрезков, чтобы можно было говорить о существенном уменьшении размерности задачи.

На основе данного алгоритма была реализована программная система, основанная на платформе Microsoft .Net Framework, позволяющая с помощью нескольких близких к оптимальному решений задачи коммивояжера получить решение, которое может оказаться лучше, чем эти решения. Для реализации системы были использованы интегрированная среда разработки Microsoft Visual Studio 2003 со встроенным в нее инструментом разработки JetBrains ReSharper 1.4. В качестве языка программирования использовался язык C#. Для хранения исходных данных был выбран формат XML. Для увеличения скорости работы запрограммированного алгоритма был использован профилировщик Ants Profiler.

Был проведен ряд вычислительных экспериментов, результаты которых позволяют говорить о целесообразности применения данного метода. Здесь описаны результаты проведения эксперимента для известной тестовой задачи Eilon75 [4], размерность задачи – 75 городов. Для экспериментов были выбраны $M = 7$, $L = 5$.

Были получены следующие 7 решений задачи коммивояжера.

1. (31 72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 56 55 54 52 51 53 57 9 10 8 7 6 12
11 18 19 20 23 22 21 45 46 49 50 48 47 43 44 42 41 40 39 38 37 36 35 34 33 32 29 27 28 26
25 24 17 16 15 14 13 5 4 3 2 1 75 74 73 30)

Длина гамильтонова цикла: 539.

2. (19 11 12 13 14 15 74 73 75 1 2 3 4 5 6 7 8 9 10 57 53 51 52 54 55 56 58 59 60 61 62
63 64 65 66 67 68 69 70 71 72 31 30 29 28 27 32 33 34 35 36 37 38 39 40 41 42 44 43 47 48
50 49 46 45 21 22 23 24 25 26 16 17 18 20)

Длина гамильтонова цикла: 537.

3. (42 41 43 44 45 46 49 50 48 47 51 52 53 54 55 56 57 9 10 8 58 59 60 61 62 63 64 65
66 67 68 70 69 31 72 71 1 2 3 4 5 7 6 12 11 19 20 18 17 16 15 14 13 75 74 73 30 29 32 33 34
36 37 38 35 27 28 26 25 24 39 40 23 22 21)

Длина гамильтонова цикла: 542.

4. (52 53 54 55 56 57 10 9 8 58 59 60 61 62 63 64 65 66 67 68 69 70 71 1 2 3 4 5 7 6 13
14 12 11 19 20 18 17 16 15 74 75 73 72 31 30 29 32 33 34 36 35 27 28 26 25 38 37 40 39 24
23 22 21 42 41 43 44 45 46 49 50 47 48 51)

Длина гамильтонова цикла: 545.

5. (74 75 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 28 27 35
34 33 36 37 38 39 40 41 42 44 43 47 45 46 49 50 48 51 52 54 53 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 31 32 29 30 73)

Длина гамильтонова цикла: 540.

6. (14 13 11 12 7 6 5 4 3 2 1 71 72 69 70 68 67 66 65 64 63 62 61 60 59 58 8 9 10 57 56
55 54 53 52 51 48 47 43 44 42 41 40 39 38 37 36 35 34 33 32 31 30 73 75 74 29 28 27 26 25
24 23 22 21 45 46 49 50 19 20 18 17 16 15)

Длина гамильтонова цикла: 541.

7. (57 56 55 54 53 52 51 47 48 50 49 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31
72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 8 7 6 5 4 3 2 1 75 74 73 30 29 28 27 26 25 24
23 22 21 20 19 18 17 16 15 14 13 12 11 10 9)

Длина гамильтонова цикла: 540.

После перехода к задаче соединения отрезков получены следующие 29 отрезков:

(1 2 3 4 5); (6 7); (8); (9 10); (11 12); (13 14 15 16 17 18); (19 20); (21 22 23);
(24 25 26); (27 28); (29); (30); (31 72 71); (32 33 34); (35); (36 37 38); (39 40);
(41 42); (43 44); (45 46 49 50); (47 48); (51 52); (53); (54 55 56); (57);
(58 59 60 61 62 63 64 65 66 67 68); (69 70); (73); (74 75)

После решения задачи соединения отрезков получено следующее решение:

(11 19 20 18 17 16 15 14 13 5 4 3 2 1 75 74 73 30 29 28 27 26 25 24 23
22 21 45 46 49 50 48 47 43 44 42 41 40 39 38 37 36 35 34 33 32 31 72 71 70
69 68 67 66 65 64 63 62 61 60 59 58 56 55 54 52 51 53 57 9 10 8 7 6 12)

Длина гамильтонова цикла: 535.

Это решение совпадает с лучшим известным решением.

В заключение можно отметить, что данный алгоритм успешно справляется с нахождением по нескольким близким к оптимальному решениям задачи коммивояжера нового решения, значение критерия оптимальности которого меньше, чем каждое из значений критериев оптимальности, достигаемых на известных решениях.

Список литературы

1. Батищев Д.И., Неймарк Е.А. Декомпозиционный подход к нахождению минимального Гамильтонова цикла. // Межвузовский сборник «Оптимизация и моделирование в автоматизированных системах». Воронеж: Воронежский политехнический институт. 1998. С. 12–19.
2. Батищев Д.И., Воробьев А.С., Перелюбский А.М. Декомпозиционный подход к решению задачи коммивояжера с помощью генетических алгоритмов. Математика и кибернетика 2003. Сборник научных статей юбилейной научно-технической конференции факультета ВМК ННГУ и НИИ ПМК, Нижний Новгород, 28–29 ноября 2003. С. 38–42.
3. Liesegang D.G. The «tube-passing problem» and the traveling salesman problem // Proc. 9th Int. Math. Program. Symp. Budapest. 1976. V. 2. P. 537–543.
4. Whitley D. Sheduling Problems and Salesman: The Genetic Edge Recombination Operator. // Proc. of the Third International Conf. of Genetic Algorithms: 1989. P. 133–140.

МЕТОД КОМБИНИРОВАННЫХ ЭВРИСТИК ПОСТРОЕНИЯ КОНВЕЙЕРНЫХ РАСПИСАНИЙ

М.Х. Прилуцкий, В.С. Власов

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается конвейерная задача построения оптимального по быстродействию расписания последовательного выполнения n работ на m станках. Пусть i – номер станка, j – номер работы, $T = \|t_{ij}\|$ – $m \times n$ действительная матрица, элемент t_{ij} которой определяет время выполнения работы j на станке i , $i = \overline{1, m}$, $j = \overline{1, n}$. Требуется найти такую матрицу $X = \|x_{ij}\|$ – размерами $m \times n$, для которой выполняются ограничения:

$$x_{ij} \geq x_{i-1j} + t_{i-1j}, \quad i = \overline{2, m}, \quad j = \overline{1, n}, \quad (1)$$

$$x_{ij} \geq x_{ik} + t_{ik}, \text{ или } x_{ik} \geq x_{ij} + t_{ij}, \quad i = \overline{1, m}, \quad j = \overline{1, n}, \quad k = \overline{1, n}. \quad (2)$$

Если

$$x_{ij} \geq x_{ik}, \text{ то } x_{sj} \geq x_{sk}, \quad i = \overline{1, m}, \quad s = \overline{1, m}, \quad j = \overline{1, n}, \quad (3)$$

$$x_{ij} \geq 0, \quad i = \overline{1, m}, \quad j = \overline{1, n}, \quad (4)$$

и достигает минимального значения критерий

$$F(X) = \max_{j=1, n} (x_{mj} + t_{mj}) . \quad (5)$$

Здесь x_{ij} – момент начала выполнения работы j на станке i , $i = \overline{1, m}$, $j = \overline{1, n}$, условия (1) означают, что выполнение работы на станке может начаться не раньше, чем эта работа завершится на предыдущем станке, условия (2) – на станке одновременно не может выполняться более одной работы, условия (3) – порядок выполнения работ одинаков на всех станках (конвейерность расписания), а условия (4) являются естественными условиями на переменные. Каждому допустимому решению соответствует своя перестановка, определяющая порядок выполнения работ на станках.

Задача относится к классу NP-трудных. Поэтому для ее решения предлагается рассматривать метод комбинированных эвристик, в основе которого лежат два алгоритма.

В первом алгоритме используется аналогия между решением оптимизационной задачи и моделированием поведения колонии «муравьев», а второй алгоритм основан на биективном соответствии между бистохастическими булевыми матрицами, определяющими решения канонической задачи о назначениях, и перестановками, определяющими конвейерные расписания.

В основе первого алгоритма колония муравьев рассматривается как многоагентная система, в которой каждый агент (муравей) функционирует по простым правилам, и при этом поведение всей системы (колонии муравьев) приводит к приемлемым результатам. Это качество многоагентной системы следует из так называемой «роевой» логики. При своем движении агент помечает свой путь, и чем больше агентов прошло по

данному маршруту, тем больше значения этих пометок и тем больше агентов этот маршрут привлекает. В конвейерной задаче построения оптимального по быстродействию расписания «длина» пути определяется значением критерия, соответствующего расписанию, построенного агентом – время завершения выполнения всех работ, включенных в строящееся расписание. В основе предлагаемого алгоритма лежит n -мерная квадратная матрица $P(t) = \left\| P_{ij}(t) \right\|_{n \times n}$, определяющая состояние системы на такт работы алгоритма t , к которому каждый из агентов уже построил t перестановок. В стратегию построения перестановки отдельным агентом заложены как элементы стохастики, так и «предыстория» жизни системы, которая отображается матрицей $P(t)$. Каждая перестановка однозначно определяет конвейерное расписание, а тем самым и значение критерия задачи, соответствующее этому расписанию. Элемент матрицы $P_{ij}(t)$ определяет величину пометок к такту t , которая соответствует тому, что работа с номером i будет входить в искомую перестановку j -ой по порядку. Система функционирует T_0 тактов.

Для описания работы второго алгоритма, обозначим через A – множество различных перестановок порядка n . Пусть $Y = \left\| y_{ij} \right\|$ – квадратная матрица порядка n , где $y_{ij} = 1$, если работа i выполняется j -ой по порядку и $y_{ij} = 0$ в противном случае, $i = \overline{1, n}$, $j = \overline{1, n}$. Рассмотрим следующую задачу о назначениях:

$$Q(Y^0) = \max \left(\sum_{i=1}^m \sum_{j=1}^n p_{ij} y_{ij} \mid \sum_{i=1}^n y_{ij} = 1, j = \overline{1, n}; \sum_{j=1}^n y_{ij} = 1, i = \overline{1, n}; y_{ij} \in \{0, 1\}, i = \overline{1, n}, j = \overline{1, n} \right) \quad (6)$$

Пусть B – множество различных решений системы ограничений задачи (1)–(5): B есть множество различных квадратных бистохастических булевых матриц. Рассмотрим соответствие $\Gamma = (Q, A, B)$, где $Q = A \times B$. В работе показывается, что соответствие Γ является функциональным, инъективным, всюду определенным и сюръективным, т.е. биективным. Отсюда исследование задачи построения конвейерных расписаний можно проводить посредством решения задач о назначениях, для чего необходимо связать значения критерия задачи (6), зависящие от коэффициентов функционала, со значениями критерия задачи (5). Это можно сделать, например, используя найденную в результате работы первого алгоритма матрицу $P(t) = \left\| P_{ij}(t) \right\|_{n \times n}$. Кроме того, так как у задачи о назначениях в общем случае существует множество оптимальных решений, то имеет смысл рассматривать все оптимальные решения задачи (6), для каждого решения находить соответствующую перестановку, строить по ней допустимое расписание и выбирать из построенных расписаний наилучшее с точки зрения критерия (5). Для поиска всех оптимальных решений (6) можно использовать схему генерации циклов в орграфе, построенном с использованием матрицы простейшей задачи о назначениях, полученной на последнем шаге работы алгоритма Куна.

МНОГОКРИТЕРИАЛЬНЫЕ ЗАДАЧИ РАСПРЕДЕЛЕНИЯ РЕСУРСОВ В ИЕРАРХИЧЕСКИХ СИСТЕМАХ

М.Х. Прилуцкий, Е.А. Куликова

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Многие объекты материального мира могут быть представлены в виде иерархических систем: это и многоуровневые организационные структуры предприятий, и разветвленные маршруты перевозок, и информационные каналы связи. При рассмотрении этих объектов возникают различные оптимизационные задачи: объемно-календарного планирования, распределения информационных ресурсов в сети городского Интернет-провайдера, транспортные задачи с промежуточными пунктами, распределения материальных, финансовых ресурсов при проектировании и изготовлении сложных изделий. Для исследования и решения подобных задач могут быть использованы модели и алгоритмы, описанные в данной работе.

Алгоритм решения задач на иерархических системах при наличии установленного лексикографического порядка рассмотрен в работе [1] и предложен метод проверки на совместность систем линейных алгебраических неравенств, учитывающий транспортную специфику задачи. В [2] и [3] показана возможность применения потоковых моделей для описания и решения задач распределения ресурсов в иерархических системах. Работа [4] содержит алгоритм решения одной из частных задач рассматриваемого класса – многоиндексной задачи объемно-календарного планирования.

Постановка задачи

Общая задача распределения однородного ресурса в многоуровневой иерархической системе формулируется следующим образом. Имеется иерархическая система, элементы которой могут передавать, производить или потреблять однородный ресурс. Эти элементы и связи между ними характеризуются ресурсными ограничениями, определяющими объемы ресурсов, которые могут циркулировать в системе. Задача заключается в нахождении таких допустимых объемов ресурсов, при которых принимают экстремальные значения критерии оптимальности, определяющие эффективность функционирования системы.

Будем моделировать иерархическую систему взвешенным связным ориентированным графом $G=(V,A)$, $A \subseteq V^2$, без петель и контуров. На множестве V вершин графа зададим разбиение $V=V_s \cup V_p \cup V_c$. В построенной модели ресурс распределяется от источников (множество вершин V_s), через передающие узлы (множество вершин V_p) пользователям (множество вершин V_c). Обозначим через $F(i)=\{j|(i,j) \in A, j \in V\}$ – множество вершин графа, непосредственно следующих после вершины i , $i \in V$; $P(i)=\{j|(j,i) \in A, j \in V\}$ – множество вершин, непосредственно предшествующих вершине j , $j \in V$. Предполагается, что $F(i)=\emptyset$, если $i \in V_c$, $P(j)=\emptyset$, если $j \in V_s$.

Пусть x_i , $i \in V$ – количество ресурса, соответствующее i -му элементу системы (количество «производимого» ресурса для источника, «передаваемого» ресурса для промежуточного элемента и «потребляемого» ресурса для потребителя ресурса). Величины x_i в общем случае могут быть ограничены как максимальными, так и минимальными значениями:

$$0 \leq A_i \leq x_i \leq B_i < \infty, i \in V. \quad (1)$$

Через y_{ij} обозначим количество ресурса, передаваемое по дуге (i,j) , $(i,j) \in A$. Пропускную способность каждой дуги определяют величины C_{ij} и D_{ij} – это нижняя и верхняя границы сегмента допустимых значений y_{ij} , $C_{ij} \geq 0$, $D_{ij} < \infty$, $(i,j) \in A$. Тогда ограничения на величины ресурса, передаваемого по дугам, определяются системой ограничений:

$$C_{ij} \leq y_{ij} \leq D_{ij}, (i,j) \in A. \quad (2)$$

Запишем условие сохранения ресурса в системе – количество ресурса, проходящего по элементу равно сумме количеств ресурса, передаваемых по входящим в него дугам, а также по исходящим из него:

$$\sum_{j \in P(i)} y_{ji} = x_i, i \in V \setminus V_s, \quad x_i = \sum_{j \in F(i)} y_{ij}, i \in V \setminus V_c. \quad (3)$$

Таким образом, задача распределения однородного ограниченного ресурса в многоуровневых иерархических системах заключается в определении таких величин x_i , $i \in V$, и y_{ij} , $(i,j) \in A$, для которых выполняются ограничения (1)-(3) и принимают экстремальное значение некоторые критерии оптимальности, определяющие эффективность функционирования системы.

Исследование математической модели

Введем дополнительные элементы, соответствующие дугам системы, и перейдем к задаче с ограничениями, определенными только для элементов системы, количество которых увеличится от $|V|$ до $m = |V| + |A|$. Перенумеруем элементы системы и будем считать, что элементу с номером i , $i = \overline{1, m}$, соответствует количество ресурса x_i . Назовем элемент иерархической системы контролируемым, если количество ресурса, которое он производит (передает или получает) влияет на эффективность функционирования системы. Не уменьшая общности, считаем, что контролируемыми являются первые n элементов системы, $n \leq m$.

Как и в [1], для каждого контролируемого элемента i рассмотрим совокупность вложенных друг в друга сегментов S_i^t , $S_i^t \subseteq S_i^{t+1}$, $t = \overline{0, p-2}$, $p \geq 1$, $i = \overline{1, n}$. Наилучшим решением с точки зрения контролируемого элемента системы i считаем решение, при котором значение x_i попадает в сегмент S_i^0 – это «отличное» качество решения. Попадание в S_i^1 будет означать «хорошее» качество, в последующие более широкие отрезки – «удовлетворительное» и т.п. Для формализации понятия качества решения введем функции $F_i(x_i) = t$, если $x_i \in S_i^t$, но $x_i \notin S_i^{t-1}$. Здесь $t = 0$ означает «отличное» решение, $t = 1$ – «хорошее» и т.д. После проделанных преобразований задача распределения однородного ресурса в иерархических системах ставится как многокритериальная задача определения такого вектора $\bar{x} = (x_1, x_2, \dots, x_m)$, который удовлетворяет системе ограничений математической модели и минимизирует критерии оптимальности $F_i(x_i) = t$, $i = \overline{1, n}$.

Представим полученную задачу, как и в [4], в матричной форме. Заданы булевые матрицы A и B размерностей $m \times k$ и $n \times k$ соответственно, действительный неотрицательный вектор \vec{c} размерности m и векторная функция $\vec{G}(\vec{v})$, определенная на множестве n -мерных векторов, со значениями вида $\vec{z} = (z_1, z_2, \dots, z_n)$, $z_i \in \{0, 1, \dots, p-1\}$, $i = \overline{1, n}$. Функция $\vec{G}(\vec{v})$ отображает пространство R^n на множество вершин n -мерного p -ичного куба. Требуется найти вектор \vec{u} , удовлетворяющий ограничениям $A\vec{u} \leq \vec{c}$, с учетом минимизируемых критериев $\vec{G}(B\vec{u})$. В рассматриваемой задаче, связанной с распределением однородного ресурса в иерархической системе, вектор-функция критериев может быть представлена в виде $\vec{G}(\vec{u})$.

Каждой вершине n -мерного p -ичного куба $\vec{z} = (z_1, z_2, \dots, z_n)$ поставим в соответствие систему ограничений $S(\vec{z}): A\vec{u} \leq \vec{c}$, $\vec{G}(\vec{u}) \leq \vec{z}$. Зададим на множестве вершин куба функцию $f(\vec{z})$, принимающую значение 1, если система $S(\vec{z})$ совместна, и 0 в противном случае. Нетрудно показать, что функция $f(\vec{z})$ обладает свойством: для вершин куба \vec{z} и \vec{v} выполняется неравенство $f(\vec{z}) \geq f(\vec{v})$, если $\vec{z} \geq \vec{v}$ (покомпонентно), т.е. функция $f(\vec{z})$ является монотонной.

Алгоритмы решения

Поставленную задачу можно разбить на две подзадачи. Первая заключается в определении существования допустимого решения системы линейных двусторонних ограничений транспортного типа. Ее решение позволит узнать значение функции $f(\vec{z})$ в определенных вершинах куба. Вторая состоит в определении среди допустимых решений наилучших с точки зрения заданных критериев: в нахождении оптимальных вершин n -мерного p -ичного куба, в которых система ограничений оказывается совместной.

Для проверки на совместность и решения систем двусторонних линейных алгебраических неравенств можно воспользоваться классическими методами решения линейных неравенств, но в данной работе используется метод, предложенный в [1]. Он является обобщением релаксационного метода ортогональных проекций Агмана-Моцкина и учитывает транспортную специфику системы. В некоторых частных классах рассматриваемых систем возможно применение алгоритма сведения задачи о совместности к задаче о допустимом потоке в транспортной сети с двусторонними пропускными способностями ([2, 3]).

В данной работе рассматриваются следующие проблемы поиска оптимальных вершин n -мерного p -ичного куба.

1. Построение множества эффективных оценок и Парето-оптимальных решений.

Под оценкой в данном случае подразумевается вершина n -мерного p -ичного куба (n -мерный вектор, содержащий определенный набор значений критериев).

Эффективной оценкой будем считать недоминируемую вершину n -мерного p -ичного куба, т.е. такую вершину $\vec{z}^0 = (z_1^0, z_2^0, \dots, z_n^0)$, в которой функция $f(\vec{z}^0)$ принимает значение 1, и при этом для любой вершины $\vec{z}^1 = (z_1^1, z_2^1, \dots, z_n^1)$, для которой $z_i^0 \geq z_i^1$, $i = \overline{1, n}$ и $\exists j : z_j^0 > z_j^1$ $f(\vec{z}^1) = 0$.

Для нахождения всего множества эффективных оценок задачи предлагается сначала ограничить область поиска, найдя граничное (минимальное возможное) значение по каждому критерию, при котором функция $f(\vec{z})$ равна 1.

Затем для поиска всех эффективных оценок предлагается проводить в направлении одной координаты поиск, последовательно изменяя остальные и проходя таким образом всю $(n-1)$ -мерную гиперплоскость. Тогда на каждом шаге будет найдена вершина, лежащая на границе множества допустимых решений задачи. После этого из полученных вершин необходимо выбрать недоминируемые.

2. Построение представительного подмножества эффективных оценок.

Пусть имеется некоторая недоминируемая вершина n -мерного p -ичного куба. Рассмотрим ситуацию, в которой необходимо найти только определенные эффективные оценки для данной задачи (например, те оценки, которые являются ближайшими к выбранной начальной вершине по некоторым заданным направлениям).

Подобная задача возникает, если найденное начальное решение задачи не устраивает пользователя. В этом случае можно применить метод последовательных уступок для поиска ближайших оптимальных вершин. Пользователем должны быть заданы номера контролируемых элементов, по которым значение критерия может быть ухудшено. Для каждого из этих элементов уменьшим значение критерия на единицу и зафиксируем его. Получим некоторый набор неэффективных оценок. Будем производить поиск эффективных оценок в гиперплоскостях, определяемых значениями этой фиксированной компоненты, т.е. значениями критерия, по которому сделана уступка. Решениями данной задачи будут все найденные в полученных гиперплоскостях недоминируемые вершины куба. Затем делается следующая уступка, и процесс повторяется, пока не будут найдены устраивающие пользователя решения.

3. Решение задачи с использованием минимаксной свертки критериев.

В этом случае множество рассматриваемых вершин куба можно ограничить до вершин, имеющих одинаковые компоненты. Производя двоичный поиск среди них, оптимальное решение задачи будет найдено за число шагов порядка $\log_2 p$. Полученная таким образом вершина куба часто является достаточно грубым решением. Его можно улучшать, налагая дополнительные условия или добавляя в рассмотрение другие критерии задачи.

Заключение

В работе рассмотрены различные подходы к решению многокритериальных задач в иерархических системах. Предложены схемы решения подобных задач для поиска всех эффективных оценок или некоторого их представительного подмножества, а также для минимаксной свертки критериев.

Список литературы

1. Прилуцкий М. Х. Многокритериальное распределение однородного ресурса в иерархических системах // Автоматика и телемеханика. 1996. С. 149–156.
2. Прилуцкий М. Х. Многоиндексные задачи объемно-календарного планирования транспортного типа // SICPRO06 . Труды 5 Международной конференции «Идентификация систем и задачи управления» SICPRO 2006. Москва 30 января – 2 февраля 2006. С. 503–510.
3. Прилуцкий М. Х., Картомин А. Г. Потоковые алгоритмы распределения ресурсов в иерархических системах. Электронный журнал «Исследовано в России». 2003. 39. С. 444–452, <http://zhurnal.ape.relarn.ru/articles/2003/039.pdf>.
4. Прилуцкий М. Х. Многокритериальные многоиндексные задачи объёмно-календарного планирования // Теория и системы управления. 2007. № 1. С. 78–82.

РАСПАРАЛЛЕЛИВАНИЕ МЕТОДА ВЕТВЕЙ И ГРАНИЦ ДЛЯ ЗАДАЧИ МНОГОРЕСУРСНОГО СЕТЕВОГО ПЛАНИРОВАНИЯ

М.Х. Прилуцкий, В.В. Слободской

Нижегородский госуниверситет им. Н.И. Лобачевского

В работе рассматривается задача многоресурсного сетевого планирования, для решения которой предложен способ распараллеливания метода ветвей и границ.

Общая математическая модель задачи многоресурсного сетевого планирования

Исходные параметры математической модели

Пусть J – множество работ всех изделий, которые необходимо изготовить, I – множество агрегатов. Обозначим через $K(j)$ – множество работ, непосредственно предшествующих работе с номером j , $K(j) \subset J \cup \{\emptyset\}$, $j \in J$, где $K(j) = \emptyset$ – означает, что работа j не имеет предшествующих; $\varphi(j)$ – номер агрегата, на котором должна выполняться работа j , $\varphi(j) \in I$, $j \in J$, t_j – время выполнения работы j , $t_j \geq 0$, $j \in J$. Пусть J^0 – множество работ, не имеющих предшествующих (начальные работы), J^D – множество работ, не имеющих последующих (директивные работы), $J^0 \subseteq J$, $J^D \subseteq J$. Обозначим через q_j – ранний срок выполнения начальной работы j , $j \in J^0$, d_j – поздний срок окончания выполнения директивной работы j , $j \in J^D$.

Варьируемые параметры математической модели

В качестве варьируемых параметров математической модели определим вектор \vec{x} , который будем называть расписанием выполнения работ на агрегатах, $\vec{x} \in R^{|J|}$, где x_j – момент начала выполнения работы j , $j \in J$.

Ограничения математической модели

$$x_j \geq \max_{l \in K(j)}(x_l + t_l), j \in J \setminus J^0. \quad (1)$$

Начало выполнения работы j возможно лишь после завершения всех непосредственно предшествующих ей работ.

$$x_j \geq q_j, j \in J^0 \quad (2)$$

– ограничения для начальных работ,

$$x_j \geq x_m + t_m \text{ или } x_m \geq x_j + t_j, \varphi(m) = \varphi(j), j \in J, m \in J \quad (3)$$

– одновременно на одном и том же агрегате не может выполняться более одной работы,

$$x_j \geq 0, j \in J \quad (4)$$

– естественные ограничения на варьируемые переменные.

Задача минимизации нарушений директивных сроков

Требуется найти такое расписание, которое задается допустимым решением системы ограничений (1)-(5), и на котором принимает минимальное значение критерий, определяющий суммарный штраф за нарушения сроков завершения директивных работ:

$$F(\bar{x}) = \sum_{j \in J^D} \frac{\alpha_j}{d_j} \max(x_j + t_j - d_j, 0) \times 100.$$

Здесь α_j задает величину штрафных санкций, которые понесет система за один процент нарушения директивного срока d_j , $\alpha_j \geq 0, j \in J^D$.

Метод ветвей и границ для решения задачи многоресурсного сетевого планирования

В основу метода заложены следующие основные индивидуальные процедуры: оценок и ветвлений.

Процедура оценок

В качестве верхней (достижимой) оценки значения критерия оптимальности принимается значение критерия на решении задачи жадным алгоритмом фронтального типа с линейной оценкой сложности. Нижняя оценка находится на оптимальном решении задачи (1)-(2), (4), которое определяется с помощью соотношений:
 $x_j^0 = q_j, j \in J^0, x_j^0 = \max_{l \in K(j)} (x_l + t_l), j \in J \setminus J^0$.

Процедура ветвления

Пусть U – множество вершин дерева ветвления, $S(u)$ – множество работ, ещё не включенных в расписание, в случае, когда обрабатывается вершина u , $u \in U$, $M(u)$ – множество работ, для которых включены в расписание все предшествующие работы $M(u) \subseteq S(u)$, $u \in U$. Ветвление осуществляется по всем работам из множества $M(u)$, определяя для очередной работы момент ее самого раннего возможного начала выполнения с учётом ограничений (1)-(4), $u \in U$.

Формализация метода ветвей и границ для многопроцессорной вычислительной системы

Начало работы алгоритма осуществляется в вершине u^0 , для которой $S(u^0) = J$, а $M(u^0) = J^0$. Пусть $T(u)$ – список вершин, для которых вершина u является предшествующей в строящемся дереве ветвлений. Будем обрабатывать вершины дерева ветвления – определять для них оценки, проводить процедуру отсева и генерировать множество вершин, для которых эти вершины являются предшествующими – по уровням. Первый уровень будет содержать лишь вершину u^0 , второй – вершины из множества $T(u^0)$ и т.д. Специфика рассматриваемого метода заключается в том, что, во-первых, обработку любой вершины, находящейся на любом уровне, можно делать независимо от других вершин. Во-вторых, если будут обработаны все вершины некоторого уровня, то не в ущерб оптимальности, во всех вершинах всех предыдущих уровней можно не считать верхнюю и нижнюю оценки. Пусть система обладает n равнomoщными процессорами. Вычислительная схема реализации метода ветвей и границ для многопроцессорной системы включает в себя следующие процедуры.

Распределение вершин по процессорам. На каждом шаге работы алгоритма один из процессоров выполняет координационные функции. Пусть на очередном шаге работы алгоритма рассмотрен уровень дерева ветвления, включающий в себя совокупность из k необработанных вершин. Если $n=k$, то для обработки вершин происходит их распределение по процессорам в произвольном (заранее определенном) порядке. Если $n>k$, то осуществляется переход на следующий уровень, при этом вершины предыдущего уровня обработаны не будут. Это не повлияет на оптимальность алгоритма, т.к. при дальнейшей работе алгоритма все вершины какого-то очередного уровня будут обработаны. Так до тех пор, пока k - количество необработанных вершин одного уровня станет больше n - числа процессоров. Тогда на $n - k + \lceil k/n \rceil * n - 1$ процессоров для обработки назначаются по $\lceil k/n \rceil$ вершин. Если при этом k/n – целое число, то оставшемуся процессору назначаются оставшиеся $\lceil k/n \rceil$ вершин и координационные функции. Если k/n не целое число, то на оставшиеся свободные $k - \lceil k/n \rceil * n$ процессоров назначается для обработки по $\lceil k/n \rceil + 1$ вершине, а оставшийся свободным процессор обрабатывает остальные вершины и выполняет функции координатора. Обработка процессором назначеннной ему совокупности вершин заключается в определении для каждой вершины верхней и нижней оценок, осуществлении процедуры отсева неперспективных направлений среди назначенных ему вершин и генерации вершин, входящих в следующий уровень дерева ветвления.

Функции координатора. Процессор, выполняющий функции координатора, обрабатывает назначаемые ему вершины, осуществляет распределение вершин по процессорам и процедуру отсева для всей совокупности вершин, запоминает лучшую найденную достижимую оценку. При ограниченности времени работы алгоритма, вычисления могут быть остановлены, и решение, соответствующее лучшей найденной достижимой оценке, принимается за искомое.

СИСТЕМА ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ НА БАЗЕ ЭЛЛИПТИЧЕСКОЙ КРИВОЙ

В.В. Пылин

Марийский государственный технический университет

В работе приводится краткое описание разработанной криптосистемы на базе эллиптической кривой (ЭК). Данная криптосистема есть результат анализа криптографических алгоритмов выработки и проверки электронно-цифровой подписи (ЭЦП) с целью их совершенствования. Актуальность данных исследований обусловлена следующими факторами.

1) Асимметричные криptoалгоритмы на базе ЭК с одной стороны являются стандартами асимметричного шифрования в России и за рубежом, а с другой стороны криптографические свойства, прежде всего стойкость, этих алгоритмов требуют дополнительных исследований [1].

2) Криптосистемы на ЭК являются сложнореализуемыми, это связано, прежде всего, со сложностью реализации математических операций в группе точек ЭК. Указанная

и другие особенности налагаю дополнительные [2] ограничения на функциональность данных крипtosистем.

Одним из упомянутых ограничений является использование в крипtosистеме фиксированного набора эллиптических кривых с последующим выбором одной из них для выполнения криптографических операций. Это связано со сложностью реализации и трудоёмкостью работы алгоритмов генерации/выбора ЭК [3]. Это ограничение безусловно является слабым местом крипtosистем на базе ЭК, т.к. предполагаемому злоумышленнику данный набор так же может быть известен и доступен для анализа. Учитывая «юный» возраст эллиптической криптографии, можно предположить, что злоумышленнику удастся найти эффективный алгоритм взлома секретного ключа именно на данной ЭК. В разработанной крипtosистеме указанный недостаток устранён с помощью использования усовершенствованного алгоритма Чуфа [4] для генерации ЭК. Использование данной методики «случайной» генерации ЭК позволяет широко варьировать параметры ЭК, что повышает надёжность всей системы.

Кратко приведём общее описание крипtosистемы.

Компоненты системы:

сервер ЭЦП;

подсистема управления;

клиент ЭЦП.

Сервер ЭЦП реализует следующие функции:

- криптографическая аутентификация рабочих станций сети, на которых установлен Клиент ЭЦП;
- ведение центральной базы данных ключей ЭЦП с хранением полномочий пользователей по доступу к информационным ресурсам системы;
- поддержку синхронизации центральной базы данных защиты и локальных баз данных защищаемых рабочих станций и серверов;
- отображение состояния защищаемых рабочих станций и серверов ЛВС;
- объединение пользователей в группы для упрощения управления их доступом к совместно используемым ресурсам;
- ведение журналов работы крипtosистемы.

Подсистема управления для генерации ключевых носителей¹ выполняет следующие функции:

генерация параметров ЭЦП;

создание ключей шифрования и изготовление ключевого носителя;

ведение базы созданных ключей на жестком диске компьютера;

проверку ключевого носителя;

уничтожение ключей из базы.

Клиент ЭЦП реализует функции:

создания электронной цифровой подписи файлов;

проверки электронной цифровой подписи файлов.

взаимодействия с «Сервером ЭЦП»;

ведения локальной базы данных ключей ЭЦП.

В качестве особенностей функционирования крипtosистемы можно отметить следующее.

¹ Как правило, ключевым носителем является дискета или съемный накопитель flash drive.

В качестве локальных баз данных используется СУБД Microsoft Access. В данной базе хранятся данные о пользователях криптосистемы, а так же их открытые ключи. Для исключения возможности подмены локальной базы открытых ключей в подсистеме управления предусмотрена функция выгрузки базы открытых ключей для последующего тиражирования вручную на ПК клиентов системы. Такой механизм используется в том случае, если *Клиент ЭЦП* не связан с *Сервером ЭЦП*, т.е. работает в режиме *offline*. В качестве особенности так же можно отметить использование в качестве ключевого носителя не только дискеты, но устройства *flash drive*, на которых секретный ключ хранится в виде бинарного файла заданного формата. Очевидно, что для обеспечения безопасности на хранение ключевого носителя налагаются дополнительные условия.

Особенностью функционирования криптосистемы, как отмечалось выше, является использование в *Подсистеме управления* методики генерации ЭК. При этом администратор системы имеет возможность использовать либо методику на основе «случайной» генерации ЭК, что с точки зрения безопасности лучше, но более трудоёмко, либо с помощью методики комплексного умножения на ЭК [5]. Это не так безопасно, но гораздо менее трудоёмко. Кроме этого, у администратора системы есть возможность не генерировать ЭК заново, а выбрать из набора уже имеющихся, сгенерированных заранее. Перечень данных ЭК хранится в центральной базе.

Кроме указанных особенностей криптосистемы в разработке находится ещё одна компонента системы – подсистема криптоанализа. Главным функцией данной подсистемы будет являться анализ криптографических свойств ЭК, сгенерированных *Подсистемой управления*. При этом каждой ЭК будет поставлено в соответствие некоторый набор параметров, характеризующих её стойкость. Таким образом, администратор системы может осуществлять выбор ЭК на основании данного набора параметров. Анализ криптографических свойств ЭК, а, прежде всего, это стойкость ключей, сгенерированных на данной ЭК, будет производиться с помощью алгоритмов дискретного логарифмирования, модифицированных для решения задачи вычисления кратности точки ЭК.

Отметим, что в качестве СУБД для хранения данных пользователей системы используется СУБД Microsoft SQL Server 2000. Использование данной СУБД обусловлено тем, что функционирование данной криптосистемы обеспечивается под ОС семейства Windows. И в данном случае указанная СУБД идеально подходит для реализации функций хранения данных пользователей. При этом возможно использование встроенных механизмов аутентификации SQL Server-а для обеспечения безопасного доступа к данным.

Подсистема управления и Клиент ЭЦП разработана в среде программирования Microsoft Visual Studio .NET 2003 v. 7.1. на языке программирования C++. Обе компоненты работают в среде .NET Framework. Использование данного средства разработки обусловлено широкими функциями, предоставляемыми программисту для решения прикладных задач. Кроме этого оболочка имеет очень удобный интерфейс, что избавляет программиста от необходимости решения низкоуровневых прикладных задач при проектировании системы. Высокая степень интеграции оболочки со средствами настройки и программирования СУБД MS SQL Server является дополнительным фактором в пользу выбора данного средства разработки. Использования языка программирования C++ обусловлено и необходимостью совместимости с библиотекой MIRACLE (*Multiprecision Integer and Rational Arithmetic C/C++ Library*), которая используется в криптографической системе и реализует базовые криптографические операции с точками ЭК. Библиотека свободно распространяется для некоммерческого использования. Главным преимуществом является наличие в её составе классов, описывающих свойст-

ва и методы работы с таким объектом как точка ЭК. Классы включают в себя базовые математические операции с точками ЭК: сложение точек и умножение точки на целое число. Более того, при этом поддерживается работа с точками, координаты которых есть «большие числа»: длина порядка 256 бит.

В заключении следует отметить, что данный программный продукт представляет собой не только инструмент для исследования криптографических свойств указанных алгоритмов постановки и проверки ЭЦП. Он может успешно использоваться в качестве криптографической системы для осуществления электронного документооборота с использованием средств ЭЦП на базе эллиптической кривой.

Список литературы

1. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография. – Спб.: АНО НПО «Профессионал», 2005. 480 с., ил.
2. Hankerson D., Menezes A., Vanstone S.A. Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.
3. Lencier R., Morain F. Counting the number of points on elliptic curves over finite fields: strategies and performances // Advances in Cryptology–EUROCRYPT ’95. Lecture Notes in Computer Science. Springer-Verlag, 1995. V. 921. P. 79–94.
4. Schoof R. Elliptic curves over finite fields and the computation of square roots mod p // Math. Comp. 1985. V. 44. P. 483–494.
5. Atkin A.O., Morain F. Elliptic curves and primality proving // Mathematics of Computation. 1993. V. 61. P. 29–68.

3D-МОДЕЛИ В ТЕХНОЛОГИЯХ AUTODESK ДЛЯ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА ИЗДЕЛИЙ

Л.И. Райкин, М.Н. Вязанкина

*Нижегородский областной центр новых информационных технологий
Нижегородского государственного технического университета*

Дано расширенное представление о геометрических моделях (ГМ). Показано, что ГМ играют ключевую роль на всех этапах технологии информационной поддержки жизненного цикла (ЖЦ) изделий (ИПИ-технологии). Являясь продуктом системы автоматизированного проектирования (САПР), единая ГМ используется на этапах маркетинга, конструкторской и технологической подготовки производства, производственной и постпроизводственной стадиях ЖЦ.

Комплексная информатизация технической деятельности предприятия определяется информационной поддержкой ЖЦ изделий (ИПИ-технологии) или, в другой интерпретации, – концепцией управления жизненным циклом изделия (*Product Lifecycle Management, PLM*).

Главная цель ИПИ-технологий – создание в рамках предприятия единого информационного пространства (ЕИП) или интегрированной информационной среды (ИИС), охватывающих все этапы ЖЦ выпускаемой этим предприятием продукции [1].

Общее представление об ИИС целесообразно расширить в отношении ГМ. Важной отличительной особенностью ИПИ-технологий становится широкое использование электронной (информационной) модели (ЭМ) изделия, его составных частей, технологической оснастки для изготовления и т.п. на большинстве этапов ЖЦ. Математической основой ЭМ служит ГМ, выступающая первоисточником для ряда других видов ЭМ. Такими видами могут быть:

- электронный макет,
- мастер-модель,
- технологический электронный макет,
- электронная компоновка.

К этапам ЖЦ изделия, на которых используются ЭМ, относятся: маркетинг и определение требований к изделию, конструкторская технологическая и организационно-экономическая подготовка производства, производство, постпроизводственная стадия. При этом ЭМ формируется единой для всех этапов, позволяя использовать ее одновременно разными структурами. ГМ, как правило, формируется с помощью САПР или в зарубежном обозначении *Computer Aided Design (CAD)*.

Трехмерная ГМ может быть сформирована различными CAD-системами. На рис. 1 представлен интерфейс CAD-системы *Autodesk Inventor* [2] – системы объемного машиностроительного конструирования. В ней имеется полный набор средств для 3D-моделирования, управления информацией, совместной работы над проектами и обеспечения технической поддержки. Эта CAD-система позволяет:

- создавать 2D- и 3D эскизы, трехмерные модели изделий и производственные чертежи;
- создавать адаптивные конструктивные элементы, детали и узлы;
- формировать кинематические снимки расположения составных частей изделий;
- настраивать отображение изделия, управляя видимостью его компонентов;
- управлять сложными изделиями, состоящими из тысяч деталей;
- запускать сторонние приложения, базирующиеся на функциях интерфейса прикладного программирования (*API*);
- использовать язык *VBA* для доступа к API-интерфейсу *Autodesk Inventor*;
- импортировать в *Autodesk Inventor* файлы форматов *SAT*, *STEP*, *AutoCAD* и *Autodesk Mechanical Desktop* (формата *DWG*) для использования в приложении *Autodesk Inventor*;
- экспортить файлы *Autodesk Inventor* в форматы *AutoCAD*, *Autodesk Mechanical Desktop*, *IGES* и *STEP*;
- сотрудничать с другими конструкторами в работе над проектами;
- использовать Интернет для доступа к проектам и данным, а также для общения с коллегами;
- пользоваться интегрированной системой поддержки.

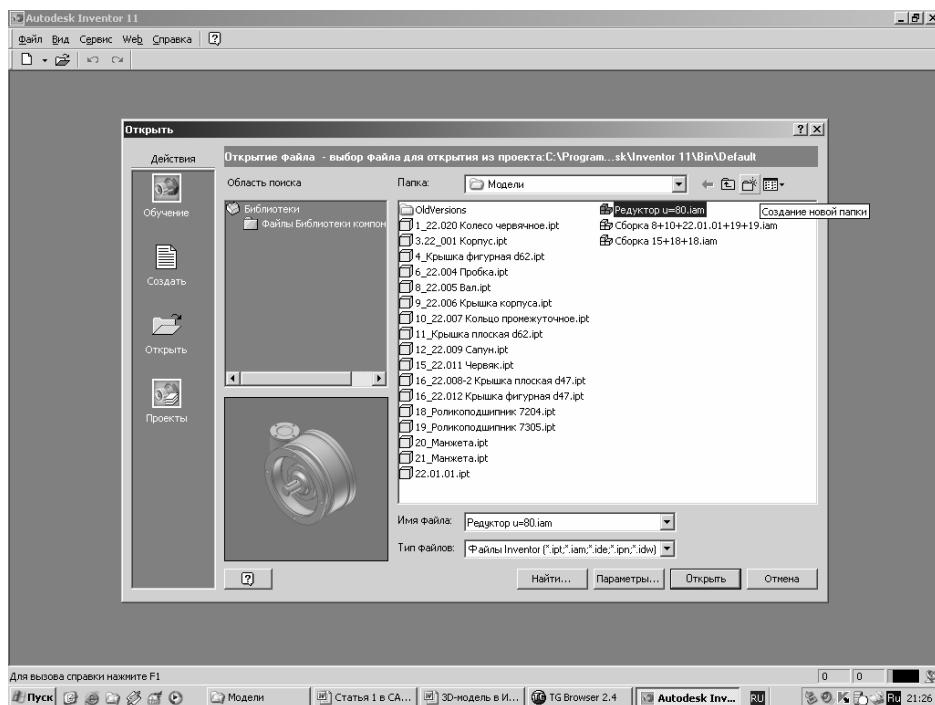


Рис. 1. Интерфейс CAD-системы Autodesk Inventor

При этом в пользу предпочтительного использования трехмерного моделирования в ИПИ-технологиях существует ряд доводов, среди которых:

Улучшенное конструктивное оформление. Трехмерная модель для конструктора – более удобный и эффективный способ воспроизведения замысла. Одним из наиболее очевидных отличий твердотельного моделирования от двумерного черчения является построение точной по размерам трехмерной модели. Благодаря графическим возможностям современных компьютеров, модель можно рассматривать на экране со всех сторон, манипулируя ею, как реальным предметом.

Автоматизированное производство чертежей. Одним из главных преимуществ программ 3D-моделирования является их способность быстро создавать точные 2D-чертежи.

Упрощенная модификация чертежей. Программы 3D-моделирования позволяют легко изменять чертежи, а по ним – уже существующие конструкции. Интеллектуальные функции пакетов 3D-моделирования ускоряют процесс модификации. Параметрический подход позволяет конструктору задать новые размеры, после чего программа пересчитает все изменения, касающиеся тех деталей модели, для которых определены эти размеры и автоматически обновит всю модель.

Интеграция с другими программами. Достоинством технологии твердотельного моделирования является возможность последующей обработки полученных результатов с помощью других программ, связанных, например, с анализом и производством.

Укороченный цикл проектирования. Преимущество технологии твердотельного моделирования связано с возможностью поддержания своей конкурентоспособности за счет сокращения цикла проектирования.

На этапе маркетинго-рекламной деятельности создается ЭМ, демонстрируемая заказчику и, в случае необходимости, корректируемая под его требования. Предпочтительными для этой цели служат технологии виртуального и анимационного моделирования, встраивания в окружающую среду и демонстрация функционирования изделия. Важными компонентами здесь являются: фотorealистичность, текстура, цвета, освещение и т.п. На рис. 2 представлена твердотельная модель редуктора, выполненная в технологии *Autodesk Inventor*, которая может быть использована на этапе маркетинго-рекламной деятельности.

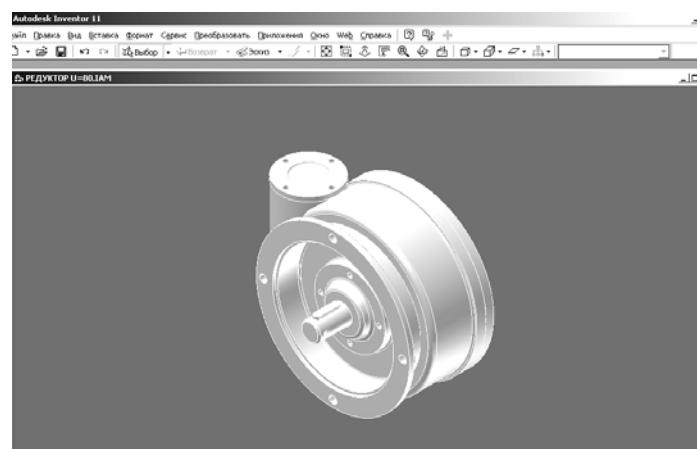


Рис. 2. Твердотельная модель редуктора
(выполнено в технологии *Autodesk Inventor*)

На этапе **подготовки производства** ЭМ играет многофункциональную роль. При **конструкторской подготовке производства**, кроме твердотельных ГМ, выполняются ГМ другого типа: каркасные и поверхностные. На рис. 3 и 4 модель редуктора отображена соответственно в каркасном режиме и в режиме с невидимыми линиями. При этом ГМ может быть выполнена как в неизменяемом, так и в параметрическом виде. В последнем случае величина и форма ГМ зависит от параметров, причем изменение каждого параметра оказывает влияние на ГМ. ГМ хранятся в базе данных изделия и предназначаются для различных инженерных задач проектирования и производства. Возможно создание по 3D-модели 2D-чертежа (рис. 5 и 6).

На стадии конструкторской проработки производится **инженерный анализ** изделия (ИА) и его составных узлов и деталей (*Computer Aided Engineering – CAE*). Разновидностями ИА являются: прочностной, кинематический, тепловой, гидроаэродинамический, акустический и другие. При проведении ИА во всех случаях первоисточником служит та или иная ГМ. Современный ИА проводится с использованием промышленных программных комплексов, базирующихся на конечно-элементных расчетах. При этом ГМ может быть сформирована как в самом программном комплексе (если такой модуль в нем предусмотрен), так и сторонней *CAD*-системой. Последний вариант с точки зрения ИПИ-технологии и формирования ЕИС представляется предпочтительным. Созданная однажды ГМ используется на всех стадиях ЖЦ изделия, включая и ИА.

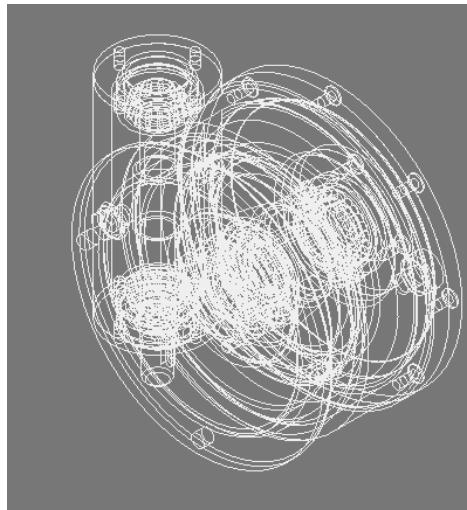


Рис. 3. Модель редуктора отображена в каркасном режиме

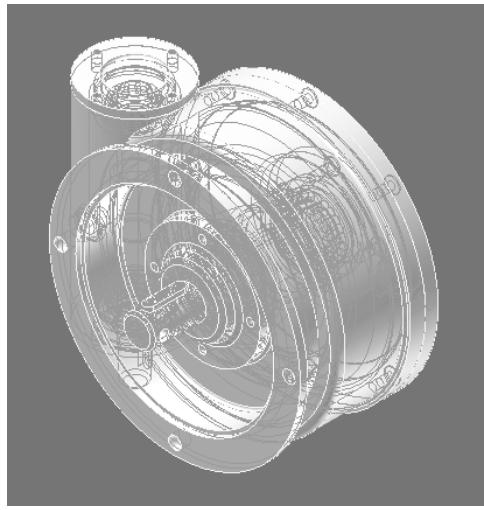


Рис. 4. Модель редуктора отображена в режиме с невидимыми линиями

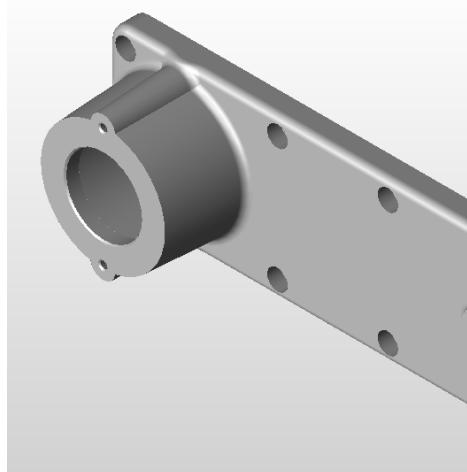


Рис. 5. 3D-модель детали

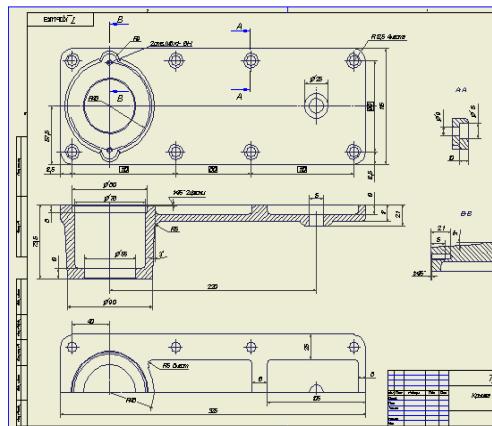


Рис. 6. 2D-чертеж детали (получено из 3D-модели инструментарием Autodesk Inventor)

Некоторые из систем ИА (например, *MSC.visualNastran Desktop 4D* – приложение для теплового, кинематического и прочностного анализа отдельных элементов и сборок проектируемых конструкций; *MSC.Dynamic Designer* – модуль для динамического и кинематического анализа механизмов корпорации *MSC.Software Corp*) полностью интегрированы с CAD-системами, напрямую встраиваются в них (*Autodesk Mechanical Desktop* и *Autodesk Inventor*, *Pro/Engineer*, *Solid Edge*, *Solid Works*) и динамически от-

слеживают все введенные конструктором изменения. Геометрические интерфейсы поддерживают форматы *ACIS*, *Parasolid*, *STEP*, *IGES*, *STL*. Конечно-элементный расчет напряженно-деформированного состояния, собственных частот, устойчивости, стационарного и нестационарного теплообмена в элементах конструкции, а также автоматическая оптимизация геометрии и топологии выполняются в тесном взаимодействии с *CAD*-системой.

При построении динамической модели механизма шарниры и связи ГМ автоматически преобразуются в шарнирные соединения динамической модели с возможностью последующей корректировки со стороны пользователя.

Препроцессор пакета *MSC.Nastran for Windows* той же корпорации, решающий широкий класс задач механики деформирования твердого тела и анализа тепловых процессов, включает в себя различные методы формирования ГМ (в том числе и твердотельных), автоматическую и полуавтоматическую генерацию конечно-элементной сетки (рис. 7), а также интерфейсы с *CAD*-системами [3].

ГМ может быть использована для анализа структуры течений в трехмерном пространстве изделия [4] (рис. 8).

ГМ играют ключевую роль и на этапе **технологической подготовки производства** (*Computer Aided Design, CAM*). Речь идет о программном обеспечении для создания технологических электронных макетов, подготовки процесса механообработки и получения готовых изделий или сопутствующей оснастки на станках с числовым программным управлением (ЧПУ) и их контроля на координатно-измерительных машинах. При этом используются данные двумерных чертежей или трехмерных моделей, содержащих поверхностную, твердотельную или гибридную геометрию. Существуют многочисленные технологические программы, однако их структура и принципы работы во многом похожи. Среди них *SolidCAM* – полностью локализованное технологическое решение для подготовки в графической среде *Autodesk Inventor* управляющих программ механообработки на станках с ЧПУ (рис. 9). Программа *SolidCAM* [5] является собой инструмент создания управляющих программ для токарной, фрезерной, токарно-фрезерной и электроэррозионной обработки. Она поставляется как с собствен-

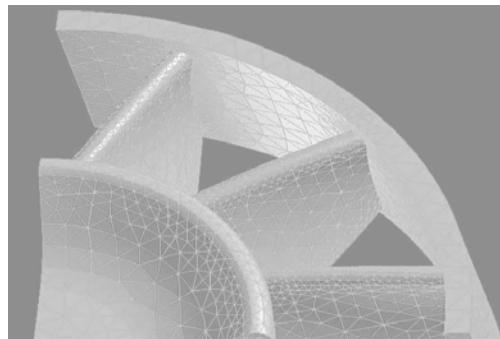


Рис. 7. Конечно-элементная сетка на ГМ изделия [3]

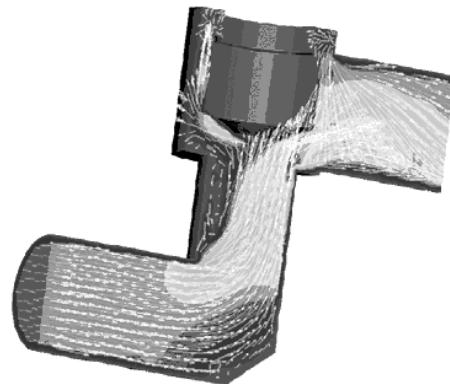


Рис. 8. Расчет 3D-течения жидкости в кране с частично закрытым проходным сечением (АО ПК «Сплав») [4]

ным графическим редактором, обеспечивающим каркасное, поверхностное и твердотельное моделирование, так и в виде интегрированного приложения к конструкторским программам, таким, как *Autodesk Inventor*, *AutoCAD*, *Mechanical Desktop* и др.

Важной особенностью совместного использования ГМ и функциональных возможностей *Autodesk Inventor* по различному представлению трехмерной ГМ в границах допусков позволяет *SolidCAM* рассчитывать траекторию обработки не по номинальному размеру, а с учетом заданных размерных припусков. Для трехмерных твердотельных, поверхностных, смешанных моделей и сборочных узлов обеспечена поддержка различных типов стратегий черновой и чистовой обработки (растровая обработка с заданным или автоматически определяемым углом раstra, обработка по эквидистанте, проекционная обработка, обработка по ватерлиниям и др.). Отметим также такие специальные типы обработки, поддерживаемые программой, как черновая обработка врезанием (рис. 10) и обработка отверстий на трехмерной модели.

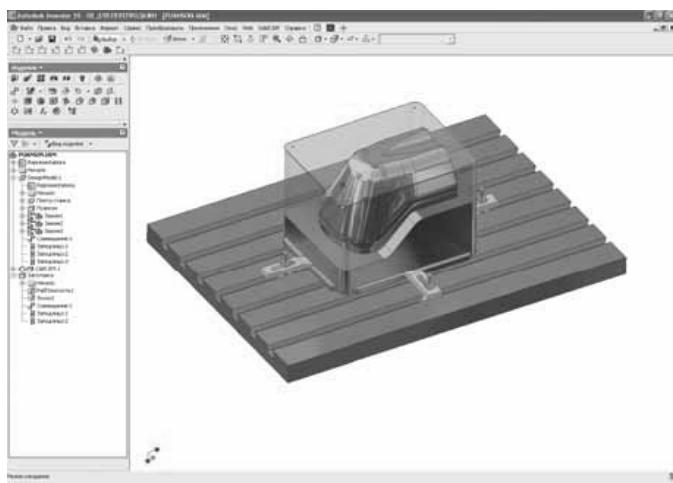


Рис. 9. ГМ в окне интерфейса программы *SolidCAM* [5]



Рис. 10. Черновая обработка врезанием на трехмерной модели [5]

На постпроизводственной стадии для осуществления интегрированной логистической поддержки (ИЛП) изделия эксплуатирующая организация использует созданное проектантами и переданное ей интерактивное электронное техническое руководство (ИЭТР). В структуру ИЭТР составной частью, как правило, включается ГМ изделия и его компоненты [6]. При этом в ИЭТР могут быть включены точные модели деталей и их чертежи, а также модели сборочных узлов и схемы разного содержания и назначения. Наилучшей наглядностью для обслуживающего персонала обладают ролики с анимационными моделями «сборки – разборки» узлов изделий, разработанные на базе их виртуальных моделей.

Таким образом, 3D-модели в ИПИ-технологиях используются практически на всех стадиях ЖЦ изделий и их применение все более расширяется.

Список литературы

1. Информационное обеспечение, поддержка и сопровождение жизненного цикла изделия/ Бакаев В.В., Судов Е.В., Гомозов В.А. и др. / Под ред. В.В. Бакаева. – М.: Машиностроение-1, 2005. – 624 с.
2. www.autodesk.ru
3. www.consistent.ru
4. Девятов С. Компьютерные технологии инженерного анализа в новом тысячелетии // CADmaster. 2002. – № 5. – С. 2–10.
5. Благодаров А. Мечты сбываются, или как сказка стала былью // CADmaster. – № 5. 2005. – С. 20–27.
6. Власов С.Е., Перенков С.А., Сидорук Р.М. и др. Разработка ремонтной и эксплуатационной документации программно-технических средств АСУ технологическими процессами АЭС на основе интерактивных электронных технических руководств. // Информационные технологии в проектировании и производстве. 2005. – № 3. – С. 90–94.

ФОРМИРОВАНИЕ ЭЛЕКТРОННОГО АРХИВА ИНФОРМАЦИОННО-ГРАФИЧЕСКИХ РАБОТ ВЫПУСКНИКОВ ТЕХНИЧЕСКОГО УНИВЕРСИТЕТА

Л.И. Райкин, К.В. Лупанов, А.В. Лексиков

*Нижегородский областной центр новых информационных технологий
Нижегородского государственного технического университета*

Наиболее сложными для большинства организаций остаются проблемы создания и ведения архивов документов, причем представленных как в электронном, так и бумажном виде. Любое современное предприятие, использующее информационные технологии (ИТ) в своей повседневной деятельности, разрабатывающее и производящее изделия, нуждается в создании и ведении электронных архивов. В качестве архивных материалов выступают архивные и вновь разработанные электронные модели, чертежи, конструкторско-технологическая документация, извещения об изменениях, документация

для заказчика и от партнеров, субподрядчиков и т.п. Создание такого электронного архива возможно с помощью программных средств управления данными об изделии (Product Data Management – PDM), являющихся составной частью интегрированного информационного пространства (ИИС) технологий информационной поддержки жизненного цикла изделий (ИПИ-технологий).

В основе PDM-системы лежит представление инженерных данных об изделии в виде древовидного или сетевого графика. Вершинами графа являются либо компоненты изделия, либо связанные с ними бизнес-процессы и используемые ресурсы [1]. С вершинами графа могут быть связаны документы и характеристики объектов (изделий, процессов, ресурсов, документов).

На современном рынке существуют сотни интегрированных и автономных PDM-систем, хотя приемлемой методики их выбора на данный момент не разработано. Известно [2], что целью PDM-системы является обеспечение полноты, целостности и актуальности информации об изделии и доступность ее всем участникам жизненного цикла (ЖЦ) изделия в соответствии с имеющимися у них правами. Для достижения этой цели из шестнадцати функций PDM-системы применительно к электронным архивам должны выполняться по крайней мере три:

- хранение различных документов;
- поиск объектов баз данных;
- генерация отчетов.

Назначение разработанной в НОЦ НИТ НГТУ информационной системы (ИС) «Электронный архив информационно-графических ресурсов» – собрать всю информацию о дипломных работах студентов кафедры ГИС в интегрированной базе данных, обеспечить ее актуализацию и использование в процессе научно-исследовательской и учебно-методической деятельности.

При разработке данной ИС в качестве технологической платформы использована компьютерная система управления данными о машиностроительном изделии PDM STEP Suite (PSS). В ее основе лежит международный стандарт ISO 10303 (STEP), определяющий схему (модель) данных в базе данных (БД), набор информационных объектов и их атрибутов, необходимых для описания изделия. PSS определяет интерфейс доступа к данным: через текстовый обменный файл или программный интерфейс (API). БД может содержать информацию о структуре, вариантах конфигурации изделия и необходимости компонентов в различные изделия, идентификационную информацию об изделии и его компонентах, технологии изготовления, геометрические модели различных типов и/или электронные образы бумажных документов (чертежей), данные об организационной структуре предприятия и соподчиненности ее элементов, ролях и полномочиях людей, данные о процессе разработки: статусах, присвоенных результатам работы, проведенных изменениях. Кроме того БД может содержать ассоциированные с элементами изделия документы, а сами элементы – иметь набор функциональных или технических характеристик, измеряемых в различных единицах.

PSS представляет собой трехуровневую информационную систему, состоящую из сервера СУБД (Oracle Server 8.i), сервера приложений (Oracle Client 8.i & PSSOraSrv) и клиентского модуля (PSS). Клиентский модуль обеспечивает диалоговое взаимодействие с БД через сервер приложений. Трехуровневая архитектура обеспечивает эффективное распределение вычислительной нагрузки при одновременной работе большого числа пользователей.

Пользователь работает с БД, представляя ее в виде дерева изделия (или пересекающегося семейства деревьев), ветви которого декомпозируются на сборочные единицы (узлы), агрегаты и отдельные детали.

С элементами дерева связаны документы, характеристики и присвоенные им статусы.

Система PSS может хранить данные, необходимые для подготовки электронных публикаций (интерактивных электронных технических руководств на изделие), и взаимодействовать с системой автоматизированной подготовки электронных руководств. В этом случае обеспечивается централизованное управление всеми данными проекта.

В отличие от большинства других систем, уровень доступа определяется не для класса (типа) информационных объектов, а для конкретного информационного объекта, что обеспечивает большую гибкость при организации параллельного проектирования.

В системе имеется встроенный двухуровневый программный интерфейс удаленного доступа, низший уровень которого соответствует спецификации международного стандарта ISO 10303-22 SDAI (Standard Data Access Interface), а высший включает в себя набор высокуровневых функций доступа к данным из разрабатываемых приложений.

Разработанная ИС позволяет реализовать следующие функции:

- регистрацию и учет поступающих документов;
- ввод комплектов электронных документов по разделам в соответствии со структурой архива;
- поддержку версий документов;
- управление структурой и физическим размещением данных для записи в архив электронных комплектов документов;
- поиск проектов и их частей, включенных документов по тематике и другим регистрационным атрибутам описания проектов;
- выполнение заказов на создание копий документов.

В качестве исходных данных для разработанной ИС были взяты информационные ресурсы с компакт-дисков дипломных работ студентов кафедры ГИС (группы 94-КГ, 95-КГ, 96-КГ, 97-КГ, 98-КГ, 99-ИСТ, 00-ИСТ). Материал был полностью переработан, отформатирован и приведен к необходимым стандартам.

На первом этапе проектирования ИС в соответствии с поставленной задачей были определены её необходимые функции. Далее была разработана общая структура системы (структура категорий, организационная структура, типы документов, статусы) и назначены роли.

На последнем этапе подготовленные исходные данные с помощью архиватора WinRAR и программного продукта PSS были объединены в общую систему соответственно разработанной структуре.

ИС позволяет управлять различными справочниками и классификаторами изделий. Справочники строятся при помощи объекта «Категория». Под категорией понимается объект, назначением которого является объединение изделий в группу по какому-либо признаку. Для организации электронного архива кафедры ГИС в качестве категорий использованы названия групп с вложенными в них категориями студентов.

Для категорий студентов в качестве наименования используются фамилия, имя и отчество, а для обозначения используется номер зачетной книжки студента, что обеспечивает необходимую уникальность обозначения.

При занесении документа в электронный архив будет запрошено уникальное обозначение документа. Обозначение используется в базе данных для однозначной иден-

тификации объекта. В качестве обозначения документа предлагается использовать следующий код:

*{Номер зачетной книжки студента, выполнившего данную работу}
{разделитель(точка)} {код предмета} {разделитель(точка)}
{порядковый номер работы}.*

Например: 991488.123.1 – означает, что эту работу выполнил студент с номером зачетной книжки 991488, работу по предмету с кодовым номером 123, порядковый номер работы – 1.

Порядковый номер работы является необязательной частью кода.

Для электронного архива кафедры ГИС в организационную структуру включены преподаватели. Роль сотрудника служит для того, чтобы сотрудники получали доступ к данным в соответствующем контексте. В структуре архива *контекст* служит для удобства представления информации. В качестве контекста выступает тот или иной *предмет*. Типы документов обозначают тип работы студента (дипломная работа, презентация, пояснительная записка и т.д.).

Статусы предлагается использовать в качестве оценок – после внедрения документа в БД преподаватель присваивает статус (ставит оценку) этой работе.

Вышеописанная структура базы данных исключает потерю и неумышленную порчу документов, хранящихся в электронном архиве, вследствие неумелого обращения, а также обеспечивает удобный поиск по всем атрибутам документов.

В целях экономии места и для удобства занесения в электронный архив, предлагается архивировать заносимые в БД документы. После того как преподаватель проверит и оценит дипломную работу студента, последний должен будет заархивировать ее и сдать преподавателю для занесения в электронный архив.

Чтобы запустить программу электронного архива необходимо:

- отработать цепочку Пуск-> Программы-> PDM STEP Suite-> PDM модуль или дважды щелкнуть по иконке PDM модуль на рабочем столе;
- в окне «Установка соединения с БД» в поле со списком «Пользователь» выбрать свою фамилию и инициалы (если они уже существуют) или написать их самостоятельно, если вы заходите в систему первый раз;
- в поле «Пароль» ввести свой пароль, который можно получить у системного администратора. В дальнейшем с целью предотвращения захода в систему под вашим именем вы можете изменить пароль;
- Оставить значение по умолчанию (Ist), если в поле со списком «Подключение» указана база данных, с которой вы желаете установить соединение.
- нажать OK (Появляется окно модуля PDM).

Предусмотрена возможность занесения дипломной работы студента (презентации, пояснительной записки, технического задания, рецензии, титульного листа) в электронный архив. Для этого необходимо:

- распахнуть дерево «Папки», щелкнув на плюсе рядом со значком папки;
- распахнуть папку с названием того предмета, преподавателем которого Вы являетесь;
- распахнуть категорию с названием группы, щелкнув по плюсу, и выбрать студента, работу которого Вы собираетесь занести в электронный архив.

Для корректной работы ИС необходима установка следующего программного обеспечения:

Сервер

Операционная система Windows 2000 и выше;

СУБД Oracle 8i и выше;

Система управления электронным документооборотом PSS.

Клиент

Операционная система Windows 2000 и выше;

Архиватор WinRAR.

Аппаратное обеспечение электронного архива

Разработка подсистемы долгосрочного хранения является достаточно сложной проблемой. Это связано с большими объемами электронной информации, включая информационно-графический контент. В качестве устройств долговременного хранения могут быть использованы роботизированные CD-DVD.

Библиотеки под управлением специализированного программного обеспечения. Система долгосрочного хранения является одной из важнейших во всей структуре электронного архива и выполняет следующие задачи:

- организации хранения больших объемов информации;
- организации доступа к большим объемам информации;
- организации пополнения хранящейся информации по мере её накопления.

К модулю долгосрочного хранения предъявляются следующие требования:

- надежность хранения информации (физическая, техническая надежность и возможность разграничения по правам доступа);
- возможность расширения объемов хранения;
- возможность увеличения производительности при увеличении числа пользователей;
- наращиваемость системы;
- возможность дальнейшего развития решения при дальнейшем развитии информационных технологий при исключении больших финансовых затрат.

Роботизированная CD-DVD библиотека под управлением специализированного программного обеспечения является высокотехнологичным накопителем информации. Носителями информации являются CD и DVD диски, являющиеся в настоящее время самым надежным и развивающимся носителем, позволяющим иметь удельную стоимость хранения единичного объема на порядок ниже по отношению к HDD и Raid-массивам. Кроме того, применяемые носители менее критичны к условиям эксплуатации (магнитные поля, температура и т. д.), не требуют перемагничивания и прочих процедур, необходимых при обслуживании жестких дисков. Время хранения информации по оценкам разных экспертов от 50 до 100 лет. Все носители размещены в одном корпусе и при помощи специализированного программного обеспечения объединены в один локальный или сетевой ресурс сверхбольшого объема, администрируемый по правам пользователей и позволяющий обращаться к нему как в «файл-серверном» режиме, так и через вышестоящее программную надстройку (СУБД, систему автоматизации и т. д.). При обращении к тому или иному файлу робот, размещенный внутри корпуса, извлекает носитель и вставляет его в привод, далее, через SCSI – интерфейс, осуществляется его передача на управляющий компьютер и при помощи программного обеспечения (ПО) файл отдается в сеть.

Роботизированные библиотеки позволяют:

о разместить в одном корпусе ~ от 120 до 600 CD-DVD носителей; в зависимости от модели число дисков может быть ~ 135, 300, 400 или 600. Каждый диск размещается в отдельном слоте внутри корпуса библиотеки. Особенностями накопителей предлагаемого модельного ряда в организации размещения носителей являются:

- использование технологии «No Touch», заключающейся в том, что каждый носитель хранится в отдельном пластиковом конверте, исключающем механическое прикосновение к диску, и, как следствие, не допускаются повреждения, влияющие на работоспособность системы;
- возможность подключения дисков через mail-slot (по одному отдельно хранимому диску) или при помощи дополнительных магазинов. Каждый дополнительный магазин содержит по пятнадцать дисков, необходимость постоянной работы с которыми отсутствует. Магазины могут храниться отдельно (вне корпуса библиотеки и подключаться при необходимости). Важной особенностью подключения магазинов является отсутствие необходимости выключать или перезагружать систему;
- о представить весь массив носителей в виде локального или сетевого ресурса (проще говоря, в виде одного диска размером до ~ 3 терабайт;
- о организовать доступ к хранимой информации, как к сетевому ресурсу;
- о организовать запись на CD и DVD-носители при увеличении объемов информации (пополнении архива);
- о повышать производительность системы (модульность устройств позволяет наращивать число приводов). Так, например, можно установить базовый модуль на 620 слото-мест с одним приводом. При наполнении архива и увеличении интенсивности доступа к нему число приводов может быть увеличено до 14;
- о переходить на новые, постоянно развивающиеся технологии (например, сейчас наивысшим стандартом является DVD-4.7 GB. Приводы указанного стандарта читают CD, DVD-2.6, DVD-4.7. При появлении новых стандартов библиотека не «устаревает». Необходимо лишь установить новый привод);
- о увеличивать объемы хранимой информации. Так, например, при установке пишущих DVD приводов, возможна перезапись с существующих CD носителей на DVD, что позволит при использовании той же модели, при том же числе слото-мест значительно повысить «емкость» хранилища.

Задача создания подсистемы долгосрочного хранения подразумевает высокую надежность системы и большое время наработки на отказ.

Список литературы

1. Судов Е. В. Интегрированная информационная поддержка жизненного цикла машиностроительной продукции. Принципы. Технологии. Методы. Модели. – М.: ООО Издательский дом «МВМ», 2003. – 264 с.
2. Информационное обеспечение, поддержка и сопровождение жизненного цикла изделия / Бакаев В.В., Судов Е.В., Гомозов В.А. и др. / Под ред. В.В. Бакаева. – М.: Машиностроение-1, 2005. – 624 с.
3. Руководство пользователя PDM STEP Suite «Модуль PDM», «Настройка системы», «Системное администрирование», «Техническое описание».

КОРРЕКЦИЯ НЕЛИНЕЙНОСТИ ФОТО- И ВИДЕОИЗОБРАЖЕНИЙ

М.О. Рубцов

Мордовский государственный университет им. Н.П. Огарева

Постановка задачи

Обработка изображений на ПК становится одним из привычных занятий. Источником изображения могут быть сканированные фотографии и негативы, оцифрованные видеозаписи, цифровые фотографии и видеозаписи цифровых камер. Качество зависит от того, насколько сильно проявляются стандартные дефекты таких изображений. К ним относятся:

1. Добавленные шумы. Процесс оцифровки вносит определенные шумы, так же как и процесс сжатия. Этот дефект, как правило, не поддается контролю со стороны пользователя.

2. Смазанное изображение, плохое освещение сцены. Эти дефекты устраняются улучшением условий самой съемки, приемов использования съемочной аппаратуры.

3. Хроматическая аберрация. Проявляется в «радужности» контуров светлых контрастных объектов. Дефект характерен для простых объективов. В объективах со сложной оптической системой хроматическая аберрация компенсирована.

4. Потеря резкости на краях изображений. Дефект также присущ простым объективам. Полностью восстановить резкость теоретически невозможно из-за наличия шумов оцифровки и сжатия. В объективах со сложной оптической системой этот дефект также компенсирован.

5. Нелинейность изображения. Дефект дают сложные оптические системы, в которых компенсированы два предыдущих дефекта. Незначительная нелинейность не нарушает восприятия человеком сцены. Проявляется в том, что прямые линии сцены на изображении становятся дугами. Существует особенность: для устройств с переменным фокусным расстоянием нелинейность также переменная.

6. Равномерное сжатие. Чаще всего возникает при сканировании фотографий и негативов (причина – сканер), а также при конвертировании видео из одного стандарта в другой (например, NTSC в PAL, причина – видеопроцессор). Не нарушает линейность изображения.

Данная работа посвящена коррекции двух последних дефектов в постобработке изображений. Работа является актуальной, поскольку наличие нелинейности очень усложняет добавление к изображениям спецэффектов средствами трехмерной графики. Некоторыми средствами ротоскопии может проводиться коррекция равномерного сжатия. Нелинейность изображений, как правило, не компенсируется. В зависимости от величины этой нелинейности последствия бывают:

- легкие – точки, находящиеся в реальности на одной плоскости (например, пол или стена) после ротоскопии будут лежать на ощутимо выпуклой поверхности. Использование такой опорной поверхности при производстве спецэффектов доставит множество проблем;

- тяжелые – ротоскоп может вообще не найти удовлетворительного решения, когда например, нужно «вписать» в видеосъемку уже готовую модель помещения.

Проблемы с искаженным изображением возникнут и на этапе синтеза спецэффектов. Синтезированное системой рендеринга изображение, как правило, строго линейно.

Смешивание его с искаженным изображением приводит к тому, что объекты разной природы будут в кадре «разъезжаться». Это особенно заметно при использовании фантомов-двойников (например, при получении сложных теней от новых объектов). Для таких объектов важно точное совпадение их границ с отснятыми границами их прототипов. Нелинейные искажения приводят к тому, что границы «сползают» при движении камеры. Таким образом, компенсация нелинейных искажений является важнейшим этапом при производстве спецэффектов и в других приложениях фотометрии. Основной задачей данной работы является создание эффективного средства для восстановления линейности изображения. Линейность восстанавливается при съемке тестового изображения.

Краткое описание проведенного исследования

Все измерения на изображении производятся относительно характерного размера изображения. Это связано с тем, что изображения с одного физического объектива могут быть оцифрованы с разным разрешением либо преднамеренно уменьшены с целью создания черновых вариантов спецэффектов. В качестве такого характерного размера принята высота изображения (из соображения фиксированного числа строк развертки для форматов видео), она соответствует единице. Дефект сжатия описывается как отношение ширины исходного изображения к ширине, которую будет иметь изображение после коррекции этого дефекта. Формализация нелинейного искажения немного сложнее.

Рассмотрим модель прямой трассировки (рис. 1). Для простоты изобразим весь объектив, как одну линзу, и выделим луч, проходящий от некоторой точки S через центр внешней линзы в объективе.

Рассмотрим расстояние от центра фоточувствительного слоя до положения изображения точки S . В случае идеального линейного проецирования это расстояние было бы равно r . В действительности оно равно R . Для сохранения линейности преобразования необходимо, чтобы $R = kr$, где k не зависело бы от r . В действительности $k = k(r)$.



Рис. 1

Из практических наблюдений известно, что в окрестности главной оптической оси нелинейные искажения отсутствуют. Также известно, что $k(r)$ – гладкая функция (иначе гладкие реальные линии на изображении были бы с изломами, чего никогда не наблюдается, они остаются гладкими). Поэтому функция $k(r)$ аппроксимируется полиномом:

$$k(r) = 1 + a_1 r + a_2 r^2 + a_3 r^3 + \dots + a_n r^n.$$

Наибольший порядок полинома, который теоретически имеет смысл вычислять для данного изображения, строго говоря, зависит от размеров изображения и его характеристик зашумленности. Ввиду сложности данный расчет здесь не приводится. Заметим лишь, что для практических задач редко требуется полином выше третьего порядка.

Для примера на рис. 2 представлены расчеты в системе MATLAB для модели нелинейного искажения с пропорциональным углом передачи и ее аппроксимация полиномом четвертой степени.

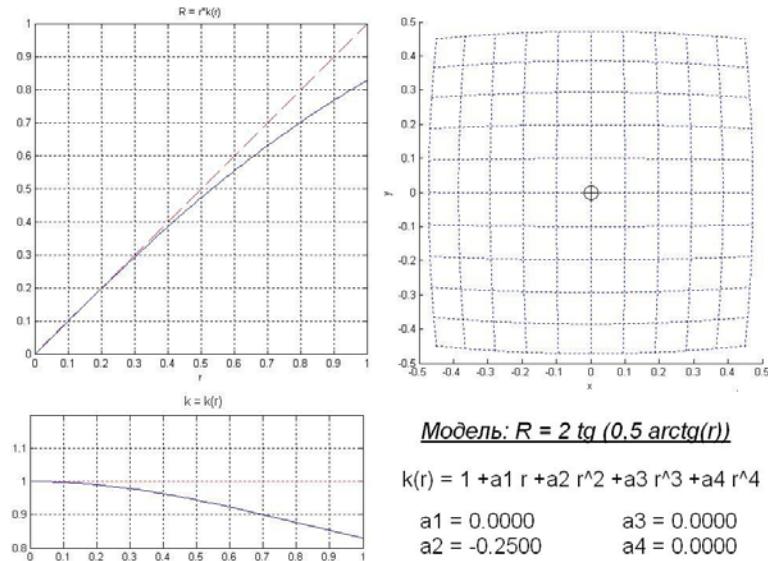
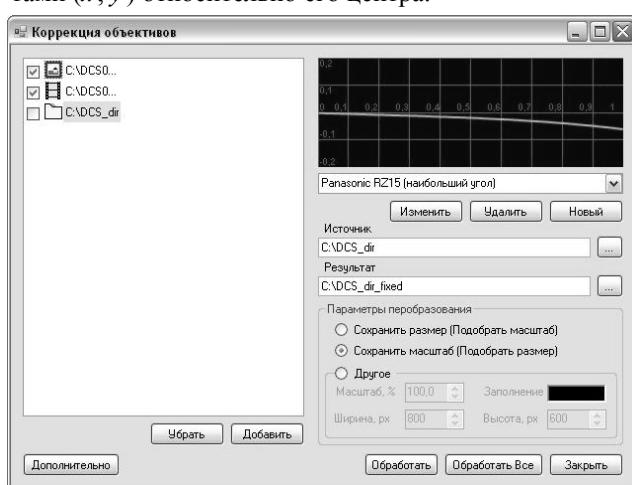


Рис. 2

При известных коэффициентах полинома процесс коррекции искажения не представляет особых сложностей. Для каждой точки восстанавливаемого изображения нужно вычислить:

- 1) ее координаты x и y относительно центра,
- 2) величину r – ее нормированное расстояние до центра,
- 3) $k = k(r)$ по заданным коэффициентам,
- 4) $x' = w * k * x$; $y' = k * y$ – искаженные координаты, w – коэффициент равномерного сжатия.

Цвет точки принимается равным цвету точки исходного изображения с координатами (x', y') относительно его центра.



Алгоритм коррекции достаточно прост. Он хорошо реализуется с использованием фрагментных шейдеров для повышения производительности корректирующей программы (в случае платформы с мощной видеокартой).

Результаты вычислительных экспериментов

Корректирующая программа имеет простой и функциональный интерфейс, кото-

рый позволяет выбрать множество заданий для обработки в пакетном режиме. В качестве задания может быть выбран один файл неподвижного изображения, файл видео либо каталог с последовательностью изображений одного размера, которые при обработке упорядочиваются по имени и рассматриваются как один видеопоток.

Для каждого из заданий можно указать объектив из набора, выбрать формат сохранения результата, а также дополнительные параметры преобразования, связанные с изменением масштаба и размера выходного изображения. После настройки всех заданий их можно запустить на обработку. Программа обработает помеченные задания (и удалит их из списка, если это необходимо).

Характеристики объективов хранятся на диске в специализированном файле. Интерфейс позволяет удалять объективы из набора, добавлять в набор новые объективы и изменять характеристики. Получение характеристик новых объективов наряду с коррекцией изображений является одной из главных возможностей программы. Изменять характеристики можно в двух режимах.

- Режим «эксперта». В этом режиме пользователь самостоятельно подбирает характеристики объектива. Это происходит следующим образом. В окно «эксперта» пользователь загружает тестовое изображение. Далее он размещает на нем отрезки прямых, концы которых «прикрепляются» программой к изображению. Пользователь изменяет коэффициент равномерного сжатия, вид кривой масштаба или непосредственно коэффициенты кривой. При этом он видит в окне просмотра, как будет выглядеть скорректированное изображение, и по расставленным отрезкам оценивает, насколько правильно производится коррекция. Отрезки на правильно скорректированном изображении располагаются строго вдоль прямых границ на тестовом изображении.

- Автоматический режим. В автоматическом режиме пользователю нужно лишь выбрать одно или несколько тестовых изображений. Далее программа их анализирует. Сначала производится векторизация исходного изображения. Программа распознает на изображении настроечное «шахматное поле», подбирая четырехугольные границы каждого из одноцветных полей. Далее производится выделение горизонталей и вертикалей поля, в действительности – прямых линий. Производится их анализ, в результате которого определяются неизвестные коэффициенты характеристики, а также анализируется точность расчета. Хотя каждое изображение обрабатывается независимо, результаты всех этих обработок статистически суммируются для получения надежной и устойчивой характеристики объектива. Более детальное описание алгоритмов работы автоматического режима не приводится ввиду его объемности и сложности для восприятия.

Основой работы обоих способов является тестовое изображение. Для получения качественных результатов оно должно представлять собой большое клетчатое черно-белое (высококонтрастное) поле, клетки которого ограничены ровными прямыми линиями. Если такого поля в студии нет, его несложно изготовить. Программа помогает пользователю получить такое поле, синтезируя его части и выводя их на печать. Пользователю остается лишь аккуратно склеить эти части и разместить данное поле на ровной поверхности, например на стене.

Приложение разрабатывается на языке C# в среде Microsoft Visual Studio 2005 с использованием Microsoft SDK. Выбор в пользу данной платформы сделан в связи с легкой переносимостью и малым объемом .NET-приложений. На выбор также повлияла возможность применения Managed DirectX для использования фрагментных шейдеров. На момент написания тезиса приложение находилось на стадии перехода от обработки изображений Unmanaged кодом к привлечению аппаратных ресурсов ПК. В работе используется NVIDIA Corporation FX Composer 1.8, NVIDIA SDK 8.0, Autodesk Maya Learning Edition.

Заключение

В соответствие с поставленной задачей в результате проведенного исследования и проектирования было сделано формальное описание дефектов линейности изображения. Было проведено теоретическое исследование возможностей восстановления линейности, разработан алгоритм коррекции изображения. Выполнена реализация с помощью `unmanaged C#`, разработан корректирующий шейдер с помощью FX Composer. Разработан метод анализа нелинейности по тестовому изображению, построения характеристики с точностью, достаточной для выполнения коррекции. Разработано приложение для обработки множества разноформатных заданий. Результаты обработки были подвергнуты ротоскопическому анализу, в ходе которого не было замечено каких-либо характерных признаков нелинейности. Таким образом, поставленная задача выполнена.

О СЕРИАЛИЗАЦИИ ОБЪЕКТНОЙ МОДЕЛИ В БИБЛИОТЕКЕ COLORER

И.В. Русских

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Сегодня объектно-ориентированное программирование (ООП) является основной парадигмой разработки и поддержки программного обеспечения. Развившаяся из структурного программирования, объектная модель наиболее естественным образом описывает структуру и поведение программных систем и позволяет оптимально и безболезненно взаимодействовать с аппаратной моделью вычислительных систем.

Одной из главных черт ООП является модульное построение, в котором независимые компоненты, выполняющие каждый свои задачи, затем компонуются в единую систему. При этом комплексные приложения обычно оперируют огромным числом динамических объектов, многие из которых создаются при активации приложения и существуют длительное время. Загрузка приложения может занимать длительное время в связи с необходимостью создания объектов динамически в памяти и их инициализации. На примере анализа объектной модели библиотеки Colorer 0 можно рассмотреть возможные пути решения подобных проблем на разных этапах создания приложения.

Библиотека Colorer. Обзор [1–6]

Colorer является специализированным синтаксическим анализатором, предназначенный для разбора текста и вносимых в него изменений в реальном времени, дальнейшей подсветке синтаксиса и предоставлению в среде редактирования информации о структуре редактируемого текста. Одной из особенностей, обеспечивающих гибкость и масштабируемость системы, является разработанная модель хранения синтаксических правил и описаний разбора в специальном формате HRC 0. Формат основан на XML и позволяет быстро и эффективно описывать требуемые правила для целевых разбираемых языков программирования. HRC создает дополнительный уровень абстракции над

стандартными средствами синтаксического разбора – регулярными выражениями и контекстами. В модели HRC введены понятия наследуемых контекстов (схем), типов языков программирования. Это позволяет легко описывать сложные, смешанные языки – такие как JSP, PHP, ASP, HTML. Структура этих и многих других языков может содержать в себе смесь более примитивных синтаксисов, и для эффективного их описания введены такие понятия, как наследование контекстов, условное изменение их поведения (виртуализация) и др.

Система Colorer эффективно отличается от решений со схожими задачами 0, 0 эффективностью и простотой описания синтаксисов, реализуя в то же время мощный инструментарий по поддержке работы программиста.

Активное развитие описаний синтаксисов в библиотеке привело к некоторым «отрицательным» последствиям: для загрузки, компиляции и преобразования HRC моделей библиотеке Colorer требуется все больше и больше времени. В абсолютном значении это время не так велико – на данный момент время загрузки всей базы в память средней машины (PIV, 3GHz) занимает около 10 секунд. При том, что библиотека подгружает модели HRC по мере необходимости, загрузка каждого отдельного языка программирования занимает не больше 1 секунды. В то же время такие параметры системы, как общая нагрузка, активное использование виртуальной памяти (swapping), файловой подсистемы, увеличивают среднее время загрузки приложения – для пользователя оно может возрасти до 4-5 секунд.

В случае с библиотекой Colorer возникает еще один фактор, критичный ко времени начальной загрузки. Существуют интерфейсы библиотеки, нацеленные на использование ее в WEB серверах для расцветки в реальном времени исходных текстов в сети интернет. В таких средах нагрузка на сервер зависит от посещаемости ресурса и в некоторых случаях может быть очень велика. Существующее время инициализации библиотеки Colorer совершенно неприемлемо для подобных сред, в которых число запросов может колебаться от единиц до десятков и сотен в секунду.

Конечно, в конкретных окружениях можно избежать подобных проблем. В случае с WEB-средами существуют решения, позволяющие легко и быстро интегрироватьрезидентные сервисы в систему web-сервера (такие как FastCGI для IIS).

Но все же в общем случае подобные решения являются обходными маневрами, никак не раскрывающими проблему изначальной сложности программной среды. Так, подобные решения не применимы во встраиваемых системах, практически всегда они требуют осторожного обращения в многопроцессорных, многопотоковых средах.

Сериализация как решение проблем с производительностью

В случае с библиотекой Colorer описание всей модели, подлежащей загрузке, представлено в форме, удобной человеку. Это XML файлы, которые удобно редактировать вручную (или в XML редакторе) и которые затем на лету компилируются при загрузке во внутреннюю форму. Такие инструменты, как регулярные выражения, синтаксические структуры, также выражены в удобной для человека форме и лишь при загрузке библиотеки компилируются.

Одним из возможных вариантов уменьшения времени загрузки является сериализация модели – частичная или полная. Сериализация предполагает исключение стадии преобразования данных из формы, доступной человеку, в форму, доступную машине. Обычно для этого требуемые объекты в модели доопределяются методами, позволяющими объекту сохранить свой образ в постоянное хранилище (файл, поток) и извлечь его оттуда.

Задача сериализации усложняется при наличии в объектной модели сложных связей между объектами, что практически всегда встречается в реальных приложениях. Для сериализации таких групп объектов необходимо прилагать дополнительные усилия по идентификации объектов, сериализации идентификаторов и ссылок на объекты. Дополнительно к этому ООП добавляет сложности, связанные с полиморфизмом объектов из иерархии наследования. Для поддержки таких моделей необходимо вводить искусственную идентификацию типа сериализуемого объекта. После того, как все эти особенности будут учтены и поддержаны, задача сводится в общем случае к обходу и сериализации графа объектов 0.

На примере библиотеки Colorer предлагается иной подход к решению подобных проблем. Это концепция «заморозки» модели, основная идея которой – автоматическое слежение за создаваемой объектной моделью и связями между объектами. С точки зрения языка C++ каждый объект в модели занимает некоторое пространство в динамической памяти. Побайтовое клонирование не отличает изолированное поведение клона от поведения исходного объекта (с некоторыми допущениями). Клонирование уже загруженной объектной модели целиком и сохранение ее в постоянное хранилище позволяет восстановить этот клон единым блоком. Все связи между объектами, какими бы сложными они ни были, восстанавливаются автоматически на основе информации о положении исходных динамических объектов.

Решение по концепции напоминает функцию заморозки (*hibernation*), позволяющую быстро сохранить и восстановить после выключения состояние персонального компьютера. Отличие в том, что представленное решение действует на уровне модели программной системы, а не на уровне аппаратной модели или операционной системы. Собственно реализация этого решения совершенно не связана с целевой областью приложения и в библиотеке Colorer представлена в виде изолированного компонента. Заморозка действует на низком уровне (уровне связей указателей) и не зависит от структуры модели и логическими связями между объектами.

Реализация основывается на подмене системных методов выделения и освобождения динамической памяти (это возможно в модели языка C++). Для создания замороженного образа модели приложение начинает загрузку в обычном режиме. Одновременно с этим переопределенные методы работы с памятью контролируют местоположение всех запрошенных в модели объектов. После завершения инициализации модели готовое к работе приложение активирует процедуру сериализации. Объекты побайтово копируются и упаковываются в единый блок, связи между объектами автоматически анализируются и строится таблица относительных ссылок между объектами. Такое построение возможно, т.к. модуль заморозки владеет информацией об адресах объектов в памяти и способен эвристически отличить данные от указателей на объекты.

Схожие идеи в мобильных и real-time системах иногда именуются «Memory Blasting», или «ROMizing». В библиотеке же Colorer замороженная модель фактически является кэшем скомпилированных символьных описаний HRC и позволяет существенно сократить время загрузки библиотеки. Восстановление полной базы из замороженного состояния занимает времени меньше, чем даже частичная динамическая инициализация: при тестировании решения в компиляторе Microsoft Visual Studio десериализация полной объектной модели HRC с устройства хранения занимала не более 0,3 с. Время восстановления базы из ROMized образа пропорционально лишь размеру этого образа и фактически равно времени его чтения с устройства хранения.

Частичная заморозка модели, которая также возможна, уменьшает время инициализации до значения 70 мс. Такой режим может использоваться для работы библиотеки

в режиме веб-сервиса, когда заранее замораживается ограниченный набор наиболее часто запрашиваемых типов данных, а синтаксисы, не попадающие в этот набор, загружаются обычным способом.

Реализация заморозки не требует абсолютно никаких изменений в готовом приложении. Платой за это является довольно жесткий набор ограничений и условий применимости данного решения.

В общем случае замораживаемая модель должна быть «закрытой». В объектах модели не должно присутствовать активных ссылок на какие-либо внешние ресурсы, не поддающиеся сериализации. К таким ресурсам относятся, например, открытые дескрипторы файловых объектов, объектов операционной системы.

В случае с языком C++ его спецификация не разрешает побайтового копирования сложных объектов. Такое копирование может выполняться только встроенным конструктором клонирования, поэтому может существовать риск возникновения проблем при использовании некоторых реализаций языка C++. Реализация заморозки была успешно протестирована на компиляторах Microsoft (платформа Windows).

Следующее ограничение – привязка сериализованной модели не только к аппаратной платформе, но и к версии компилятора, и даже к версии самого приложения. Минимальное изменение в объектной модели и перекомпиляция приложения делает замороженную модель некорректной. Для обнаружения несоответствия между объектной моделью приложения и существующим ROMized образом можно использовать версионность или присвоение временных меток.

При интеграции данного решения в конечные среды могут возникнуть особенности, связанные с необходимостью обработки таблиц виртуальных функций объектов. Во многих операционных средах исполняемый образ приложения может занимать различное место в адресном пространстве. Часто это зависит от способа загрузки приложения, иных факторов (DLL в ОС Windows). Это в свою очередь оказывается на адресах виртуальных функций, которые в реализациях C++ хранятся в контексте объекта. Для корректного функционирования загруженной модели необходимо корректировать эти таблицы с адресами, а это требует наличия у операционной системы интерфейса по определению базового адреса приложения.

Заключение

Несмотря на описанные ограничения, предлагаемое решение является действенной альтернативой иным способам сериализации объектной модели, в первую очередь, из-за простоты применения и внедрения: решение не требует никаких структурных и архитектурных изменений объектной модели приложения.

В отличие от традиционных способов сериализации, заморозка никак не вредит целостности и объектной структурированности модели. Она не требует отказываться от абстракции и объектного представления модели. Решение оптимально подходит для систем с проблемами производительности, которые невозможно разрешить иначе.

Многие из ограничений заморозки модели исчезают при рассмотрении встраиваемых или мобильных систем, что связано с их спецификой (единое адресное пространство, централизованная инициализация). Помимо улучшения производительности, заморозка модели улучшает показатели работы с памятью. Вся модель представляется единым блоком, а это избавляет систему от ресурсоемкой задачи динамического выделения памяти 0.

Возможность реализации подобной системы на других языках программирования зависит от динамических свойств этих языков. Так, система тривиальна и стабильна в языке С: исчезают особенности, связанные с поддержкой виртуальных функций. В иных объектных системах (например, в системах со сборкой мусора) реализация описанного способа невозможна без дополнительной поддержки со стороны среды выполнения (компилятора, интерпретатора).

Предоставление средой информации о внутренней структуре объекта (ссылки, типы полей) потенциально позволило бы убрать эвристическую часть из работы описанного алгоритма, связанную с поиском ссылок и адресов методов в таблицах виртуальных функций. Подобные reflection-механизмы, дающие доступ к структуре самого языка, к сожалению, не стандартизированы языком С++ и недоступны напрямую.

Список литературы

1. Andrew J. Ko and Brad A. Myers. Barista: An Implementation Framework for Enabling New Tools, Interaction Techniques and Views in Code Editors. CHI 2006 Proceedings, 387–396.
2. Boshernitsan M. Harmonia: A Flexible Framework for Constructing Interactive Language-Based Programming Tools, University of California, Berkeley, Technical Report CSD-01-1149, 2001.
3. Jiri Soukup. Taming C++: Pattern Classes and Persistence for Large Projects. Addison-Wesley, 1994.
4. Русских И.В. Основные принципы разработки библиотеки Colorer // Методы и средства обработки сложной графической информации: Тез. докл. – Н.Новгород, 2003. – С. 80–82.
5. Русских И.В. Оптимизация системы динамического распределения памяти в приложениях на примере библиотеки Colorer-take5 // Вестник Нижегородского университета, сер. Математическое моделирование и оптимальное управление. – Н.Новгород. – 2004. – Вып. 1(27).– С. 234–242.
6. Russkih Igor. Colorer-take5 library, HRC Language Reference. <http://colorer.sf.net/hrc-ref/>

ВИЗУАЛИЗАЦИЯ НЕБОЛЬШИХ ВОЛН НА ВОДНОЙ ПОВЕРХНОСТИ СРЕДСТВАМИ MICROSOFT DIRECTX С ИСПОЛЬЗОВАНИЕМ ШЕЙДЕРОВ

A.В. Рябчиков

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Задача фотorealистичного моделирования воды в реальном времени является одной из классических задач компьютерной графики. Это связано с тем, что вода является сложным природным объектом, и ее внешний вид зависит от многих факторов, таких, как окружающий мир или положение наблюдателя.

Задачу визуализации водной поверхности можно разделить на две части, каждая из которых представляет определенный интерес:

- 1) физика поверхности (моделирование волн, воздействий на поверхность, оптимальная триангуляция поверхности),
- 2) имитация оптических эффектов, характерных для воды.

Визуализация воды в целом предполагает решение обеих подзадач, но в некоторых случаях достаточно лишь имитации оптических эффектов, а в качестве поверхности воды можно использовать плоскость. Однако такое возможно только тогда, когда деформация водной поверхности невелика (река, озеро, бассейн, резервуар с водой). В случае сильных деформаций поверхности ее визуализация невозможна без имитации физического поведения, иначе изображение получится не слишком реалистичным.

Возможно, имитация оптических эффектов является более важной проблемой, нежели моделирование точного физического поведения воды. Ведь наблюдатель, смотрящий на воду, в первую очередь обращает внимание на ее неповторимый внешний вид и только потом на физическое поведение.



Постановка задачи

Для создания фотoreалистичной водной поверхности необходимо, прежде всего, определить основные оптические свойства воды, без которых невозможно обойтись при ее визуализации. Далее требуется систематизировать и изучить методы имитации этих свойств. Сюда же включается рассмотрение всех тонкостей данного процесса, выявление проблем и способов их устранения (если такие имеются).

В работе рассматриваются: способы имитации отражений и преломлений, характерных для водной поверхности, их сочетание с использованием коэффициента Френеля, коррекция цвета воды, использование карт нормалей для анимации возмущенной поверхности (эффекты моделируются с помощью пиксельного шейдера). Кроме этого, ставится вопрос об оптимизации шейдера водной поверхности с целью увеличения скорости работы приложения [1–9].

Описание проекта

Результатом проведенных исследований является приложение, использующее *Microsoft DirectX* для визуализации воды. Фактически его основой является пиксельный шейдер, который осуществляет всю работу по вычислению конечного цвета точек воды. Приложение совершают необходимую подготовку данных для их дальнейшей обработки шейдером.

Создание текстур

$$matReflect = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 2h & 0 & 1 \end{pmatrix}$$

Перед визуализацией воды необходимо подготовить текстуры отражения и преломления, так как именно на их основе мы и получаем конечный результат. При их создании мы опираемся на тот факт, что поверхность воды можно представить в виде плоскости с незначительной погрешностью за счет того, что присутствуют лишь небольшие деформации. Для простоты будем считать, что уравнение поверхности воды имеет вид $y = h$. Теперь мы должны зеркально отразить камеру относительно данной плоскости. Камера задает матрицу вида $matView$. Чтобы получить матрицу вида зеркально отраженной камеры, необходимо перемножить 2 матрицы:

$$matReflectView = matReflect \times matView.$$

Теперь можно производить рендеринг объектов сцены, которые будут отражаться в воде. Перед отрисовкой объектов необходимо изменить режим отсечения невидимых граней (из-за зеркального отражения в матрице вида сменится порядок обхода вершин геометрических примитивов).

При создании карты преломления нужно сразу произвести отрисовку сцены в буфер из исходной точки (т.е. в качестве матрицы вида нужно использовать матрицу $matView$).

При создании текстур можно установить их в качестве буфера вывода (render target), но лучше сначала произвести вывод в буфер экрана, а затем скопировать его содержимое в текстуру. Это связано с тем, что при использовании сглаживания рендеринг в текстуру невозможен.

Проекционное текстурирование

$$matReatReming = \begin{pmatrix} 0,5 & 0 & 0 & 0 \\ 0 & -0,5 & 0 & 0 \\ 0 & 0 & 1,0 & 0 \\ 0,5 & 0,5 & 0 & 1,0 \end{pmatrix}$$

Теперь, когда текстуры с отраженными и преломленными объектами готовы, возникает вопрос о том, как наложить их на поверхность воды. Здесь используется метод проекционного текстурирования. Он заключается в том, что текстура проецируется на поверхность

воды из фиксированной точки (в нашем случае из положения камеры). Фактически необходимо лишь вычислить координату точки в пространстве экрана, что делается путем применения преобразования, заданного матрицами вида и проекции. Но кроме этого, необходимо к итоговому результату применить преобразование, которое описывается матрицей (эта матрица служит для преобразования координат точек пространства экрана в текстурные координаты с учетом специфики метода формирования текстур).

Для правильного наложения текстур на поверхность воды необходимо использовать матрицу следующего вида:

$$matProjectTex = matReflectView \times matProject \times matRemapping.$$

Это преобразование применяется к вершине поверхности воды в вершинном шейдере. Результат обрабатывает пиксельный шейдер:

$$ProjectCoord = TexCoord / TexCoord.w$$

(ху-компоненты полученного вектора используются в качестве координат текстуры).

Основные проблемы

В процессе создания карт отражения и преломления необходимо отсекать «лишние» объекты или части сцены, которые не должны отражаться или преломляться.

1. User Clip Planes – не всегда удобны из-за специфики использования.
2. Oblique Frustum Culling – идея метода состоит в преобразовании матрицы проекции так, что ближняя отсекающая плоскость переводится в плоскость воды.

Если не использовать отсечение, то появятся артефакты (для карты отражения это «призраки» подводных объектов, которые будут присутствовать в отражении).

Другая проблема касается объектов, пересекающихся с водой. За счет последующего искажения текстур на их границе с водой будут видны «разрывы». Чтобы избежать этого, можно использовать следующие методы:

1. Небольшой сдвиг отсекающей плоскости. В этом случае снова возникает проблема объектов-«призраков».
2. Введение зависимости силы искажения от глубины. Можно использовать высоту точек ландшафта для определения их глубины. Либо использовать специальную текстуру (fade texture), которая задает силу искажений в разных областях водной поверхности. На ее основе можно так искажать созданные карты отражения и преломления, что рядом с объектами, расположенными в воде, разрывов не будет.

Для динамических объектов нет эффективного способа устранения вышеописанных артефактов.

Анимация водной поверхности

На основе карты нормалей водной поверхности производится расчет смещения каждой точки водной поверхности. Чтобы создавалось впечатление, что вода движется, нужно каким-то образом анимировать нормали. Этого можно достичь:

- 1) использованием одной карты нормалей, которая движется по поверхности воды в заданном направлении и с заданной скоростью (метод «скроллирования»).
- 2) движением нескольких карт нормалей в разных направлениях и/или с разной скоростью.
- 3) использованием заранее сгенерированного циклического набора карт нормалей, из которого на каждом такте выбирается одна в качестве активной.

На основе карты нормалей производится искажение текстурных координат, что приводит к возникновению эффекта «ряби» на воде. Также можно ввести зависимость силы искажения от расстояния (от камеры до точки поверхности)

$$Offset = Normal.xzxz \cdot ReflectRefractScale \cdot clamp(TexCoord.z, 0.5, 5.0)$$

$$ReflectRefractTexCoord = saturate(ProjectCoord.xy + Offset)$$

Цвет воды

В простейшем случае цвет точки водной поверхности можно вычислить на основе смешивания текстур отражения и преломления с помощью приближения формулы Френеля: $R(\alpha) = R(0) + (1,0 - R(0)) \cdot (1,0 - \cos(\alpha))^5$, α – угол между нормалью и направлением взгляда. Для воздуха и воды $R(0) = 0,02037$. Таким образом, цвет воды в данной точке вычисляется интерполяцией двух значений с помощью коэффициента Френеля:

$$Color = lerp(RefractionColor, ReflectionColor, Fresnel).$$

Для большей реалистичности нужно учитывать оттенок воды. Одним из способов реализации этого эффекта является рендеринг объектов с учетом тумана при создании карты преломления. Тогда цвет воды будет зависеть от глубины в каждой конкретной точке. Но можно просто задать два оттенка воды – для отраженных и преломленных объектов, и перед смешиванием цветов, взятых из соответствующих текстур, умножить их соответственно на эти оттенки (цвет дна не будет зависеть от глубины).

$$ReflectionColor = ReflectionColor \cdot ReflectTint$$

$$RefractionColor = RefractionColor \cdot RefractTint$$

Шейдеры и их оптимизация

Весь процесс визуализации воды строится на основе шейдеров, написанных на HLSL. Для большего удобства, вместо отдельных шейдеров используется один файл эффектов (это особенность DirectX, начиная с версии 9.0).

В результате объединения всех методов, мы получаем весьма внушительный по количеству арифметических инструкций пиксельный шейдер версии 2.0 (вершинный шейдер довольно прост и имеет версию 1.1). Для увеличения быстродействия приложения стоит немного оптимизировать пиксельный шейдер. Все оптимизации сводятся к уменьшению количества используемых инструкций за счет замены арифметических действий на выборку значений из текстур (используются текстурные шейдеры для процедурного создания текстур, хранящих коэффициент Френеля, и нормализованные векторы для упрощения процесса нормализации). Таким образом, быстродействие шейдера заметно возрастает.

Результаты экспериментов

При запуске тестового приложения на CPU AthlonXP 1800+ и GPU GeForce 6600GT в оконном режиме 800x600 производительность составила примерно 150 FPS. При этом внешний вид водной поверхности весьма реалистичен, что доказывает адекватность применения описанного подхода во многих приложениях.

Список литературы

1. Материалы сайтов для разработчиков игр [HTML] (www.gamedev.ru, www.gamedev.net).
2. Примеры и материалы с сайта разработчиков NVIDIA и ATI [HTML] (<http://developer.nvidia.com>, <http://www.ati.com/developer>).
3. Lasse Staff Jensen and Robert Golias. Deep-Water Animation and Rendering. Gamasutra, September 26, 2001 [HTML] (<http://www.gamasutra.com/gdce/2001/jensen/>).
4. Rim van Wersch. Reflection using projective textures on the fixed function pipeline. [HTML] (<http://www.mdxinfo.com/tutorials>).
5. Pieter Germishuys. How to simulate water using shaders. [HTML] (<http://www.mdxinfo.com/tutorials>).
6. Елыков Н. А., Белаго И. В., Кузиковский С. А. Методы визуализации и анимации моделей протяженных водных поверхностей в системах виртуальной реальности. GRAPHICON 2005 [PDF].
7. John Isidoro, Alex Vlachos, Chris Brennan. Rendering Ocean Water. ShaderX [PDF].
8. John Isidoro, Alex Vlachos, Chris Oat. Rippling Reflective and Refractive Water. ShaderX [PDF].
9. Ben Humphrey. Realistic Water Using Bump Mapping and Refraction. [PDF] (<http://www.gametutorials.com>).

ПРИМЕНЕНИЕ МОДУЛЯ РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ДЛЯ ПОИСКА ОПТИМАЛЬНОГО ПУТИ В ГРАФЕ, ОПРЕДЕЛЯЮЩЕГО ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗУЧЕНИЯ ОБРАЗОВАТЕЛЬНОГО КУРСА

А.Е. Рыбаков, И.Г. Сидоркина, Р.О. Савельев

Марийский государственный технический университет

Задача по построению аддаптивной траектории изучения курса в системах управления контентом является в настоящее время очень актуальной, обеспечивающей реализацию индивидуального обучения. Для ее решения предлагается использовать модель предметной области в виде семиотической сети [1]. В ее основе лежит орграф. Он позволяет использовать аппарат теории для анализа модели предметной области (МПО) [2].

Пусть G – орграф, описывающий МПО; $VG = \{v_i\}$ – множество вершин и $VE = \{e_i\}$ – множество дуг этого орграфа. Орграф G не должен содержать:

- петель, т.е. дуг (v_i, v_i) ;
- циклов, т.е. таких маршрутов $v_1, e_1, v_2, e_2, \dots, e_k, v_{k+1}$, в которых $v_1 = v_{k+1}$ (где v_i, e_i – соответственно номера вершин и дуг, входящих в маршрут);
- несвязных вершин или подграфов.

Для определения наличия несвязных вершин (подграфов) орграф рассматривается как неориентированный граф. Неориентированный граф является связным тогда и только тогда, когда для произвольной фиксированной вершины v существует маршрут (v, \dots, u) , где u – любая другая вершина графа.

Использование графов дает возможность решить оптимизационную задачу, связанную с определением целесообразной последовательности изучения тем как в рамках курса, так и нескольких взаимосвязанных курсов (дисциплин). МПО отражает логические связи между темами (понятиями), которые относятся, возможно, к разным дисциплинам. Соответственно последовательность освоения тем должна быть такова, чтобы к началу изучения темы i все предшествующие ей темы (понятия) были уже изучены.

Эта задача сводится к задаче раскраски вершин графа. Напомним, что раскраской графа G называется произвольная функция вида $f: VG \rightarrow \{1, 2, \dots, k\}$, где k – количество различных красок. В данном случае решением задачи определения последовательности изучения тем является такая раскраска орграфа G , при которой для любого маршрута $v_1, e_1, v_2, e_2, \dots, e_k, v_{k+1}$, вершины которого раскрашены цветами l_1, l_2, \dots, l_k , верно утверждение $l_i < l_j$, если $i < j$.

Чтобы решить задачу раскраски, необходимо знать кратчайший путь из одной вершины в другую. Рассмотрим решение данной задачи (рис. 1) с помощью надстройки Excel для пары вершин $v_{\text{нач}}, v_{\text{кон}}$. Множество дуг VE представим в виде матрицы смежности $S = [s_{ij}]$ порядка n , где n – число вершин графа, и

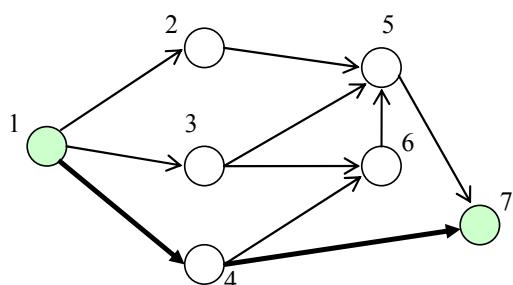


Рис. 1. Кратчайший путь в орграфе

$$s_{ij} = \begin{cases} 1, & \text{вершина } v_i \text{ соединена с } v_j, \\ 0, & \text{в противном случае.} \end{cases}$$

Введём целочисленные переменные x_{ij} , $i = 1, \dots, n, j = 1, \dots, n$, где

$$x_{ij} = \begin{cases} 1, & \text{кратчайший путь включает переход из вершины } v_i \text{ в } v_j, \\ 0, & \text{в противном случае.} \end{cases}$$

Целевая функция имеет следующий вид:

$$\sum_{i=1}^n \sum_{j=1}^n s_{ij} x_{ij} \rightarrow \min$$

В ней вычисляется количество переходов между вершинами в кратчайшем пути. Минимизация означает, что решением является путь, содержащий минимальное количество переходов.

Первая пара ограничений задаёт условия для начальной вершины пути $v_{\text{нач}}$. В исскомом пути в эту вершину не должно быть входа, но должен быть один выход:

$$\sum_{i=1}^n s_{i \text{ нач}} x_{i \text{ нач}} = 0, \quad \sum_{j=1}^n s_{j \text{ нач}} x_{j \text{ нач}} = 1.$$

Вторая пара ограничений задаёт условия для конечной вершины пути $v_{\text{кон}}$. В ней должен быть один вход, но не должно быть выхода:

$$\sum_{i=1}^n s_{i \text{ кон}} x_{i \text{ кон}} = 1, \quad \sum_{j=1}^n s_{j \text{ кон}} x_{j \text{ кон}} = 0.$$

Для всех остальных вершин (кроме $v_{\text{нач}}$ и $v_{\text{кон}}$) устанавливаются ограничения, за дающие равенство количества входов и выходов в каждую из них в исскомом кратчайшем пути:

$$\sum_{i=1}^n s_{ik} x_{ik} - \sum_{j=1}^n s_{kj} x_{kj} = 0, \quad k = \overline{1, n}, k \neq \text{нач}, k \neq \text{кон}.$$

Количество входов и выходов для каждой вершины не должно быть более одного:

$$\sum_{i=1}^n s_{ik} x_{ik} \leq 1, \quad k = \overline{1, n}; \quad \sum_{j=1}^n s_{kj} x_{kj} \leq 1, \quad k = \overline{1, n}.$$

В качестве области допустимых значений переменных используем значения 0 и 1: $0 \leq x_{ij} \leq 1, i = 1, \dots, n, j = 1, \dots, n$.

Подготовим данные для решения задачи на рабочем листе Excel в соответствии с рис. 2. Ячейки B3:H9 содержат матрицу смежности вершин графа, а ячейки B13:H19 – исковые переменные x_{ij} . В ячейки I13:J19 вписываются формулы для вычисления суммы произведений, соответствующих строк матрицы смежности и матрицы переменных: СУММПРОИЗВ (B3:H3;B13:H13) ... СУММПРОИЗВ (B9:H9;B19:H19). В ячейки B19:H19 вписываются формулы для вычисления суммы произведений соответствующих столбцов этих же матриц: СУММПРОИЗВ (B3:B9;B13:B19) ... СУММПРОИЗВ (H3:H9;H13:H19). В ячейку I20 (на рисунке выделена штриховкой) введём формулу для вычисления целевой функции: СУММПРОИЗВ(B3:H9;B13:H19).

A	B	C	D	E	F	G	H	I
1			Матрица смежности для вершин графа					
	1	2	3	4	5	6	7	
3	1		1	1	1			
4	2					1		
5	3				1	1		
6	4					1	1	
7	5						1	
8	6				1			
9	7							
10								
11			Матрица переходов - искомые переменные					
12	1	2	3	4	5	6	7	Суммы произв:
13	1							0
14	2							0
15	3							0
16	4							0
17	5							0
18	6							0
19	7							0
20	Суммы произв:	0	0	0	0	0	0	0
21								

Рис. 2. Исходные данные

Откроем окно «Поиск решения» (пункт меню «Сервис/Поиск решения») и установим следующие значения:

- Целевая ячейка – I20.
- Равной – минимальному значению.
- Изменяемые ячейки – B13:H19.
- Ограничения:

$$\begin{aligned} B20 &= 0; I13 = 1; H20 = 1; I19 = 0; I14:I18 = C20:G20; \\ I13:I19 &\leq 1; I13:I19 \geq 0; B20:H20 \leq 1; B20:H20 \geq 0; \\ B13:H19 &= \text{целое}; B13:H19 \geq 0; B13:H19 \leq 1. \end{aligned}$$

- Параметры – линейная модель.

После ввода всех параметров нажатием кнопки «Выполнить» находим решение. Возможны два результата:

- Решение найдено. Все ограничения и условия оптимальности выполнены. Значит, кратчайший путь существует и найден. В целевой ячейке мы увидим полученное значение его длины: 2. А нижняя часть таблицы будет заполнена значениями (рис. 3), описывающими найденный путь: $v_1 \rightarrow v_4 \rightarrow v_7$.
- Поиск не может найти подходящего значения. Значит, кратчайший путь не существует и не найден.

11			Матрица переходов - искомые переменные						
12	1	2	3	4	5	6	7	Суммы произв:	
13	1	0	0	0	1	0	0	0	1
14	2	0	0	0	0	0	0	0	0
15	3	0	0	0	0	0	0	0	0
16	4	0	0	0	0	0	0	1	1
17	5	0	0	0	0	0	0	0	0
18	6	0	0	0	0	0	0	0	0
19	7	0	0	0	0	0	0	0	0
20	Суммы произв:	0	0	0	1	0	0	1	2
21									

Рис. 3. Результат поиска

Таким образом, в работе предложен метод решения задачи поиска оптимальной последовательности при изучении курса с использованием нахождения кратчайшего пути в орграфе, сводя его к линейной оптимизационной задаче. Также предложен способ использования средств системы Excel для поиска оптимального решения. Рассмотренный метод имеет ограничения на количество вершин графа – не более 14. Оно связано с количеством изменяемых ячеек в Excel, которое не может превышать значение 200.

Список литературы

1. Поспелов Д.А. Семиотические модели в управлении. В кн.: Кибернетика. Дела практические. – М.: Наука, 1984. – С. 70-87.
2. Лекции по теории графов / Емеличев В.А., Мельников О.И., Сарванов В.И. и др. – М.: Наука, Гл. ред. физ.-мат. лит., 1990. – 384 с.
3. Майника Э. Алгоритмы оптимизации на сетях и графах – М: Мир, 1981.
4. Excel 2003 и VBA. Справочник программиста. – М.: Вильямс, 2006. – 1088 с.
5. Гилл Ф., Мюррей У. Практическая оптимизация. – М.: Мир, 1985. – 509 с.

ПРОЕКТ: «РАСЧЕТ КРЕДИТНЫХ СТАВОК ПО СЕЛЬСКОХОЗЯЙСТВЕННОЙ ТЕХНИКЕ И ЕЕ РАЦИОНАЛЬНОЕ ИСПОЛЬЗОВАНИЕ»

М.Р. Рязапов, Е.П. Коляда

Саратовский государственный аграрный университет им. Н.И. Вавилова

Данный проект выполнен в приложении Microsoft Office Excel 200, составлен для расчета процентных ставок по кредиту с/х техники и рациональному ее использованию.

Целью данного исследования является максимальное облегчение расчета кредита и его рациональное использование заемщиками. Из поставленной цели вытекают следующие задачи:

1. Помощь предпринимателю в выборе с/х техники и расчет стоимости ее доставки.
2. Облегчение выбора банка для займа кредита и оценка способности его выплаты.
3. Помощь предпринимателю с выбором посадочной культуры и расчет оптимальной площади посевов.
4. Решение проблемы с выплатой заработной платы сотрудникам с целью рационального использования фонда заработной платы.

Данный проект особо актуален в рамках недавно принятого национального проекта «Развитие АПК», так как поставленная в проекте цель поможет решить многие проблемы заемщиков, такие как: неспособность воспользоваться кредитом вследствие устоявшихся стереотипов о дорогоизнене его использования, обновление устаревшего парка техники и оборудования, экстенсивное ведение сельского хозяйства. Следует отметить, что национальный проект, принятый государством, помогает решить эти и другие проблемы с минимальными расходами самого предпринимателя, в частности, государство погашает проценты по кредиту, предоставляет льготы на ГСМ.

Актуальность выбранного проекта также объясняется тем, что современное сельское хозяйство находится в упадочном состоянии и требуется срочное его оздоровление.

ние. К примеру: парк техники изношен на 90-95%, земля не удобряется, не ведутся работы по сохранению используемых земель. Все это было вызвано бездействием государства, дефицитом денежных средств у сельхоз производителей.

Рациональное использование государственной поддержки денежными средствами позволит решить наболевшие проблемы, а данный проект поможет предпринимателям в этом.

Проект разбит на 6 частей (листов), каждый из которых выполняет свою определенную функцию. Рассмотрим каждый лист по отдельности.

Лист 1

Данном листе представлен прайс-лист, в котором указан весь ассортимент сельскохозяйственной техники, которую можно приобрести. Прайс-лист построен таким образом, что потенциальному заказчику не приходится просматривать весь лист целиком, а с помощью сводной таблицы (в форме которой выполнен прайс-лист), покупатель может открыть лишь ту часть документа, которая ему действительно интересна. Ассортимент распределен по тематическим разделам: тракторы, комбайны, запасные части, с/х оборудование, химикаты, удобрения и т.д.

Лист 2

На данный лист дает возможность добавления интересующего товара в вашу «корзину» и позволяет выяснить, сколько будет стоить доставка различными видами транспорта выбранного товара до любого районного центра Саратовской области, который можно выбрать из раскрывающегося списка. Так же в зависимости от выбранного районного центра вы будете иметь возможность видеть график, на котором будет отражаться стоимость доставки до выбранного районного центра по сравнению с другими. На данном листе использованы следующие возможности Excel:

- 1) работа с элементами управления: флаги, переключатели, счетчики, поле со списком;
- 2) функции для работы с массивами: ВПР, ИНДЕКС, ПОИСКПОЗ и др.

Лист 3

На данном листе потенциальный покупатель видит процентные ставки всех банков и условия предоставления кредита банками Саратовской области. Здесь также представлен график диапазона процентных ставок банков, чтобы покупатель мог наглядно оценить их разброс. На этом листе использованы: графики и элементы управления в виде раскрывающегося списка.

Лист 4

На данном листе предоставляется возможность расчета кредита. В зависимости от суммы покупки, выбранной процентной ставки банка, условий предоставления кредита и способности выплатить кредит, покупатель может рассчитать оптимальный срок выплаты кредита, дабы не стать банкротом. В данном разделе решается задача оптимизации с помощью поиска решения.

Лист 5

На этом листе дан сравнительный анализ рентабельности с/х культур, которая непосредственным образом влияет на уровень прибыльности хозяйства. В зависимости от выбранной для засева культуры и ее потенциальной урожайности, предоставляется возможность расчета оптимальной площади посадки культуры, что сводит к минимуму

дополнительные издержки. В текущем разделе использованы следующие средства Excel: подбор параметров, решение задачи оптимизации, построение диаграмм.

Лист 6

Данный лист полностью посвящен расчету оплаты труда и расходам на горючесмазочные материалы (ГСМ). Предприниматель имеет возможность рассчитать оптимально возможную продолжительность рабочего дня, время работы техники в зависимости от расхода топлива, а также оптимально возможный размер оплаты труда рабочим, задействованным в работе с/х предприятия, в зависимости от имеющихся возможностей хозяйства. На данном листе приложения Microsoft Excel используются следующие средства: строится линейная модель расчета заработанной платы, за основу которой берется заработка плата механизатора, которая находится путем подбора параметров.

Вывод. Данный проект позволяет решить наиболее значимые проблемы предпринимателей, занятых в сельском хозяйстве и стремящихся за счет покупки новой техники в кредит повысить прибыльность и рентабельность своего хозяйства.

АВТОМАТИЗАЦИЯ ПЕРЕВОДА ТЕКСТА НА РУССКОМ ЯЗЫКЕ В ДОРЕВОЛЮЦИОННУЮ ОРФОГРАФИЮ

Я.А. Седова, А.В. Морозов

*Астраханский государственный технический университет
Институт информационных технологий и коммуникаций*

Задача перевода текста на русском языке в дореволюционную орфографию приобрела актуальность в последнее время в связи с быстрым развитием сети Интернет и появлением в ней ряда сайтов, ориентированных на дореволюционную культуру. Как неоднократно отмечалось и многими русскими писателями, и представителями русской язычной общественности, последствия реформы русской орфографии, упразднившей ряд букв старого алфавита и отменившая ряд правил, являются неоднозначными. В результате этой реформы русский язык был существенно обеднен и частично потерял свою индивидуальность.

Целью предлагаемого программного продукта является автоматизация перевода текста в дореволюционную орфографию. Другими словами, задача заключается в восстановлении исходного строения текста, падежных форм слов, связей между словами и, по возможности, логики автора текста. Например, до революции окончания «-ие» и «-ые» в женском и среднем роде прилагательных заменялись на «-ия» и «-ыя»: «большія поля», «большія рѣки», но «большіе дома». Существительные первого склонения в дательном и предложном падежах и существительные второго склонения в предложном падеже единственного числа оканчивались на «ѣ»: «приду въ полѣ», но «видѣль поле». Прилагательные мужского и среднего родов имели в родительном падеже окончания «аго» и «яго»: «большаго дома», «синяго моря». Очевидно, по форме самого слова не всегда можно определить его грамматические признаки (именно поэтому все предыдущие

примеры давались словосочетаниями). Поэтому программа должна находить связи между словами.

Подобные связи в тексте не всегда удается установить, поскольку для синтаксического анализа текста на русском языке в настоящее время не разработано стопроцентно работоспособных алгоритмов. Тем не менее, в данном программном продукте реализован специально разработанный алгоритм, который для большого количества предложений находит верный вариант грамматического разбора. В тех случаях, в которых программа не может определить, стоит ли изменять написание определенного слова, она предпочитет вообще не применять правила дореволюционной орфографии к данному слову и сообщить об этом в итоговом отчете. Таким образом, процесс перевода текста в дореволюционную орфографию не является автоматическим, однако работа пользователя существенно упрощается. Помимо анализа связей между словами, программа автоматически прибавляет твердые знаки к словам мужского рода на согласную, заменяет букву «и» на «ї» и применяет ряд других правил, в частности, находит корни с буквой «ять» («ъ»). Для этой цели в программном продукте реализован алгоритм морфологического разбора слов. Кроме того, к программе подключен небольшой словарь, который был составлен на основе словаря Зализняка, но сокращен также при помощи разработанных алгоритмов для устранения избыточности данных.

В результате разработанный программный продукт реализует следующие алгоритмы:

- алгоритм морфологического разбора текста,
- алгоритм синтаксического разбора текста.

Составлен словарь русского языка, удобный для реализации данных алгоритмов.

Работа программы происходит следующим образом: она подключается к Microsoft Word, выбранному в качестве наиболее распространенного текстового редактора, и по требованию пользователя переводит ту часть текста, которую может разобрать гарантированно верно, в дореволюционную орфографию. Относительно тех случаев, в которых программа не пришла к окончательным выводам, она формирует отчет, руководствуясь которым пользователь может вручную завершить редактирование текста.

Программный продукт написан с использованием лицензионной среды разработки Microsoft Visual Studio 2005.

Писать в дореволюционной орфографии – дело привычки, разобрать же текст на естественном языке – задача компьютерной программы. Предлагаемый программный продукт позволяет автоматизировать достаточно трудоемкий процесс перевода текста на русском языке в дореволюционную орфографию [1–3].

Список литературы

1. Русское правописание. Руководство, составленное по поручению второго отделения Императорской академии наук академиком Я. К. Гротом. Одиннадцатое издание. СПб, 1894 г.
2. Люгер Джордж Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. – М.: Вильямс, 2003.
3. Шенк Р. Обработка концептуальной информации. – М.: Энергия, 1980.

АНИМАЦИЯ ВИРТУАЛЬНОЙ ГОЛОВЫ ЧЕЛОВЕКА С ИСПОЛЬЗОВАНИЕМ MANAGED DIRECTX

С.В. Сидоров, Ю.В. Горюнов, Е.С. Городецкий

Нижегородский госуниверситет им. Н.И. Лобачевского

Постановка задачи

Моделирование человеческого лица является важной задачей, которая находит применение в самых разных видах программного обеспечения: от компьютерных игр до телекоммуникаций. При этом исходными данными для такого рода алгоритмов являются последовательности положений тех или иных частей лица, полученные из видеопотока через web-камеру или из файла. Чтобы получить достаточно реалистичную модель головы необходимо, чтобы она имитировала движения, характерные для человека: повороты, наклоны, кивания, а также открытие/закрытие рта, перемещение зрачков и моргание. Таким образом, перед нами встает задача создания реалистичной модели головы и разработки алгоритмов анимации по исходным данным, описывающим все перечисленные движения.

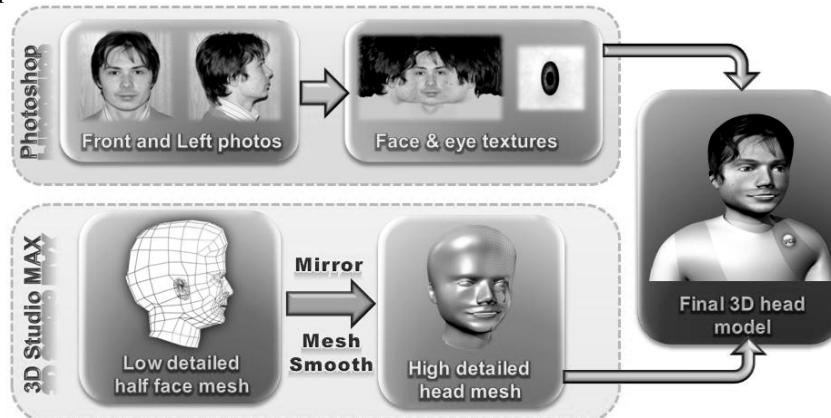
Создание модели головы конкретного человека

Поставленная задача создания модели головы конкретного человека требует обеспечения достаточного сходства с индивидуумом. Решение этой задачи выполнялось вручную с использованием современных систем компьютерной графики.

Вначале были сделаны две фотографии лица человека: фас и профиль. Затем задача была решена в два независимых этапа, результатом которых стала окончательная модель:

1. В редакторе Adobe PhotoShop по исходным фотографиям была создана текстура лица, предназначенная для наложения на объёмную модель головы, с помощью цилиндрических координат. Создание текстуры выполнялось путём склеивания фотографий профиля и фаса лица, сглаживания изображения в местах склейки и дополнительной цветовой коррекции. Кроме того, было создано изображение текстуры глаза.

2. С использованием системы трёхмерного моделирования 3D Studio MAX создавалась объёмная модель головы человека. В начале был сделан низкополигональный каркас одной половины головы. Затем с помощью модификаторов Mirror (отражение) и MeshSmooth (сглаживание) получена высокополигональная модель головы и наложены текстуры.



Полученная модель головы представлена на изображении снизу. Для дальнейшего использования модели в программе на DirectX трёхмерные объекты были конвертированы из формата 3D Studio MAX в формат X-файла с помощью утилиты Kilowatt X-Port.

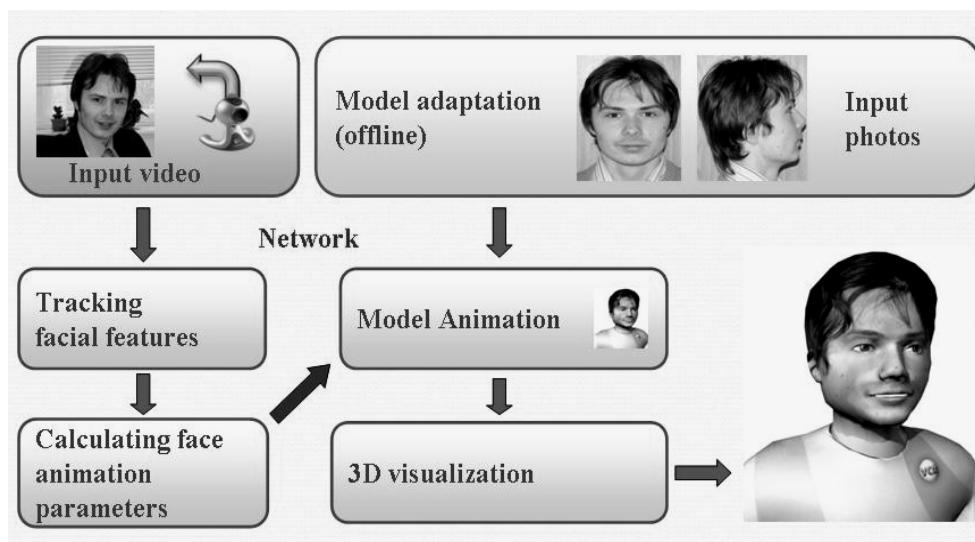


Входные данные анимации модели головы

Входными данными для осуществления анимации были выбраны углы горизонтального и вертикального поворота головы, наклона головы, открытия, горизонтального и вертикального поворота левого и правого глаз, открытия рта (губ).

Система расчета параметров головы по изображению с камеры

Используя полученное с web-камеры изображение, следящий алгоритм определяет положение прямоугольников, содержащих зрачки и губы человека. На основе этой информации вычисляются входные данные для анимации модели. Такого рода информация может быть использована для анимирования созданной вышеописанным способом модели непосредственно на данном компьютере или передана через компьютерную сеть для удаленной анимации. В случае дополнительной передачи по сети аудиопотока от человека, снимаемого камерой, возможно осуществить распределенную конференцию с низкой загруженностью сетевого канала. Применение такой технологии обеспечивает снижение передаваемого траффика по сети около 500 раз по отношению к передаче «живого» видео.



Анимация модели человеческой головы

Основная идея анимации головы состоит в разделении модели на отдельные объекты: глаза, лицо, веки. Каждый объект представлен 3D-моделью, загружаемой из X-файла. Задачу анимации человеческой головы можно разделить на две части: выполнение вращательных движений и деформация модели при открытии и закрытии рта.

Повороты, наклоны, кивания головы реализовывались путем трансформации мировой матрицы по входным данным.

Достижение эффекта открытия/закрытия рта обеспечивалось за счёт линейной интерполяции между заранее подготовленными моделями с закрытым, наполовину открытым и полностью открытым ртом. Первоначально загружались все три модели лица из соответствующих X-файлов в объекты типа Mesh. С использованием механизма блокировки (Locking) создавались массивы, содержащие информацию о вершинах модели (координаты вершины, нормали, текстурные координаты). Также создавался массив, содержащий информацию о связях вершин в модели. При каждом изменении открытия рта создавалась новая модель: координаты положения вершин высчитывались с учетом угла открытия рта. При этом текстурные координаты оставались без изменения, а нормали пересчитывались.

Движение глаз обеспечивалось путем трансформации мировой матрицы. Для привязки глаз к голове матрица трансформации глаза умножалась на матрицу сдвига глаз относительно модели головы и на матрицу трансформации лица.

Для реализации эффекта моргания были созданы модели век. Они представляли собой полусферы радиуса чуть большего, чем глаза. В момент моргания полусфера «наезжает» на глаз, скрывая его. Это делается путем умножения на матрицу трансформации. Этого достаточно, т.к. моргание быстрый процесс и такой способ не дает впечатления чего-то искусственного.

Начальная привязка положения глаз, век к модели головы сохраняется в отдельном config-файле. В качестве этого файла выступает XML-файл, чтение из которого осуществляется встроенными методами .NET Framework.



Программа анимации модели головы, повторяющей движения пользователя

В качестве иллюстрации вышеописанных подходов к анимации была реализована тестовая программа, позволяющая обрабатывать входящее с web-камеры видеоизображение, определять значения анимационных параметров и отображать модель человеческой головы.

Заключение

Результаты данной работы были использованы в проекте, выполняемом студентами ННГУ.

Список литературы

1. Kang S.B., Jones M. Appearance-based structure from motion using linear classes of 3-d models // *Manuscript*. 1999.
2. Parke F. I. Computer generated animation of faces // *ACM National Conference*. November, 1972.
3. Parke F. I., Waters K. *Computer Facial Animation*. AKPeters, Wellesley, Massachusetts, 1996.
4. Jilin Tu, Thomas Huang, Hai Tao. Face as Mouse Through Visual Face Tracking // Proceedings of CRV. 2005. P. 339–346.

БИБЛИОТЕКА CONFERENCEXP КАК .NET-КАРКАС ДЛЯ ПРИЛОЖЕНИЙ, ОБЕСПЕЧИВАЮЩИХ ПРОВЕДЕНИЕ АУДИО- И ВИДЕОКОНФЕРЕНЦИЙ

**(общий обзор возможностей библиотеки CXR и описание возможностей
её адаптации и встраивания в конкретные приложения
для передачи видео или аудио)**

С.В. Сидоров, А.Н. Половинкин, А.А. Латышев

Нижегородский госуниверситет им. Н.И. Лобачевского

В настоящей работе рассматривается библиотека Microsoft ConferenceXP, обеспечивающая проведение аудио- и видеоконференций как через локальную сеть, так и через Интернет.

Описание библиотеки

Библиотека ConferenceXP разрабатывается в Microsoft Research. Она предназначена для организации как аудио-, так и видеоконференций, когда их участники располагаются удаленно друг от друга и связаны только через Интернет или локальную сеть. Применение такого рода конференций получило широкое распространение при организации дистанционного образования, в медицине и т.д. Таким образом, используя возможности этой библиотеки, легко организовать удаленное общение студентов и преподавателей или, например, совместную игру музыкантов через Интернет.

К особенностям библиотеки относятся:

- поддержка высокоскоростных сетевых каналов (до 10 гигабит);
- поддержка дисплеев высокого разрешения;
- поддержка Tablet PC;
- поддержка Microsoft® Windows® XP и Windows® Vista.

ConferenceXP – это платформа, позволяющая исследователям и разработчикам программного обеспечения создавать распределенные приложения, которые используют ее преимущества, такие, как интерфейс взаимодействия с Tablet PCs и беспроводными сетями. Она также позволяет создавать собственные библиотеки на своей основе. Проект ConferenceXP объединяет мировое академическое сообщество в педагогических науках с технологиями Microsoft's в области коммуникаций. В 2005 году на конкурсе Imagine Cup победил проект omniMusic, который использовал данную платформу.

Использование библиотеки

Библиотека ConferenceXP представляет собой набор классов на основе платформы .NET.

Функционально библиотеку можно разделить на 2 части:

- клиентская (реализованная на основе .NET Framework 2.0);
- серверная (реализованная на основе ASP.NET 2.0).

Клиентская часть обеспечивает механизм создания новой распределенной конференции, подключение к уже существующей, захват, передачу и прием аудио- и видеопотоков.

Необходимо отметить, что библиотека также позволяет организовать распределенный показ презентации всем участникам конференции, а также отрисованных на слайдах пометок.

Серверная часть представляет собой web-сервис Venue Service, который обеспечивает прием и передачу аудио- и видеопотоков всем участникам конференции, хранит информацию о подключившихся пользователях.

В составе пакета ConferenceXP находится реализованное клиентское приложение, которое использует функциональность библиотеки для организации конференций.

Библиотека поставляется в виде готовых приложений (клиента и web-сервиса), а также в виде исходных кодов, которые можно скачать с сайта [1].

Для разработки собственных приложений на основе данной библиотеки необходимо скачать ее с официального сайта [1] и подключить ее dll к собственному проекту.

Применение библиотеки для организации аудиоконференции

Возможности данной библиотеки были использованы для передачи аудиопотоков в проекте, выполняемом студентами ННГУ [2]. С результатами можно ознакомиться на сайте проекта.

Список литературы

1. www.conferencexp.net
2. <http://www.itlab.unn.ru/?dir=366>.

ОБЩАЯ СХЕМА РЕАЛИЗАЦИИ ОСТАНОВКИ И ВОЗОБНОВЛЕНИЯ СЧЕТА ИНДЕКСНОГО МЕТОДА ПОИСКА ГЛОБАЛЬНО-ОПТИМАЛЬНЫХ РЕШЕНИЙ

С.В. Сидоров, А.В. Сысоев

Нижегородский госуниверситет им. Н.И. Лобачевского

В настоящей работе рассматривается подход к реализации схемы остановки и продолжения счета в последовательной и параллельной версии индексного метода [1, 2], реализованного в программной системе «Абсолют Эксперт» [3, 5].

Необходимость остановки счета

В настоящей работе рассматривается расширение функциональности параллельного индексного метода с одной разверткой, который является частью набора методов системы поиска глобально-оптимальных решений «Абсолют Эксперт». Указанный алгоритм представляет собой итерационную схему, в которой выбор следующей точки итерации происходит на основе информации, накопленной на всех предыдущих итерациях. Таким образом, под остановкой счета понимается остановка итерационного алгоритма с сохранением всей накопленной информации о решении задачи, а также с сохранением исходных данных о решаемой задаче. Все данные после остановки счета располагаются во внешней памяти и могут быть перенесены с одного вычислительного узла на другой с возможностью продолжения счета на нем. При возобновлении счета происходит загрузка всех данных о решаемой задаче с возможным некоторым изменением критерии задачи и продолжение выполнения ранее остановленных итераций.

Необходимость остановки счета обусловлена рядом факторов.

Наличие ее влияет на эффективность алгоритма поиска, обеспечивая возможность при решении реальных прикладных задач прекратить в определенный момент счет с получением текущей оценки глобального минимума. В дальнейшем можно использовать полученную оценку либо сделать послабление в границе области поиска и продолжить нахождение новой оценки.

Возможность остановки и возобновления счета серьезно расширяет функциональность алгоритма, позволяя начать вычисления на одном вычислительном узле, потом при необходимости прекратить их с сохранением всей накопленной информации и продолжить вычисления на этом или другом узле через любой промежуток времени. Таким образом, в случае окончания квоты на использование вычислительных ресурсов можно прекратить вычисления и продолжить их при появлении таких ресурсов или на других ресурсах.

Описание схем остановки и возобновления

Первоначально рассмотрим стандартный процесс прекращения вычислений алгоритма при выполнении одного из критериев остановки: точность или число выполненных итераций.

В случае прекращения вычислений происходит выход алгоритма из цикла итераций и вывод результатов о текущей найденной оценке глобального минимума.

В случае остановки алгоритма происходит дополнительное сохранение во внешней памяти всех параметров решаемой задачи, а также всей накопленной поисковой информации.

Необходимо отметить, что общая схема сохранения параметров для последовательного и параллельного алгоритмов существенно отличается, так как в зависимости от реализации параллельного алгоритма возможна ситуация распределения всего набора поисковой информации между вычислительными узлами. В этом случае необходимо выполнить первоначальную аккумуляцию поисковой информации на одном узле, а потом выполнить на нем сохранение. Примером метода с распределенной поисковой информацией является параллельный индексный с множественной разверткой. Реализация остановки и возобновления такого метода наиболее сложна.

В данной работе рассматривается как последовательная, так и параллельная реализация индексного метода с одной и множественной разверткой. При этом наиболее простой является реализация сохранения последовательной версии индексного метода с одной разверткой. Параллельная версия индексного метода с одной разверткой предпо-

лагает дублирование поисковой информации на вычислительных узлах, поэтому после остановки вычислений необходимо только сохранить всю накопленную информацию и параметры решаемой задачи одним из вычислительных узлов. Параллельная версия индексного метода с множественной разверткой требует первоначального сбора поисковой информации и параметров поискового метода на одном узле.

Сохраняется поисковая информация, включая состояние матрицы поиска и текущую оценку константы Липшица для всех используемых разверток. Также сохраняются следующие параметры решаемой задачи и метода:

- идентификатор объекта оптимизации;
- число задач оптимизации, использующих данный объект;
- параметры всех задач оптимизации, такие, как область поиска, список дискретных параметров;
- число ограничений;
- свойства метода: точность, максимальное число итераций, точность развертки, параметр ϵ -резервирования и т.д.

Необходимо отметить, что возобновление после остановки счета параллельного метода серьезно отличается от возобновления последовательного метода, так как в последовательном методе необходимо только произвести восстановление поисковой информации и информации о решаемой задаче, а дальше запустить цикл итераций.

Особенностью возобновления счета параллельного метода является то, что вся сохраненная информация сосредоточена только на одном узле, поэтому необходимо после восстановления условия задачи, а также поисковой информации осуществить пересылку всей этой информации всем остальным узлам и произвести восстановление параметров для продолжения счета.

Реализация

Описанная выше схема остановки и возобновления счета на данный момент реализована как в последовательном, так и параллельном индексном методе с одной и множественной разверткой в рамках системы «Абсолют Эксперт». Сохранение и восстановление отдельных компонентов выполнено в качестве методов соответствующих классов.

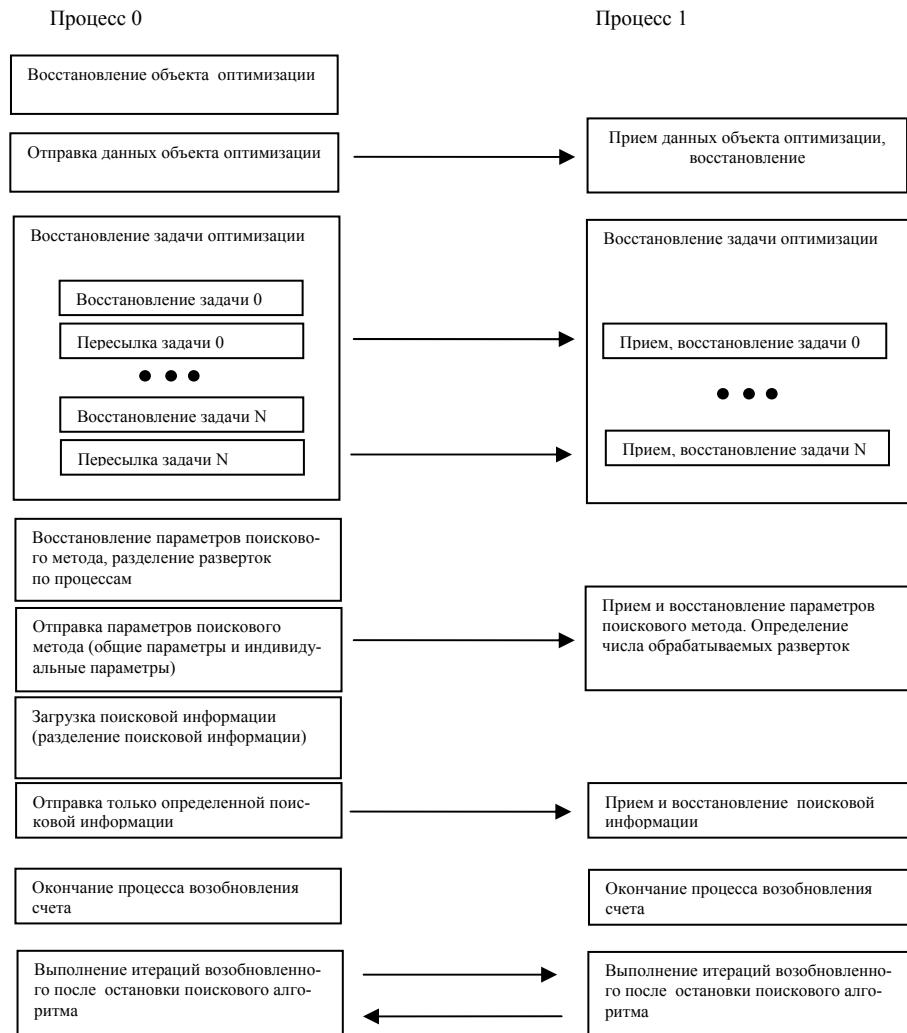
Реализация сохранения всех параметров решаемой задачи для последовательных версий индексного метода является относительно простым процессом в силу специфики поискового метода, обусловленной содержанием всей поисковой информации на одном узле. Таким образом, процесс сохранения представляет последовательный вызов методов по сохранению компонентов поискового процесса.

Более высокий интерес представляет реализация возобновления вычисления параллельного поискового алгоритма. Ниже приведена схема реализации восстановления параллельного индексного метода с множественной разверткой.

В данной схеме предполагается, что вся сохраненная информация содержится на процессе с номером 0 и возобновляются вычисления для двух процессов.

Результаты экспериментов

Для проверки корректности выполнения процесса остановки и возобновления счета последовательного и параллельного индексного метода с одной и множественной разверткой была проведена серия экспериментов, заключающаяся в сравнении результатов работы алгоритма, остановившегося и возобновившего счет после остановки, и алгоритмом, который выполнил такое же количество итераций, но не останавливался. Таким образом, считалось, что в ситуации совпадения результатов приостановление и возобновление прошли корректно.



Такого рода эксперименты были проведены на функциях Гришагина 97 и 98. Эксперименты подтвердили корректность работы представленных алгоритмов.

Настоящая работа поддержана грантами Netherlands Organization for Scientific Research (NWO) № 047.016.014 и Российского фонда фундаментальных исследований (РФФИ) № 04-01-89002-HBO_a.

Список литературы

1. Стронгин Р.Г. Поиск глобального оптимума // Математика и кибернетика. – Знание. 2. 1990.
2. Strongin R.G., Sergeev Ya.D. Global optimization with non-convex constraints: Sequential and parallel algorithms. – Kluwer Academic Publisher, Dordrecht, 2000.
3. Sysoyev A.V. Program system of parallel computations for solving the tasks of global-optimum choice // VI International Congress on Mathematical Modeling. Book of abstracts. 2004. P. 62.
4. Сысоев А.В., Сидоров С.В. Использование чисел расширенной точности в реализации индексного метода поиска глобально-оптимальных решений // Материалы Пятого международного научно-практического семинара «Высокопроизводительные параллельные вычисления на кластерных системах». 2005. С. 208-215.
5. Гергель В.П., Сысоев А.В. О реализации параллельной версии индексного метода поиска глобально-оптимальных решений в многомерных задачах оптимизации в программном комплексе «Абсолют Эксперт» // Труды всероссийской конференции «Научный сервис в сети Интернет: технологии параллельного программирования». 2006. С. 115-118.

АВТОМАТИЧЕСКИЙ АНАЛИЗ ИЗОБРАЖЕНИЯ

Е.В. Смирнов, Д.В. Егоров, П.В. Иванов

Чувашский государственный университет им. И.Н. Ульянова

1. Введение и постановка задачи

Задачи анализа изображений – одни из основных в различных областях научных и прикладных исследований. Трудности решения этих задач связаны с большим количеством информации, содержащимся в изображении и, соответственно, большой трудоемкостью ее извлечения, часто встречающейся необходимостью анализа изображений в реальном времени (в системах контроля и управления).

В данной работе представлен разработанный нами подход и компьютерная программа, которые позволяют автоматизировать ряд задач анализа изображения.

2. Методика решения

Рассмотрим возможности разработанной компьютерной программы на примере автоматического анализа модельного монохромного изображения ансамбля частиц: фон – белый, частицы – черные (рис. 1).

После запуска программы, начиная с левого верхнего угла изображения сверху вниз исследуется каждый пиксель изображения (т. е. определяется его цвет), пока не «встретится» пиксель черного цвета. Таким образом обнаруживается «первая» частица. Затем программа окрашивает в зеленый цвет все граничащие друг с другом пиксели черного цвета, то есть выделяет всю частицу (рис. 2), и определяет координаты всех пикселей выделенной частицы.

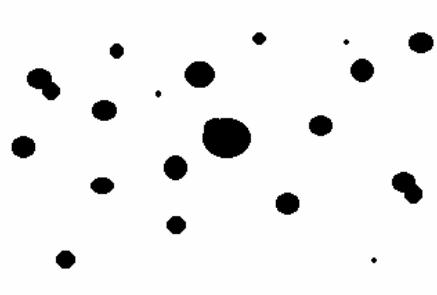


Рис. 1. Модельное монохромное изображение:
черные частицы на белом фоне

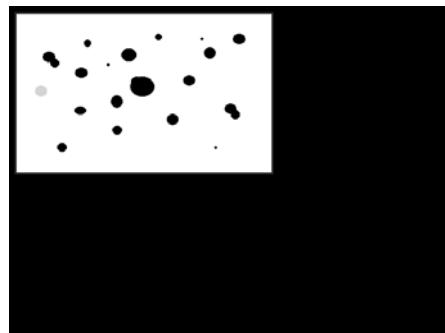


Рис. 2. Один из моментов работы программы:
первая обнаруженная частица окрашена
и проанализирована

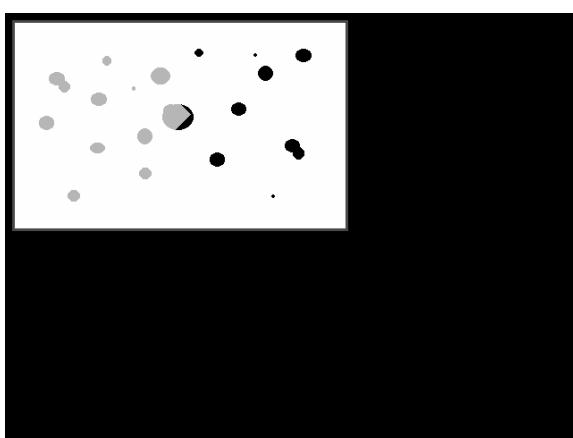


Рис. 3. Один из моментов работы программы:
обнаружено и проанализировано по отдельности
уже несколько частиц

После этого, через полученные значения координат, определяются: габариты (т. е. максимальная высота и ширина частицы), площадь, периметр, отношение площади к периметру (табл. 1). Далее программа приступает к поиску следующей частицы (рис. 3), причем уже проанализированная частица автоматически исключается из «работы», так как она окрашена в другой (зеленый) цвет.

Данный принцип является основным в нашем подходе, который можно сформулировать так: «отметил – проанализировал – исключил». Именно он позволяет автоматизировать процесс анализа изображения.

Таким образом, сканируется все изображение, последовательно проводится анализ каждой обнаруживаемой частицы, и для каждой из них получаются соответствующие вышеупомянутые геометрические характеристики, а также подсчитывается общее количество частиц. Далее программа проводит статистический анализ: получаются гистограммы распределения частиц по ширине, высоте, площади и периметру (рис. 4, 5).

Помимо этого также определяются максимальные значения ширины, высоты, площади, периметра, суммарная площадь и суммарный периметр частиц (табл. 2).

Таблица 1

Данные анализа модельного монохромного изображения

№ частицы	Ширина частицы, пиксель	Высота частицы, пиксель	Площадь частицы, пиксель	Периметр частицы, пиксель	Отношение площади частицы к периметру
1	17	16	216	59	3.66
2	24	23	332	80	4.15
3	14	13	142	42	3.38
4	17	12	160	48	3.33
5	18	15	214	59	3.62
...
11	35	29	824	117	7.04
...
17	4	4	12	10	1.20
18	22	23	317	75	4.22
19	18	15	214	59	3.62

Таблица 2

Дополнительные данные анализа

Общее количество частиц	19
Суммарная площадь, пиксель	3953
Суммарный периметр, пиксель	966
Максимальная ширина, пиксель	35
Максимальная высота, пиксель	29
Максимальная площадь, пиксель	824
Максимальный периметр, пиксель	117

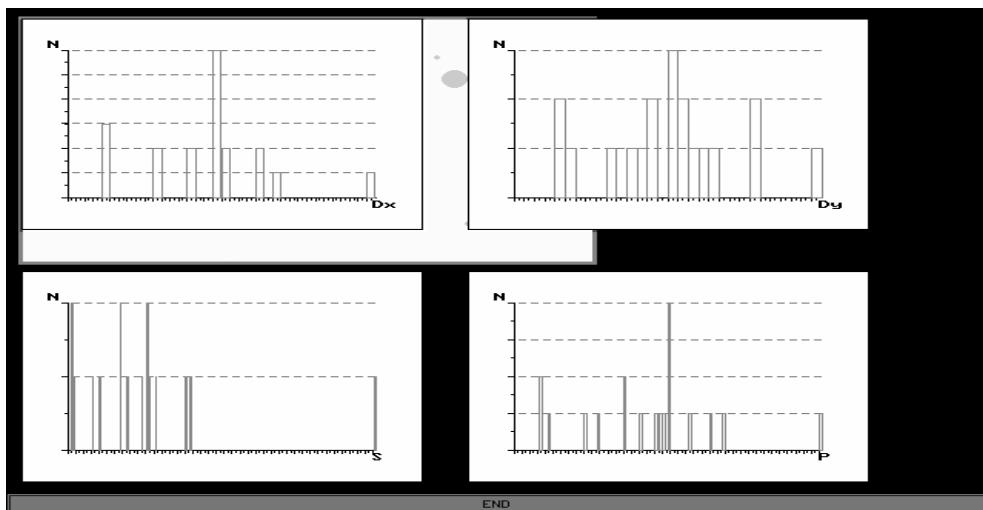


Рис. 4. Гистограммы распределения частиц по ширине, высоте, площади и периметру

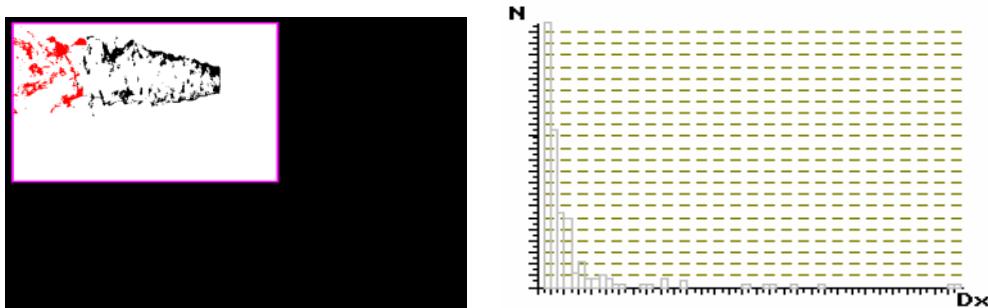


Рис. 5. Экраны двух моментов работы программы при расшифровке изображения распыления жидкости: процесс анализа изображения и гистограмма распределения частей распыленной жидкости по ширине

3. Заключение и выводы

На рисунках приведены экраны первоначального варианта компьютерной программы (написанной под DOS). В настоящее время имеется программа, написанная под Windows (среда разработки Visual C++). Она обладает дружественным интерфейсом и может анализировать любые изображения, переведенные в 24-разрядные изображения bmp-формата.

Данный подход к автоматизации анализа изображения и разработанная компьютерная программа могут использоваться при создании автоматических систем диагностики, тестирования, контроля (а в перспективе – и управления) в различных областях науки и техники, в частности при решении таких задач, как:

1. Обработка изображения и его фильтрация, например с целью устранения шума, увеличения качества изображения.
2. Регистрация и распознавание образов, например поиск частей изображения с заранее заданными геометрическими параметрами и их статистический анализ (регистрация и распознавание движущихся целей).
3. Задачи криминалистики (например, распознавание отпечатков пальцев).
4. Контроль формы деталей, сравнение их геометрических параметров со «стандартными», обнаружение дефектов (машиностроение, приборостроение, системы контроля качества продукции и ее сортировки).
5. Задачи аэрофотосъемки и картографии, например определение изменения береговой линии, мониторинг сезонных изменений поверхности земли и лесных пожаров.
6. Оцифровка графиков, анализ и распознавание временных диаграмм на бумажном носителе.
7. Анализ медицинских и биологических изображений (изображения хромосом, рентгенограммы, томограммы, Кирлиан-изображения и т. д.).
8. Анализ интенсивности и характерных масштабов турбулентных потоков.
9. Определение распределения концентрации частиц в факеле распыления жидкости.
10. Нахождение «центра масс» частиц или системы частиц.
11. Статистический анализ и контроль характеристик поверхности материала (металлургия, материаловедение).
12. Определение формы пламени, статистический анализ горящей поверхности пороха.
13. Определение температурного поля пламени по интерферограммам и другим изображениям (при совместном применении технологий искусственных нейронных сетей).

**ВЫСОКОПРОИЗВОДИТЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА
ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ МЕТОДОМ ОБРАТНОГО
РАСПРОСТРАНЕНИЯ ОШИБКИ С ИСПОЛЬЗОВАНИЕМ СРЕДСТВ
РАЗРАБОТКИ INTEL И MS VISUAL STUDIO**

А.Ю. Соколов, А.В. Гребенщиков

Вятский госуниверситет

Постановка задачи: разработать демонстрационное приложение, с помощью которого можно моделировать работу многослойного персептрона.

Алгоритм обратного распространения ошибки определяет стратегию подбора весов синапсов многослойной сети с применением градиентных методов оптимизации. Минимизируемой целевой функцией является сумма по всем выходам сети (M) и примерам из обучающей выборки (p) квадратов разностей между фактическими и ожидаемыми значениями выходных сигналов:

$$E(w) = 0,5 \sum_{j=1}^p \sum_{k=1}^M (y_k^{(j)} - d_k^{(j)})^2. \quad (1)$$

Обучение нейронной сети (НС) с использованием алгоритма обратного распространения ошибки проводится в несколько этапов:

1. Подать на входы сети один из возможных образов и в режиме обычного функционирования НС, когда сигналы распространяются от входов к выходам, рассчитать значения последних. Формула для расчёта выхода одного нейрона выглядит следующим образом:

$$u_i = f \left(\sum_{j=0}^{N-1} w_{ij} x_j \right). \quad (2)$$

2. Рассчитать значение ошибки и изменение весов для выходного слоя.
3. Рассчитать значение ошибки и изменение весов для слоев от предшествующего выходному до первого.

4. Скорректировать все веса в НС.

5. Если ошибка сети существенна, перейти на шаг 1. В противном случае – конец.

Сети на шаге 1 попеременно в случайном порядке предъявляются все тренировочные образы, чтобы сеть, образно говоря, не забывала одни по мере запоминания других.

Рассмотрим обобщённую структуру нейронной сети.

Информационный граф распространения сигнала в сети аналогичен структуре, представленной на рис. 1. Из анализа данной структуры можно сделать вывод, что вычисление обладает **горизонтальным параллелизмом**. Следовательно, для повышения производительности может быть использована библиотека Intel® Integrated Performance Primitives (Intel® IPP). Данная библиотека включает в себя набор высокопроизводительных программных функций, использующих SIMD-расширения архитектур процессоров Intel и общий для всех этих архитектур интерфейс прикладного программирования, благодаря чему реализованные в ней функции будут работать на платформах на базе различных процессоров Intel без какого-либо изменения исходного кода. Как дополнительный плюс ко всему вышеперечисленному, данная библиотека реализована под следующие операционные системы: Windows, Linux и MacOS.

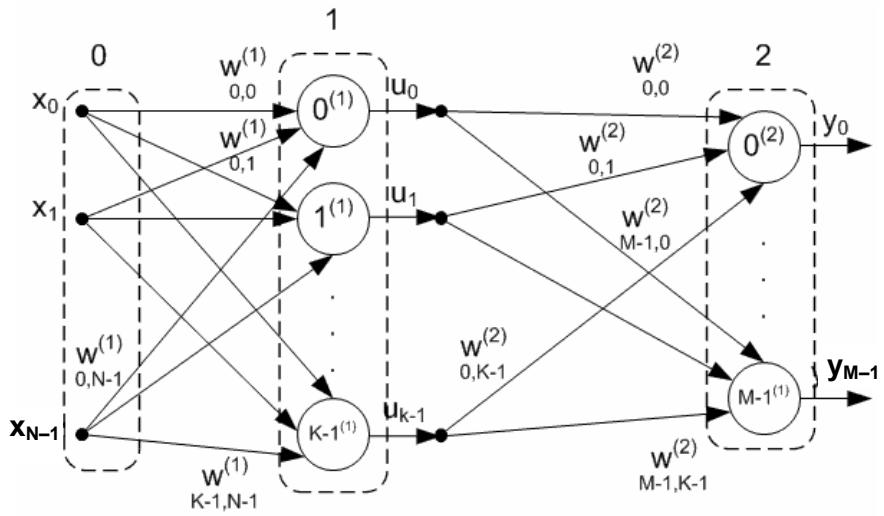


Рис. 1. Обобщённая структура двухслойной сигмоидальной нейронной сети
(с одним скрытым слоем)

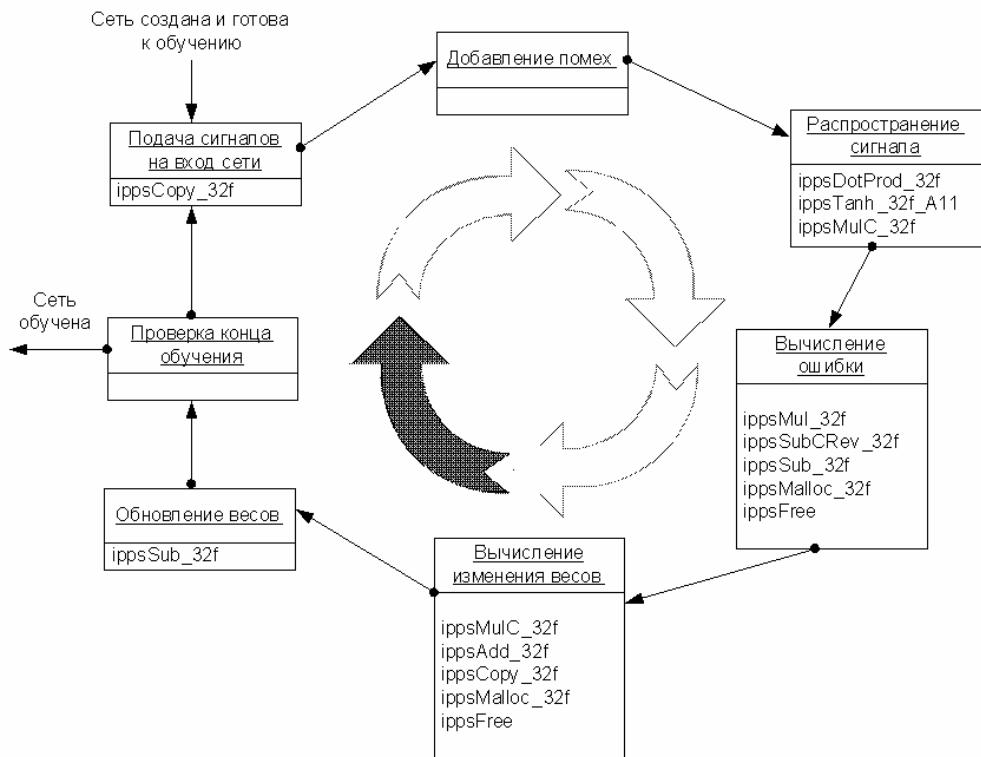


Рис. 2. Цикл обучения и используемые функции

Рассмотрим реализацию вычисления выходов одного слоя с использованием Intel® IPP и без нее.

Без использования Intel® IPP	С использованием Intel® IPP
<pre>for (int i=1; i<n; i++) { for (int j=0; j<m; j++) { axon[i] = axon[i] + w[i][j] * x[j]; axon[i] = 0.5 * tanh(axon); } }</pre>	<pre>for (int i=1; i<n; i++) ippsDotProd_32f(w[i], x, length1, axon); ippsTanh_32f_A11(axon, axon, length2); ippsMulC_32f(axon, 0.5, length2);</pre>

n – количество нейронов в слое; **m** – количество входов (синапсов) нейрона;
axon – выходной вектор (аксоны) слоя нейронов; **w** – матрица весов;
x – входной вектор (аксоны предыдущего слоя) для слоя; **tanh** – гиперболический тангенс;
length1 – количество выходов предыдущего слоя; **length2** – количество выходов текущего слоя.

Таким образом, почти все операции над векторами, которые осуществлялись в цикле, будут реализованы с помощью вызова соответствующей функции из библиотеки Intel® IPP. В случае использования IPP производительность возрастает примерно в 3 раза.

Рассмотрим обобщённый алгоритм цикла обучения сети и наборы функций, которые использовались на каждом шаге алгоритма (рисунок 2).

Разработанное приложение является многопоточным. Во время обучения нейронной сети процесс разделяется на два потока. Один поток выполняет вычисления, а другой отвечает за взаимодействие с пользователем. Для того чтобы пользователь знал, что выполняется обучение, предусмотрена специальная полоса процесса (progress bar). Если изменять состояние полосы процесса после каждого цикла обучения, то инструмент профилирования многопоточных приложений Intel® Thread Profiler покажет, что значительное время тратится на взаимодействие потоков. Для сокращения этих непроизводительных накладных расходов при неизменной функциональности программы можно изменять состояние полосы процесса в 20 раз реже. При этом время обучения сети сокращается на 15%.

Список литературы

- Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2004. – 344 с.: ил.
- Intel® Integrated Performance Primitives for Intel® Architecture. Reference Manual.

СИСТЕМА КОНТРОЛЯ ВЕРСИЙ ЭЛЕКТРОННЫХ КАРТ

Е.С. Сорокин, С.В. Жерзев

Нижегородский госуниверситет им. Н.И. Лобачевского

Обеспечение точности и актуальности данных электронных карт (ЭК) является важной проблемой в геоинформационных системах (ГИС), с помощью которых оперативно принимаются решения на основе этих данных. Вследствие различных факторов (природных, техногенных и пр.) со временем пропадает согласованность содержимого карты и фактического состояния отображаемых объектов. Для избежания принятия на

основе устаревших данных ЭК ошибочных решений необходима модификация документа, т. е. его изменение в соответствии с современным состоянием местности [2].

Смежной актуальной задачей является организация репликации изменений, то есть оперативной доставки обновленных данных от поставщиков ЭК до потребителей. Это задача синхронизации информационного содержания ЭК на стороне клиента с современным фактическим состоянием объектов и явлений на местности решается, например, внедрением системы контроля версий, объектами хранения которой являются ЭК.

Поскольку реальные файлы ЭК имеют размер порядка десятков мегабайт даже при использовании эффективных методов сжатия данных, одной из задач, возникающих при работе с такими картами в глобальных вычислительных сетях (ГВС), является уменьшение объема передаваемой информации вследствие низкой скорости и/или высокой стоимости передачи данных. В связи с этим особое значение приобретает разработка эффективных методов, алгоритмов и технологий передачи графической информации [3].

Жесткие требования к размеру сетевого трафика делают невозможным применение существующих программных продуктов (таких, как Rational ClearCase, CVS, Microsoft SourceSafe и пр. [4–6, 8]). Эти системы ориентированы в первую очередь на текстовые (последовательно организованные) данные и не учитывают специфику представления изображений и применяемые форматы хранения. К причинам разработки собственной системы управления версиями помимо редукции объема передаваемой информации нужно отнести смежные задачи: необходимость взаимодействия с существующим архивом морских карт, требование обеспечить защиту против недобросовестных пользователей, реализация дополнительных опций.

В докладе рассматривается разработанный программный комплекс, ориентированный на большеформатные изображения в формате BIG [1]. Особенностью этого формата, на которых основывается предлагаемая технология, является разбиение сверхбольшого изображения на множество независимых фрагментов и удобство доступа к каждому из них. Учитывая типичное свойство версионных объектов – незначительное различие модификаций одного файла (редакторская правка чаще всего затрагивает со средоточенные локально области карты, а, значит, большая часть документа остается неизменной), предлагается выделять множество измененных фрагментов ЭК.

Сформированный пакет картографических обновлений содержит только те фрагменты, на которых непосредственно расположены изменяемые объекты, и эти фрагменты независимы, а значит, отпадает необходимость передачи через ГВС всей обновленной карты, достаточно пересылать только эти изменения.

Размер этого пакета выявленной разности между парой версий карт гораздо меньше (что, как уже упоминалось, критически важно при передаче через сеть), нежели исходного документа. Поэтому проблема синхронизации большеформатных ЭК при удаленном доступе к ГИС посредством ГВС сводится к выделению изменений, организации оперативной доставки этих пакетов картографических данных до потребителей, а также созданию алгоритма, восстанавливающего по клиентской версии ЭК и пакету обновлений актуальный документ.

Решение было реализовано в виде программного комплекса, композиционно разделенного на следующие подсистемы:

- Корневая библиотека, формирующая файл с изменениями, выделенными между двумя версиями одной карты, и актуализующая ЭК по базовой версии и скомпонованному пакету разности.
- Подсистема-сервер, в задачи которого входит реагирование на запросы программ-клиентов, проведение необходимой верификации, компоновка и рассылка обновлений.

- Библиотека суммирования обновлений, реализующая функцию слияния изменений, что бывает необходимо при пересылке данных пользователю.
- Подсистема-клиент – программа с дружественным интерфейсом, позволяющая запрашивать, получать обновления и применять их к существующей версии карты.

Экспериментальная апробация подтвердила эффективность предложенного метода; тестирование, проведенное на ряде специально подготовленных ЭК, с изменениями, затрагивающими порядка 4% общей площади документа, показало существенный выигрыш объема: размер пакета обновлений составлял около 2–4% от исходных данных.

Таким образом, на практике подтвердилась эффективность рассмотренного выше способа синхронизации версий сверхбольшого изображения в формате BIG.

Программное обеспечение выполнено на платформе Microsoft .NET [7, 8] с использованием языка C#.

Список литературы

1. Кудин А.В. Технология эффективного хранения и оперативного отображения картографической растровой информации: Дис. ... канд. технических наук. Нижний Новгород, 2000.
2. Крючков А.Н., Боричев С.П. Программно-информационный комплекс обновления цифровых карт по аэрокосмическим снимкам / Минск, Объединенный институт проблем информатики НАН Беларуси // Методы и средства обработки сложной графической информации: VIII Всероссийская научная конференция: Тезисы докладов. Нижний Новгород, 2005. 105 с.
3. Васин Ю.Г., Жерзев С.В. Технология хранения и передачи данных ГИС / НИИ ПМК ННГУ, Нижний Новгород // Методы и средства обработки сложной графической информации: VII Всероссийская с участием стран СНГ конф.: Тез. докл. Нижний Новгород, 2003. – 104 с.
4. ClearCase – система конфигурационного и версионного контроля, Александр Новичков, www.citforum.ru.
5. CVS – система управления версиями. Алексей Махоткин. www.citforum.ru
6. Developing Software with ClearCase, Document Number 800-023814-000 November 2000 Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421
7. Рихтер Дж. Программирование на платформе Microsoft .NET Framework. Мастер-класс: Пер. с англ. – 3-е изд. – М.: Издательско-торговый дом «Русская Редакция»; СПб.: Питер, 2005. – 512 стр.: ил.
8. Материалы сети Internet:
<http://www.perforce.com/>
<http://www.rational.com/>
<http://subversion.tigris.org/>
<http://better-scm.berlios.de/comparison/>
<http://www.microsoft.com/>
<http://www.c-sharpcorner.com/>
<http://www.gotdotnet.ru>

МЕСТО ЗАДАЧИ КОМПОНОВКИ В ПРОЦЕССЕ ПРОЕКТИРОВАНИЯ БМК

Н.В. Старостин, А.В. Филимонов

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В настоящее время развитие цифровой техники характеризуется широким внедрением в радиоэлектронную аппаратуру (РЭА) больших и сверхбольших интегральных схем (БИС и СБИС). Проектирование подобных схем, особенно с учетом постоянного

увеличения их размеров и сложности, – трудная техническая и математическая задача. Существует несколько типов БИС и процессов их разработки, отличающихся прежде всего сложностью и уровнем специализации, т.е. технико-экономической эффективностью реализации.

БИС по способу проектирования делятся на два класса – заказные и полузаизанные. Заказная микросхема – это схема, разработанная на основе стандартных или специально созданных элементов и узлов по функциональной схеме заказчика. Полузаказные БИС представляют собой совокупность заранее спроектированной постоянной части и переменной – заказной – части, структура которой определяется заказчиком. К полузаизанным БИС относятся базовые матричные кристаллы (БМК) и программируемые пользователем логические интегральные схемы (ПЛИС). При использовании БМК специализация полузаизанной БИС осуществляется на последнем этапе производства за счет нанесения переменных слоев межсоединений.

В данной работе рассматривается одна из задач проектирования БМК. Обычно БМК содержит центральную часть с матрицей регулярно расположенных базовых ячеек (БЯ) и каналы трассировки связей между нескоммутированными элементами БМК. По периферии расположены ячейки ввода-вывода и контактные площадки, предназначенные для организации ввода-вывода сигналов. Каждая БЯ может представлять собой как функционально законченный узел, выполняющий конкретную операцию, например И-НЕ, так и набор нескоммутированных элементов – транзисторов и резисторов, из которых формируется тот или иной библиотечный элемент.

Будем рассматривать процесс проектирования БМК, который подразумевает, что БЯ – это набор нескоммутированных элементов. Каждая БЯ или группа БЯ путем коммутации внутренних элементов может быть превращена в один из библиотечных элементов, из которых состоит функциональная схема заказчика. Таким образом, проектирование БМК состоит из двух этапов – определения, какие именно БЯ будет занимать тот или иной элемент функциональной схемы, и трассировки соединений между этими элементами на кристалле. Эти два этапа формализуются в задачи размещения и трассировки соединений.

Суть задачи трассировки – в проведении всех трасс между элементами функциональной схемы на кристалле. Трассы при этом могут проходить только в специально отведенных зонах БМК, называемых каналами трассировки. Очевидно, что поскольку место для трассировки ограничено, то с увеличением количества связей между элементами вероятность успешной прокладки всех трасс уменьшается. Кроме того, успех решения задачи трассировки напрямую зависит от того, каким именно образом элементы функциональной схемы (ФС) расположены на кристалле. Поэтому цель задачи размещения – расположить элементы ФС так, чтобы максимизировать вероятность успешного решения задачи трассировки.

Нетрудно заметить, что при размещении элементов ФС существуют ситуации, возникновение которых резко снижает вероятность успешной трассировки. К таким ситуациям можно отнести возникновение так называемых «горизонтальных связей»¹, т.е. связей между различными рядами БЯ. Это означает, что трассировщик должен будет проложить горизонтальную трассу через весь канал трассировки. Это приведет к тому, что возможности прокладки других трасс в этом канале станут намного меньше. В частности, в этом канале не смогут быть проложены любые трассы, соединяющие элемент выше горизонтальной трассы и ниже. Очевидно, что для предотвращения подобных

¹ Обычно считается, что ряды БЯ в БМК располагаются вертикально.

ситуаций необходимо размещать элементы ФС, имеющие связи, в пределах одного столбца матричных БЯ, т.е. необходимо максимально сократить число связей между элементами, расположенными на разных столбцах матричных БЯ. Таким образом, одним из этапов решения задачи размещения может являться решение задачи компоновки элементов ФС в ряды с минимизацией связей между рядами. Необходимость формализации задачи компоновки подразумевает выбор формальной модели. Одной из самых удобных моделей в данном случае является гиперграфовая модель.

Формализация задачи компоновки

Гиперграф является удобным средством моделирования электронных схем, в том числе и интегральных. Это связано с тем, что поскольку гиперграф является обобщением классического графа, то значительная часть методов и приемов теории графов может быть перенесена на гиперграфы, часто с сохранением семантики той или иной операции.

С другой стороны, гиперграфы – более общий аппарат и предоставляют несравненно большие возможности моделирования электронных схем в различных задачах проектирования. В частности, предложенная выше задача компоновки элементов подразумевает достаточно простую формализацию в терминах гиперграфовой модели.

Формализация электронной схемы происходит следующим образом: элементам ФС ставятся в соответствие вершины гиперграфа, а цепям, соединяющим элементы ФС, – гиперребра. Пусть мы имеем элементы L_5 и L_6 и соответствующие им вершины гиперграфа v_5 и v_6 , тогда в гиперграфе будет существовать гиперребро $\{v_5, v_6\}$, когда существует цепь, соединяющая L_5 и L_6 . Цепи, естественно, могут соединять более двух элементов.

Предложенная модель позволяет формализовать описанную задачу компоновки в задачу разбиения гиперграфа.

Постановка задачи разбиения

Проблема k -разбиения гиперграфа состоит в распределении множества вершин V по k подмножествам V_1, \dots, V_k таким образом, чтобы удовлетворялся набор ограничений, называемых декомпозиционными ограничениями, а оптимизируемая функция (функция цели, критерий оптимальности, функционал), определенная над разбиением, принимала экстремальное значение. Распределение предполагает, что множества V_1, \dots, V_k попарно не пересекаются и при объединении составляют исходное множество V :

$$\begin{aligned} V_i \cap V_j &= \emptyset, \quad i, j = \overline{1, k}, \quad i \neq j, \\ \bigcup_{i=1}^k V_i &= V. \end{aligned} \tag{1}$$

Подмножества разбиения определяют подграфы разбиения. Значение k фиксируется и задается как параметр задачи.

Определим декомпозиционные ограничения так, чтобы вес каждого подграфа разбиения $w(V_i) = \sum_{v_q \in V_i} w(v_q)$, лежал в следующих пределах:

$$L_i \leq w(V_i) \leq U_i, \quad i = \overline{1, k}, \tag{2}$$

L_i и U_i – минимальный и максимальный вершинные веса подграфов.

В ограничениях (2) константы L_i и U_i определяют пределы, в которых могут варьироваться веса подграфов разбиения.

Критерий оптимальности, как и декомпозиционные ограничения, определяются спецификой конкретной задачи. Один из наиболее часто используемых критериев – минимизация веса сечения.

Вес сечения определяется суммой весов ребер, которые целиком не принадлежат ни одному подграфу разбиения:

$$W_c = \sum_{e_c \in E_c} w(e_c) \rightarrow \min, \\ E_c = \left\{ e \in E \mid \exists v_i, v_j \in e : v_i \in V_a, v_j \in V_b, a \neq b, a, b = \overline{1, k} \right\}. \quad (3)$$

Задачу (1)–(3) будем называть задачей k -разбиения или k -декомпозиции гиперграфа.

Решение задачи декомпозиции гиперграфа в терминах исходной задачи компоновки следует интерпретировать следующим образом: попадание какой-либо вершины в подграф разбиения говорит о том, что соответствующий элемент ФС должен быть определен в вертикальный ряд БЯ БМК, который соответствует этому подграфу.

Дальнейшее размещение элементов ФС, т.е. определение, какие именно БЯ будет занимать тот или иной элемент, выполняется алгоритмом размещения, частью входной информации которого являются данные о том, какому ряду принадлежит тот или иной элемент. Таким образом, цель алгоритма размещения – определение местоположения элемента ФС в априорно определенном алгоритмом компоновки ряду БЯ, что является несравненно более простой задачей, чем полное размещение, т.е. размещение элемента ФС в пределах всего кристалла.

Общая схема работы многоуровневого алгоритма декомпозиции

Исследуемые задачи декомпозиции гиперграфа NP-трудны и на практике, как правило, имеют большие размерности. Алгоритмы, получающие точные решения таких задач, оказываются неприменимыми в силу большой вычислительной сложности. Одним из эвристических подходов к решению декомпозиционных задач на гиперграфах большой размерности является многоуровневый подход. Ключевая идея многоуровневого алгоритма декомпозиции¹ состоит следующем: вместо того чтобы строить k -разбиение непосредственно для исходного гиперграфа, сначала строится ряд его приближений (загрублений). Каждое загрубление уменьшает (редуцирует) размерность исходной задачи. Процесс редукции продолжается до тех пор, пока порядок гиперграфа не снизится до сотен или даже десятков. На гиперграфе таких размерностей и отыскивается так называемое начальное разбиение. Полученное решение используется для построения решения для исходной задачи. Это достигается путем выполнения серии переносов решения задачи меньшей размерности на задачу большей размерности с последующим улучшением перенесенного решения. Таким образом, работу многоуровневого алгоритма можно разделить на три фазы: первая – фаза загрубления, когда производится ряд последовательных редукций размерности задачи, вторая – фаза поиска начального разбиения и третья – фаза восстановления решения, когда производится серия

¹ В дальнейшем будем употреблять термин «декомпозиция» как синоним термина «разбиение», за исключением специально оговоренных случаев.

последовательных переносов решения задачи меньшей размерности на менее редуцированную задачу с последующим улучшением спроектированного решения.

Такой метод позволяет:

- Решать декомпозиционные задачи на гиперграфах большой размерности, содержащих десятки и сотни тысяч вершин.
- Контролировать качество решения в ходе работы алгоритма.
- Контролировать вычислительную сложность алгоритма.
- Осуществлять подстройку многоуровневой эвристики путем подбора алгоритмов загрубления, начального разбиения и улучшающих алгоритмов, входящих в ее состав.

Описанный выше многоуровневый алгоритм был реализован и опробован на кластерных и сеточных гиперграфах большой размерности (100000 и более) и показал способность находить приемлемые, а на классе кластерных гиперграфов – оптимальные решения.

Список литературы

1. Пономарев М.Ф., Коноплев Б.П. Микроэлектроника: Учеб. пособие для вузов. – М.: Высшая школа, 1987.
2. Мурога С. Системное проектирование сверхбольших интегральных схем. Кн. 2. – М.: Мир, 1985.
3. Бершадский А.М. Применение графов и гиперграфов для автоматизации конструкторского проектирования РЭА и ЭВА. - Саратов: СГУ, 1983.
4. Зыков А.А. Гиперграфы // Успехи математических наук. 1974. № 6 (180).

РАЗРАБОТКА ПРОГРАММНОЙ РЕАЛИЗАЦИИ ВЫЧИСЛЕНИЯ ИНТЕГРАЛОВ МЕТОДОМ МОНТЕ-КАРЛО

А.Н. Степина

Саратовский государственный социально-экономический университет

Задача интегрирования достаточно часто встречается на практике. Одним из самых известных численных методов интегрирования является метод Монте-Карло. Он нашел применение в экономике и оптимизации. В физике данный метод применяется для моделирования поведения отдельных элементов физических систем, в теории планирования метод Монте-Карло позволяет рассчитывать общую продолжительность реализации проекта. Этот метод является статистическим и позволяет рассчитывать любой класс интегралов, благодаря чему широко используется в математике.

Целью данной работы является создание программы для вычисления интегралов методом Монте-Карло.

Для достижения данной цели необходимо решить следующие задачи:

- изучить математические обоснования алгоритма вычисления интеграла методом Монте-Карло;
- определить основные функции разрабатываемой программы, ее входные и выходные данные, требования к интерфейсу;
- реализовать программу на языке C++ в интегрированной среде разработки приложений Borland C++ Builder Enterprises 7.0;

- продемонстрировать результаты работы программы на известных примерах.

В ходе выполнения работы были изложены принципы расчета с использованием метода Монте-Карло, рассмотрено математическое обоснование алгоритма, а также преимущества данного метода вычисления интегралов перед другими. Создана реализация данного метода на языке C++.

Разработанная программа обеспечивает выполнение следующих функций:

- ввод всех необходимых параметров;
- генерация псевдослучайных чисел для дальнейшего расчета интеграла;
- расчет интеграла;
- построение графика точек, попавших в область интегрирования для трехмерных интегралов;
- предоставление полного отчета о проведенных вычислениях;
- последующее перенесение отчета в MS Excel.

Входными данными для программы служат: размерность области интегрирования, параметры области интегрирования, тип интегрируемой функции, тип и величина погрешности, доверительная вероятность, стартовое значение генератора псевдослучайных чисел.

Выходными данными служат: значение вычисленного интеграла, целевая и достигнутая погрешности, доверительная вероятность, квантиль нормального распределения, объем объемлющего параллелепипеда, число итераций, объем выборки на последней итерации, число точек, попавших в область интегрирования, доля точек, попавших в область интегрирования (%), выборочное среднее, выборочная дисперсия, время счета (с), время расчета.

Интерфейс программы обеспечивает максимальную простоту для пользователя при работе с данной программой и одновременно гибкую систему ввода параметров для наиболее точного задания интегрируемой функции.

В программе присутствует справочная система, которая включает в себя подсказку по интерфейсу программы, справку о системе сообщений от программы, а также краткие замечания о назначении каждого из вводимых параметров. Данная программа предназначена для работы в операционной системе Windows 98/NT/XP.

В качестве тестовых примеров был рассчитан типовой интеграл от квадратичной функции по 3-мерному симплексу

$$I = \int_G (x_1^2 - 2x_1x_2 + x_3 + 1) dx_1 dx_2 dx_3 = \frac{7}{30} = 0,23333\dots$$

Результаты вычислений приведены в табл. 1 и 2.

Приближенное значение $I \approx 0,233365218610308$ найдено для целевой абсолютной погрешности 0,0001.

Погрешность: 0,000034416630896 или 0,014749984670 %.

Поставленная в работе задача – разработка программы для вычисления интегралов методом Монте-Карло – была полностью выполнена. То есть в интегрированной среде разработки приложений Borland C++ Builder Enterprises 7.0 была создана программа, названная «CarloS», предназначенная для расчета интегралов от многомерных скалярных функций, реализующая все функции, перечисленные выше.

Функциональность созданной программы позволяет использовать ее как в качестве учебной программы для проведения занятий для студентов высших учебных заведений экономических и физико-математических специальностей, так и для специализированных расчетов.

Таблица 1

Входные данные

Параметр	Значение
Размерность области интегрирования	3
Тип интегрируемой функции	Квадратичная функция
Параметры интегрируемой функции	$A = \begin{pmatrix} 2 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, c = 1$
Объемлющий параллелепипед	x_{\min} x_{\max} 0 1 0 1 0 1
Вид ограничения	Квадратичное
Параметры системы ограничения	$A = (1 \ 1 \ 1), \ b = 1$
Тип погрешности	Абсолютная
Величина погрешности	0.0001
Доверительная вероятность	0.95
Стартовое значение генератора случайных чисел	1000000

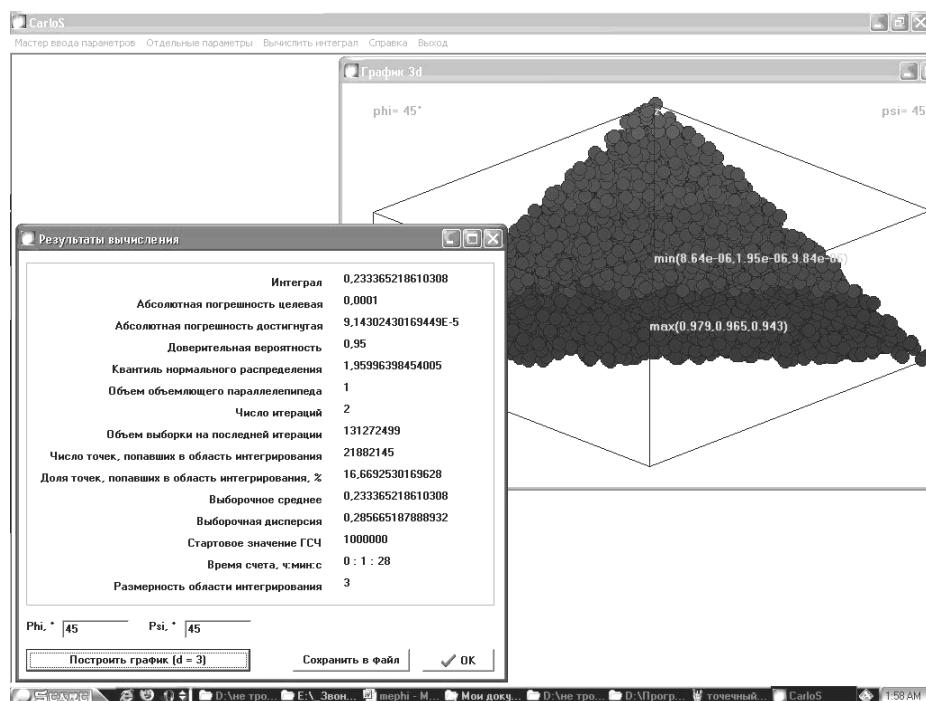


Рис. Форма вывода результатов вычислений и графика

Таблица 2
Выходные данные

Результат	Значение
Интеграл	0,233365218610308
Абсолютная погрешность целевая	0,0001
Абсолютная погрешность достигнутая	9,14302430169449E-5
Доверительная вероятность	0,95
Квантиль нормального распределения	1,95996398454005
Объем объемлющего параллелепипеда	1
Число итераций	2
Объем выборки на последней итерации	131272499
Число точек, попавших в область интегрирования	21882145
Доля точек, попавших в область интегрирования, %	16,6692530169628
Выборочное среднее	0,233365218610308
Выборочная дисперсия	0,285665187888932
Стартовое значение ГСЧ	1000000
Начало счета	13/10/2006 20:08:25
Окончание счета	13/10/2006 20:09:57
Время счета, с	92
Время счета, ч:мин:с	0:1:32
Размерность области интегрирования	3
Интегрируемая функция:	$f(x) = x'Ax + b'x + c$ $A = \begin{pmatrix} 2 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, b = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, c = 1$
Объемлющий параллелепипед ($x_{\min} \leq x \leq x_{\max}$):	$x_{\min} \quad x_{\max}$ 0 1 0 1 0 1
Ограничения:	$A = (1 \ 1 \ 1), \ b = 1$

Форма вывода результатов вычислений и график для квадратичной функции по 3-мерному симплексу приведены на рисунке.

Список литературы

1. Бережная Е.В., Бережной В.И. Математические методы моделирования экономических систем. – М.: Финансы и статистика, 2001. – 368 с.
2. Мюллер П., Нойман П., Шторм Р. Таблицы по математической статистике. – М.: Финансы и статистика, 1982. – 278 с.
3. Теннант-Смит Дж. Бейсик для статистиков. – М.: Мир, 1988. – 208 с.
4. Baranger J. Analyse numérique. Hermann, 1991.
5. Маделунг Э. Математический аппарат физики. Справочное руководство. – М.: Наука, 1968. С. 287.
6. Гмурман В.Е. Теория вероятностей и математическая статистика. – М.: Высшая школа, 2003.

РЕАЛИЗАЦИЯ ПОДДЕРЖКИ СТРАНИЧНОЙ ПАМЯТИ В АЛГОРИТМАХ ПОИСКА ГЛОБАЛЬНО-ОПТИМАЛЬНЫХ РЕШЕНИЙ

Е.В. Субботина, С.В. Сидоров, В.П. Гергель

Нижегородский госуниверситет им. Н.И. Лобачевского

В настоящей работе рассматривается схема организации структуры хранения поисковой информации, порождаемой в процессе решения задачи глобально-оптимального выбора в программной системе «Абсолют Эксперт» [2].

Постановка задачи

Многомерные многоэкстремальные задачи глобальной оптимизации являются типичной математической моделью процессов принятия решений. Аналитически такие задачи могут быть решены лишь в весьма простых частных случаях – обычным способом поиска глобально-оптимального решения являются различные численные итерационные процедуры. Одним из эффективных методов глобального поиска является индексный метод [1], накапливающий и использующий в ходе решения поисковую информацию. Поскольку в процессе поиска объем поисковой информации непрерывно растет, эффективная организация ее структуры хранения является существенным фактором скорости работы метода.

Понятие матрицы состояния поиска (МСП)

Для организации процесса оптимизации на каждой итерации поиска запоминается вся информация, получаемая о решаемой задаче оптимизации. Данная информация включает:

- x – координату точки итерации;
- z – значение функции в точке x ;
- d – расстояние до точки следующей итерации поиска;
- ix – номер итерации поиска;
- iz – индекс функционального значения.

Набор перечисленных параметров будем далее именовать *информационным элементом* или *столбцом* поисковых данных. Таким образом, поисковая информация есть множество информационных элементов: $A_k = (\alpha_1, \alpha_2, \dots, \alpha_k)$, где

$$\alpha_i = (x_i, z_i, d_i, ix_i, iz_i), \quad 1 \leq i \leq k,$$

$$d_i = (x_{i+1} - x_i)^{1/N}, \quad 1 \leq i < k, \quad x_1 < x_2 < \dots < x_k, \quad N - \text{размерность задачи.}$$

В процессе поиска элементы упорядочиваются по координатам точек итераций поиска.

Рассмотренное представление поисковой информации будем именовать *матрицей состояния поиска* (МСП).

Необходимо также отметить, что для решения многомерных задач в реализацию индексного метода введена схема редукции размерности, которая позволяет свести задачу поиска оптимального решения в многомерном пространстве к задаче в одномерном пространстве. При этом в задачах, где $Nm > 52$ (m – точность разбиения), возникает необходимость использования чисел расширенной точности для хранения координаты точки итерации и расстояния между точками итерации [3], в связи с чем матрица состояния поиска должна обеспечивать эффективное использование чисел расширенной точности.

Необходимость использования архивов

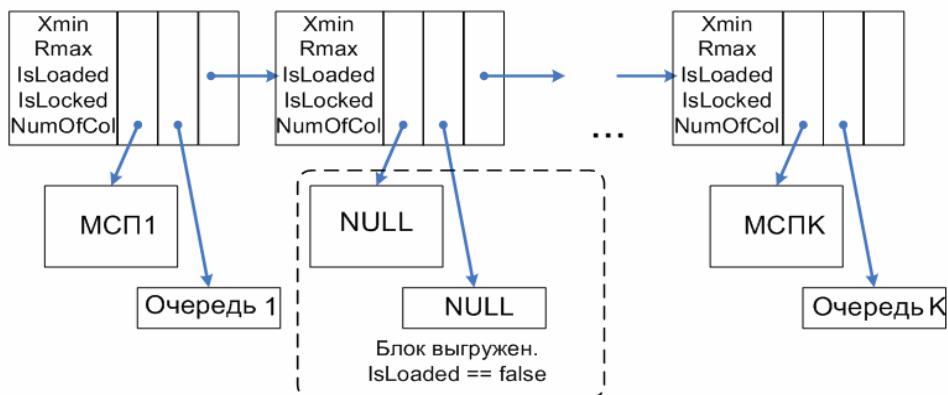
Подсистемой архивов называется организация матрицы состояния поиска, обеспечивающая хранение части поисковой информации во внешней памяти и эффективные стратегии подкачки данной информации.

В процессе решения задачи объем поисковой информации непрерывно растет, причем тем больше, чем выше размерность задачи. Возникает проблема организации вычислений при условии использования очень большого объема памяти.

Реализация эффективной структуры хранения поисковой информации и методов ее частичного вытеснения во внешнюю память является необходимым условием работы поисковых методов.

Реализация системы архивов

В основу реализации была положена идея страничной памяти. В системах со страничной организацией основная память делится на блоки (страницы) фиксированной длины, а внешняя память (главным образом дисковое пространство) представляет собой набор файлов, обеспечивающих сохранение отдельных страниц внешней памяти. Когда основная память заканчивается, некоторые страницы, в зависимости от используемой стратегии вытеснения, размещаются во внешней памяти, а зарезервированная ими память высвобождается для дальнейшего использования. И наконец, в тех случаях когда страница, к которой обращается процесс, не находится в физической памяти, нужно организовать ее подкачуку с диска. Для организации страничной памяти используется МСП следующего вида.



Заголовок каталога теперь содержит следующую информацию:

- *Xmin* – минимальная координата *x* среди всех столбцов блока;
- указатель на блок, соответствующий данному заголовку;
- указатель на следующий заголовок;
- указатель на локальную очередь;
- *Rmax* – максимальная характеристика в блоке (учитывает также правый граничный интервал);
- *IsLoaded* – загружен ли блок в память;
- *IsLocked* – флаг блокировки выгрузки;
- *NumOfCol* – число столбцов в блоке.

Поскольку часть блоков выгружена, локальные очереди для них также должны быть неактивны. Но мы должны знать максимальную характеристику для каждого блока для того, чтобы корректно находить интервал с максимальной характеристикой, поскольку он может располагаться в выгруженном блоке. Поэтому для каждого блока в заголовке поддерживается параметр Rmax, который содержит в себе максимальную характеристику в блоке, причем он должен зависеть еще и от правого граничного интервала, т. к. следующий блок тоже может быть выгружен.

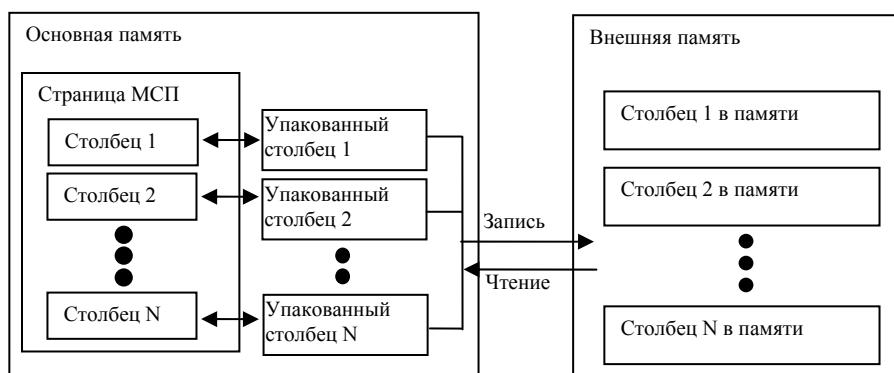
Менеджер памяти

Основным действием менеджера памяти является выделение кадра оперативной памяти для размещения в ней виртуальной страницы, находящейся во внешней памяти. Причем при необходимости выделения страницы основной памяти (например, при делении блока) с большой вероятностью не удастся найти свободный страничный кадр. В этом случае менеджер памяти, в соответствии с заложенными в него критериями, должен:

- найти некоторую занятую страницу основной памяти;
- переместить в случае необходимости ее содержимое во внешнюю память;
- переписать в этот страничный кадр содержимое нужной виртуальной страницы из внешней памяти;
- должным образом модифицировать необходимый элемент соответствующей таблицы страниц.

Упаковка и распаковка столбцов

При сохранении и восстановлении отдельных блоков МСП, состоящих из столбцов, содержащих числа расширенной точности, необходимо реализовать упаковку и распаковку элементов блоков в участки непрерывной памяти, так как данные числа не в одном куске памяти. Ниже представлена схема сохранения и восстановления блоков МСП.



Настоящая работа поддержана грантами Netherlands Organization for Scientific Research (NWO) № 047.016.014 и Российского фонда фундаментальных исследований (РФФИ) № 04-01-89002-HBO_a.

Список литературы

1. Strongin R.G., Sergeev Ya.D. Global optimization with non-convex constraints: Sequential and parallel algorithms. Kluwer Academic Publisher, Dordrecht, 2000.
2. Sysoyev A.V. Program system of parallel computations for solving the tasks of global-optimum choice // VI International Congress on Mathematical Modeling. Book of abstracts. 2004. P. 62.
3. Сидоров С.В., Сысоев А.В. Использование чисел расширенной точности в параллельных алгоритмах глобально-оптимального поиска // Материалы международной конференции «Высокопроизводительные вычисления на кластерных системах». Нижний Новгород, 2005. С. 233.
4. Субботина Е.В., Сысоев А.В. Способы организации хранения поисковой информации в программной системе параллельного решения задач глобально-оптимального выбора //Технологии Microsoft в теории и практике программирования. Нижний Новгород, 2006. С. 284.

ОБ ОПЫТЕ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ CLUSTER OPENMP ДЛЯ СОЗДАНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ

А.В. Сысоев, А.А. Лабутина, А.М. Камаев, А.А. Коваленко, А.А. Сиднев

Нижегородский госуниверситет им. Н.И. Лобачевского

Написание параллельной программы с использованием низкоуровневых механизмов: потоков, сокетов или разделяемой памяти – требует весьма значительных усилий. В качестве альтернативы сегодня в распоряжении разработчиков стандартизованные технологии OpenMP и MPI, предназначенные для систем с общей и распределенной памятью соответственно. В работе рассматривается представленная в 2006 году технология Cluster OpenMP, позволяющая создавать параллельные программы для систем с распределенной памятью, используя конструкции OpenMP в качестве языкового средства и компилятор в качестве инструмента. Обсуждаются способы создания программ на Cluster OpenMP, проводится сравнение трех упомянутых технологий на примере портирования библиотеки вероятностных сетей PNL.

1. Введение

Настоящая работа представляет результаты исследований по применению новой технологии Cluster OpenMP [2], предназначенной для разработки параллельных программ для систем с распределенной памятью. При этом, в полном соответствии с названием, технология предлагает создавать параллельные программы, используя стандарт OpenMP и поддержку со стороны компилятора (в настоящий момент такая поддержка присутствует в компиляторах Intel® C++ Compiler for Linux версии 9.1 и Intel Fortran Compiler for Linux версии 9.1). Объектом, на котором в рамках данной работы исследовалась технология Cluster OpenMP, являлась библиотека обработки вероятностных сетей Probabilistic Network Library [1].

2. Технология Cluster OpenMP

В 2006 году с выпуском очередных версий (9.1) компиляторов для языков C\С++ и Fortran компания Intel представила новую технологию для разработки параллельных программ для кластерных систем, получившую наименование Cluster OpenMP.

Технология Cluster OpenMP поддерживает выполнение OpenMP-программ на нескольких вычислительных узлах, соединенных сетью. Поскольку в этом случае узлы не обладают общей (shared) памятью, необходимой OpenMP-программе, ее наличие обеспечивается в Cluster OpenMP программным путем в библиотеке времени исполнения. В терминологии Cluster OpenMP этот механизм именуется распределенной общей памятью (distributed shared memory).

Программная имитация общей памяти приводит к необходимости синхронизации данных между вычислительными узлами в моменты чтения/записи. Таким образом, программист должен стремиться к уменьшению числа разделяемых данных, то есть данных, размещенных как shared.

Помимо общей памяти Cluster OpenMP имитирует и единое пространство потоков внутри параллельной программы, скрывая тот факт, что на каждом вычислительном узле исполняется некоторое количество процессов (их число, так же как и число потоков в каждом процессе, определяется в Cluster OpenMP явным указанием в специальном конфигурационном файле).

Написание параллельной программы на Cluster OpenMP в целом происходит так же, как и разработка OpenMP-программы, а значит, обеспечивает для систем с распределенной памятью те же преимущества, что и OpenMP для систем с общей памятью. Основным отличием между данными технологиями является необходимость в Cluster OpenMP-программе явно определять, какие данные должны быть размещены в разделяемой памяти.

Cluster OpenMP-программа может быть запущена и на SMP-системе, в этом случае она будет работать аналогично OpenMP-программе (при наличии необходимых указаний в конфигурационном файле).

В руководстве приводятся два условия, наличие которых позволит получить эффективную Cluster OpenMP-программу:

- хорошее ускорение, показываемое реализацией с использованием технологии OpenMP;
- хорошая локальность данных и невысокие требования по синхронизации.

Однако, поскольку описываемая технология является альтернативой MPI, в настоящей работе нас интересовало не только портирование существующих OpenMP-версий алгоритмов PNL, но и сравнение сложности разработки MPI- и Cluster OpenMP-вариантов решения одной и той же задачи, а также возможность портирования MPI-программы на Cluster OpenMP.

Не менее интересным представляется и сравнение эффективности различных вариантов параллельной программы: OpenMP и Cluster OpenMP на SMP-системе, MPI и Cluster OpenMP на кластере. Однако данная задача выходит за рамки настоящего исследования.

Более подробные сведения о технологии Cluster OpenMP представлены в [2].

3. Создание параллельной программы с использованием Cluster OpenMP

Разработка параллельной программы с использованием технологии Cluster OpenMP практически не отличается от написания такой же программы, использующей техноло-

гию OpenMP. Основное различие состоит в том, что кроме синхронизации обращений к общим данным программисту теперь надо следить ещё и за наличием этих данных у обрабатывающего их процессора. При разработке Cluster OpenMP-программы такие данные надо явно помечать в коде специальными конструкциями, в чём нередко может помочь компилятор. Такая особенность позволяет сравнительно легко реализовывать на Cluster OpenMP алгоритмы, ранее распараллеленные с использованием OpenMP, – нужно лишь портировать их, указав компилятору, каким способом следует размещать данные, используемые в процессе вычислений.

3.1. Портирование с OpenMP на Cluster OpenMP

Портирование кода с OpenMP на Cluster OpenMP заключается в явном разделении всех данных на общие (shared) и собственные (private) на каждом потоке [2]. Несмотря на то, что распределение данных на shared и private можно свести к нескольким стандартным ситуациям, компилятор не может гарантировать корректность автоматического выбора необходимого способа хранения данных для каждого процесса. Поэтому портирование с OpenMP на Cluster OpenMP необходимо осуществлять под контролем программиста – в полуавтоматическом режиме.

Технология Cluster OpenMP предоставляет в распоряжение программиста средства для явного размещения данных в разделяемой памяти. В первую очередь это директива компилятора `#pragma intel omp sharable (<variable_list>)` и макрос `kmp_sharable`, а также набор расширений API и набор рекомендуемых решений (к примеру, для STL), подробно описанных в User's Guide.

3.2. Портирование с MPI на Cluster OpenMP

Если по способу создания Cluster OpenMP-программа максимально близка к программе на OpenMP, то по модели исполнения она все же существенно ближе к программе на MPI. Узким местом всех параллельных программ для систем с распределенной памятью, вне зависимости от того, на основе какой технологии они написаны, является обмен данными через сеть. Именно этот момент должен быть проверен в первую очередь в том случае, если после портирования с OpenMP ускорение получившейся Cluster OpenMP-программы при запуске на кластере из N вычислительных узлов оказывается существенно меньше, чем ускорение исходной OpenMP-версии при запуске на SMP-системе с тем же числом процессоров.

Если замедление действительно вызвано синхронизацией данных между узлами, то его причинами могут быть, во-первых, количество объектов, расположенных в sharable-памяти, во-вторых, объем передаваемых данных.

Повысить эффективность в первом случае можно, сделав private все объекты, для которых это допустимо. Возможности данного подхода ограничены сложностью отделения в реальных программных системах, содержащих тысячи объектов, данных, которые должны быть sharable, от тех, которые должны быть private.

Повышение эффективности путем уменьшения объема пересылаемых данных доступно программисту на Cluster OpenMP лишь частично. Разделяемая (sharable) память организована в Cluster OpenMP страничным образом [2], а синхронизация выполняется целыми страницами. Доступа к параметрам внутренней организации разделяемой памяти у программиста нет, таким образом, уменьшения объема передаваемой информации можно добиться лишь путем «упаковки» sharable-объектов (максимально близкого их размещения), что не всегда возможно в принципе.

Решение указанных проблем состоит в использовании для написания Cluster OpenMP-программы подхода, характерного для MPI. При наличии готовой MPI-программы мы можем говорить о фактическом портировании с MPI на Cluster OpenMP. Дополнительный плюс такого портирования состоит в том, что в хорошо написанной MPI-программе количество пересылок данных обычно сведено к минимуму.

Итак, задача состоит в следующем: имея готовую реализацию алгоритма на MPI, портировать её на Cluster OpenMP. В рамках описываемого исследования была найдена и практически проверена процедура портирования с MPI на Cluster OpenMP.

Логика MPI-программы предполагает параллельную работу процессов с самого начала выполнения. В то же время параллельные секции в Cluster OpenMP-программе (так же как и в OpenMP) используются только в тех участках кода, где это необходимо. Таким образом, для воссоздания MPI-модели требуется создать параллельную секцию в самом начале кода.

Для организации взаимодействия параллельных программ MPI предлагает специальный API (`MPI_Send`, `MPI_Recv`, `MPI_Allreduce`, ...), следовательно, первый шаг портирования состоит в реализации необходимых функций средствами Cluster OpenMP. При аккуратном написании такого API, sharable-данные будут находиться только в части кода, реализующей аналоги MPI-функций, во всей остальной программе данные (за исключением статических) будут `private`. При написании API особое внимание следует уделить выравниванию данных в памяти. Данные разных процессов должны располагаться на разных страницах памяти. Для этого нужно использовать функцию `kmp_aligned_sharable_malloc`.

После выполнения указанных действий могут появиться проблемы со статическими объектами. Статические объекты по умолчанию располагаются в `private` памяти, а значит, доступ к ним в параллельной секции может приводить к возникновению ошибок сегментирования. Поскольку, как было указано выше, вся наша программа состоит теперь из одной параллельной секции, данная ошибка будет возникать при доступе к любому статическому объекту. Таким образом, статические объекты необходимо сделать `sharable`, используя соответствующую директиву Cluster OpenMP.

Представленная схема портирования является универсальной, но не оптимальной. В каждом конкретном случае в зависимости от существа задачи полученный код может быть оптимизирован. Например, может быть выполнена замена группы функций `MPI_Send`, `MPI_Recv` на конструкции Cluster OpenMP, дающие ту же функциональность, но с меньшим числом пересылок по сети.

4. Использование Cluster OpenMP для распараллеливания алгоритмов вывода и обучения библиотеки обработки вероятностных сетей Probabilistic Network Library (PNL)

Описанные выше подходы к созданию Cluster OpenMP-программы были применены в процессе портирования библиотеки PNL на Cluster OpenMP. Далее представлено описание процесса распараллеливания алгоритмов вывода и обучения библиотеки PNL с использованием каждого из подходов.

4.1. Библиотека PNL – краткая характеристика

Библиотека PNL (Probabilistic Network Library) – средство для работы с графовыми моделями общего назначения. Библиотека позволяет пользователю работать с моделями в виде ориентированных и неориентированных графов, с дискретным и непрерывным

распределением вероятности на вершинах сети. Библиотека содержит высокопроизводительные реализации алгоритмов вывода и обучения байесовских и марковских сетей.

Несколько алгоритмов обучения и вывода в библиотеке PNL помимо последовательных имеют также параллельные версии, разработанные на основе технологий OpenMP и MPI. К числу таких алгоритмов относятся алгоритмы вывода Junction Tree Inference Engine, Loopy Belief Propagation, Gibbs Sampling Inference и алгоритм обучения параметров байесовской сети на основе максимизации ожидания Generalized EM Learning.

4.2. Портирование алгоритмов PNL с OpenMP на Cluster OpenMP

Портирование алгоритмов библиотеки PNL являлось достаточно трудоемкой задачей ввиду ее размеров. Особенностями OpenMP-версий алгоритмов вывода и обучения библиотеки являются активное использование STL-контейнеров (преимущественно `std::vector`) и явное планирование вычислительной нагрузки (*explicit work-sharing*). Помимо самих алгоритмов пришлось значительное внимание уделить коду, отвечающему за создание и чтение с диска исходных данных (вероятностных сетей и наблюдений), т.к. эти данные были необходимы всем процессам для проведения расчетов.

На первом этапе была использована предлагаемая User's Guide [2] последовательность шагов по портированию с целью получения на основе имеющихся OpenMP-версий алгоритмов библиотеки их Cluster OpenMP-вариантов. Однако в силу весьма высокой сложности и объемности библиотеки предлагаемые в User's Guide шаги к успеху не привели.

Следующим действием стала попытка заменить аллокаторы в STL-контейнерах, активно используемых в библиотеке. Это действие также было описано в User's Guide, как раз для подобных объектов. В силу нескольких небрежного обращения с STL-контейнерами в PNL данный шаг процедуры портирования также не привел к успеху: постоянно возникали ошибки компиляции с диагностикой, указывающей в стандартную библиотеку языка C++.

В результате в силу невозможности напрямую разместить в разделяемой памяти большую часть динамически создаваемых объектов, активно использовавшихся в параллельных вычислениях, было принято решение всю динамическую память выделять в разделяемой области памяти – «сделать всю кучу sharable». Для того чтобы реализовать такой подход, была использована возможность глобальной перегрузки операторов `new/delete`.

После выполнения данного действия были получены стабильно работающие версии трех алгоритмов, четвертый (Loopy Belief Propagation) по-прежнему завершался аварийно. На этот раз причиной оказалась передача параметров по ссылке внутри алгоритма. Заменой в необходимых местах ссылок на указатели данная ошибка была устранена и получена работающая версия четвертого параллельного алгоритма библиотеки PNL.

4.3. Портирование алгоритмов PNL с MPI на Cluster OpenMP

Все параллельные алгоритмы библиотеки PNL имеют две реализации: OpenMP и MPI. После успешного портирования OpenMP-версий алгоритмов следующей целью настоящей работы стало исследование возможности портирования версий, использующих технологию MPI.

Первым алгоритмом, портированным с MPI на Cluster OpenMP, был Loopy Belief Propagation.

Первым шагом стало преобразование MPI-версии в OpenMP:

- Конструкции MPI (структуры, функции) были заменены на аналоги, реализованные на OpenMP. Были реализованы только те функции, которые используются в MPI-версии алгоритма.

- В самое начало тестовой программы была добавлена параллельная секция (директива препроцессора `#pragma omp parallel`).

В результате была получена работоспособная OpenMP-версия. Далее решалась задача ее сборки под Cluster OpenMP. Первый шаг состоял в простой замене ключа `-openmp` на `-cluster-openmp`. Сборка прошла успешно, однако при запуске полученной программы возникла ошибка сегментирования. Было обнаружено, что причина – в использовании статических объектов класса `string` (из библиотеки STL). Чтобы устранить эту ошибку, в обычной ситуации достаточно разместить статические объекты в `sharable`-памяти, однако в данном случае это решение не помогло. Класс `string` сам содержит поля, для которых используется динамическое выделение памяти через функцию `malloc` или оператор `new`. Память, выделяемая таким способом, расположена в `thread private` области, поэтому вызов любого метода из класса `string`, который обращается к этим `private` полям, вызывает ошибку сегментирования. В данном случае было решено отказаться от использования объектов класса `string` в ситуациях, которые вели к ошибкам, и заменить их на переменные типа `char*` (альтернативным решением могла быть замена аллокатора в классе `string`).

Далее полученные OpenMP-версии MPI-функций были портированы на Cluster OpenMP (для размещения данных в них по отдельным страницам `sharable`-памяти использовалась функция `kmp_aligned_sharable_malloc`). Кроме того, все использующиеся в алгоритме статические переменные были сделаны `sharable`.

В результате была получена стабильно работающая Cluster OpenMP-версия.

Описанная последовательность шагов была затем успешно применена еще к двум параллельным алгоритмам – Gibbs Sampling Inference и Generalized EM Learning.

4.4. Результаты портирования

Как и любая технология программирования, Cluster OpenMP имеет целью предоставить разработчику средства и методы, позволяющие сократить затраты на создание программных продуктов. При этом в силу выбранного подхода Cluster OpenMP предлагает весьма высокоуровневые механизмы для программирования под кластерные системы, чем выгодно отличается от MPI.

По результатам предварительных экспериментов два из четырех портированных алгоритмов (Gibbs Sampling Inference и Generalized EM Learning) показали ускорение, близкое к тому, что имели исходные OpenMP- и MPI-версии. Результаты двух других алгоритмов оказались менее хорошими.

Заключение

В рамках выполнения настоящей работы получен положительный опыт использования новой технологии разработки параллельных программ для систем с распределенной памятью Cluster OpenMP. В ходе исследований последовательность портирования OpenMP-программ на Cluster OpenMP, представленная в [2], опробована на примере библиотеки обработки вероятностных сетей Probabilistic Network Library и дополнена новыми пунктами. Разработана и практически опробована на алгоритмах PNL последовательность действий по портированию MPI-версий параллельных программ на Cluster OpenMP. Получены сравнительные оценки затрат на портирование с OpenMP и MPI (только на примере библиотеки PNL).

Работа поддержана грантом компании Intel.

Список литературы

1. Belov S.A., Gergel V.P., Sysoyev A.V. Scalable parallel inference algorithms in probabilistic networks // Preproceedings of UK–Russia Workshop on Proactive Computing. N. Novgorod, 2005. P. 5 – 10.
2. Cluster OpenMP User’s Guide Version 9.1: [<http://developer.intel.com>].
3. Gergel V.P. Teaching Course: CS338. Introduction to Parallel Programming: [<http://www.msdnaacr.net/curriculum/pfv.aspx?ID=6594>].
4. Официальный форум MPI: [<http://www mpi-forum.org>].
5. Официальный сайт OpenMP: [<http://www.openmp.org>].

ПРОЕКТ АВТОМАТИЗИРОВАННОГО ИНФОРМАЦИОННО-ТОРГОВОГО КОМПЛЕКСА УПРАВЛЯЮЩИХ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Д.П. Тамбовцев, С.В. Шибанов

Пензенский госуниверситет

Постановка задачи

В рамках проекта создается автоматизированный информационно-торговый комплекс систем, предназначенный для осуществления торговой деятельности на фондовых и валютных биржевых площадках России и всего мира. Главной задачей разработки данного комплекса систем является автоматизация процесса торговли на заданной биржевой площадке. В процессе работы на фондовой или валютной бирже полностью исключается влияние человеческого фактора, а именно психологии человека (его чувств и эмоций), на принятие торговых решений. Далее важно понимать, что целью разработки этой совокупности систем не является создание автоматической торговой системы, которая сама выставляет заявки, совершает сделки. Отличие от «торгового робота» велико и заключается в том, что разрабатывается лишь инструмент автоматизации, а сама логика работы и принципы совершения сделок – это центральная и отдельная часть комплекса, это обучаемая база знаний, которая настраивается и регулируется человеком, ее устройство и принципы работы являются коммерческой тайной данного проекта.

Краткое описание проекта

Для дальнейшего знакомства с составными частями информационно-торгового комплекса необходимо иметь хотя бы смутное представление об организации торговой деятельности на фондовых и валютных биржевых площадках.

Основными игроками на любой бирже являются профессиональные участники торгов – брокеры, которые имеют штат специалистов и необходимые лицензии. Рядовой человек или организация может участвовать в торгах посредством брокера, который будет выставлять заявки клиента и совершать сделки от его имени за комиссию. На сегодняшний день этот процесс достаточно автоматизирован: у брокеров есть серверы, которые подключены через специальные шлюзы к биржам, у клиентов есть торговые

платформы, с помощью которых осуществляется взаимодействие с брокером и через него собственно с биржей. Передаваемые данные, естественно, шифруются торговой платформой, предоставляемой брокером. Передача данных осуществляется через Интернет.

Взаимодействующими частями информационно-торгового комплекса являются:

1. Торговая платформа, предоставляемая брокером. Функции и возможности:
 - обеспечение доступа к торговам на фондовом, валютном и срочном рынках;
 - получение биржевой информации в режиме реального времени, включая очереди котировок ценных бумаг;
 - обеспечение участника торгов и его клиентов информацией о собственных заявках и сделках;
 - возможность подачи стоп-заявок, отложенных заявок с их пакетным выставлением в торговую систему биржи;
 - мониторинг показателей маржинального кредитования встроенными средствами и многое другое.
2. Управляющая программа, которая реализует логику работы на бирже. Функции:
 - принятие управляющей информации о заявках и сделках от базы знаний и ее последующая обработка;
 - выдача управляющих сигналов для торговой платформы и таким образом заключение сделки или выставление заявки;
 - наполнение базы знаний данными торгов, поступающими в режиме реального времени от торговой платформы, на основе которых принимаются управляющие решения;
 - наполнение базы знаний историческими данными, которые также участвуют в процессе принятия управляющего решения.

Управляющая программа реализована с помощью средств, предоставляемых средой программирования Visual Studio .NET 2005, на языке Visual C#.

3. Библиотека интерфейсов взаимодействия управляющей программы и торговых платформ.

По своей сути торговая платформа является лишь инструментом в руках трейдера в работе на рынке. В настоящее время существует множество торговых платформ: TRANSAQ, QUIK, METATRADER и т.д. Выбор зависит от возможностей брокера, от торговой площадки, т.е. биржи, и от предпочтений трейдера. Библиотека интерфейсов взаимодействия является универсальным расширением информационно-торгового комплекса и представляет собой набор интерфейсов взаимодействия, каждый из которых, в свою очередь, является библиотекой служебных функций контроля для конкретной торговой платформы. Таким образом, для каждой торговой платформы нужен свой интерфейс управления, функциями которого управляющая программа и «заставляет» торговую платформу работать на рынке. На практике для торговли используется не так много торговых платформ, например в России около 90% трейдеров, работающих на фондовых площадках ММВБ и РТС, используют торговую платформу QUIK и примерно 80% игроков валютного рынка FOREX используют METATRADER.

В нашем случае существуют два варианта интеграции двух взаимодействующих программ (управляющей программы и торговой платформы):

- если в торговой платформе предусмотрены такие средства взаимодействия с ней, как, например, встроенный алгоритмический язык QPILE торговой платформы QUIK, с помощью которого мы можем «подсовывать» программе действия, мы вполне можем воспользоваться этими средствами;

– второй способ универсальный – взаимодействие через внешний интерфейс программы, т.е. в библиотеке находятся функции, которые управляют программой как пользователь, с помощью функций и системных сообщений WIN32 API.

Библиотека интерфейсов является набором dll-файлов, т.е. библиотек управляемых функций, написанных на языке программирования Visual C#, в среде Visual Studio .NET 2005 с использованием функций и системных сообщений WIN32 API.

4. База знаний информационно-торгового комплекса управляющих систем. Функции:

- хранение исторических данных торгов (цены, объемы торгов и т.д.);
- хранение торговых систем, каждая из которых является отдельной логикой работы на фондовом или валютном рынке;
- выработка управляющих сигналов на основе данных и определенной логики работы.

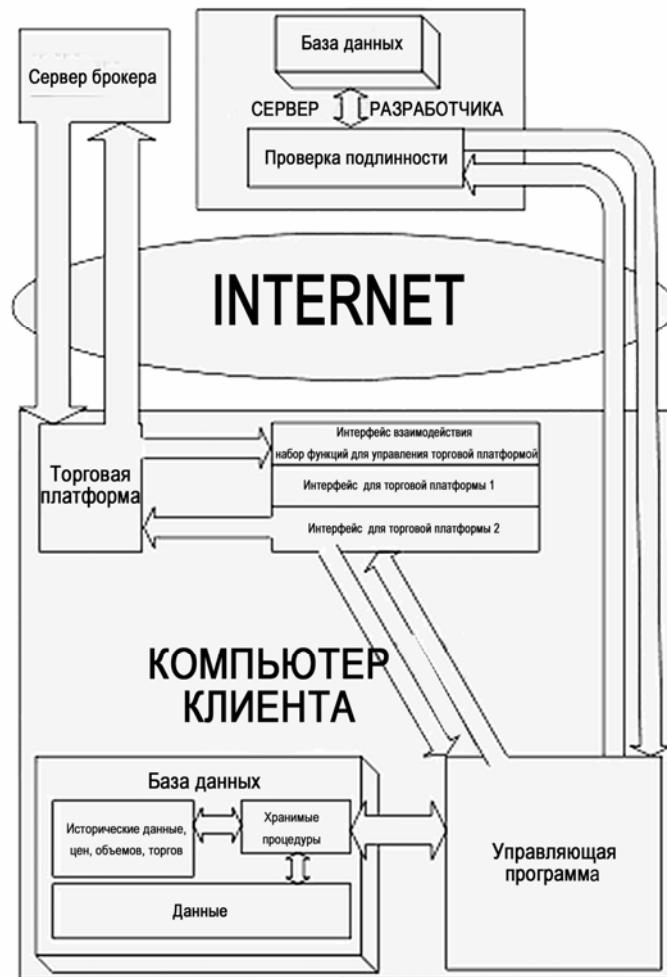


Рис. Схема взаимодействия логических частей автоматизированного информационно-торгового комплекса управляющих систем реального времени

База знаний информационно-торгового комплекса управляющих систем реализуется с использованием Microsoft SQL Server 2000. Выбор СУБД не случаен: в процессе тестирования, при закачке данных в режиме реального времени MS SQL Server 2000 показала один из лучших результатов по сравнению с другими СУБД. Так как остальные части комплекса пишутся на Visual C# (тоже продукт Microsoft), было решено в качестве СУБД выбрать MS SQL Server 2000. Во-первых, продукты одного производителя будут взаимодействовать оптимально, во-вторых, данная СУБД показала неплохой результат при тестировании.

5. Система проверки подлинности. Данная система является частью управляющей программы, но выделена в отдельную функциональную единицу.

На данный момент создан некий каркас этого информационно-торгового комплекса, но уже сейчас наблюдается заметная экономическая отдача от его применения. Поэтому в будущем, после соответствующего тестирования и возможного расширения функциональности комплекса, планируется коммерческое использование данного проекта. И как любой коммерческий проект, данный комплекс имеет систему защиты от копирования и распространения, которая постоянно совершенствуется. Проверка подлинности производится через Интернет, путем взаимодействия с сервером разработчика, который дает разрешение на работу данного комплекса или запрещает ее, если запущена копия.

ИСПОЛЬЗОВАНИЕ НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ ПРИ ПОСТРОЕНИИ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ БАНКОВСКОГО КРЕДИТНОГО СКОРИНГА

Д.П. Ульянов, Е.А. Палькин, В.Н. Гусятников

Саратовский государственный социально-экономический университет

В докладе рассматриваются возможности нейросетевых технологий при построении скоринговых карт в современном российском банке, их практическая реализация, апробация и корректировка.

Задача работы: построить адекватную автоматизированную систему скоринга заемщиков, провести практическую реализацию системы принятия решений, на основе нейронной сети, оценить результаты и провести корректировку в случае необходимости.

В упрощенном виде скоринговая модель представляет собой взвешенную сумму определенных характеристик клиента. В результате получается интегральный показатель (score); чем он выше, тем выше надежность клиента, и банк может упорядочить своих клиентов по степени возрастания кредитоспособности.

Интегральный показатель каждого клиента сравнивается с неким числовым порогом, или линией раздела, которая, по существу, является линией безубыточности и рассчитывается из отношения, сколько в среднем нужно клиентов, которые платят в срок, для того, чтобы компенсировать убытки от одного должника. Клиентам с интегральным

показателем выше этой линии выдается кредит, клиентам с интегральным показателем ниже этой линии – нет.

Философия скоринга заключается не в поиске объяснений, почему этот человек не платит. Скоринг выделяет те характеристики, которые наиболее тесно связаны с ненадежностью или, наоборот, с надежностью клиента. Банк не знает, вернет ли данный заемщик кредит, но зато банк знает, что в прошлом люди этого возраста, этой же профессии, с таким же уровнем образования и с таким же числом изживенцев кредит не возвращали. Поэтому давать кредит этому человеку рискованно.

Задача предсказания степени надежности конкретного клиента на основе кредитных историй предыдущих клиентов может эффективно решаться с применением нейросетевых технологий. Однако современные скоринговые системы, применяемые в российских банках, не используют их возможности.

В качестве нулевого приближения работы автоматизированной скоринговой системы выбран стандартный алгоритм принятия решения в банке.

Первый этап разработки автоматизированной системы: определение входных параметров. В банковской практике – это прежде всего анкета клиента, информация из кредитного бюро и данные по счетам, если речь идет об уже действующем клиенте банка.

Второй этап: назначение веса каждой характеристике или «построение скор-карты». Для осуществления задачи адекватной оценки веса каждого параметра были использованы анкетные данные 100 заемщиков. Из этого числа экспертным путем были отсеяны те заемщики, которые по оценке кредитных инспекторов должны получить решение «отказать в выдаче». Анкета с максимальным количеством баллов из числа «отказных» установила границу «одобрения». Теперь анкеты, набравшие количество баллов ниже данной отметки, автоматически получают решение «отказать», а, соответственно, выше – решение «одобрить».

Третий этап – это апробация полученной скор-карты и корректировка весов, в случае необходимости, при рассмотрении новых заявок.

Для выполнения задачи данного этапа было взято 500 кредитных заявок на получение автокредита. Результат их обработки выявил: 132 заявки набрали количество баллов выше зоны одобрения (положительное решение); 49 заявок набрали количество баллов в пределах зоны одобрения, поэтому были реструктуризованы (уменьшена сумма кредита). По остальным 319 заявкам было получено решение «отказать».

После получения скорингового решения кредитная заявка дополнительно пересматривалась кредитным комитетом, и статус заявки подтверждался или опровергался. Если происходило опровержение статуса заявки, то веса критериев пересматривались заново. Таким образом, если рассматривать скоринг как адаптивный сумматор нейрона, то процедура пересмотра весов является обучением нейрона методом градиентного спуска (обратного распространения ошибки).

Полученным после корректировки весов заявкам присваивался статус «одобрено к выдаче» и ссуды финансировались. Таким образом, был сформирован базовый кредитный портфель скоринговой карты.

Результат работы системы скоринга в нулевом приближении: 136 кредитов выдано, из них 14 кредитов впоследствии признаны безнадежными к взысканию.

Недостатком данной системы скоринга является отсутствие гибкости, так как один нейрон может разбивать множество решений только на две гиперплоскости. Для принятия решения о выдаче кредита необходимо построение областей во множестве решений. Данная задача может реализоваться только нейронной сетью.

Деревья решений могут строиться в скоринг-модели в виде правил. При этом дерево решений способно перестраиваться при добавлении новых примеров, игнорировать несущественные признаки (Николай Паклин. BaseGroup Labs. Deductor:Loans – комплексное скоринговое решение в области потребительского кредитования. <http://www.basegroup.ru/solutions/loans.htm>).

Возможна и несколько иная концепция, когда определяется набор ключевых параметров, которые в скоринг-модели будут играть роль узлов при ветвлении дерева (см. рис. 1).

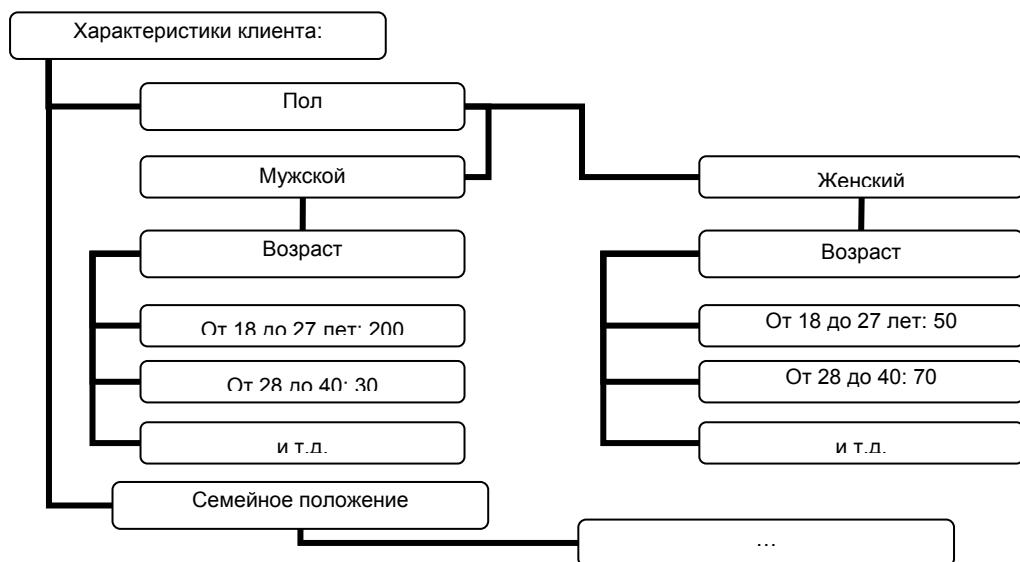


Рис. 1. Дерево решений скоринговой системы

Для решения данной задачи в работе использована нейронная сеть (НС) обратного распространения (back propagation) ошибки, так как НС такого типа являются эффективным инструментом поиска закономерностей, прогнозирования, качественного анализа.

Нейронная сеть обратного распространения состоит из нескольких слоев нейронов, причем каждый нейрон слоя i связан с каждым нейроном слоя $i + 1$, т. е. речь идет о полностью связной НС.

Средой разработки нейронной сети была выбрана Borland Delphi 7. Сеть была построена на основе свободно распространяемых модулей, реализующих работу многослойной нейронной сети. На вход нейронной сети подавались числовые параметры анкетных данных. На выходе формировалось числовое значение, соответствующее оценке кредитоспособности данного заемщика.

Для проверки адекватности работы новой скоринг-карты были повторно проведены все этапы: прохождение новых заявок, получение результата, анализ результатов экспертами, корректировка, корректировка области одобрения. В результате из множества заявок, рассмотренных при первом опыте, было дополнительно «одобрено» еще 53 заявки. Новая скоринг-карта позволила более гибко анализировать анкеты и выборочно менять веса входных параметров, что привело к увеличению баллов у данных 53 анкет. Например, теперь значимость возраста у женщин от 18 до 27 лет стала большей, т.к. риск поступления женщины в

армию ничтожен, а у мужчин высок. Или вес семейного положения «разведен» у женщин стал более высоким, чем у мужчин, т.к., по статистике, разведенные мужчины чаще теряют работу и свой социальный статус, чем женщины, и т.п.

Результат работы второго варианта скринговой системы: 188 кредитов – выдано, из них 14 кредитов впоследствии признаны безнадежными к взысканию.

Следующим этапом работы стал поиск дополнительных входных параметров для проверки непротиворечивости данных и обнаружение скрытых факторов возможного дефолта в имеющихся данных.

Поиск и извлечение скрытой информации из имеющейся в анкете данных является практической реализацией технологии раскопки данных Data Mining, когда новая информация образуются в виде правил «если..., то...». Подробный анализ анкетных данных клиентов, по которым имелась длительная просрочка, позволил выявить следующие параметры, корелирующие с вероятностью дефолта:

- географическое место проживания клиента;
- взаимосвязь возраста заемщика и занимаемой должности;
- точка обращения клиента за кредитом (офис банка, конкретный магазин или конкретный автосалон) и т.д.

После введения процедур дополнительного анализа скринг-карта сократила количество заявок, получивших положительное решение, на 8 анкет, что на первый взгляд сократило кредитный портфель. Однако и количество дефолтных ссуд теперь снизилось до количества 6 кредитов. На рис. 2 ломаная линия схематично показывает границу между зонами «одобрения» и «отклонения» в разработанной скринговой системе.

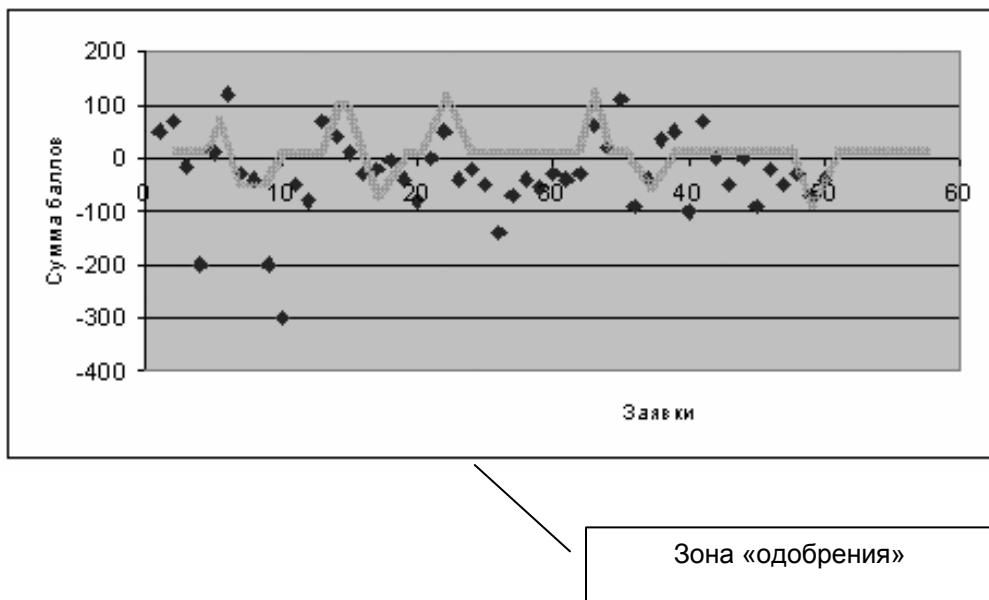


Рис. 2. Изменение границы между зонами «одобрения» и «отклонения»
при использовании нейронной сети

Результат работы третьего варианта построенной автоматизированной скринговой системы: 180 кредитов – выдано, 6 кредитов впоследствии признаны безнадежными к взысканию.

В нашем проекте получил реализацию новый альтернативный подход: методика оценивает не столько кредитоспособность заемщика, сколько строит прогноз его современной и будущей возможности выполнять свои обязательства, проверяет данные клиента на правдивость. Другими словами, строятся априорные оценки его будущих вероятностных доходов и расходов (<http://www.consumerlending.ru/products/Application Scoring/MacroScoring/methodology/scoring/>). В данной системе большее значение приобрели факторы, которые ранее если и рассматривались, то их влияние на результат было ничтожным. Среди них район проживания, уровень образованности, точка обращения за кредитом и др.

Стоит заметить, что в работе принималось несколько важных допущений, одним из которых является правдивость информации. Подобное допущение является очень существенным, ведь если информация, подаваемая для проверки, будет изначально неверной, то и результат получится искаженным. В работе рассматривался только один этап принятия кредитного решения – скоринг. В реальных условиях необходима проверка сотрудниками службы безопасности достоверности указываемой информации, непротиворечивости декларируемых данных. Без этой работы функционирование скоринговой системы мало того, что будет неэффективным, но и впоследствии вообще приведет к «засорению» системы. Скоринговая карта, основанная на принципе запоминания кредитной истории «прошлых клиентов» со сходными характеристиками, через определенное время будет отказывать даже хорошим клиентам только потому, что когда-то клиент, имеющий похожие характеристики, оказался мошенником.

Именно поэтому скоринговая система постоянно должна подвергаться «чистке». Из множества одобренных анкет должны впоследствии отсеиваться анкетные данные мошенников, чтобы их характеристики не оказывали влияния на общий результат.

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ КРУГЛЫХ ПРОТЯЖЕК В СРЕДЕ T-FLEX CAD С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ СОМ И БИБЛИОТЕКИ MFC

А.А. Федченко

*Рузаевский институт машиностроения
Мордовского госуниверситета им. Н.П. Огарева*

Проектирование сложного металлорежущего инструмента связано с выполнением большого объема вычислительной работы. Расчет такого инструмента, как протяжка, производится по известным аналитическим формулам. Эти формулы включают в себя большое количество справочных коэффициентов, что существенно повышает трудоемкость расчета. Кроме того, для проектирования этого инструмента большое значение имеет выбор наиболее эффективной схемы резания.

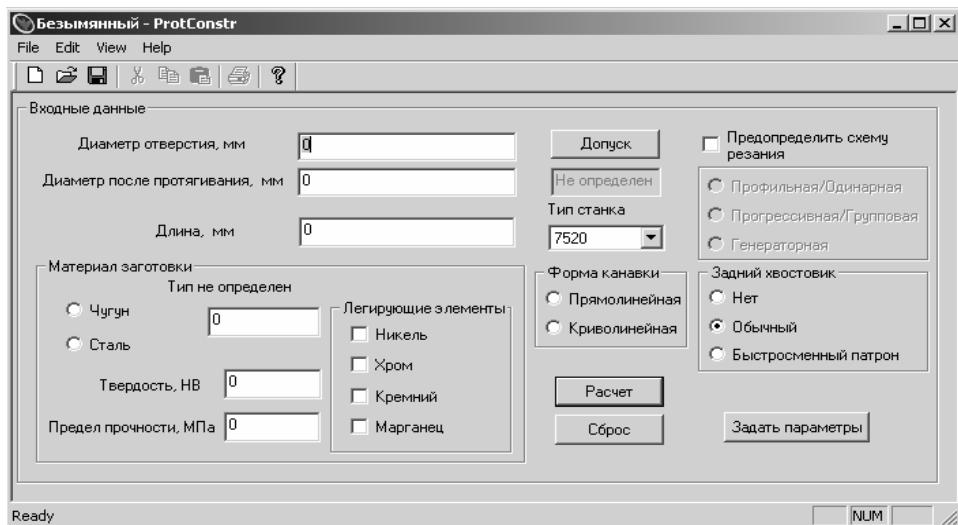


Рис. 1. Диалоговая панель ввода данных

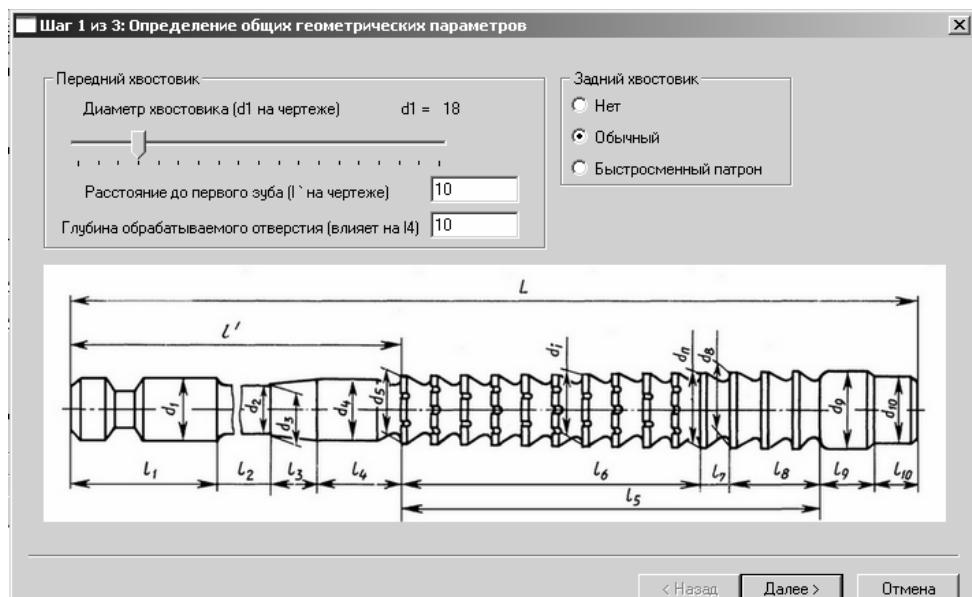


Рис. 2. Мастер задания параметров

Для автоматизации процесса проектирования протяжек на кафедре металлообрабатывающих станков и комплексов Мордовского государственного университета была разработана программа для расчета и конструирования круглых протяжек в среде Visual C++ 6.0.

Программа предназначена для проектирования и создания чертежей круглых протяжек для отверстий среднего диаметра (от 18 до 65 мм). При этом возможен расчет протяжек как с прямолинейной, так и с криволинейной формой канавки.

В процессе проектирования программа решает следующие задачи:

- 1) основная – расчет необходимых геометрических параметров протяжки;
- 2) вспомогательная – определение наиболее эффективной схемы резания;
- 3) выполнение рабочего чертежа.

Для удобства в приложении реализована возможность предопределения некоторых важных геометрических параметров по выбору пользователя.

Программа имеет достаточно простой и удобный интерфейс. Ввод всех необходимых данных реализован на диалоговой панели (рис. 1).

Кроме возможности расчета инструмента с заданием некоторых ключевых параметров обработки, есть вариант задания параметров протяжки «напрямую» с помощью специального мастера (рис. 2). Процесс проектирования инструмента этим способом включает в себя три этапа, каждый из которых отвечает за определенную часть протяжки.

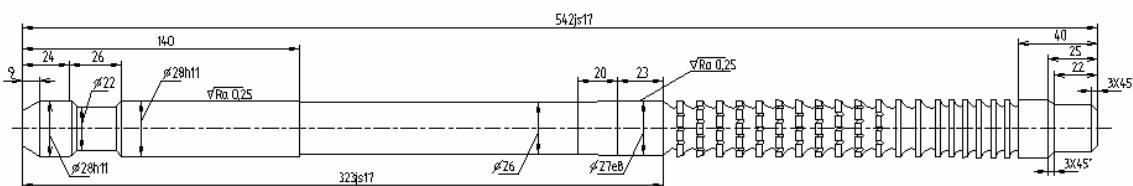


Рис. 3. Чертеж протяжки

Построение чертежа реализуется в среде T-FLEX CAD с помощью механизма OLE Automation. Доступ к функциям T-FLEX осуществляется через библиотеку Tfw32.tlb. В программе используется класс ITFSERVER для работы с приложением и класс ITFLEX – для работы с документом. Фрагмент рабочего чертежа представлен на рисунке (рис. 3).

Среда разработки проекта – Microsoft Visual C++ 6.0.

Характеристики программы. Объем exe-файла – 1,14 Мб.

УЧЕБНО-ИССЛЕДОВАТЕЛЬСКИЙ ПРОЕКТ «ПАРАМЕТРИЧЕСКАЯ ОПТИМИЗАЦИЯ» НА БАЗЕ Т-FLEX CAD И MFC

А.А. Федченко, А.А. Прохоров, М.В. Чугунов, И.Н. Полунина

*Рузаевский институт машиностроения
Мордовского госуниверситета им. Н.П. Огарева*

Учебно-исследовательский проект «Параметрическая оптимизация» реализован на основе методов и алгоритмов, разработанных на кафедре ТУиП ННГУ под руководством В.П. Малкова [1]. Проект представляет собой учебно-методический и исследовательский комплекс, предназначенный для изучения теоретических основ оптимизации, а также практических аспектов оптимального проектирования. Комплекс включает в себя следующие модули: инструментальные средства решения задач оптимизации в виде нелинейного математического программирования (exe-файлы), готовые к использованию в рамках лабораторного практикума; электронный учебник (скомпилированный html-файл); видеолекции (*.avi); базу данных учета самостоятельной работы студентов, модуль формирования моделей для решения «типовых» задач оптимизации (рис. 1).

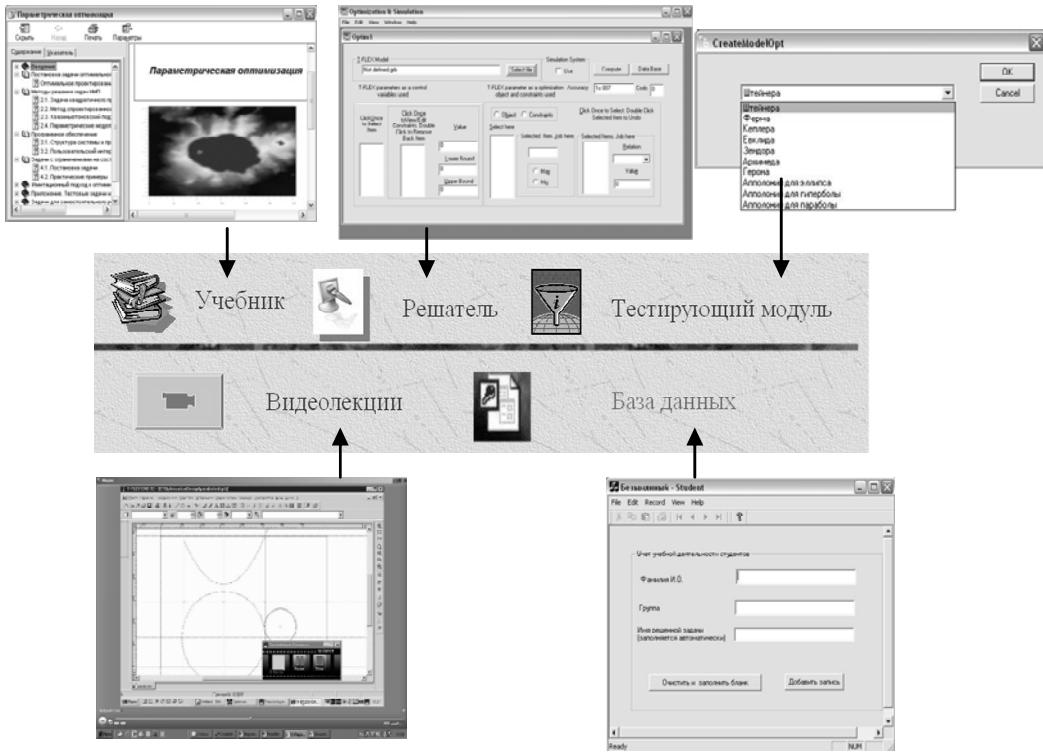


Рис. 1

Программное обеспечение реализовано в виде многодокументного приложения Windows с использованием архитектуры документ-вид на базе MFC и включает в себя: препроцессор, процессор (решатель), программный интерфейс, интегрирующий систему в среду T-FLEX CAD на основе использования механизмов OLE Automation. Исходные данные (функции оптимизации, начальная точка, параметры оптимизационного алгоритма), а также результаты решения представляют собой объекты сериализуемых классов и могут быть сохранены в соответствующих файлах [2]. Среда разработки Microsoft Visual C++.

Формирование модели оптимизационной задачи может быть осуществлено как в аналитической, так и в графической (геометрической) форме. При этом механизмы параметризации, поддерживаемые T-FLEX, позволяют в процессе оптимизации сохранять некоторые из геометрических свойств созданной модели (касание, принадлежность, параллельность, пересечение геометрических примитивов и т.д.), что дает возможность «избавиться» от некоторых ограничений в виде равенств и неравенств, а также сделать более наглядными постановку задачи и анализ результатов.

На базе API T-FLEX был разработан модуль автоматического формирования моделей оптимизационных задач, традиционно рассматриваемых в курсах оптимизации в качестве типовых (задачи Штейнера, Ферма, Кеплера, Евклида, Зендора, Архимеда, Герона, Аполлония для эллипса, Аполлония для гиперболы, Аполлония для параболы и др.).

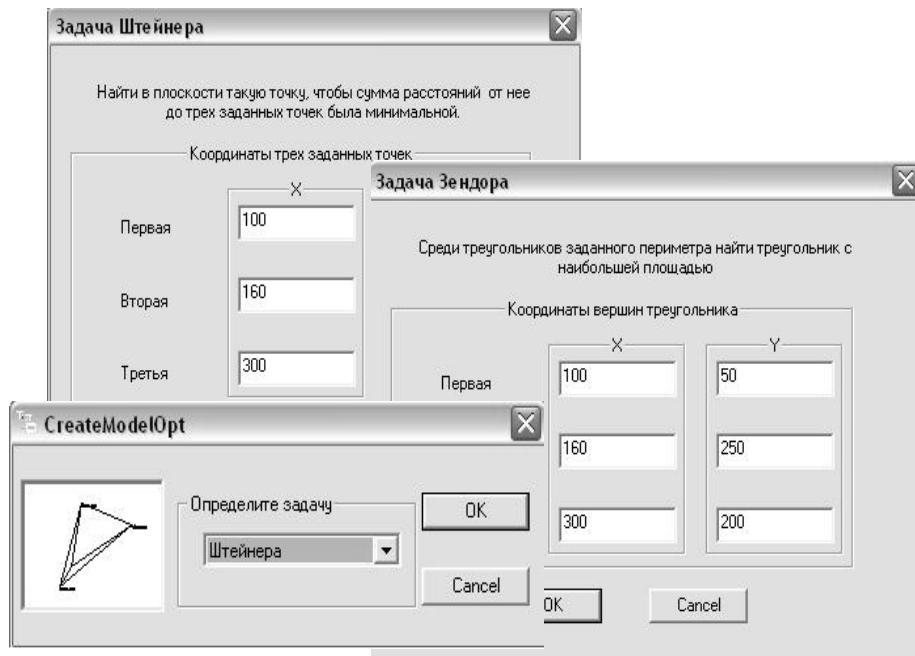


Рис. 2

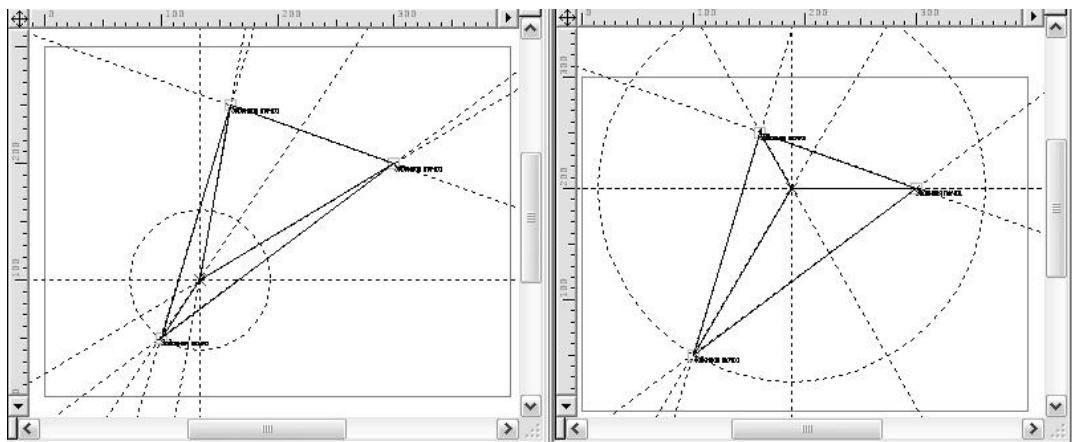


Рис. 3

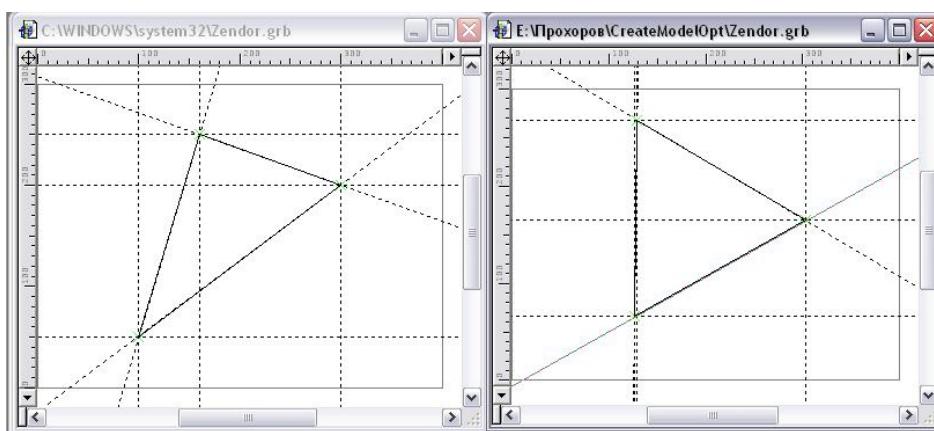


Рис. 4

В качестве примеров рассмотрим задачи Штейнера и Зендора (рис. 2). Пользователь выбирает из списка имя задачи, при этом на экран выводится диалоговая панель для определения исходных данных, а геометрическая модель отображается в специальном окне.

Исходный и оптимальный варианты для задач Штейнера и Зендора представлены на рис. 3 и 4 соответственно.

Если функции оптимизации являются алгоритмически заданными, то с целью снижения вычислительных затрат используется имитационный подход, позволяющий поэтапно заменять эти функции упрощенными аналитическими моделями на основе применения методов планирования эксперимента [1, 2].

При этом в окне имитационной системы отображается двумерное сечение пространства оптимизации, исходная точка, оптимальная на этапе точки, функции оптимизации, план эксперимента и подобласть поиска. На немодальной диалоговой панели отображаются координаты текущего положения курсора, а средства диалогового графического интерфейса позволяют деформировать и перемещать подобласть поиска на разных двумерных сечениях в направлении движения от исходной точки к оптимальной на этапе, а также перемещать точки плана эксперимента. Таким образом, рассматриваемый подход дает возможность создавать дополнительные параметрические связи между геометрической моделью оптимизируемого объекта и моделью оптимизационной задачи, а также выбирать проектные решения, изменяя положение текущей точки в пространстве проектирования, контролировать соответствующее значение целевой функции и степень удаленности проектного решения от предельного состояния.

Список литературы

1. Малков В. П., Торопов В. В., Филатов А. А. Имитационный подход к оптимизации деформируемых систем // Прикладные проблемы прочности и пластичности. Статика и динамика деформируемых систем: Всесоюз. межвуз. сб. – Горький, 1982. – С. 62–69.
2. Федченко А.А. Система параметрической оптимизации и имитационного моделирования. Технологии Microsoft в теории и практике программирования. Материалы конференции/ Под ред. проф. Р.Г. Стронгина. – Нижний Новгород: Изд-во Нижегородского университета, 2006. – С. 306–307.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА ФОРМИРОВАНИЯ ЛИЧНЫХ ДЕЛ АБИТУРИЕНТОВ

Н.И. Филандыш, В.В. Лаптев

Астраханский государственный технический университет

В высших учебных заведениях ежегодно производится зачисление абитуриентов. Во время данного процесса приемная комиссия принимает и оформляет большое количество разнообразных документов, каждый из которых оформляется по определенным правилам. Оператор приемной комиссии по мере поступления документов должен определять, какие документы следует оформлять далее в соответствии со сложными правилами приема. Правила приема могут изменяться каждый год.

В связи с большими объемами документации работа оператора приемной комиссии должна быть автоматизирована. Автоматизированная система должна вести полный

учет всех поступивших документов каждого абитуриента и обеспечивать поддержку принятия решений относительно дальнейшего оформления документов. Так как правила приема и состав необходимых документов могут меняться из года в год (например, в скором времени обязательным документом для приемной комиссии станут результаты ЕГЭ), система должна обеспечивать простоту модификации состава документов и правил их оформления.

Разработка программы, соответствующей указанным требованиям, была начата с проектирования модели используемых сущностей (рис. 1). Так как программа предназначена для оформления документов, центральной сущностью является документ. Каждый документ обладает некоторым содержанием, а также свойствами, наличие которых определяется содержанием документа. Однотипные документы объединяются в множества. Состав данных множеств и свойства документов используются в правилах оформления документа для определения того, в какой последовательности необходимо оформлять документы.

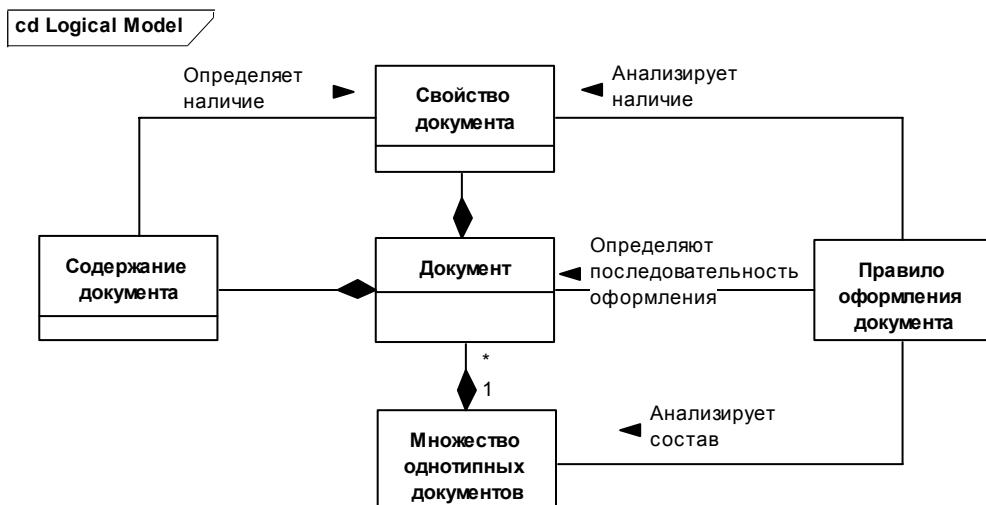


Рис. 1. Основные сущности системы

Правила, определяющие состав и порядок оформления документов, представлены набором логических выражений. Переменными в логических выражениях являются элементарные условия, такие как требование наличия документа в конкретном множестве или наличия определенного свойства документа.

Для реализации программы была выбрана клиент-серверная архитектура, так как она наилучшим образом обеспечивает совместную работу пользователей с единой базой данных. Серверная часть была реализована в виде базы данных Microsoft SQL Server.

В базе данных хранятся документы со своим содержанием, множество документов и свойства документов. При вводе в базу данных нового документа автоматически посредством триггеров происходит добавление данного документа в соответствующее множество и назначение ему свойств, соответствующих его содержанию.

Правила оформления документов также хранятся в базе данных в виде логических выражений, приведенных к дизъюнктивной нормальной форме. При этом отдельная таблица выделена для атомарных выражений вместе с возможными операторами отри-

зания и две другие таблицы – для входящих в выражение конъюнкт и дизъюнкт. Вычисление значений данных выражений происходит на сервере при помощи хранимых процедур Transact SQL. Вычисление осуществляется пошагово, от таблицы с атомарными выражениями до таблицы с операторами дизъюнкции, представляющими последние действия в выражении, приведенном к дизъюнктивной нормальной форме.

Клиентская часть была реализована на языке C# в среде Microsoft Visual Studio.NET. При этом была использована технология объектно-реляционного отображения с помощью библиотеки Gentle.NET. Клиентская часть разделена на две подсистемы: подсистему оформления документов и подсистему администрирования.

Интерфейс подсистемы оформления документов содержит дерево документов различных типов. Возле каждого типа документов отображается значок, сигнализирующий о возможности, невозможности или необходимости оформления документа. Данные сигналы обеспечивают оператору приемной комиссии поддержку принятия решений относительно выбора документа для оформления.

Также интерфейс подсистемы оформления документов содержит элемент управления Document Manager, позволяющий загружать документы с различными интерфейсами и предоставляющий доступ к любому из них в нужный момент времени. При открытии или создании нового документа интерфейс оформления данного документа загружается в данный элемент управления.

Интерфейс подсистемы администрирования содержит доступные для редактирования списки типов документов и для каждого типа документов – списки используемых свойств документов и правил оформления. Для редактирования правил представлен удобный интерфейс, позволяющий скомпоновать соответствующее логическое выражение из готовых элементов по методу Drag&Drop. После создания логическое выражение приводится к дизъюнктивной нормальной форме и сохраняется в базу данных.

Разработанное программное обеспечение автоматизирует работу операторов приемной комиссии по оформлению документов, входящих в состав личных дел абитуриентов. Определение документа как основной сущности в системе позволяет вводить новые типы документов, не затрагивая остальные части системы. Правила в виде логических выражений обеспечивают поддержку принятия решений. Использование Microsoft SQL Server не только обеспечило быстрый и надежный способ доступа к данным, но и позволило реализовать основные расчеты в виде триггеров и хранимых процедур, избавив от необходимости создавать подсистему для работы с базами данных. Использование для разработки клиента среды Microsoft Visual Studio.NET и разработанных для нее библиотек сторонних производителей позволило сократить время разработки и предоставить пользователям удобный интерфейс.

Список литературы

1. Закон Российской Федерации «Об образовании».
2. Закон Российской Федерации «О высшем и послевузовском профессиональном образовании».
3. Мамаев Е. Microsoft SQL Server 2000. Наиболее полное руководство. – БХВ-Петербург, 2001.
4. Троелсен Э. C# и платформа .NET. Библиотека программиста. – СПб.: Питер, 2002.
5. Фролов А. В., Фролов В. Г. Визуальное проектирование приложений C#. – Кудиц-Образ, 2003.

ВИРТУАЛЬНЫЙ ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО МЕХАНИКЕ

К.В. Фомин, Д.О. Голубинский

Астраханский государственный технический университет

Курс общей физики входит в программу обучения практически любой технической специальности. При изучении дисциплины требуется выполнять ряд лабораторных работ. В частности, при изучении раздела механики обычно выполняют лабораторные работы с различными механизмами, например маятниками. Однако в настоящее время в вузах (кстати, и в школах тоже) наблюдается значительный дефицит необходимых приборов: старые пришли в негодность, а новых практически не выпускают. В условиях тотального дефицита естественным решением проблемы выполнения лабораторных работ является моделирование лабораторных установок на компьютере. Такой путь имеет несколько достоинств: во-первых, позволяет решить собственно проблему необходимых приборов, так как для каждого вида лабораторных работ можно реализовать отдельную программу, которую предоставить каждому студенту; во-вторых, на модели становится возможным выполнять работы с гораздо большей точностью, чем в реальных условиях; в-третьих, моделирование позволяет осуществлять невозможные в реальности режимы выполнения, например приостановку эксперимента. Становится возможным наблюдать экстремальные и необычные физические условия: например, можно задать ускорение свободного падения как на Луне.

Таким образом, реализация лабораторных работ в виде обучающих программ представляется весьма перспективной. В Астраханском государственном техническом университете реализована часть лабораторных работ по разделу механики. Комплекс получил название «Виртуальная механика» и реально использовался на кафедре физики для выполнения лабораторных работ. Программа состоит из 6 модулей, 5 из которых предназначены для моделирования следующих лабораторных установок:

1. Машина Атвуда.
2. Маятник Обербека.
3. Пружинный маятник.
4. Физический маятник.
5. Крутильный маятник.

Последний модуль реализует интерфейс пользователя для выбора лабораторных работ и вызова справки. Модули, предназначенные для проведения лабораторных работ, обладают удобным интуитивно понятным интерфейсом. Каждый интерфейс представлен окном визуализации, панелью параметров системы, панелью отображения динамических параметров системы и управляющими кнопками.

Окно визуализации отображает графическое представление лабораторной установки. Оно является интерактивным и позволяет пользователю передвигать мышью подвижные части установки. Также оно позволяет наблюдать за изменением состояния системы с течением времени эксперимента. На рис. 1 показано окно интерфейса для проведения лабораторных работ с пружинным маятником.

Панель параметров системы отображает предопределённые параметры установки, а также позволяет задавать конкретные значения параметров в допустимых пределах. С ее помощью устанавливаются различные режимы работы системы, предназначенные для различных лабораторных работ, проводимых с использованием данной установки.

Панель отображения динамических параметров отображает текущее значение параметров во время эксперимента.

Интерфейс содержит следующие управляющие кнопки: пауза, сброс, справка, выбор лабораторной работы. Кнопка «Пауза» позволяет приостановить течение эксперимента с последующей возможностью возобновления его хода. Кнопка «Сброс» останавливает эксперимент, что позволяет задать новые значения параметров системы. Кнопка «Справка» вызывает справку, содержащую методические указания по использованию

программного продукта и проведению эксперимента, теоретический материал по лабораторным работам. Кнопка «Выбор» лабораторной позволяет перейти к главному меню программы.

Главное меню позволяет выбирать лабораторную и вызывать общую справку.

В качестве платформы реализации была выбрана система Visual Studio.NET – новая революционная платформа, созданная компанией Microsoft. Программный продукт был реализован в среде Microsoft Visual Studio 2005, значительно упрощающей жизнь программиста. Данная среда выбрана, потому что она обладает интуитивно понятным интерфейсом и широким кругом возможностей.

Текстовый редактор весьма удобен и функционален. Он может интеллектуально обнаруживать ошибки и подсказывать в процессе ввода, какой именно

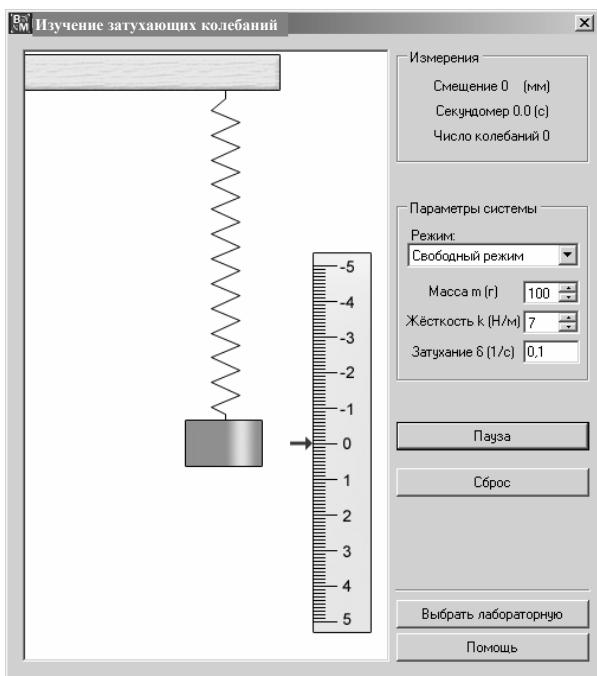


Рис. 1. Окно для выполнения лабораторных работ с пружинным маятником

код необходим. Приложение использует Windows Forms, для создания которых Visual Studio предоставляет весьма удобные и эффективные инструменты визуальной разработки, позволяющие создавать приложения путем простого перетаскивания мышью. Visual Studio автоматически выполняет все шаги, необходимые для компиляции исходного кода.

В состав системы входит несколько вспомогательных программ, которые позволяют автоматизировать выполнение наиболее распространенных задач; причем многие из этих программ могут добавлять необходимый код в уже существующие файлы, так что программисту не придется беспокоиться (а в некоторых случаях и вообще не вспоминать) о соблюдении синтаксических правил.

Visual Studio обеспечивает использование совершенных методов отладки при разработке проектов: например, пошаговое выполнение кода, когда выполняется один оператор за раз, что дает возможность следить за текущим состоянием приложения.

Распространять приложения в VS столь же просто, как и писать их: VS облегчает передачу кода клиентам и позволяет им инсталлировать его без каких-либо проблем.

VS имеет большое количество мощных инструментов, благодаря которым можно просматривать отдельные элементы проекта или осуществлять в них поиск, независимо от того, являются ли эти элементы файлами с кодами на языке C# или представляют собой какие-либо иные ресурсы, например двоичные графические или звуковые файлы.

Многие типы проектов, создание которых возможно на C#, могут разрабатываться на основе «каркасного кода», заранее включаемого в программу. Вместо того чтобы каждый раз начинать с нуля, VS позволяет использовать уже имеющиеся файлы с исходным кодом, что уменьшает временные затраты на создание проекта.

Для развёртывания приложения был создан установочный пакет. Для его разработки был использован Windows Installer, интегрированный в Microsoft Visual Studio, что значительно упростило процесс его создания.

Программный продукт был реализован на языке C#, специально разработанном для .NET. Это современный, хорошо продуманный язык, учитывающий ряд ошибок, которые присутствуют в существующих языках программирования.

Для создания графики был использован интерфейс GDI+. Благодаря этому интерфейсу написание графических приложений становится лёгким и увлекательным занятием.

Для создания элементов графического интерфейса был использован редактор растровой графики Adobe Photoshop 8.0 CS, позволяющий создавать графические объекты любой сложности.

ПРОБЛЕМЫ АРХИТЕКТУРЫ СОВРЕМЕННЫХ КОМПЬЮТЕРНЫХ СИСТЕМ ПОДДЕРЖКИ ЖИЗНЕННОГО ЦИКЛА ИЗДЕЛИЙ И ПУТИ ИХ РЕШЕНИЯ

В.И. Швецов, Я.В. Ефремов

Нижегородский госуниверситет им. Н.И. Лобачевского

Обзор существующих решений

Последние годы активно развиваются системы информационной поддержки жизненного цикла изделий (ИПИ-системы, CALS). Их развитие обусловлено острой необходимостью автоматизации процессов разработки и производства изделий на крупных машиностроительных предприятиях. Однако реализации многих современных программных комплексов, составляющих CALS-систему различных типов (CAD/CAM, PLM, PDM, ERP и т.д.), имеют ряд недостатков. К ним можно отнести как архитектуру всего программного комплекса в целом, так и ряд проблем в реализации. В этой статье мы обозначим некоторые недостатки архитектуры современных программных систем и наметим пути их устранения и повышения эффективности.

На сегодняшний день практически любой крупный программный комплекс, в котором необходима реализация прикладной логики и хранение больших объемов данных, построен на базе классической многоуровневой клиент-серверной архитектуры.

Серверная часть состоит из так называемой платформы, метаданных и прикладных данных. Под платформой мы понимаем серверный программный компонент (или компоненты), отвечающий за доступ к прикладным данным, обработку метаданных и работу прикладной логики (исполнение прикладного кода). Под метаданными мы понимаем описание структуры прикладных данных и программного кода, реализующего прикладную логику. То есть платформу можно представить как виртуальную машину, исполняющую прикладной код и обеспечивающую высокоуровневый доступ этого кода к прикладным данным.

Плюсы такой архитектуры платформы для решения прикладных задач хорошо известны – это абстрагирование от низкоуровневой реализации системных функций и деталей доступа к данным СУБД. Таким образом, мы получаем независимость прикладного кода и данных от операционной системы и конкретной СУБД. Однако у такой архитектуры есть и минусы: медленная скорость выполнения прикладного кода, невозможность получить прямой доступ к функциям ядра платформы, ограниченные возможности расширения базовых конструкций платформы, доступных прикладному сценарию. Также необходимо отметить, что прикладные сценарии, разработанные для разных платформ, несовместимы или ограниченно совместимы, что снижает эффективность повторного использования кода. Многие системы вообще поддерживают только собственный язык сценариев.

Очевидно, что использование собственного встроенного языка программирования не только замедляет процесс обучения программистов, но и делает невозможным использование имеющихся наработок. Справедливости ради стоит заметить, что многие встроенные в платформы языки сценариев имеют возможность взаимодействовать с исполняемым кодом, написанным на другом языке программирования, используя технологии динамических библиотек, COM и др., однако скорость такого взаимодействия и ограниченность возможностей не позволяет считать это приемлемым решением.

Многие системы используют рассмотренную выше архитектуру, несмотря на разную область применения (например, 1С, BAAN ERP и др.), и используют собственные встроенные языки программирования для разработки прикладной логики. Конечно, эти языки упрощают разработку в области применения этим программных комплексов (например, обилие финансовых функций или возможностей по созданию отчетов), однако требуют изучения их синтаксиса, а также методов работы с редактором, отладчиком и т.д. В программном комплексе BAAN ERP вообще два встроенных языка: 3GL и 4GL. 3GL – процедурно-ориентированный язык, не имеющий средств взаимодействия с пользовательским интерфейсом и поэтому имеющий ограниченную область применения. Сценарии на языке 4GL являются событийно-ориентированными. Они всегда связаны с некоторым сеансом, за которым закреплено несколько экранных форм. Сценарий делится на секции, отвечающие за обработку различных событий. Такой метод разработки потребует привыкания программиста, работавшего с другими языками, например C++. Оба эти языка существенно ограничены как в функционале, так и возможностях среды разработки. Методы разработки с использованием CASE-средств вообще практически неприменимы.

Выделение проблем и постановка задачи

Таким образом, можно отметить проблемы, связанные с разработкой прикладной логики: слабый функционал встроенных языков сценариев, практическая невозможность повторного использования кода, необходимость обучения новому языку и методам разработки. Эти проблемы присутствуют во многих современных программных комплексах, входящих в систему поддержки жизненного цикла изделий.

Идеальным решением этих проблем была бы возможность использовать в CALS-системах любой язык программирования (или хотя бы набор широко известных и распространенных) на выбор разработчика. Отметим, что поддержка сценариев необходима не только на уровне взаимодействия прикладной логики с данными, но и при доступе сценария к ядру платформы программного комплекса.

Можно обозначить некоторые подходы к решению поставленной задачи. Добавим некоторые дополнительные условия, позволяющие конкретизировать задачу. Необходимо разработать гибкую объектную модель самой платформы, позволяющую расширять ее в любых направлениях (например, добавление новых типов данных, базовых объектов, провайдеров доступа к данным различных СУБД, модулей коммуникации, шифрования и т.п.). Также эта платформа должна обеспечивать прикладному коду простой доступ к данным, скрывая сложности низкоуровневой работы с данными, поскольку работа с большими объемами информации особенно характерна для ИПИ-систем.

Самым часто используемым подходом предоставить возможность разработки на нескольких языках программирования и повторного использования кода являются динамически подключаемые библиотеки и технология COM. В данной статье мы не будем подробно останавливаться на этом методе, т.к. он хорошо изучен и широко используется.

Еще одной возможностью предоставить возможность разработки на нескольких языках программирования является технология WSH – Windows Scripting Host. По сути это сервер сценариев (виртуальная машина), предоставляющий возможность подключения к другому приложению, в нашем случае к разрабатываемой платформе. Однако у WSH есть несколько ограничений, уменьшающих ее практическую ценность для наших задач. Во-первых, WSH – это интерпретатор кода, а значит, скорость работы прикладного кода будет невелика по сравнению с машинным кодом. Во-вторых, эта технология зависит от операционной системы, т. к. является компонентой Microsoft Windows. В-третьих, предоставить доступ к функциям ядра платформы достаточно сложно, т.к. для этого необходимо разработать прокси-функции и прокси-классы для взаимодействия с соответствующими функциями и классами ядра. К плюсам Windows Scripting Host можно отнести относительную простоту использования, а также возможность разработки прикладного кода на любом языке, поддерживаемом сервером сценариев.

Наиболее перспективными на сегодняшний день являются технологии Microsoft .NET и основанные на языке Java технологии (J2EE и др.). Существенным положительным отличием этих технологий является независимость от операционной системы. Как компиляторы, так и виртуальные машины Java существуют практически для всех распространенных операционных систем. Поддержка технологии .NET существует не только в Windows-системах, но в некоторых Unix-подобных системах, а поскольку эта технология основана на открытых стандартах, то может быть расширена сторонними производителями.

Решение

В качестве основной программной платформы для разработки ИПИ-системы, лишенной некоторых вышеперечисленных недостатков, выберем Microsoft.NET Framework. К преимуществам этого программного каркаса можно отнести встроенную поддержку во всех современных операционных системах от Microsoft. Это важно, т.к. именно эта ОС используется большинством пользователей как в России, так и во всем мире. К тому же эта технология является платформонезависимой и открытой для расширения разработчиками других операционных систем или компиляторов.

Также необходимо отметить широкие возможности, которые дает технология отражения (reflection). При компоновке сборок или модуля компилятор .NET Framework создает таблицы определений типов, полей, методов и т.д. По сути, отражение – это технология

синтаксического разбора этих таблиц. То есть объекты, реализующие отражение, реализуют некоторую модель для доступа к метаданным сборки или модуля. Обычно отражение используется в библиотеках классов, которым нужно понять определение некоторого типа, чтобы дополнить его (например, механизм сериализации). Отражение используется и когда во время выполнения приложению нужно загрузить определенный тип из некоторой сборки. Например, для решения нашей задачи можно загружать в платформу объекты, разработанные сторонними производителями, для расширения возможностей ядра системы. Также можно загружать и исполнять прикладной код.

Рассмотрим архитектуру программного комплекса. Система состоит из клиента, сервера бизнес-логики и сервера баз данных. Тонкий клиент, под которым часто понимают веб-браузер, имеет как известные преимущества, так и недостатки. Основной недостаток заключается в том, что с таким клиентом практически невозможно реализовать модель портфеля (briefcase model), т.е. режим работы с данными, при отсутствии подключения к серверу. Таким образом, для реализации модели портфеля, а также для применения других функций, таких как шифрование, передача данных в бинарном (т.е. более компактном) формате и т.п., клиент будет реализован в виде приложения с графическим интерфейсом Windows Forms. Сервер бизнес-логики реализуем в качестве службы Windows, т.к. служба может автоматически запускаться при загрузке компьютера, а также есть возможность назначить пользователя, с правами которого будет выполняться процесс. В качестве хранилища данных мы выберем реляционную СУБД, т.к. данные, которые мы предполагаем хранить, достаточно просто представить в нормальной форме. Благодаря технологии ADO.NET мы можем абстрагироваться от деталей подключения и работы с различными СУБД. Однако для разработки и тестирования системы мы выберем Microsoft SQL Server, т.к. в .NET Framework существует специализированный провайдер доступа к данным, использующий низкоуровневый доступ к SQL Server, что положительным образом сказывается на производительности.

В качестве технологии обмена данными между клиентом и сервером выберем .NET Remoting. .NET Remoting обеспечивает сильную связность клиента и сервера, поскольку ими разделяются одни и те же типы объектов. .NET Remoting переносит функциональность объектов CLR на методы, вызываемые между разными доменами приложений. К достоинствам этой технологии относится гибкость архитектуры, которая может быть использована с приложениями любого типа через любые транспортные протоколы, используя любое шифрование. .NET Remoting поддерживает удаленный доступ к объектам с клиентской или серверной активацией, управление временем жизни удаленных объектов, передачу по ссылке и по значению.

Серверную часть программного комплекса можно разделить на платформу и конфигурацию. Платформа состоит из инфраструктурной части, отвечающей за обслуживание клиентских подключений и прочих системных функций, и набора компонент. Каждый компонент должен реализовывать функциональность для создания различных экземпляров данного типа, для работы с данными этих экземпляров, находящимися в СУБД, для отображения и изменения пользовательских данных. С точки зрения пользователя он работает с совокупностью объектов различных типов (например, справочники, журналы, отчеты и т.п.), которые составляют конфигурацию. Таким образом, конфигурация – это набор метаданных, описывающих входящие в нее экземпляры объектов различных типов, предопределенных базовыми компонентами, и программный код, реализующий прикладную логику. Например, некоторый компонент может содержать тип «справочник», представляемый простой таблицей в базе данных. Т.к. различные справочники могут иметь множество реквизитов (отображаемых на поля таблицы в базе

данных), каждый экземпляр некоторого типа описывается своими метаданными и может содержать некоторые пользовательские данные.

Чтобы обеспечить функциональность для расширения платформы, нужно разработать ряд открытых интерфейсов для добавления компонент сторонних разработчиков. Так как в качестве программного каркаса был выбран .NET Framework, сторонние разработчики могут использовать любые языки программирования, для которых существуют .NET-компиляторы. Чтобы взаимодействовать с платформой, каждый компонент должен корректно регистрироваться, т.е. создавать метаданные в понятном формате, а также поддерживать передачу данных с помощью .NET Remoting и иметь формы пользовательского интерфейса. Формы пользовательского интерфейса нужны для их передачи клиентскому приложению для отображения и редактирования данных. При использовании различных компонентов появится возможность разрабатывать различные конфигурации, создавая экземпляры типов различных компонентов и программный код, решающие конкретные прикладные задачи.

При разработке конфигурации, как и при разработке компонент платформы, программист сможет выбрать любой язык программирования, для которого существует .NET-компилятор. Таким образом, он сможет использовать имеющиеся наработки либо библиотеки, разработанные другими разработчиками. Основным плюсом использования .NET Framework для разработки прикладной бизнес-логики является то, что код, написанный для .NET будет работать гораздо быстрее интерпретаторов типа Windows Scripting Host и т.п.

Мы наметили решение некоторых проблем в архитектуре и реализации современных сложных компьютерных систем информационной поддержки жизненного цикла изделий.

КОМБИНИРОВАННЫЙ АЛГОРИТМ СИНТЕЗА СТРАТЕГИЙ ДВУХРЕЙСОВОГО ОБСЛУЖИВАНИЯ СТАЦИОНАРНЫХ ОБЪЕКТОВ В РАЗВЕТВЛЕННОЙ РАБОЧЕЙ ЗОНЕ

А.Ю. Шлюгаев, Ю.С. Федосенко

Волжская госакадемия водного транспорта

Экономические условия эксплуатации ресурсов крупномасштабных грузообразующих районов водного транспорта (КГР), в которых плавучими дизель-электрическими комплексами (ПДК) осуществляется массовая русловая добыча нерудных строительных материалов (НСМ), предъявляют повышенные требования к качеству оперативного управления транспортно-технологическими процессами. В разные периоды навигации в КГР работает от нескольких единиц до нескольких десятков единиц ПДК. Русловая топология района представляет собой существенно протяженную главную магистраль (200–400 км) с боковыми ответвлениями, простирающимися на относительно небольшие расстояния (1–30 км). Дислокация добывающих комплексов в КГР зависит как от географии русловых месторождений НСМ, так и от технических характеристик добывающих комплексов: на главной, глубоководной магистрали располагаются существенно более мощные ПДК по сравнению с комплексами, которые работают

на боковых, относительно мелководных ответвлениях. На рис. 1 приведен пример русловой топологии фрагмента КГР, на котором главная магистраль и расположенные на ней пиктограммы ПДК выделены более темным тоном по сравнению с боковыми ответвлениями и расположенными на них добывающими комплексами.

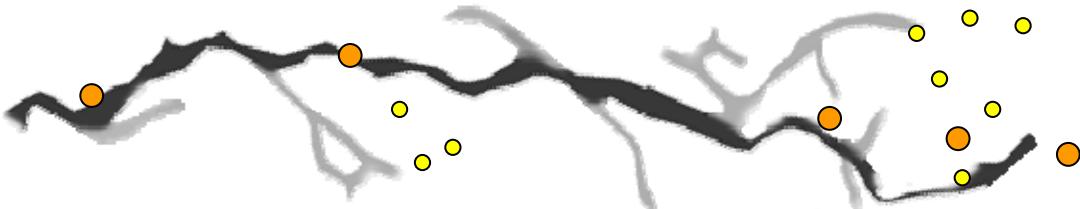


Рис. 1

Ежедневная суммарная потребность парка ПДК в дизельном топливе, доставляемом закрепленным за КГР специализированным танкером-заправщиком, составляет 100–500 тонн. Снабжение ПДК топливом производится в процессе перемещения заправщика вдоль главной магистрали в прямом и обратном направлениях; при этом осуществляются заходы танкера и на боковые ответвления для бункеровки расположенных на них добывающих комплексов.

Одним из актуальных направлений повышения эффективности функционирования КГР является формирование стратегий оперативного управления процессами снабжения ПДК дизельным топливом на базе адекватных математических моделей и их реализация средствами современных информационных технологий.

В данной работе построение математической модели и её исследование выполнено в рамках дискретного формализма. Выбору именно такого подхода способствовали как результаты анализа особенностей процессов снабжения ПДК в КГР, так и определенный опыт, накопленный авторами при изучении транспортно-технологических процессов на внутреннем водном транспорте.

Итак, имеется n -элементная группа независимых стационарных объектов $O_n = \{o(1), o(2), \dots, o(n)\}$, расположенных в соответствующих вершинах i ($i = \overline{1, n}$) неориентированного односвязного графа G . Все объекты группы O_n подлежат однократному однофазному обслуживанию автономно перемещающимся *mobile*-процессором P , для которого граф G является рабочей зоной. Пример графового представления рабочей зоны для $n = 15$ представлен на рис. 2.

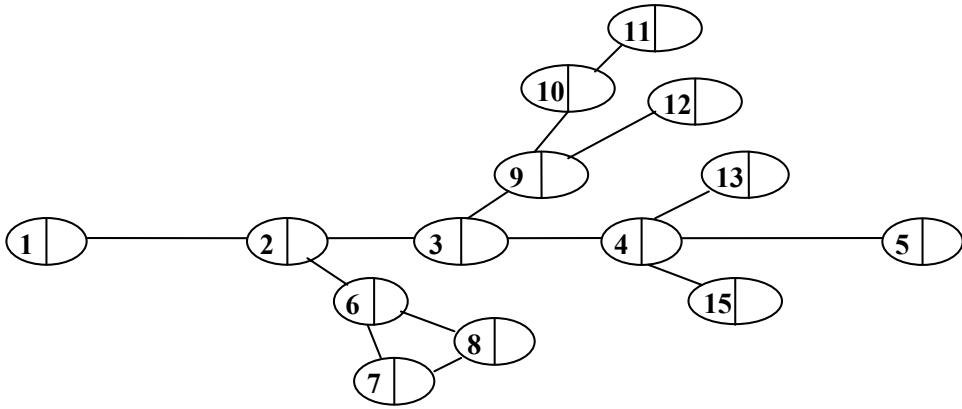


Рис. 2

Относительно графа G известно, что он может быть разбит на m непересекающихся подграфов (ветвей) G_1, G_2, \dots, G_m ($m \leq n$), которым соответственно принадлежат вершины $1, 2, \dots, m$, так, что каждые две соседние ветви G_y и G_{y+1} соединены ребром $(y, y+1)$ графа G ($y = \overline{1, m-1}$). Будем говорить, что последовательность ребер $(1, 2), (2, 3), \dots, (m-1, m)$ и множество инцидентных им вершин образуют главную магистраль M графа G ; при этом ветви G_j исходят из соответствующих узловых вершин j главной магистрали ($j = \overline{1, m}$). Предполагается, что на структуру графа G налагается следующее ограничение: ветви графа G могут содержать циклы, однако последние не могут включать в себя вершины разных ветвей и ребра главной магистрали. В частном случае ветвь может состоять из единственной вершины.

Обозначим через B_j множество вершин ветви (подграфа) G_j . Под *обслуживанием ветви* G_j понимается обслуживание всех объектов, расположенных в вершинах множества B_j , начинающееся с объекта $o(j)$.

Соответственно принятой на практике технологии обслуживание объектов группы O_n осуществляется в процессе реализации двух рейсов – прямого и обратного: вначале процессор P перемещается от стартовой вершины 1 к вершине n , выполняя в некоторой очередности обслуживание части ветвей (движение в прямом рейсе λ_+), а затем возвращается к стартовой точке (движение в обратном рейсе λ_-), завершая обслуживание оставшихся ветвей аналогичным образом.

Обслуживание $o(i)$ осуществимо начиная с момента $t(i)$ ($i = \overline{1, n}$) готовности к обслуживанию. По каждому объекту заданы функции: $\tau(i, t)$ – продолжительность обслуживания, если оно начинается в момент времени t , и $\varphi(i, t)$ – индивидуальный штраф за обслуживание, завершающееся в момент времени t ($i = \overline{1, n}$). Обе введенные функции являются монотонно возрастающими (в нестрогом смысле) по второму аргументу. Считается известной матрица $W = \{w(i, j)\}$ продолжительностей перемещений процессора P из вершины i в j , если они смежные.

Стратегию обслуживания процессором P группы объектов O_n определяем как набор $H = (V, \rho_{B1}, \rho_{B2}, \dots, \rho_{Bm})$, в котором приняты следующие обозначения.

Под $V = \{j^+_1, \dots, j^+_{s^+}\}$ понимается совокупность номеров ветвей, которые обслуживаются в прямом рейсе λ_+ ; при этом ветвь $G(j^+_k)$ в прямом рейсе обслуживается k -й по очереди ($j^+_k \in \{1, \dots, m\}, k = \overline{1, s^+}, s^+ \leq m$). Остальные ветви ($\{1, \dots, m\} \setminus V = \{j^-_1, \dots, j^-_{s^-}\}$) обслуживаются в рейсе λ_- . Для определенности считаем, что ветвь $G(m)$ обслуживается в прямом рейсе, т.е. $m \in V$.

Через $\rho_{B_j} = (i_1(B_j), \dots, i_{|B_j|}(B_j))$ обозначена перестановка номеров объектов ветви B_j , определяющая расписание ее обслуживания ($i_{z_j}(B_j) \in B_j, z_j = \overline{1, |B_j|}, i(B_j, 1) = j, j = \overline{1, m}$).

Для объекта $o(i)$ ($i = \overline{1, n}$) считаем $t_{\text{нач}}(H, i)$ и $t^*(H, i)$ соответственно моментами начала и завершения обслуживания при реализации стратегии H . Полагаем, что все допустимые реализации стратегии H компактны, т.е. имеют место следующие связи:

$$\begin{aligned} t_{\text{нач}}(H, i_1(B_{j^+_1})) &= \max \{l(1, i_1(B_{j^+_1})); t(i_1(B_{j^+_1}))\}; \\ t^*(H, i_k) &= t_{\text{нач}}(H, i_k) + \tau(i_k, t_{\text{нач}}(H, i_k)), k = \overline{1, n}; \\ t_{\text{нач}}(H, i_{z_j}(B_j)) &= \max \{t^*(H, i_{zj-1}(B_j)) + l(i_{zj-1}(B_j), i_{zj}(B_j)); t(i_{zj}(B_j))\}; z_j = \\ &\quad 2, \dots, |B_j|, j = \overline{1, m}; \\ t_{\text{нач}}(H, i_1(B_{j^+_k})) &= \max \{t^*(H, \bar{i}(B_{j^+_{k-1}})) + l(\bar{i}(B_{j^+_{k-1}}), i_1(B_{j^+_k})); \\ &\quad t(i_1(B_{j^+_k}))\}, j^+_k \in \{1, \dots, m\}, k = \overline{2, s^+}; \\ t_{\text{нач}}(H, i_1(B_{j^-_1})) &= \max \{t^*(H, \bar{i}(B_m)) + l(\bar{i}(B_m), i_1(B_{j^-_1})); t(i_1(B_{j^-_1}))\}, \\ t_{\text{нач}}(H, i_1(B_{j^-_k})) &= \max \{t^*(H, \bar{i}(B_{j^-_{k-1}})) + l(\bar{i}(B_{j^-_{k-1}}), i_1(B_{j^-_k})); t(i_1(B_{j^-_k}))\}, \\ &\quad j^-_k \in \{1, \dots, m\}, k = \overline{2, s^-}. \end{aligned}$$

В приведенных соотношениях $\bar{i}(B_j) = i_{|B_j|}(B_j)$ означает номер последнего обслуживаемого объекта ветви B_j , а через $l(i, j)$ обозначены элементы матрицы $L = \{l(i, j)\}$, получаемой из W путем применения алгоритма Флойда [1].

Задача заключается в отыскании стратегии обслуживания, минимизирующей величину суммарного штрафа по всем объектам, т.е. $\min_{H \in H^0} Q(H)$, где

$$Q(H) = \sum_{i=1}^n \varphi(i, t^*(i, H)).$$

В работе предлагается алгоритм Λ_k синтеза оптимальной стратегии обслуживания группы O_n , активно учитывающий в рамках рассматриваемой модели специфику русской топологии КГР и расстановки ПДК. Алгоритм Λ_k является комбинированным, сочетающим в себе процедуры синтеза оптимальных стратегий обслуживания линейно распределенной [2, 3] и пространственно рассредоточенной [4] групп объектов, и состоит из двух этапов.

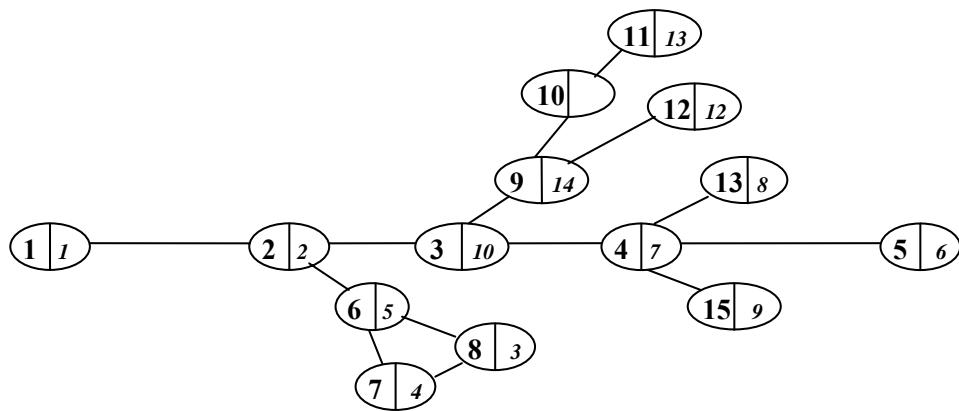


Рис. 3

На первом этапе работы алгоритма Λ_k из рассмотрения исключаются объекты, не принадлежащие главной магистрали M , и к оставшейся совокупности применяется алгоритм синтеза оптимальной стратегии обслуживания линейно рассредоточенной группы объектов. Таким образом определяется, какие из объектов главной магистрали обслуживаются в рейсе λ_+ , а какие – в рейсе λ_- , т.е. определяется совокупность V , соответствующая оптимальной стратегии обслуживания группы O_n .

На втором этапе определяются оптимальные последовательности обслуживания объектов ветвей, не принадлежащих главной магистрали: последовательно, в порядке обслуживания узловых объектов V , полученном на первом этапе, для каждой ветви применяется алгоритм синтеза расписания обслуживания пространственно рассредоточенной группы объектов. Результатом выполнения второго этапа являются перестановки $\rho_{B1}, \rho_{B2}, \dots, \rho_{Bm}$ в оптимальной стратегии обслуживания H .

Для алгоритма Λ_k выполнена программная реализация на языке C#, с помощью которой были проведены вычислительные эксперименты для типовых значений параметров КГР: $\tau_i \in [0,5; 1]$; $w_{ij} \in [15, 20]$ – соответствует объектам главной магистрали и $w_{ij} \in [0,1; 3]$ для объектов, расположенных на боковых ответвлениях.

На рис. 3 для графовой модели и вышеуказанных типовых значений параметров приведен пример оптимальной стратегии обслуживания объектов, синтезированной путем реализации алгоритма Λ_k : порядковые номера обслуживания объектов обозначены курсивом в правых секторах вершин графовой модели.

Список литературы

1. Асанов М. О. Дискретная математика. Графы, матроиды, алгоритмы / М.О. Асанов, В.А. Баранский, В.В. Расин. – Ижевск: НИЦ «Регулярная и хаотическая динамика», 2001.
2. Коган Д. И. Задачи оптимального обслуживания группы стационарных объектов, расположенных в одномерной зоне / Д.И. Коган, А.В. Синий, Ю.С. Федосенко // Вестник ВГАУТ. Межвузовская серия: Моделирование и оптимизация сложных систем. – Н. Новгород: Изд-во ФГОУ ВПО ВГАУТ, 2004. Вып. 9. – С. 27–34.
3. Синий А. В. Математические модели и алгоритмы синтеза оптимальных расписаний бункеровки добывающих комплексов в крупномасштабных русловых районах / А.В. Синий, Ю.С. Федосенко // Технологии Microsoft в теории и практике программирования: сб. материалов

конференции, Н. Новгород, 21–22 марта 2006 г. – Н. Новгород: Изд-во Нижегородского госуниверситета, 2006. – С. 276–279.

4. Федосенко Ю.С. Общая задача однопроцессорного обслуживания пространственно распределенной группы стационарных объектов / Ю.С. Федосенко, А.Ю. Шлюгаев // Математическое моделирование и оптимальное управление: Вестник ННГУ. – 2007 (в печати).

ИССЛЕДОВАНИЕ ИНДИКАТОРОВ ТЕХНИЧЕСКОГО АНАЛИЗА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПРОГНОЗИРОВАНИЯ ВРЕМЕННЫХ РЯДОВ

Д.С. Шумков, Е.С. Кузнецов, И.Г. Сидоркина

Марийский государственный технический университет

На сегодняшний момент задачами анализа временных рядов занимаются исследователи из разных областей, используя различные алгоритмы, методы и технологии. Одна из основных проблем, возникающих при работе над этим направлением, – получить качественный прогноз будущих значений временного ряда, основываясь на его предыдущих данных. Эта задача является актуальной при прогнозировании цен акций и котировок валют, потому что динамику изменения цены во времени можно представить в виде временного ряда, что позволяет применить все известные методы для его анализа. Одной из технологий, которая применяется при выявлении закономерностей в изменении цен акций и котировок валют, является технический анализ. В его основе лежат методы прикладной статистики, основанные на скользящих средних и осцилляторах.

Целью данной работы является исследование технических индикаторов для создания «оптимального» прогноза различных временных рядов. Другими словами, ставится задача прогнозирования временных рядов разной природы, а не только котировок акций и валют, с помощью индикаторов технического анализа.

Ниже представлен предложенный алгоритм решения задачи прогнозирования на основе технических индикаторов:

1. Выбор нескольких технических индикаторов.
2. Расчет их нормализованных значений.
3. Анализ полученных данных с целью выявления закономерностей в числовом ряду.
4. Создание таблицы правил.

Рассмотрим предложенный алгоритм прогнозирования подробнее. На первом этапе необходимо определиться с набором технических индикаторов, которые предполагается использовать для анализа. Для каждой задачи наиболее оптимальный набор индикаторов может отличаться, тем не менее следует постараться выбрать индикаторы различного типа, например, взвешенное скользящее среднее, ленту Джона Боллинджера, индекс относительной силы (RSI) и стохастический осциллятор [1].

Второй этап включает расчет значений каждого индикатора и его последующую нормализацию, подразумевающую, что значения могут изменяться в некотором определенном интервале, например от 0 до 100 или от –1 до 1. Нормализация является необходимым требованием, потому что позволяет сделать исследуемый процесс стационарным. Значения

некоторых индикаторов, например индекс относительной силы, могут с самого начала изменяться в диапазоне от 0 до 100, поэтому в нормализации не нуждаются.

В расчете многих индикаторов технического анализа присутствует значение периода. В математической статистике под этим понятием подразумевают «лаг», или временной сдвиг. Поэтому для получения более качественных результатов расчет значений каждого индикатора необходимо проводить с различным числом периодов, например, для вычисления взвешенного скользящего среднего воспользоваться интервалами 3, 5, 11 и т. д. Это позволит выявить различные связи элементов ряда.

Следующим шагом анализа является поиск закономерностей во временном ряде, заключающийся в том, что необходимо обнаружить, при каких нормализованных значениях индикатора значения временного ряда ведут себя предсказуемо. Например, при значении «80» первого индикатора и «10» второго индикатора следующее значение временного ряда всегда больше предыдущего, а при «90» и «5» – больше не менее чем на 5%. Этот шаг является наиболее трудоемким и долгим потому, что приходится иметь дело с большим объемом данных. Для того чтобы упростить эту задачу, необходимо постараться обнаружить все возможные закономерности для каждого индикатора отдельно. Сохранить их в отдельном месте и только после этого проводить поиск закономерностей для нескольких индикаторов вместе. Это позволит значительно снизить объем данных, задействованных при анализе, и найти решение быстрее.

Закономерности, которые будут выявлены на третьем этапе, представляют собой набор правил, позволяющих прогнозировать поведение исследуемого временного ряда. Таким образом, накопив все правила с частотой их срабатывания, необходимо выбрать только те из них, которые встречались достаточно часто. После этого они заносятся в специальную таблицу правил. Данный этап будет заключительным в том случае, если набор накопленных правил является вполне достаточным для «оптимального» прогноза. Иначе необходимо добавить новые технические индикаторы и повторить все пункты исследования.

Одним из средств, позволяющих проводить исследования с помощью предложенного алгоритма прогнозирования, являются электронные таблицы Excel компании Microsoft. Данное средство обладает значительным количеством встроенных функций, которые выполняют различный спектр вычислений. Достоинством является возможность подключения дополнительных модулей или надстроек. Для работы с техническими индикаторами в среде Excel существует несколько надстроек: TA-Lib и TraderXL Pro компании Analyzer LLC. Последняя из них представляет собой комплекс связанных надстроек, позволяющих производить расчет значений не только технических индикаторов, но и имеет дополнительные функции – это загрузка исторических данных о ценах акций из Интернета, прогнозирование временных рядов, тестирование эффективности торговых стратегий, классификация данных и отслеживание изменений портфеля акций. Общее число реализованных технических индикаторов в данной надстройке составляет 146, поэтому данный набор охватывает все типы известных индикаторов. К тому же у пользователя есть возможность создания и использования своих собственных индикаторов.

Исследования предложенного алгоритма прогнозирования временных рядов проведены с использованием программного средства Excel и надстройки TraderXL Pro. Данный алгоритм применим для анализа временных рядов различной природы. Необходимым условием правильной его работы является достаточная для исследования длина ряда, позволяющая выявлять основные его закономерности.

Список литературы

1. Стратегии лучших трейдеров мира. – М.: Тора-Центр, 1997. – 173 с.

ПАРАЛЛЕЛЬНАЯ МОДЕЛЬ ВЫЧИСЛЕНИЙ ДЛЯ ОТОБРАЖЕНИЯ БОЛЬШИХ УЧАСТКОВ ЛАНДШАФТА НА ОСНОВЕ ПРОГРЕССИВНОГО КВАДРОДЕРЕВА И МАКСИМАЛЬНОГО ИСПОЛЬЗОВАНИЯ ГРАФИЧЕСКОГО ПРОЦЕССОРА СРЕДСТВАМИ MICROSOFT DIRECTX

Е.А. Юсов

Нижегородский государственный университет им. Н.И. Лобачевского

Введение

Проблема интерактивной визуализации открытых участков ландшафта большой площади в фотографическом качестве является актуальной для многих областей современных информационных технологий, от геоинформационных систем и авиатренажеров до компьютерных игр. Хотя производительность современных графических процессоров растет очень быстро и к настоящему времени уже достигла отметки в миллиард вершин в секунду, не менее быстро растут и объемы данных, задающих форму ландшафта, который необходимо визуализировать. Подобные данные могут представлять собой сетку высотных отметок, которая покрывает тысячи квадратных километров с разрешением всего в несколько метров и содержит огромное число отсчетов. Поэтому нагрузка на графическую подсистему должна контролироваться путем уменьшения числа отображаемых примитивов при минимальном ущербе качеству изображения.

Постановка задачи

Чтобы уменьшить сложность выводимой модели без негативного влияния на визуальную точность, триангуляция ландшафта должна быть адаптирована к характеристикам поверхности: она должна содержать меньшее число больших треугольников в гладких областях поверхности и большее число маленьких треугольников для участков с резкими особенностями. Кроме того, она также должна учитывать пространственное положение камеры, потому что удаленные зоны требуют меньшей детализации, чем участки, расположенные близко к камере. Среди других важных требований, которым должна удовлетворять модель триангуляции, также можно выделить следующие: 1) быстрое построение адаптированной триангуляции; локальные особенности поверхности не должны вести к глобальному усложнению модели; 2) отсутствие разрывов в геометрии, освещении и текстуре; 3) поддержание стабильной скорости генерации кадров; 4) сведение к минимуму скачкообразных изменений геометрии, связанных со смешной уровня детализации; 5) модель должна обеспечивать компактное представление и хранение данных.

Обзор работ в данной области

Для решения поставленной задачи в последние годы было разработано большое число различных алгоритмов, позволяющих адаптивно управлять уровнем детализации поверхности ландшафта в зависимости от ее локальных особенностей и характеристик наблюдения. Первые методы были описаны в [1] и [3] и были основаны на построении квадродерева вершин или бинарного дерева треугольников. Основной характеристикой предложенных методов является то, что в качестве наименьшего элемента представления модели ландшафта, на базе которого осуществляется упрощение и адаптация триангуляции, они используют отдельную вершину или треугольник. Десять лет назад такой

подход был вполне оправдан, потому что производительность центрального процессора намного превосходила возможности графического и было необходимо, насколько возможно, уменьшить число выводимых примитивов. Но в начале текущего десятилетия ситуация начала меняться и к настоящему моменту уже графические процессоры намного превосходят центральные по производительности (выполнения операций над вещественными числами). Поэтому на современных аппаратных средствах методы, описанные в [1] и [3], оказываются неэффективными, поскольку они перегружают работу центральный процессор, оставляя графический практически незадействованным. Начиная с 2000 года начали появляться работы, являющиеся расширением описанных в [1] и [3], которые в качестве наименьшего элемента представления ландшафта используют рельефный блок вместо единичной вершины или треугольника [4–8]. Переход от наименьшего элемента (треугольника или вершины) к более крупному позволяет существенно сократить время, требуемое на определение уровней детализации различных участков рельефа. При этом возрастает нагрузка на графический процессор, т.к. триангуляция оказывается хуже адаптированной. Однако нагрузка на различные элементы системы (CPU и GPU) выравнивается и общая производительность возрастает.

Описание алгоритма

С представленными в [4–8] подходами связано несколько проблем. Во-первых, поскольку они продолжают идеи, изложенные в [1] и [3], им свойственны те же ограничения – триангуляции блоков (патчей) должны удовлетворять ограничению, что уровни детализации соседних узлов должны отличаться не более чем на один. Вторая сложность связана с хранением подготовленных триангуляций. Во всех перечисленных методах триангуляции хранятся либо в основной памяти, либо на вспомогательных запоминающих устройствах в исходном виде и занимают очень много места, что не является эффективным. Предлагаемый в данной работе алгоритм преодолевает эти ограничения.

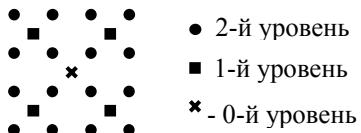


Рис 1. Квадродерево вершин

Он является развитием метода, представленного в [9]. В основе метода лежит квадродерево вершин, изображенное на рис. 1. В качестве минимального элемента упрощения модели предлагаемый метод использует рельефный блок. Алгоритм состоит из двух основных фаз – предобработки и отображения. Подобно [6] и [8] на первой фазе осуществляется создание адаптивных триангуляций патчей и построение квадродерева, узлам которого соответствуют патчи различных уровней детализации. Но, в отличие от представленных методов, сами триангуляции не запоминаются на диске. Вместо этого составляется список вершин квадродерева родительского патча, которые должны быть разбиты для увеличения детализации, по аналогии с операциями разбиения вершин в прогрессивных сетках, представленных в [2] (рис. 2). (Отсюда название метода – прогрессивное квадродерево.) Подобная «разностная» информация требует всего 1 бит на вершину, вследствие чего занимает очень небольшой

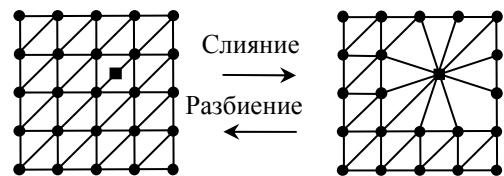


Рис. 2. Разбиение/слияние вершины

объем и может быть сохранена в оперативной памяти. Решение о возможности слияния вершин осуществляется путем сравнения пространственной погрешности, вводимой слиянием, с пороговым значением. Построение списка треугольников блока осуществляется один раз при его создании на этапе отображения и занимает крайне мало времени. Как и в [5], построенные триангуляции и патчи кэшируются в видеопамяти графической карты и используются в нескольких последующих кадрах, что снижает необходимость в постоянной пересылке данных через шину. В отличие от [5], построение триангуляций не включает в себя анализ пространственных и экранных погрешностей вершин, что позволяет сократить необходимое для построения время. Таким образом, предлагаемое решение является компромиссным вариантом между методами из [6] и [5] – на этапе предподготовки определяется компактная структура данных, на основе которой на этапе выполнения осуществляется построение триангуляций.

Реализация метода в среде Microsoft Visual Studio .NET средствами DirectX

Предложенный алгоритм был реализован в среде Microsoft Visual Studio .NET, в качестве базового графического программного обеспечения использовалась Microsoft DirectX 9.0. Для обеспечения стабильной работы алгоритма он был реализован в многопоточном асинхронном варианте с предсказанием положения камеры в пространстве. Принципиальная схема реализации алгоритма изображена на рис. 3.

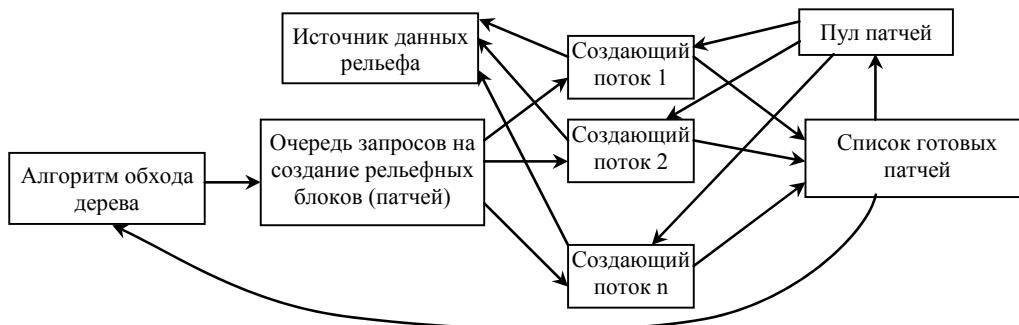


Рис. 3. Схема построения системы

Алгоритм обхода дерева определяет необходимые уровни детализации на основе экранной погрешности блоков, а также предсказывает, какие блоки потребуются в ближайшем будущем. Он помещает запросы на создание требуемых блоков в очередь, которые выбираются и асинхронно обрабатываются несколькими потоками. Созданные патчи помещаются в список готовых к отображению, незадействованные перемещаются в пул патчей, откуда извлекаются создающими потоками.

Результаты экспериментов

На рис. 4 приведены графики, демонстрирующие производительность системы при облете сильногористого рельефа с очень большой скоростью. Измерение показателей производилось с интервалом в 1/2 секунды. Рисунок содержит графики числа треугольников, отображаемых в текущем кадре (квадратные отметки), текущей частоты кадров (ромбовидные отметки) и производительности системы в миллионах треугольников, обрабатываемых за секунду (треугольные отметки).

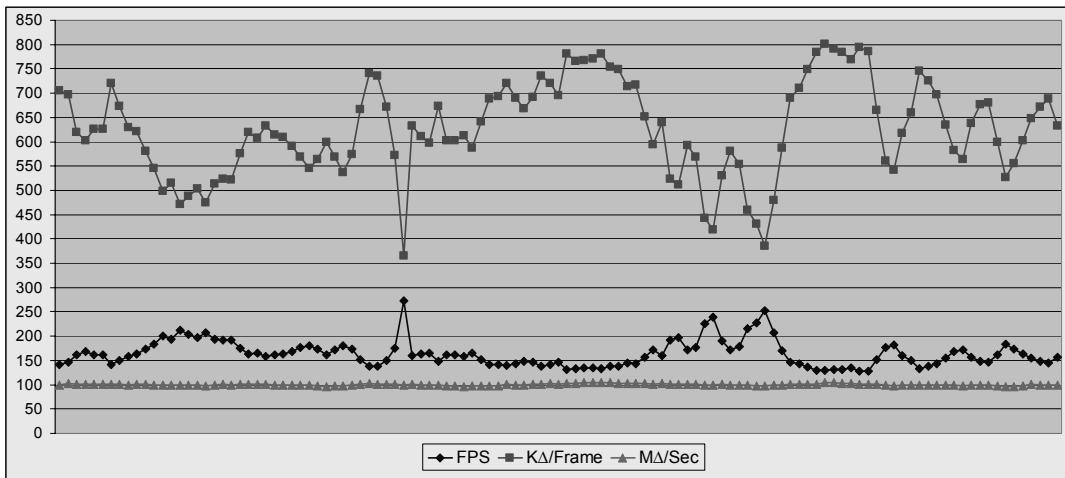


Рис. 4

Тестовый рельеф представлял собой восстановленный по данным спутникового сканирования реальный участок местности. Размер поля высот – 8192×8192 вершин. Траектория камеры представляла собой кривую с множеством резких поворотов (о чем свидетельствуют перепады графика количества треугольников на один кадр). Тестовая машина была оснащена процессором Intel Pentium D 3.4GHz, 2 Gb оперативной памяти и графическим процессором NVIDIA GeForce 7950 GT с 512 Mb локальной видеопамяти. Для данного теста экранная погрешность была выбрана равной 1 пикселеу. (Следует отметить, что поскольку алгоритм поддерживает геоморфинг по принципу, описанному в [9], метод может давать хорошие визуальные результаты и при существенно больших значениях погрешности.)

Рис. 4 подтверждает эффективность предлагаемого аппарата. Метод максимально использует возможности графического процессора, разгружая основной, и обеспечивает стабильную производительность системы. Благодаря эффективной параллельной модели и предсказанию положения камеры в пространстве, издержки, связанные с построением триангуляции, генерацией нормалей, загрузкой данных в видеопамять графического процессора, никак не сказываются на производительности системы. Как показывает график, число треугольников, обрабатываемых за одну секунду практически все время работы сохранялось на уровне 100 М, что близко к максимальным возможностям видеокарты. Частота кадров при этом никогда не падала из-за нагрузки на графическую систему ниже 100 кадров в секунду, что явно достаточно для систем реального времени.

Список литературы

1. Lindstrom P., Koller D., Ribarsky W., Hodges L. F., Faust N., and G. A. Turner. Real-time, continuous level of detail rendering of height fields. In Proceedings SIGGRAPH 96, pages 109–118. ACM SIGGRAPH, 1996.
2. Hoppe H. Progressive meshes. In Proceedings SIGGRAPH 96, pages 99–108. ACM SIGGRAPH, 1996.
3. Duchaineau M., Wolinsky M., Sigeti D. E., Miller M. C., Aldrich C., and M. B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. In Proceedings Visualization 97, pages 81–88. IEEE, Computer Society Press, Los Alamitos, California, 1997.
4. Pomeranz A. A. ROAM Using Surface Triangle Clusters (RUSTiC) Master's thesis, University of California at Davis, June 2000.

5. Joshua Levenberg Fast View-Dependent Level-of-Detail Rendering Using Cached Geometry In Proceedings IEEE Visualization'02 (Oct 2002), IEEE. P. 259–266.
6. Cignoni P., Ganovelli F., Gobbetti E., Marton F., Ponchio F. 2003 BDAM – Batched Dynamic Adaptive Meshes for High Performance Terrain Visualization Computer Graphics Forum, Volume 22(Sept. 2003), Number 3, pages 505–514.
7. Roberto Lario, Renato Pajarola, Francisco Tirado 2003 HyperBlock-QuadTIN: Hyper-Block Quadtree based Triangulated Irregular Networks IASTED International Conference on Visualization, Imaging and Image Processing (VIIP 2003). P. 733–738.
8. Gobbetti E., Marton F., Cignoni P., M. Di Benedetto, and F. Ganovelli 2006 C-BDAM – Compressed Batched Dynamic Adaptive Meshes for Terrain Rendering Computer Graphics Forum, Volume 25(2006), Number 3.
9. Юсов Е., Турлапов В. Динамическая оптимизация ландшафта на базе преобразования Хаара и квадрдерева вершин. Материалы конф. по комп. граф. и ее приложениям – GraphiCon'2006, Новосибирск, 1–5 июля 2006.

РАВНОМЕРНОЕ ПЕРЕРАСПРЕДЕЛЕНИЕ РЕСУРСОВ В ИЕРАРХИЧЕСКИХ СИСТЕМАХ ТРАНСПОРТНОГО ТИПА

Л.Г. Афраймович, М.Х. Прилуцкий

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается задача равномерного перераспределения ресурсов в иерархических системах транспортного типа. Иерархическая система представляет собой совокупность элементов и связей между ними [1], каждый элемент системы обладает определенным объемом распределяемых ресурсов. Требуется перераспределить ресурсы так, чтобы все элементы системы обладали равными объемами ресурсов, минимизируя при этом затраты на передачу ресурсов по связям иерархической системы.

Подобного рода оптимизационные задачи возникают, например, при решении задачи сбалансированной загрузки распределенной вычислительной системы [2]. Здесь иерархическая система моделирует распределенную вычислительную систему.

Элементы представляют собой процессоры; связи – коммутационные связи неограниченной пропускной способности между процессорами; ресурсы – независимые, равнозначные (с точки зрения вычислительных затрат) задачи; начальные объемы ресурсов – начальные загрузки процессоров. Требуется равномерно переназначить задачи, минимизируя затраты на переназначение задач с одного процессора на другой.

Иерархическую систему будем моделировать ориентированным графом $G=(V,A)$ без петель, $A \subseteq V^2$, множество V , $|V|=n$, вершин которого соответствует элементам системы, а множество A дуг – связям между элементами системы.

Каждый из элементов i обладает объемом ресурсов w_i , $w_i \geq 0$, $i \in V$. Тогда, для равномерного распределения, каждый из элементов должен обладать ресурсами в объеме $\frac{1}{n} \cdot \sum_{i \in V} w_i$. Через c_{ij} , $c_{ij} \geq 0$, будем обозначать затраты на передачу единицы

ресурсов по связи (i,j) , $(i,j) \in A$. Пусть $X = \|x_{ij}\|_{n \times n}$ – матрица, элемент x_{ij} которой определяет количество ресурсов, которое будет передано по связи (i,j) , $(i,j) \in A$.

Тогда задача равномерного перераспределения ресурсов в иерархической системе транспортного типа будет иметь следующий вид:

$$\sum_{(j,i) \in A} x_{ji} - \sum_{(i,j) \in A} x_{ij} + w_i = \frac{1}{n} \cdot \sum_{i \in V} w_i, \quad i \in V, \quad (1)$$

$$x_{ij} \geq 0, \quad (i,j) \in A, \quad (2)$$

$$\sum_{(i,j) \in A} x_{ij} c_{ij} \rightarrow \min. \quad (3)$$

Как показано в [1], при решении задачи (1)–(3) могут быть применены потоковые алгоритмы [3]. То есть проблема поиска допустимого решения системы ограничений (1)–(2) сводится к задаче поиска максимального потока в канонической транспортной сети; а задача равномерного перераспределения ресурсов (1)–(3) – к задаче поиска потока минимальной стоимости заданной величины.

Таким образом, можно предложить алгоритм решения задачи (1)–(3) с вычислительной сложностью $O(n^4)$.

Список литературы

1. Прилуцкий М.Х., Афраймович Л.Г. Оптимальное распределение однородного ресурса в иерархических системах с доходами // Вестник ВГАВТ. Межвузовская серия Моделирование и оптимизация сложных систем, Вып. 9. 2004. С. 56–63.
2. Diekmann R., Frommer A., Monien B. Efficient schemes for nearest neighbor load balancing // Parallel Computing, V. 25. 1999. P. 789–812.
3. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. – М.: Мир, 1985. – 510 с.

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА УПРАВЛЕНИЯ ВУЗОМ НА БАЗЕ ЭЛЕКТРОННОЙ КАРТЫ

С.А. Вильданов, В.М. Конюхова, Л.Н. Махмутов

*Елабужский филиал Казанского государственного технического
университета им. А.Н. Туполева*

Автоматизированная информационная система (АИС) на базе электронной карты вуза предназначена для повышения эффективности управления вузом, наглядности отображаемой информации и оперативности принятия решений.

Составными частями АИС являются:

- электронная карта вуза;
- база данных;
- приложения пользователей.

Содержанием проектов в автоматизированной информационной системе являются документы (тома, книги, разделы) и чертежи в векторной и растровой форме с пояс-

няющим текстом. К преимуществам проектной документации прежде всего отнесем то, что она хорошо структурирована и идентифицирована, а к недостаткам – то, что она содержит графику, обладающую огромной неформализуемой информативностью. Большое количество внутренних и внешних связей внутри файл-документов и между ними усложняет изготовление документов и многократно затрудняет их корректировку. Однако структура ИС, четко продуманная уже на начальном этапе разработки, позволяет не только избежать «файловой свалки» документов, но и уверенно выстраивать совместную работу [1]. В связи с этим еще более оправданной является строение АИС, использующей такие технологии как .NET, MySQL, MSF и т.д.

Востребованность цифровой картографической продукции в последнее время значительно возросла. Все большее число пользователей топографической информации обращают внимание на преимущества использования в своей работе цифровых карт [2].

Данная карта представляет из себя информационную систему, содержащую информацию о подразделениях вузов, их полномочиях, подлежащей синхронизации документации. Она позволяет классифицировать и находить объекты, решать «задачу коммивояжера» [3]. Возможно применение автоматических датчиков, обеспечивающих немедленный ввод информации с портов системы.

На карту нанесены объекты, необходимые для функционирования системы управления вуза. К ним относятся: здания (учебные корпуса, общежития, административные корпуса и т. п.), дороги, системы связи, маршруты движения студентов и преподавателей, автостоянки, закрепленная территория, коммуникации (тепло-, водо-, электросети и т.п.), элементы ландшафта [4].

Информация по каждому объекту хранится в базе данных, являющейся составной частью АИС. База данных (Microsoft SQL Server) хранится на сервере локальной сети вуза. Совокупность объектов поделена на слои по типам. Предусмотрена удобная система поиска объектов в базе данных и на карте. АИС предусматривает поиск и выделение цветом объектов на карте по информации из базы данных, а также поиск информации в базе данных по выбранному на карте объекту.

На карте может отображаться любое количество слоев в зависимости от желания пользователя или от его прав доступа.

Приложения пользователей разрабатываются для различных должностных лиц вуза, в зависимости от решаемых задач.

Использование ГИС в картопостроении порождает как новые возможности, так и новые проблемы [3]. Проблемы связаны в основном со сбором информации, а также четкой алгоритмизацией поставленных задач.

Для создания электронных карт может быть использован графический редактор, позволяющий наносить на карту объекты и вносить информацию о них в базу данных.

Для разработки АИС есть возможность использовать оцифрованное изображение карты, которое является растровой подложкой при нанесении объектов пользователем и последующем их отображении. Изображения объектов могут наноситься манипулятором «мышь» и с помощью различных инструментов. Также можно предусмотреть множество стандартных изображений условных обозначений, из которого «мышью» пользователь может «перетащить» любое изображение на карту. По желанию это множество может быть дополнено другими изображениями, созданными с помощью известных векторных редакторов типа CorelDraw, Visio и т.п., имеющих возможность работать с Windows MetaFile (WMF, EMF) форматами.

Существующие геоинформационные компьютерные системы (MapInfo, GeoDraw, Нева и др.) используют полностью векторное изображение карты, т. е. такой формат, когда изо-

брожения объектов хранятся в виде математических описаний объектов. При отображении карты программы используют эти описания для формирования изображений объектов.

Векторное изображение карты может создаваться заранее с помощью отдельных программ. Карта рисуется вручную, что требует больших затрат времени и высокой квалификации специалистов. Недостатками существующих систем являются высокая стоимость и большие трудозатраты при изготовлении карты.

Формат карты в разрабатываемой АИС может быть смешанный, векторно-растровый. Для получения растрового изображения карты необходим обычновенный сканер и несколько человеко-часов трудозатрат. Для хранения многоцветного изображения этой карты потребуется около 15 Мб дискового пространства, что является незначительной величиной для современных накопителей на жестких магнитных дисках.

Использование смешанного формата в десятки раз снижает трудозатраты на изготовление и стоимость электронной карты.

Таким образом, разрабатываемая АИС позволит должностным лицам вуза практически мгновенно получать наглядную информацию о необходимых объектах управления, планировать командировки сотрудников, использовать книжный фонд вуза в соответствии с потребностями и преподаваемыми дисциплинами в каждом подразделении вуза, распределенного территориально.

Система является незаменимой для вузов, имеющих филиалы в других городах, а не только для управления отдельными зданиями, расположенными в одном городе.

Разрабатываемые при создании АИС средства могут быть использованы в других областях применения и в народном хозяйстве.

Список литературы

1. http://www.cadmaster.ru/articles/25_cad_systems.cfm, 2006.
2. <http://www.geokosmos.ru/>, 2006.
3. Алгоритмы. Том 2. Д. Кнут – М.: Мир, 1977.
4. Автоматизация управления вузом / А.Я. Савельев, Ю.Б. Зубарев, Т.А. Колоскова.– М.: Радио и связь, 1984.

ОТОБРАЖЕНИЕ СОСТАВНЫХ БОЛЬШЕФОРМАТНЫХ РАСТРОВЫХ ДОКУМЕНТОВ

И.А. Голубев

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Целью данной работы является частичное решение проблемы оперативного обновления большеформатных растровых документов.

Многолистные картографические раstry имеют сотни тысяч пикселов как по горизонтали, так и по вертикали. Для организации обработки таких огромных раstrов необходимо разрабатывать особые методики эффективного хранения, так как все общезвестные подходы к проблеме компактного хранения раstrов неприменимы для большеформатных документов.

Так, например, картографическое растровое изображение в несжатом виде может занять значительное место на жестком диске персонального компьютера, его обработка может занять длительное время и т.п.

При работе с многолистным картографическим растром никогда не требуется получение всего изображения (необходим лишь некоторый заданный фрагмент раstra, помешающийся либо на экране монитора, либо на устройстве вывода). Таким образом, необходимо использовать только такие методы хранения, которые обеспечивают доступ к фрагментам хранимого изображения без распаковки лишней графической информации за достаточно короткий временной интервал.

Разработанный в НИИ прикладной математики и кибернетики формат хранения большеформатных растровых документов BIG (Big Images Graphics) [1] позволил решить проблему организации компактного хранения, обеспечения прямого доступа и оперативного отображения сверхбольших растровых изображений.

Недостатком формата BIG является невозможность оперативного обновления электронной карты, для обновления требуется перекодировать все изображение целиком. В данной работе предложен способ хранения большеформатного растрового изображения не целиком, а отдельными листами. Все большеформатное растровое изображение разбиваем на части равномерной прямоугольной сеткой. Каждая ячейка сетки будет представлять собой прямоугольный растр, который принимается за новый элемент хранения – лист. Таким образом, возникает задача оперативной сшивки разбитого на части изображения, с целью получения исходного изображения.

В ходе исследования, был предложен способ разделения большеформатного растрового изображения на отдельные листы, предложен формат описания предложенного разбиения, разработаны алгоритмы построения итогового изображения из отдельных листов, разработана библиотека визуализации.

Реализация и внедрение библиотеки просмотра большеформатных растровых документов в НИИ ПМК, позволило существенно ускорить процесс обновления изображения. А также упростить процесс поставки электронных карт конечному пользователю, путем снижения объема передаваемой информации.

Работа выполнена при поддержке РФФИ (проект № 05-01-00590).

Список литературы

1. Кудин А.В. Технология эффективного хранения и оперативного отображения картографической растровой информации. – Дис. канд. технических наук. Нижний Новгород, 2000.

МОБИЛЬНЫЙ ПРОГРАММНЫЙ КАРТОГРАФИЧЕСКИЙ КОМПЛЕКС «BIGVIEWER CE»

А.А. Егоров

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Индустрис ГИС активно развивается. Ее развитие постепенно сближает их с офисным ПО [1]. Разработчики офисных пакетов, такие как Microsoft с ее продуктом MapPoint 2000, делают встречные шаги [2]. Все это является индикатором, свидетельствующим о большом потенциале данного направления науки и техники.

В НИИ ПМК ННГУ создана и развивается многофункциональная объектно-ориентированная топологическая геоинформационная система ГИС «Терра», позволяющая решать широкий спектр задач в разных областях деятельности. В состав ГИС «Терра» входит программный комплекс «BigViewer» для PC-совместимых компьютеров под управлением ОС из семейства MS Windows. В настоящее время рынок ГИС становится все более дифференцированным и достаточно большую его составляющую занимают портативные компьютеры (КПК, смартфоны, мобильные телефоны, коммуникаторы и т.д.). Появление таких технологий значительно расширяет как число пользователей компьютерной техники, так и область ее применения, особенно если речь идет о ГИС.

В связи с этим было создано еще одно приложение в составе ГИС «Терра» – «BigViewer CE», разработанное специально для КПК на платформе Pocket PC. Основная задача этого комплекса – помочь в навигации пользователю в различных условиях (от простого движения по маршруту на карте до поиска кратчайшего пути до нужного объекта в критических ситуациях).

«BigViewer CE» выгодно отличается от подобного ПО в скорости, как при отображении карт и навигации, так и при поиске и получении информации об объектах из БД. Масштабирование, скроллинг, послойное отображение с выбором активных и неактивных слоев – все эти преимущества «BigViewer» также реализованы в «BigViewer CE».

Основное достоинство КПК – это, конечно же, мобильность и, при наличии GPS-приемника, возможность определения текущего местоположения на местности с записью пройденного пути.

Входящая в состав практически любой программы навигации для КПК, работа с маршрутами также реализована в «BigViewer CE», со своими дополнительными функциями. Предварительная прокладка маршрута (добавление, изменение, удаление точек предполагаемого пути) и определение его длины – позволит заранее спланировать маршрут. Построение ортодромии между каждой парой точек намеченного пути – позволит вычислить кратчайший путь, например для мореплавателей. Движение по намеченному пути к последней его точке (отображение пройденных точек и указание корректировки маршрута при отклонениях от намеченного пути), а также перевод пути для движения обратно в первую точку – не даст сбиться с намеченного пути. Сохранение пройденного пути в файл и последующая работа с ним как с маршрутом – позволит запомнить дорогу до нужной точки.

В программе предусмотрена работа с заметками. В любой момент можно поставить заметку в любое место карты, сохранить или загрузить заметки в файл или отредактировать заметку. Заметка может включать в себя как текст, так и небольшую картинку. Заметки можно использовать для пометок на маршруте, для перевода надписей на карте, для пометок об изменении объектов для последующего их редактирования в ГИС «Терра».

Еще один набор функций тесно связан с БД, используемой в программе. Данные в этой базе хранятся в некоторой иерархической структуре. У каждого объекта есть описание его типа, номер и некоторые характеристики. Пользователю предоставлена возможность осуществлять поиск объектов на карте по типу и номеру, характеристикам или получение информации по указанному объекту (фактически поиск по характеристике «местоположение»).

При поиске по номеру и типу, в случае наличия объекта в БД, отображается объект (подсвечивается его граница), соответствующий запросу на карте. Возможен поиск ближайшего от указанного местоположения объекта выбранного типа, в результате отображается найденный объект и кратчайший путь до него от указанной точки. Для поиска по характеристикам (в т.ч. получение информации при указании на объект) реализовано несколько функций:

1. Получение информации об указанном объекте. Пользователь указывает на объект, по которому хотел бы получить информацию (чтение характеристик) и в появившемся списке ему выводится пять ближайших к указанной точке объектов. В этом списке пользователь выбирает нужный ему объект и читает его характеристики.

2. Поиск объекта по характеристикам с некоторыми условиями. Например, можно уточнить, с каким кодом объекты следует рассматривать, а какие нет. Можно указать, по каким характеристикам ведется поиск, какое условие должно выполняться по данной характеристике и введенному эталону (больше, меньше, равно, не равно, входит строка, совпадает строка).

Экспериментальные апробации программного комплекса подтвердили его эффективность.

Работа выполнена при поддержке РФФИ (проект № 05-01-00590).

Список литературы

1. ГИС и тенденции на будущее – gisnews.icc.ru
2. Карасев А., Трубицын А. Измеряя, вычисляя, отображая, сравнивая... – www.pcweek.ru
3. Кудин А. В. Технология эффективного хранения и оперативного отображения картографической растровой информации. – Диссертация на соискание ученой степени кандидата технических наук. Нижний Новгород, 2000.
4. Yu.G. Vasin, Yu.V. Yasakov. GIS Terra: A graphic database management system // Pattern recognition and image analysis. 2004. Vol. 14. № 4. P. 579–586.

СИНХРОНИЗАЦИЯ ИЗОБРАЖЕНИЙ В ФОРМАТЕ BIG

С.В. Жерзев

*НИИ Прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Одной из задач, возникающих при создании архивов электронных картографических документов, является поддержание их в актуальном состоянии и оперативное обновление. Особенно заметно проявляется эта проблема при работе с коллекциями морских навигационных карт из нескольких тысяч листов, поправки к которым (извещения мореплавателям) выпускаются каждые несколько дней.

Большой вклад в объем электронной карты вносит ее растровое изображение, которое, даже представленное в специализированном формате BIG [1] с использованием эффективных схем сжатия, может иметь размер от единиц до десятков мегабайт. Как следствие, актуальной является задача обновления подобных файлов при минимальных требованиях к пропускной способности каналов связи.

Предложенное решение делит данный процесс на два этапа: определение различающихся фрагментов изображений на сервере (новый вариант) и клиенте (старый вариант) и передача информации об изменениях. Для реализации второго этапа существенно используется блочная структура файла в формате BIG, которая позволяет независимо манипулировать относительно небольшими фрагментами изображения.

В основе алгоритма для первого этапа лежит технология, применяющаяся в утилитах синхронизации rsync [2], но модифицированная с учетом специфики данных (двумерная природа растрового изображения) и знания структуры BIG-файла, что позволяет существенно повысить скорость работы приложения.

Выявление участков изображения, требующих повторной загрузки, использует тот факт, что, как правило, внесенные изменения локализованы в пространстве. Алгоритм производит иерархическое деление листа карты на уменьшающиеся участки для проверки их идентичности, начиная с проверки на изменения всего листа и заканчивая выявлением конкретных измененных BIG-фрагментов.

На каждом шаге сервер и клиент вычисляют и сравнивают значения дайджестов соответствующих участков с использованием алгоритма хеширования Message Digest 5 (MD5) [3], алгоритм рекурсивно применяется для тех участков, значения дайджеста которых на сервере и клиенте отличаются. Такая схема позволяет выявить требующие обновления фрагменты с минимальными накладными расходами.

Предложенный алгоритм был реализован в виде клиент-серверного приложения для синхронизации набора электронных карт. В серверной части реализованы дополнительные функции, обеспечивающие взаимодействие с архивом электронных карт и аутентификацию пользователей.

Работа выполнена при поддержке РФФИ (проект № 05-01-00590).

Список литературы

1. Кудин А. В. Технология эффективного хранения и оперативного отображения картографической растровой информации. – Диссертация на соискание ученой степени кандидата технических наук. Нижний Новгород, 2000.
2. Материалы сайта <http://samba.anu.edu.au/rsync/>
3. Rivest R. The MD5 Message Digest Algorithm. RFC 1321, MIT and RSA Data Security, Inc., April 1992.

СЖАТИЕ ИЗОБРАЖЕНИЙ В ИЗВЕЩЕНИЯХ МОРЕПЛАВАТЕЛЯМ

С.В. Жерзев

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

В настоящее время для поддержания актуальности морских навигационных карт производится периодическая рассылка на суда так называемых извещений мореплавателям. В электронном виде этот документ представлен в формате PDF и содержит как текстовые описания исправлений, так и обновленные фрагменты карт для распечатки (вклейки). Поскольку каналы связи, как правило, являются ограниченным ресурсом, актуальной является задача уменьшения объема передаваемых данных.

Исследования показали, что максимальный вклад в объем файла вносят именно внедренные изображения и решение проблемы следует искать в первую очередь в их более компактном представлении. На практике используются два типа таких изображений: сканированные растровые изображения в формате TIFF с использованием различных алгоритмов сжатия и синтезированные векторные изображения в формате Post-Script.

Было предложено растеризовать с необходимым разрешением соответствующие страницы документа и использовать разработанный в НИИ ПМК метод иерархического сжатия растровых данных без потерь [1], показавший хорошие результаты на различных типах изображений [2].

В отличие от большеформатных документов, размеры вклеек не ставят жестких требований к скорости прямого доступа к фрагментам изображения, что позволило использовать деревья отсчетов с большим, чем обычно, числом уровней (20 вместо 14). Это обеспечило более эффективную работу модуля статистического сжатия и меньший объем результирующих архивов.

Практическое тестирование на различных примерах вклеек показало получение 5–30% выигрыша в объеме файлов для сканированных растровых вклеек (черно-белые, разрешение 600 dpi) и более чем пятикратного – для векторных изображений (3 цвета, растеризация на разрешении 600 dpi).

Работа выполнена при поддержке РФФИ (проект № 05-01-00590).

Список литературы

1. Yu. G. Vasin and S.V. Zherzdev Information Techniques for Hierarchical Image Coding // Pattern Recognition and Image Analysis, Vol. 13, No. 3, 2003. P. 539–548.
2. Васин Ю.Г., Жерзев С.В. Цифровые архивы графических документов // VII международная научно-техническая конференция «Физика и радиоэлектроника в медицине и экологии – ФРЭМЭ 2006». Доклады. Книга 1. – Владимир, Собор, 2006. С. 297.

ARAGELI – БИБЛИОТЕКА ДЛЯ СИМВОЛЬНЫХ ВЫЧИСЛЕНИЙ. ОБЗОР АРХИТЕКТУРЫ

Н.Ю. Золотых, С.С. Лялин

Нижегородский госуниверситет им. Н.И. Лобачевского,

Введение

Arageli [1, 2] – это библиотека для выполнения точных, т.е. символьных или алгебраических, вычислений. Она содержит реализацию таких базовых алгебраических структур, как целые числа произвольного размера, рациональные числа, векторы, матрицы, полиномы и др.

При проектировании были рассмотрены аналогичные библиотеки (см., например, [3, 4, 5]). Одним из требований к библиотеке является её удобство использования неподготовленными пользователями: студентами, слушателями спецкурсов по компьютерной алгебре и символьным вычислениям. Это позволяет использовать библиотеку в обучении при проведении лабораторных работ по вышеназванным курсам. Почти везде при выборе между сложной (тем более – аппаратно зависимой) оптимизацией и ясным изложением хрестоматийного алгоритма, разработчики выбирали второе.

Историческая справка

Arageli имеет многолетнюю историю развития. Первоначальная версия была создана Н.Ю. Золотых на языке Pascal и имела ограниченную функциональность. Основной причиной, побудившей разработчиков библиотеки перейти на C++, было отсутствие простого способа конструировать одни типы данных из других, как это делается с основными структурами в алгебре. Шаблоны C++ существенно упростили этот процесс, дав при этом возможность конструировать типы на этапе компиляции.

Сейчас Arageli – это библиотека, написанная на C++ и распространяемая в исходных кодах. Она использует большинство возможностей языка реализации, указанных в стандарте [6], хотя при переходе с Pascal на C++ перед разработчиками стоял выбор: 1) использовать лишь ограниченное подмножество языка для обеспечения совместимости с различными компиляторами или 2) надеяться, что у пользователя всегда будет компилятор, в достаточной степени поддерживающий стандарт [6], и использовать его при проектировании библиотеки.

Архитектура Arageli плавно развивалась от решения (1) до решения (2). Такой положительный переход вызван прежде всего тем, что в последнее время стали доступны (бесплатные) компиляторы, поддерживающие стандарт C++ в достаточной степени. Под операционной системой Microsoft Windows – это компилятор, входящий в состав Microsoft Visual C++ 2005 Express Edition, а под Linux – последние версии GCC.

Помимо авторов этой статьи в разное время в библиотеку сделали свой вклад: Евгений Агафонов, Макс Алексеев, Алексей Бадер, Анна Бестаева, Александр Пшеничников, Алексей Половинкин, Николай Санталов, Андрей Сомиков и Екатерина Щукина. Также следует отметить всех участников мини-проектов, выполненных стажёрами Лаборатории информационных технологий (ITLab), полный список которых можно найти на сайте [1].

Структуры данных и алгоритмы

Библиотека поддерживает следующие структуры (включая базовые операции над ними): целые числа произвольной величины, числа с плавающей точкой произвольной точности и величины, кольца частных (включая поле рациональных чисел и рациональные функции), полиномы от одной и нескольких переменных (разреженное и плотное представления), векторы, матрицы, кольца вычетов (модулярная арифметика), конечные поля, алгебраические числа, выпуклые многогранники, интервалы (интервальная арифметика).

Почти все перечисленные выше структуры представляются шаблонными классами. Пользователь может сконструировать любую комбинацию базовых типов для вычислений в нужной алгебраической области.

Особенностью библиотеки является то, что кроме статического (шаблонного, т.е. работающего на этапе компиляции) способа конструирования типов предоставлена возможность их динамического построения на этапе исполнения программы. Это, в частности, используется в работе оболочки библиотеки – Arageli Shell, а также может быть полезным в любой программе, в которой нельзя в момент компиляции зафиксировать конкретные алгебраические структуры. Данный способ построения объектов используется в системе GiNaC [3], но там он не дополнен статическим методом, который не содержит дополнительных накладных расходов на динамическое определение типов, хотя и служит источником слишком большого количества сгенерированного бинарного кода. При операциях над объектами допускаются смешанные вычисления. Например, при выполнении бинарной операции умножения над двумя матрицами формально типы аргументов могут различаться, но если они представляют одну и ту же структуру с алгебраической точки зрения, то такая операция откомпилируется и будет работать правильно. Для сложных типов допускаются различия на любом уровне иерархии построения объекта; библиотека содержит статические средства для определения указанного «подобия» типов.

Кратко перечислим основную алгоритмическую базу библиотеки: классический, расширенный и бинарный алгоритмы Евклида для вычисления НОД; алгоритмы Гаусса и Гаусса – Бареисса для получения ступенчатой формы матрицы над полями и целыми числами; различные алгоритмы для получения нормальной диагональной формы Смита для полиномиальных и целочисленных матриц; вероятностные тесты на простоту: Соловей – Штрассена, Миллера – Рабина и Аgravала – Кайяла – Саксенны; алгоритмы разложения числа на простые множители: ρ -метод Полларда, $(p - 1)$ – метод Полларда; симплекс-метод для решения задачи линейного программирования; алгоритмы Гомори для решения задачи линейного целочисленного программирования; алгоритм Штурма для отделения вещественных корней полинома; алгоритм RSA; алгоритм двойного описания Моцкина – Бургера; два классических алгоритма триангуляции многогранного конуса; LLL-алгоритм для отыскания приведённого базиса решётки; алгоритм Монтгомери для быстрого модулярного умножения; алгоритм быстрого преобразования Фурье (FFT); алгоритм Бухбергера для нахождения базиса Грёбнера; алгоритм Борвинка для подсчёта целых точек в политопах.

Алгоритмический состав библиотеки прежде всего определяется научной и учебной работой, которая ведётся с использованием библиотеки, и является вкладом студентов, аспирантов и других независимых участников проекта Arageli.

Контролируемые алгоритмы

Все выше перечисленные алгоритмы реализованы как шаблоны с параметризацией по типам входных, выходных и некоторых промежуточных данных. Такое оформление алгоритмов, с одной стороны, делает их реализацию независимой от конкретных типов, фикси-

руя лишь интерфейс обращения с типом. С другой стороны, позволяет проводить некоторые исследования работы алгоритмов, передавая аргументы специально сконструированных типов, которые помимо «основной работы» имеют дополнительный «побочный эффект», – это может быть подсчёт операций, производимых над объектом данного типа, контроль максимальных и минимальных значений переменных, отслеживание потоков данных, объёма занимаемой памяти и т.д. Т.е. это способ тонкого профилирования реализаций.

Приведём простой пример. Пусть стоит задача: подсчитать количество битовых операций при вычислении нормальной диагональной формы Смита (НДФ) для заданной матрицы, содержащей полиномы с рациональными коэффициентами.

Для решения этой задачи следует определить тип данных, который имел бы интерфейс целого числа произвольного размера (`big_int`), но при выполнении элементарных операций (подобно сложению, умножению и т.д.) над объектом этого типа производился бы учёт и накопление бинарной сложности производимых операций. Пусть этот класс имеет имя `bigint_bitoper`. Далее следует сконструировать тип матрицы. В обычном случае (без задачи подсчёта битовых операций) указанный тип матрицы в `Arageli` выражается следующей конструкцией: `matrix<polynom<rational<big_int>>`, при использовании же `bigint_bitoper` нужный тип выражается так: `matrix<polynom<rational<bigint_bitoper>>`.

Пренебрегая операциями с индексами матрицы и полинома, при правильном определении `bigint_bitoper` простой вызов функции НДФ для заданной матрицы даст в качестве «побочного эффекта» искомое число битовых операций. Следует отметить, что из-за несомненной полезности `bigint_bitoper` подобный класс уже реализован в `Arageli`.

Более того, механизм смешанных вычислений, о котором говорилось ранее, играет здесь большую роль, так как позволяет использовать одновременно оригинальные типы и типы с «побочным эффектом» как равноправные в выражениях. Например, два вышеупомянутых типа матриц будут признаны «подобными».

Наряду с параметризацией по входным, выходным и промежуточным данным функции, реализующие некоторые алгоритмы, принимают ещё один дополнительный аргумент, называемый *контролёром алгоритма*. Такие функции называются *контролируемыми*. Объект этого типа служит способу дополнительного контроля хода выполнения алгоритма и является расширением концепции функции обратного вызова.

Контролируемая функция в ключевых местах своей работы (определяемых спецификой реализуемого алгоритма) вызывает различные методы контролёра, передавая текущие обрабатываемые данные в качестве аргументов. Меняя типы контролёров для функции, можно добиться различных эффектов от этих вызовов: от полного игнорирования до подробного вывода на консоль или в файл всех промежуточных данных. Последнее может быть полезно не только при отладке, но и при иллюстрации работы алгоритма в учебных целях.

Для каждой контролируемой функции реализован по крайней мере один контролер: тот, который ничего не делает, когда вызываются его методы, и не использует передаваемые данные. Такой контролёр называется *idle*-контролёром. Так как тип контролёра задаётся в качестве параметра шаблона, то использование *idle*-контролёров не ведёт ни к каким дополнительным затратам на этапе исполнения, – каждый из рассмотренных нами компиляторов полностью убирал ненужные вызовы пустых методов такого контролёра.

Допускается, что контролёр, будучи вызванным из контролируемой функции, может генерировать исключение в одном из своих методов. Для выполняемой функции данная ситуация служит сигналом к аварийному завершению своей работы. Контролируемые функции, корректно обрабатывающие такие ситуации, называются *прерывае-*

мыми функциями. В реальных приложениях такой способ завершения работы функции является полезным, например, в случае, когда алгоритм выполняется ощутимый промежуток времени, а контролёр алгоритма связан с пользовательским интерфейсом, через который пользователь может прервать работу.

Некоторые алгоритмы библиотеки реализованы в виде классов. В этом случае возникает ещё одна возможность контроля хода выполнения: приостановка работы алгоритма и дальнейшее продолжение обработки. Такие алгоритмы называются *приостанавливаемыми*. Классы, реализующие такие алгоритмы, также принимают контролёры алгоритмов и реализуют через них приостановку.

Особенностью Arageli является то, что все перечисленные способы контроля алгоритмов являются полностью отключаемыми: если пользователю нужна производительность, то он передаёт *idle*-контролёры, если же ему нужен подробный диагностический вывод и сбор статистики, то он подменяет базовые типы данных или применяет нетривиальные контролёры.

Ввод/вывод

Подсистема ввода/вывода в Arageli поддерживает несколько различных способов представления объектов. Для разных типов объектов число представлений варьируется. Общие методы следующие: обычный текстовый ввод/вывод, выровненный вывод для консоли, вывод в LaTeX, двоичный ввод/вывод. Для двухмерных полиэдров поддерживается вывод, пригодный для вставки в документ на LaTeX и отображающийся как графическое представление этого полиэдра. Для трёхмерных телесных политопов поддерживается вывод в VRML-файл.

Кроме основных структур данных возможностью ввода/вывода обладают и объекты классов алгоритмов. В сочетании со свойством приостанавливаемости, это даёт возможность, например, запустить выполнение алгоритма, через некоторое время приостановить его, сохранить состояние объекта алгоритма в файл или передать по сети на другую машину, а затем, после восстановления объекта из сохранённого образа, продолжить выполнение. Данный сценарий находит своё применение при распределённых вычислениях в сетях с переменным числом узлов.

Заключение

В настоящее время библиотека является коллекцией параметризованных структур данных и контролируемых алгоритмов с поддержкой смешанных вычислений. Развитие библиотеки направлено как в сторону расширения моделируемых алгебраических областей, в которых можно производить вычисления, так и в сторону пополнения множества конкретных задач, которые можно решить с помощью библиотеки в уже моделируемых областях. Ещё одним направлением развития является улучшение совместимости с другими подобными библиотеками.

Список литературы

1. Arageli Home Page, <http://www.unn.ru/cs/arageli>.
2. Lyalin S.S. and Zolotykh N.Yu. Arageli: a Library for Algebraic Computations. Overview of Architecture. Preprint. Reported at The Second International Congress on Mathematical Software, 2006. PDF is available at http://www.unn.ru/cs/arageli/doc/Arageli_overview_announce_preprint.pdf.
3. GiNaC Home Page, <http://www.ginac.de>.
4. NTL Home Page, <http://www.shoup.net/ntl>.
5. LiDIA Home Page, <http://www.informatik.tu-darmstadt.de/TI/LiDIA>.
6. Programming languages – C++. International Standard ISO/IEC 14882:1998(E).

ОПТИМИЗАЦИЯ ОДНОЙ МОДИФИКАЦИИ АЛГОРИТМА ДВОЙНОГО ОПИСАНИЯ ДЛЯ МНОГОГРАННОГО КОНУСА

Н.Ю. Золотых, С.С. Лялин

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Задача двойного описания многогранного конуса возникает во многих прикладных задачах, востребованных вычислительной биологией и химией. Анализ и верификация аппаратных и программных систем, в частности алгоритмы анализа для автоматической оптимизации и распараллеливания программ, опирающиеся на полиздральную модель, также используют алгоритм двойного описания полиздротов.

В данной работе сообщается об успешной попытке дополнительной оптимизации существующей модификации [1, 2] алгоритма двойного описания.

Необходимые определения

Пусть дана матрица $A \in R^{m \times d}$. Многогранным выпуклым конусом C в пространстве R^d называют множество решений системы линейных неравенств $Ax \geq 0$:

$$C = \left\{ x \in R^d \mid Ax \geq 0 \right\}. \quad (1)$$

Кроме представления (1) конус C может быть записан в параметрическом виде, а именно для некоторой матрицы $B \in R^{d \times n}$:

$$C = \left\{ x \in R^d \mid x = B\lambda, \lambda \in R_+^n \right\}, \quad (2)$$

где R_+ – неотрицательные действительные числа. Известно (см., например, [3]), что для любой матрицы A из описания (1) найдётся матрица B для описания (2) такая, что оба описания будут задавать один и тот же конус. Верно и обратное. Более того, известно, что задачи перехода от (1) к (2) и от (2) к (1) эквивалентны и могут быть решены одним и тем же алгоритмом. Не уменьшая общности, везде далее будем считать, что решается прямая задача.

Минимальная по числу столбцов матрица B для данного конуса называется матрицей остова конуса, а соответствующее множество векторов-столбцов – остовом конуса C . В данном случае векторы-столбцы матрицы B называются экстремальными лучами конуса.

Алгоритм двойного описания

Алгоритм двойного описания позволяет переходить от одного описания конуса к другому, причём получаемое описание будет минимальным, т.е. алгоритм получает остов конуса. Алгоритм был впервые предложен в [4]; известны также некоторые вариации (см., например, [5, 6]). Здесь мы рассматриваем одну из последних модификаций [1, 2], которая являлась наибыстрейшей из известных существующих на момент написания данной статьи.

Коротко изложим суть алгоритма. Неравенства системы $Ax \geq 0$ рассматриваются по очереди в определённом порядке. В процессе работы поддерживается текущий остаток конуса, который соответствует уже рассмотренным неравенствам. На каждом шаге алгоритма принимается к рассмотрению одно из ещё не обработанных неравенств, и с его учётом перестраивается остаток.

Перестройка остатка состоит в удалении из старого остатка B лучей, которые не удовлетворяют вновь рассматриваемому неравенству, и добавлении новых лучей. Не вдаваясь в подробности, скажем лишь, что при этом используется понятие смежности лучей: два луча называются смежными, если они принадлежат одной 2-грани конуса. Выяснение смежности лучей – вычислительно трудная операция. Для уменьшения числа лучей, которые должны быть протестированы на смежность, модификация [1] дополнительно сохраняет часть матрицы смежности, которая перестраивается на каждом шаге работы алгоритма.

Рассматриваемая модификация [1] реализована в виде программы Skeleton [2, 7], которая и анализировалась в настоящей работе.

Анализ производительности алгоритма и сделанные улучшения

Было проведено профилирование реализации [7] с использованием продукта Intel VTune 7.2. На типичных исходных данных распределение времени работы алгоритма по основным стадиям приведено в табл. 1. Было также обнаружено, что в части, которая тестирует битовые комбинации, представляющие лучи (это 57% из указанных 80% в табл. 1), возникает много промахов в кэш L1, а на некоторых наборах данных – большое число промахов и в L2 кэш.

При создании новых лучей также наблюдается большое число кэш-промахов в L1 и в L2 кэш.

Таблица 1

Распределение времени работы по основным стадиям

Стадия алгоритма	Доля времени
Тестирование лучей на смежность и добавление ребер в граф смежных лучей	80%
Создание новых лучей и удаление ребер из графа смежных лучей	13%

Разберём выбранные стадии более подробно.

Основная по времененным затратам часть задачи тестирования лучей на смежность состоит в следующем. Дан набор из k m -битовых последовательностей. Требуется перебрать все неупорядоченные пары в данном наборе, исключая пары вида (x, x) ; для каждой пары вычислить побитовую операцию ИЛИ для соответствующих m -битовых последовательностей и подсчитать число единичных бит в результате. Данная операция занимает 57% всего времени работы программы.

Из высказанного ясно, что всё множество тестируемых пар можно представить как верхнюю треугольную матрицу с исключённой диагональю (см. рис. 1). При достаточно больших числах n и m при простом переборе всех пар из набора (в лексикографическом порядке, который имеет место в оригинальной реализации, см. рис. 1а), мы не достигаем нужной локальности обращения к памяти. Из-за этого возникает много кэш-промахов.

Была реализована блочная схема тестирования лучей, в которой пары лучей перебираются в порядке, показанном на рис. 1б. На рисунке: $k = 8$, блок содержит по четыре различных луча для каждого элемента пары.

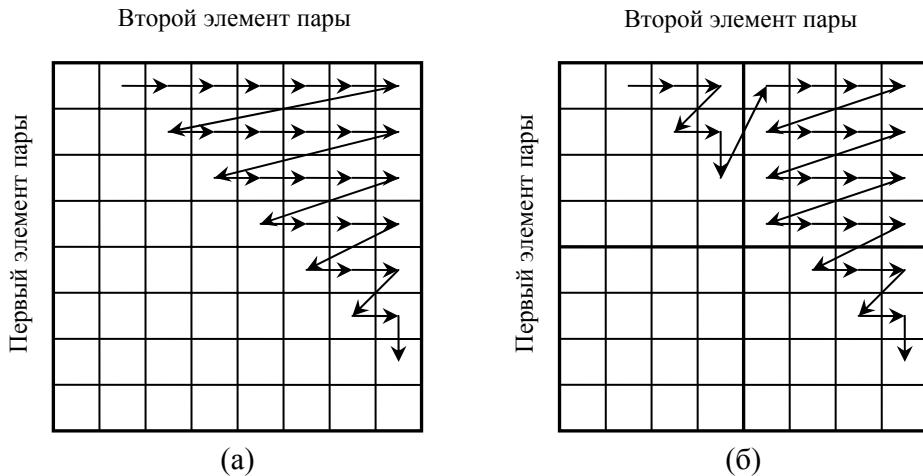


Рис. 1. Обход пар лучей: а) лексикографический,
б) блочный лексикографический

Размер блока выбран так, чтобы все соответствующие битовые последовательности убирались в L1 кэш процессора.

Вторая стадия алгоритма, выбранная для оптимизации, – это создание новых лучей (13% всего времени). При создании нового луча вычисляется линейная комбинация двух существующих векторов размера m и делается дополнительный проход по вновь созданному вектору для изменения некоторых элементов. При выполнении этой операции в оригинальной версии создавались промежуточные векторы, по которым осуществлялся многократный проход. При достаточно большом m данные, вовлечённые в создание одного луча, не помещаются в кэш L1 и при каждом новом проходе по вектору загружаются из L2.

Эта операция была преобразована в новую, которая требует однократного прохода по каждому из вовлечённых векторов и частичного второго прохода. Новая операция не использует временных переменных, размер которых зависит от m . Помимо экономии памяти это улучшило использование кэша L1.

Результаты численных экспериментов

Эксперименты проводились под операционной системой Microsoft Windows XP SP2 на двух машинах:

- Машина А. Intel Pentium III, 800 MHz, Data L1 = 16 Kb, L1 latency = 3 такта, L2 = 256 Kb, L2 latency = 23 такта, основная память 512 Mb.
- Машина В. Intel Xeon 3.4 GHz, Data L1 = 16 Kb, L1 latency = 4 такта, L2 = 1Mb, L2 latency = 32 такта, основная память 2.75 Gb.

Параметры задержки обращения к кэшу (latency) измерялись с помощью утилиты latency.exe, входящей в состав программы CPU-Z [8]. Программа компилировалась с помощью Microsoft Visual C++ 2005. Результаты представлены в табл. 2. Следует отме-

тить, что при экспериментах не применялись эвристики, которые реализованы в [7] по выбору очередного неравенства, и все неравенства брались в порядке их следования в исходной задаче.

Таблица 2
Результаты запуска неоптимизированной и оптимизированной версий

Набор данных	Параметры задачи			Неоптимизированный, сек.		Оптимизированный, сек.		Ускорение, раз	
	d	m	n	A	B	A	B	A	B
skeleton-test-1.ine	10	100	393	26.79	6.969	13.36	3.285	2.00	2.12
cube16.ine	17	32	65536	29.69	7.218	17.24	4.212	1.72	1.71
mit729-9.ine	9	729	4862	253.06	64.189	126.99	30.197	1.99	2.13

Результаты экспериментов показывают, что низкоуровневая оптимизация обращений к кэш-памяти может дать значительный прирост производительности без существенного изменения реализуемого алгоритма. Отметим, что при оптимизации стадии тестирования лучай на смежность общее число выполняемых операций увеличилось (дополнительные накладные расходы на организацию блоков), но мы всё равно получили существенное ускорение от этого преобразования.

Дальнейшие исследования

Дополнительно в процессе анализа были выявлены параллельные потоки данных как локально внутри одной итерации, так и между двумя соседними итерациями. Данная информация позволяет начать работу над параллельной реализацией алгоритма. Причём выявленная параллельность отличается от параллельности, использующейся в двух основных подходах, представленных в [6].

Список литературы

1. Золотых Н. Ю. Новая модификация метода двойного описания для построения остова многогранного конуса. III Всероссийская конференция «Проблемы оптимизации и экономические приложения». Тезисы докладов. – Омск, 2006. – С. 108.
2. Золотых Н. Ю. Skeleton – программа построения остова многогранного конуса. Информационный бюллетень Ассоциации математического программирования. № 11. Конференция «Математическое программирование и приложения». – Екатеринбург, 2007. – С. 120–121.
3. Черников С.Н. Линейные неравенства. – М.: Наука, 1969.
4. T.S. Motzkin, H. Raiffa, G.L. Thompson, and R.M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, Contributions to the Theory of Games – Volume II, number 28 in Annals of Mathematics Studies, pages 51–73. Princeton University Press, Princeton, New Jersey, 1953.
5. Черникова Н.В. Алгоритм для нахождения общей формулы неотрицательных решений системы линейных неравенств. Ж. вычисл. матем. и матем. физ. 5, № 2 (1965), 334–337.
6. Fukuda K. and A. Prodon. Double description method revisited. In M. Deza, R. Euler, and I. Manoussakis, editors, Combinatorics and Computer Science, volume 1120 of Lecture Notes in Computer Science, pages 91–111. Springer-Verlag, 1996.
7. Nikolai Yu. Zolotykh. Skeleton: Implementation of Double Description Method, <http://www.uic.mnov.ru/~zny/skeleton>.
8. CPUID. x86 Technical Resources Home Page, <http://www.cpuid.com>.

АКТУАЛЬНЫЕ ПРОБЛЕМЫ СЖАТИЯ ВИДЕО

Ю.С. Кирпичев

*НИИ прикладной математики и кибернетики
Нижегородского госуниверситета им. Н.И. Лобачевского*

Рассматриваются основные проблемы, связанные с технологией сжатия видеоинформации, а также наиболее часто используемые подходы для решения основных проблем.

Введение

Видеинформация в современном мире получила практически повсеместное применение. При этом число разнообразных форматов представления цифрового видео огромно и подавляющее большинство сегодняшних алгоритмов сжатия видео являются алгоритмами с потерей данных. Все они при сжатии используют три основных типа избыточности информации, характерных для видеоизображений: когерентность областей изображения, избыточность в цветовых плоскостях и подобие между кадрами. И хотя для обычного пользователя проблема хранения больших объемов видеинформации постепенно отходит на второй план, за счет увеличения емкости современных носителей информации, все же, как правило, стандартные подходы имеют ряд недостатков, в той или иной мере ухудшающих степень сжатия. Так, многие стандартные алгоритмы не учитывают, что видеоизображения не всегда являются идеальными. При этом в обработанных данных возникают различные характерные артефакты: шум камеры (характерен для цифровых и веб-камер), мусор, царапины, дрожание кадра, а также возникает такой неприятный визуальный эффект, как блочность.

Другие алгоритмы могут не учитывать существование альтернативных подходов для обработки кадров, имеющие сравнимые показатели, а возможно и лучшие, но лишенные некоторых из описанных выше недостатков используемых технологий.

Еще одной серьезной проблемой является то, что, как правило, при устраниении временной избыточности простая межкадровая разница работает плохо при сильном движении в кадре, что может дать непредсказуемый результат. Для решения этой проблемы используются различные алгоритмы компенсации движения. Но, как правило, алгоритмы компенсации не всегда находят оптимальное решение или же требуют больших вычислительных затрат.

Таким образом, можно выделить целый ряд направлений исследования для улучшения существующих алгоритмов обработки видеинформации.

Основные направления совершенствования

Как можно видеть, оптимизация существующих алгоритмов сжатия видео возможна в нескольких, слабо коррелирующих между собой направлениях. Поэтому просто выделим эти направления. Так, первым является выделение объектов в кадре, отражающих семантическое, а не пространственное разделение кадра. Эта задача, пожалуй, больше относится к задачам распознавания образов. Но при существующих вычислительных мощностях возможно лишь частичное решение данной задачи, с применением различных методов распознавания образов. Потому как полностью задачу распознавания на скорости 25 кадров в секунду, для изображений размером 720×576 пикселей, не в

состоянии решить на современных вычислительных мощностях ни один из имеющихся алгоритмов распознавания образов.

Вторым направлением является улучшение алгоритма сжатия, причем это может быть характерно для алгоритмов как с потерями, так и без потерь. Так, к примеру, некоторые алгоритмы используют кодирование Хаффмана, когда арифметическое кодирование является более оптимальным, или же DCT, при существовании алгоритмов, лишенных артефактов, характерных для DCT.

Другим альтернативным направлением можно выделить различные алгоритмы постфильтрации, интрафильтрации, предварительной обработки изображения. Данные алгоритмы позволяют как достичь больших коэффициентов сжатия, так и получить лучшее визуальное отображение при прочих равных условиях. Как подпункт, наверное, сюда также возможно отнести различные алгоритмы, используемые для повышения качества более значимых объектов.

Следующее направление – это улучшение алгоритмов масштабирования видео. Так, к примеру, при применении бикубической интерполяции вместо билинейной можно добиться лучшего визуального эффекта, при одинаковых степенях сжатия видео.

Одним из основных направлений возможного улучшения методов сжатия видеинформации является оптимизация алгоритмов компенсации движения. Без компенсации движения все усилия предварительной обработки могут быть сведены на нет при сильном движении в кадре, потому что действительно один из ключевых этапов процесса сжатия видео. И хотя очень много исследований уже было проведено в этой сфере и существуют множество различных алгоритмов и методов для компенсации движения, но все еще остаются шансы для дальнейшей оптимизации существующих алгоритмов. Именно в этой сфере предполагается продолжить исследования, поэтому различные аспекты, связанные с компенсацией движения, ниже будут рассмотрены более подробно.

Таким образом, мы попытались выделить основные направления для оптимизации существующих методов сжатия видео. Данный обзор не является какой-либо классификацией и ни в коей мере не претендует на полноту и может быть продолжен другими направлениями, а является лишь попыткой убедить, что оптимизация алгоритмов компенсации движения является действительно одним из приоритетных направлений дальнейших исследований. Поэтому перейдем непосредственно к рассмотрению этого вопроса.

Компенсация движения

В общем случае задача компенсации движения может быть разделена на две подзадачи – это задача выделения (распознавания) объектов на изображении и задача определения траектории движения или, в общем случае, характера деформации объекта. Но существующие решения во многом упрощают эти две задачи, останавливая свое внимание лишь на прямоугольных непересекающихся областях и на ограниченном множестве возможных преобразований. При этом для решения второй подзадачи используются различные методы: пиксельные методы (полный перебор, перебор по шаблону, трехмерный рекурсивный поиск), сопоставление блоков, параметрические и основанные на объектном подходе методы.

Так, наиболее популярными и часто используемыми являются алгоритмы компенсации движения, основанные на сравнении блоков. Но среди этих алгоритмов также есть свои особенности и различия. При этом могут использоваться как FSBMC (fixed size block motion compensation), так и VSBMC (variable size block motion compensation) технологии, а также плюсы и минусы каждой из них. Но и на этом разделение еще не заканчивается. И при использовании как FSBMC, так и VSBMC технологий, являющихся

ся всего лишь подходами к разбиению изображения на фрагменты того или иного прямоугольного размера, возможны различные методы поиска решения задачи компенсации движения.

Так, существует множество различных методов, позволяющих найти решение задачи компенсации движения. К ним относятся: трехшаговый, 2-D логарифмический и другие методы поиска. Но, как правило, существенным недостатком этих подходов является то, что они часто останавливаются, находя лишь псевдооптимальное решение, что отрицательно сказывается на эффективности дальнейшего сжатия. Поэтому был разработан целый ряд алгоритмов, решающий данную задачу в той или иной степени методом полного перебора. Данные алгоритмы могут быть разделены на две большие группы: допускающие нахождение псевдооптимального решения (иерархические алгоритмы, фильтрующие алгоритмы, алгоритмы с множественными разрешениями) и не допускающие нахождение псевдооптимального решения (различные быстрые схемы полного поиска, двоичные алгоритмы, алгоритмы исключения кандидатов и т.п.). В общем случае различные схемы полного перебора являются более рабочими по отношению к схемам, в той или иной степени ограничивающим количество возможных решений. Но схемы полного поиска являются намного более требовательными к вычислительным ресурсам системы. Поэтому появляется новая задача: задача оптимизации алгоритма компенсации движения.

В дальнейшем именно для решения проблемы оптимизации алгоритма компенсации движения предполагается использование иерархических методов представления и обработки видеинформации [1, 2]. При этом при кодировании каждого кадра изображения предполагается использовать основные особенности иерархических алгоритмов: возможность мультиразрешения, тем самым обеспечивая реализацию эффективных методов принятия решений в задачах обработки видеинформации; управляемая величина ошибки сжатия, обеспечивающая контроль качества сжатой информации [3].

Работа выполнена при поддержке РФФИ (проект № 05-01-00590)

Список литературы

1. Васин Ю. Г., Жерзев С. В. Технология иерархического кодирования видеинформации // Тезисы 12-й Международной конференции по компьютерной графике и машинному зрению Графикон'2002. Н. Новгород: Изд-во ННГАСУ, 2002. С. 326–333.
2. Yu.G. Vasin and S.V. Zherzdev. Information Techniques for Hierarchical Image Coding // Pattern Recognition and Image Analysis. Vol. 13. № 3. 2003. P. 539–548
3. Александров В. В., Горский Н. Д. Представление и обработка изображений. Рекурсивный подход. – Л.: Наука, 1985.

ИСПОЛЬЗОВАНИЕ СТАТИСТИЧЕСКИХ ДАННЫХ И ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ПРИ ОПРЕДЕЛЕНИИ ОПТИМАЛЬНОЙ КОНФИГУРАЦИИ АППАРАТНОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СУБД

В.М. Конюхова, Н.А. Шулаева

*Елабужский филиал Казанского государственного технического
университета им. А. Н. Туполева*

Целью данной работы является определение оптимальной конфигурации сервера, работающего с базами данных. Производится оценка текущей нагрузки сервера, затем моделируется нагрузка под управлением других операционных систем и/или другого оборудования. Определяются преимущества и недостатки возможных конфигураций.

В разных областях техники, в организации производства, в социальной сфере и в военном деле постоянно возникает необходимость решения вероятностных задач, связанных с работой систем массового обслуживания разного вида требований. Термин «массовое обслуживание» предполагает многократную повторяемость ситуаций в том или ином смысле (много прибывших в систему и обслуженных заявок, большое число находящихся в эксплуатации аналогичных систем) и статистическую устойчивость картины. По справедливому замечанию виднейшего советского специалиста в теории массового обслуживания (ТМО) Б.В. Гнеденко, легче указать ситуации, где не может быть использована ТМО, чем перечислить все сферы ее потенциального применения. Успешно моделируется функционирование центрального процессора и многих других элементов персонального компьютера и информационных сетей.

В качестве начального объекта исследования был выбран анализ работы веб-сервера под управлением ОС Windows Vista с установленной СУБД MySQL. Рассматривалось время выполнения запросов в типичных интернет-приложениях: форумах, чатах, новостных службах. Объект выбирался в силу своей доступности, открытости исходного кода и соответствия задачам проекта.

Первый этап: внедрение в рассматриваемый проект средств подсчета времени, необходимого на выполнение запроса к СУБД. Результаты работы приложения в течение некоторого промежутка времени записываются в статистические файлы. Основная задача этапа – определить частоту и характер используемых запросов (запросы на выборку различной сложности, добавление, редактирование).

Второй этап: определение характеристик СУБД и объема накопленных данных для дальнейшего тестирования. Также возможно указание предпочтительных направлений моделирования. На этом этапе в автоматическом режиме определяются модели, по которым пройдет трассировка.

Третий этап: выполнение набора параметров для созданных моделей. В качестве моделей выступают рассмотренные заранее компьютеры, для которых собраны статистические данные по выполнению тех или иных одиночных запросов. Этот достаточно емкий по времени этап предоставляет пользователю окончательные результаты для анализа.

Очевидно, что анализу может быть подвергнуто любое приложение, использующее SQL в качестве языка запросов к базе данных и обеспечивающее сбор статистики по выполнению запросов. Естественное ограничение накладывает лишь специфика приложения: наиболее сложными и ресурсоемкими вычислительными задачами должны быть именно задачи обращения к СУБД.

СИНТЕЗ ОПТИМАЛЬНОЙ СТРАТЕГИИ ОБСЛУЖИВАНИЯ ПОТОКА ОБЪЕКТОВ ДВУМЯ НЕЗАВИСИМЫМИ MOBILE-ПРОЦЕССОРАМИ В УЗЛОВОЙ РАБОЧЕЙ ЗОНЕ

М.Б. Резников

Волжская госакадемия водного транспорта

Доклад посвящен формированию модели и синтезу оптимальной стратегии обслуживания двумя независимыми mobile-процессорами [1] композитного потока объектов, состоящего из конечного числа подпотоков. Рабочая зона Ξ mobile-процессоров может быть представлена графом Ψ – это пучок отрезков Λ_s ($s = \overline{1, l}$) с общей точкой Ω (рис. 1). Каждый объект поступает в рабочую зону через «свою» точку входа (одну из граничных точек рабочей зоны) и перемещается поступательно с постоянной скоростью до момента выхода из Ξ через другую «свою» граничную точку согласно заданному маршруту следования.

Процессоры осуществляют сервисное обслуживание объектов, автономно перемещаясь в пределах рабочей зоны.

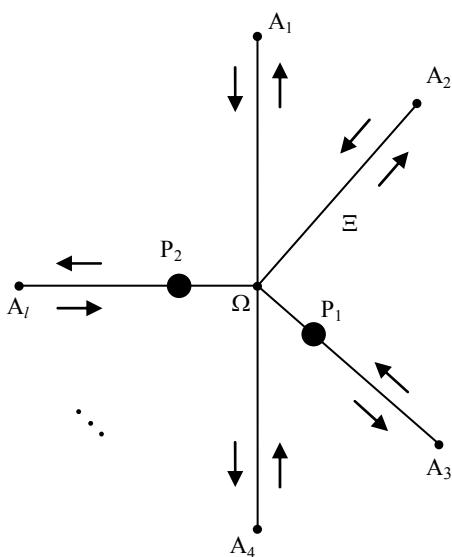


Рис. 1.

Процессор, совершающий обслуживание, перемещается совместно с объектом со скоростью последнего. При прохождении рабочей зоны каждый объект может быть обслужен последовательно обоими процессорами или одним процессором (в обоих случаях не более одного раза). Допустимы ситуации, при которых объект проходит зону Ξ без обслуживания.

На содержательном уровне формулируемая модель адекватно описывает сравнительно новую для районов интенсивного речного судоходства технологию технического обслуживания транспортных единиц, транзитно проходящих рабочую зону Ξ – область ответственности сервисного предприятия. Обслуживание транспортных единиц осуществляется «на ходу» специализированными самоходными судами, каждое из которых предназначено для выполнения только одного вида работ. Основная задача диспетчера сервисного

предприятия заключается в реализации такой стратегии управления обслуживанием нуждающихся в услугах транзитных судов, которая обеспечивает сервисному предприятию получение суммарного максимального дохода.

Введем следующие обозначения: $O_n = \{o(1), o(2), \dots, o(n)\}$ – поток объектов, проходящих рабочую зону Ξ ; P^1, P^2 – обслуживающие процессоры, автономно перемещающиеся в рабочей зоне с постоянными скоростями u^1 и u^2 соответственно. Осуществ-

вляемое процессором P^l (P^2) обслуживание каждого очередного объекта – однофазное, без прерываний.

В зоне Ξ задано m путей Ξ^k , соответствующих всем возможным сочетаниям точек входа и выхода объекта ($m = C_l^2$, $k = \overline{1, m}$). Поток O_n состоит не более чем из m линейных бинарных подпотоков O^k таких, что каждый объект подпотока O^k перемещается только по своему пути Ξ^k ($O^i \cap O^j = \emptyset$, $i = \overline{1, m}$, $j = \overline{1, m}$).

Каждый объект $o(i)$ характеризуется целочисленными параметрами:

$t(i)$ – момент поступления в зону Ξ ($0 \leq t(1) \leq t(2) \leq \dots \leq t(n)$);

$\tau^l(i)$, $\tau^2(i)$ – нормы длительности обслуживания процессорами P^l и P^2 ($\tau^l(i) > 0$, $\tau^2(i) > 0$);

$\eta(i)$ – указатель принадлежности подпотоку O^k ($\eta(i) = k$, если $o(i) \in O^k$);

$\mu(i)$ – указатель направления перемещения ($\mu(i) = 1$, если индекс точки входа в Ξ меньше индекса точки выхода из неё, и $\mu(i) = 0$, если индекс точки входа в рабочую зону больше индекса точки выхода);

$v(i)$ – скорость движения, которая измеряется числом проходимых в единицу времени элементарных участков;

$w^l(i)$, $w^2(i)$ – величина дохода за обслуживание процессорами P^l и P^2 соответственно ($w^l(i) > 0$, $w^2(i) > 0$).

Местоположение объекта на графе Ψ определяется путевым расстоянием от соответствующей ему точки входа в рабочую зону.

Для идентификации месторасположения процессора P^l в рабочей зоне введем вектор (x^j, y^j) , где x^j – значение индекса s отрезка Λ_s , на котором находится процессор ($x^j \in [1; l]$), а y^j – расстояние от Ω до точки расположения процессора P^l на графике Ψ ($j = \overline{1, 2}$).

Для описания стратегии обслуживания объектов процессором P^l введем кортеж $\rho^j = \{(a_r^j, b_r^j), (a_2^j, b_2^j) \dots (a_d^j, b_d^j)\}$, где a_r^j – номер обслуживаемого объекта, b_r^j – координата объекта в момент начала обслуживания, d^j – число обслуживаемых объектов ($r \in [1; d^j]$).

Общую стратегию обслуживания в потока объектов O_n в рабочей зоне Ξ определим как $\rho = (\rho^l, \rho^2)$. Тогда задача синтеза оптимальной стратегии ρ^* , обеспечивающей максимизацию суммарного дохода W за обслуживание, может быть записана в виде следующего соотношения:

$$W(\rho^*) = \max_{\Theta} \left(\sum_{r=1}^{d^1} w^l(a_r^1) + \sum_{r=1}^{d^2} w^2(a_r^2) \right),$$

где Θ – множество допустимых стратегий обслуживания.

Для решения сформулированной экстремальной задачи разработан и программно реализован алгоритм на основе идеологии динамического программирования. Алгоритм включен в разрабатываемый автором в среде Microsoft Visual Studio 2005 пакет для моделирования и решения задач оперативного управления технологическими процессами на внутреннем водном транспорте [2]. В табл. 1 приведен пример для четырехэлементного потока объектов, демонстрирующий эффективность использования оптимальной стратегии (ОС) по сравнению с более простой (жадной) стратегией обслуживания (ЖС), которая заключается в назначении обслуживающему процессору ближайшего к нему объекта, запрашивающего обслуживание.

Таблица 1

i	T	τ^I	τ^2	$\eta(i)$	v	w^I	w^2	r^I (ЖС)	r^2 (ЖС)	r^I (ОС)	r^2 (ОС)
1	1	30	20	1	15	1000	500	1	1	1	1
2	5	60	40	2	20	2000	700	2	2	2	2
3	8	15	10	3	-15	1000	500		3	3	
4	10	20	20	1	-12	700	1000	3			3
W								5400		6200	

Список литературы

1. Коган Д. И., Федосенко Ю. С. Задача синтеза оптимального расписания обслуживания бинарного потока объектов в рабочей зоне mobile-процессора // Вестник Нижегородского университета. Математическое моделирование и оптимальное управление. 1999. Вып. 1(20). С. 179–187.
2. Резников М. Б., Федосенко Ю. С. Синтез оптимальных расписаний обслуживания бинарного детерминированного потока объектов в двухпроцессорной системе транспортного типа // Технологии Microsoft в теории и практике программирования. Материалы конференции / Под ред. проф. Р. Г. Стронгина. – Нижний Новгород. Изд-во Нижегородского госуниверситета, 2006. С. 254–256.

ОПРЕДЕЛЕНИЕ СРОКОВ РЕАЛИЗАЦИИ ПРОЕКТА С ИСПОЛЬЗОВАНИЕМ MS EXCEL И MATLAB

Е.А. Паршкова

Нижегородский госуниверситет им. Н.И. Лобачевского

В докладе изучается возможность применения средств MS Excel и пакета MATLAB при сетевом планировании и управлении.

Одним из основных понятий системы методов СПУ является понятие проекта, или комплекса работ. Под комплексом работ понимается совокупность операций, необходимых для достижения определённой цели. Основой применения СПУ является представление комплекса работ в виде сетевого графика.

Одно из важнейших понятий сетевого графика – понятие пути (любая последовательность работ, в которой конечное событие каждой работы совпадает с начальным событием следующей за ней работы). Среди различных путей сетевого графика наибольший интерес представляет полный путь L – любой путь, начало которого совпадает с исходным событием сети, а конец – с завершающим. Наиболее продолжительный полный путь в сетевом графике называется критическим.

Граф представляется своей матрицей инцидентности. Её построению обычно предшествует упорядочение. Среди методов расслоения (упорядочения) графа выделяются методы Демукрона и метод, использующий транзитивное замыкание вершин.

Вычисление по первому методу производится с помощью MS Excel. Этот метод основан на преобразовании матрицы инцидентности графа.

Для второго метода, который представляет собой построение матрицы транзитивного замыкания, был использован MATLAB.

Расслоения, полученные этими методами, идентичны.

Одной из основных целей сетевого планирования является нахождение критического пути в сетевом графике. Для этого существует множество методов. В работе при помощи пакета MATLAB таблиц Excel реализуются алгоритмы Форда и Беллмана – Калаба.

В результате выполнения алгоритма Форда получается вектор ранних свершений всех событий графа и вершины, составляющие критический путь. Этот алгоритм достаточно просто выполняется с помощью пакета MATLAB. Для рассматриваемого численного примера была написана соответствующая программа Ford.m.

Алгоритм Беллмана – Калаба является некоторой модификацией алгоритма Форда на основании идеи динамического программирования. Для его выполнения необходимо решить систему уравнений

$$v_i = \max_j(t(i, j) + v_j), \quad i = 0, 1, \dots, n - 2, \\ v_{n-1} = 0,$$

где v_i ($i=0, 1, \dots, n-1$) представляет величину оптимального пути от вершины i до конечной вершины.

Данный алгоритм реализуется с помощью MS Excel. Вместо $-\infty$, необходимой для вычислений, требуется взять любое большое по модулю число. В данном случае в качестве такого значения было выбрано -10000 .

Классическим методом определения критического пути является нахождение резервов событий и работ – критические работы и события резервов времени не имеют. Однако этот способ достаточно трудоёмкий – необходимо просчитать 11 параметров. С помощью MS Excel это нетрудно, но увеличивается вероятность ошибки. В этом смысле лучшим представляется алгоритм Форда, так как с помощью программного пакета MATLAB он реализуется достаточно быстро и требует для своего выполнения только лишь матрицу инцидентности графа. Алгоритм Беллмана – Калаба, основанный на первом методе, также реализуется довольно легко.

Список литературы

1. Ануфриев И., Смирнов А., Смирнова Е. MATLAB 7: В подлиннике: Наиболее полное руководство. – СПб.: БХВ-Петербург, 2005.
2. Афанасьев М. Ю., Суворов Б. П. Исследование операций в экономике: модели, задачи, решения: Учеб. Пособие. – М.: Инфра-М, 2003.
3. Додж М., Стinson К. Эффективная работа с Microsoft Excel 2000. – СПб.: Питер, 2002.
4. Исследование операций в экономике: Учеб. пособие для вузов / Под ред. проф. Н.Ш. Кремера. – М.: ЮНИТИ, 2005.
5. Кофман А., Дебазей Г. Сетевые методы планирования и их применение. – М.: Прогресс, 1968.

РАЗРАБОТКА ЛАБОРАТОРНОГО ПРАКТИКУМА ПО СОЗДАНИЮ ЭЛЕМЕНТОВ ЭКОНОМИЧЕСКИХ ИС С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ .NET

С.А. Тюрина, И.Г. Чернопятова, О.П. Макарова, Е.Ю. Малышева

Тольяттинский государственный технический университет сервиса (ТГУС)

Постановка задачи

Переход на платформу .NET в настоящее время актуален для многих информационных систем, в том числе и для экономических. Невозможно представить себе современную экономическую информационную систему без клиент-серверных и распределенных баз данных, без использования Internet/Intranet технологий, без многоуровневой архитектуры. Большие возможности технологии .NET, ее огромная библиотека классов, возможность использования различных языков программирования и ASP.NET, работа на разных платформах делают задачу применения технологии .NET в экономических информационных системах безусловно интересной, актуальной и востребованной.

Нашей задачей была разработка лабораторного практикума по типовым ситуациям, которые встречаются практически во всех экономических информационных системах. Практикум содержит разделы по созданию баз данных, вводу и обработке информации, работе с локальными, клиент-серверными и распределенными базами данных, работа в локальной сети и с использованием сети Internet, элементы имитационного моделирования для подсистем поддержки принятия решений. Дополнительные требования – практикум должен быть реализован на сквозном примере, проходящем через все разделы и программная реализация различных разделов по возможности должна опираться на один и тот же базовый набор объектов и классов библиотеки .NET.

Этапы разработки и содержание практикума

Этапы создания лабораторного практикума в целом соответствуют структуре разработанного курса. Первым был создан раздел «Работа с локальными базами данных», далее разделы «Работа с клиент-серверными и распределенными базами данных в локальных сетях в VB.NET», «Работа с клиент-серверной базой данных с использованием ASP.NET» и «Элементы имитационного моделирования в VB.NET».

Работа с локальными базами данных в VB.NET

Исторически раздел возник на базе соответствующих лабораторных работ в среде Delphi с технологией ADO. В качестве СУБД используются Access и Microsoft SQL Server. При формулировке заданий акцент делался на подчеркивании сходства и различия разработки программных продуктов в разных средах и на особенностях использования объектов ADO и ADO.NET для работы с базами данных – представление данных в памяти и отображение на форме, минимизация открытых соединений, ввод, обработка и редактирование данных через SQL-запросы и хранимые процедуры, особенности навигации по наборам данных.

Работа с клиент-серверными и распределенными базами данных в локальных сетях в VB.NET

Это центральный раздел. Он предусматривает создание элементов распределенной информационной системы с распределенной базой данных. Раздел «Работа с клиент-серверными и распределенными базами данных в локальных сетях в VB.NET» состоит из двух типов приложений (администратора офиса и операторов подразделений) и нескольких баз данных (клиент-серверная в офисе и локальные в подразделениях), между которыми организован обмен информацией. В приложении администратора заполняются справочные таблицы, данные из которых копируются в справочные таблицы подразделений и впоследствии периодически обновляются, а также имеется возможность просмотра операционных таблиц, которые обновляются данными из локальных баз данных подразделений. В подразделении на основании поступивших справочных данных ведется работа с операционными таблицами и передача данных в базу администратора.

Работа с клиент-серверной базой данных с использованием ASP.NET

Вариант предыдущей задачи, но вся информация хранится в единой базе данных (Microsoft SQL Server), работа с которой реализована через Web-приложение, созданное в ASP.Net. Обращается внимание на сходство и различие использования компонентов ADO.Net в Windows-приложениях и в Web-приложениях.

Элементы имитационного моделирования в VB.NET

Имитационное моделирование реализовано для систем массового обслуживания – магазины, транспорт, поликлиники, служба скорой помощи и т.п. Для моделирования системы используются принципы объектно-ориентированного проектирования и язык UML, программная реализация в виде Windows-приложения в VB.NET. Для хранения параметров моделирования, данных моделируемого процесса, получаемых характеристик системы используется базы данных Access.

Выводы

Практикум реализован в дисциплинах «Проектирование ИС» и «Системы автоматизированной обработки экономической информации», т.о., студенты специальности «Прикладная информатика в экономике» третьего и четвертого курса приобретают опыт разработки элементов экономических информационных систем в среде .NET.

В лекционном курсе содержатся разделы по основам программирования в VB.NET и C#, поэтому некоторые студенты выполняют практикум на языке C#. В дальнейшем планируется все примеры приводить на двух языках – VB и C#.

Список литературы

1. Шумаков П.В. ADO.NET и создание приложений баз данных в среде Microsoft Visual Studio.NET. М.: Диалог-МИФИ , 2003 – 528 с.
2. Гандэрлой. ADO и ADO.NET. Полное руководство. М.: Корона, 2003. – 912 с.

МНОГОАГЕНТНАЯ ПЛАТФОРМА ДЛЯ СОЗДАНИЯ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ

Э.В. Шогулин, В.В. Андреев

Чувашский госуниверситет

Постановка задачи

Перед разработчиками ставилась задача спроектировать и разработать многоагентную платформу, позволяющую распараллеливать ресурсоемкие задачи на различные хосты и иметь возможность использования разработанной платформы для изучения применения многоагентных платформ для решения различных задач.

Основные положения

В данной работе под агентом понимается сущность, которая объединяет данные, код и способна перемещаться между различными окружениями, в которых выполняется код агента. Многоагентная платформа прозрачна для разработчика алгоритма вычисления. Для реализации агента достаточно реализовать класс, пронаследованный от базового класса агентов Agent, и перегрузить необходимые методы.

На рис. 1 показан простейший агент, выполняющий параллельно три свои части на различных хостах. Так, Function1 и Function2 выполняются на других хостах. Перед началом их выполнения агент портируется на эти хосты и после выполнения вызвавший их агент получает результат их работы. Рис. 2 содержит схему портирования агента на примере задачи, состоящей из трех шагов, второй из которых позволяет распараллелить вычисление. Функционирование многоагентной платформы происходит следующим образом. Агент запускается на хосте № 1. После вычисления первой подзадачи происходит распараллеливание подзадачи на четыре части, одна из которых выполняется на том же хосте. Платформа портирует агента на хосты под номерами 2, 3 и 4 и запускает соответствующие подзадачи.

Как видно на рис. 2, при выполнении шага № 2.2 происходит ещё одно распараллеливание вычисления на три части, одна из которых выполняется на том же хосте № 3. После того как платформа портирует того же агента на хосты № 5 и № 6, происходит выполнение шагов № 2.2.1 и № 2.2.2. После завершения каждого шага результат возвращается вызвавшему агенту, выполнившему свою часть подзадачи и ожидающему выполнения всех остальных распараллеленных подзадач. Так результаты вернутся агенту на хосте № 3 и на хосте № 1. После этого задача продолжит вычисляться на шаге № 3.

Архитектура

Разработчики многоагентной платформы поставили перед собой задачу сделать архитектуру платформы максимально гибкой и понятной. Любые компоненты платформы могут быть замены пользователями, и технология их замены проста и однотипна.

```

public class CalculationAgent: Agent
{
    #region private
    private Object Function1(Object param)
    {
        Thread.Sleep(60000);
        return "Function1 result";
    }
    private Object Function2(Object param)
    {
        Thread.Sleep(40000);
        return "Function2 result";
    }
    private Object Function3(Object param)
    {
        Thread.Sleep(20000);
        return "Function3 result";
    }
    #endregion

    public override Object Run(Object param)
    {
        FunctionEvent functionEvent1 = StartThread(
            new RunFunctionHandler(Function1));
        FunctionEvent functionEvent2 = StartThread(
            new RunFunctionHandler(Function2));
        Object functionEvent3Value = Function3(null);

        Wait(functionEvent1, functionEvent2);

        return Convert.ToString(functionEvent1.Value) +
            Convert.ToString(functionEvent2.Value) +
            Convert.ToString(functionEvent3Value);
    }
    public override void Stop()
    { }
}

```

Рис. 1

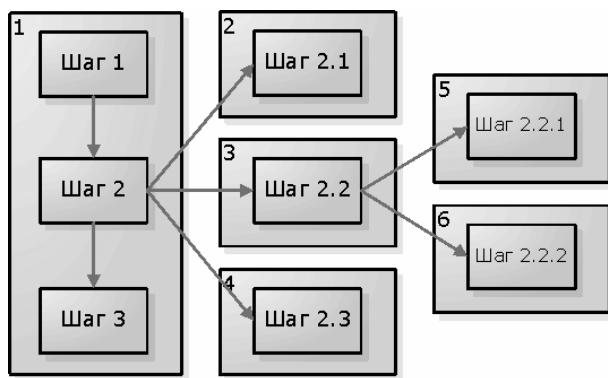


Рис. 2

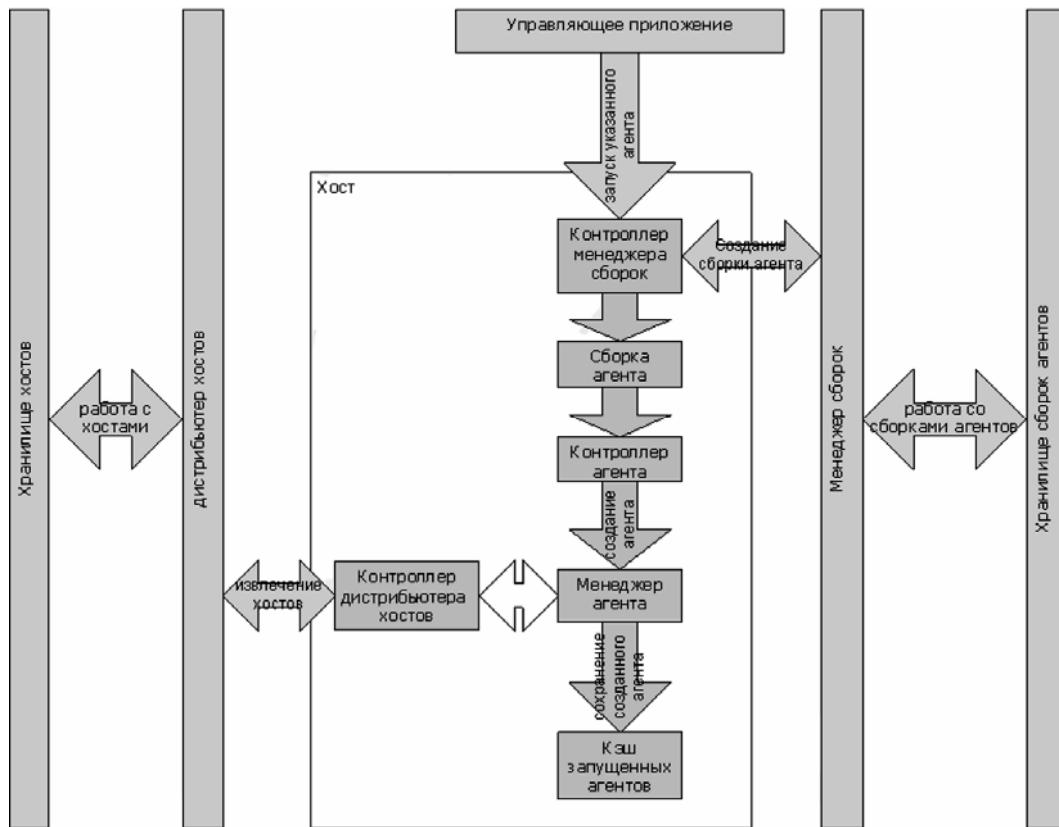


Рис. 3

Также это означает, что при удалении или неактивности какого-либо компонента многоагентной платформы сможет далее функционировать, за исключением функциональности удаленного или неактивного компонента. При этом гибкость и понятность архитектуры не должна быть реализована за счет потери производительности.

Программная платформа .NET, на которой была построена рассматриваемая многоагентная платформа, является компонентно-ориентированной. За счет этого разработчикам удалось добиться преследуемых целей.

На рис. 3 приведены основные компоненты разработанной многоагентной платформы, используемые хостом. Стрелками показаны взаимодействия различных компонентов. Направление стрелок указывает направление взаимодействия. Светлыми стрелками показано взаимодействие с внешними компонентами – наиболее дорогое взаимодействие. Темными стрелками показаны внутренние взаимодействия – наиболее дешёвое взаимодействие. Светлыми прямоугольниками показаны внешние компоненты, темными прямоугольниками – внутренние компоненты.

Использование платформы для распараллеливания ресурсоемких вычислений

Платформа предоставляет возможность распараллеливания ресурсоемких задач с использованием математического пакета MATLAB. Для этого была разработана управляемая оболочка для функций MATLAB.

Платформа для построения бизнес приложений

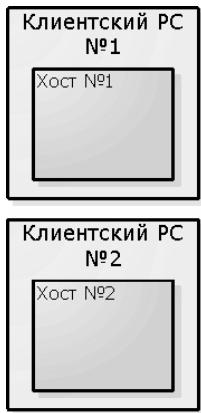


Рис. 4

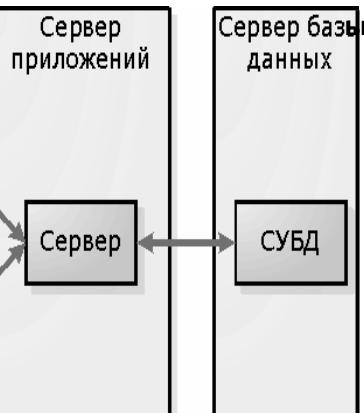
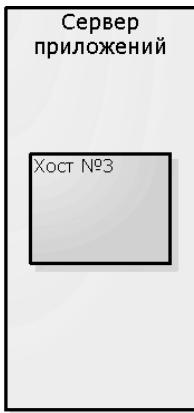


Рис. 5

Для большинства бизнес-приложений мы можем выделить некоторую общую функциональность, задача которой заключается в предоставлении пользователю возможности сохранять, читать и отображать некоторые сущности, такие как, например: клиент, счет, заказ, договор и так далее. Эти сущности, как правило, известны под названием бизнес-сущности. Типичное бизнес-приложение, как правило, имеет трехзвенную архитектуру: клиент, сервер приложений, СУБД. При этом каждое звено делится на слои. На рис. 4 отображена архитектура обычного трехзвенного бизнес-приложения. Различные звенья в бизнес-приложении связывает потоки данных. Каждое из звеньев уникально и выполняет отведенную ему роль. Если же бизнес-приложение выполнено на многоагентной платформе, то её архитектура будет выглядеть, как представлено на рис. 5. Основные отличия заключаются в статичности компонентов приложения в случае типичной реализации и в динамике компонентов приложения в случае реализации на базе многоагентной платформы. Рассмотрим многоагентную платформу в работе. При запуске агента платформа выглядит, как показано на рис. 6. При вводе данных формируется агент, который выполняет свою работу на клиентском компьютере № 1. Например, здесь может осуществляться первичная проверка введенных данных с целью уменьшения сетевого трафика. Если данные верны, то агент перемещается на сервер приложений согласно рис. 7. Здесь происходит основная работа агента. После отработки агента на сервере приложений агент, согласно бизнес-логике, сохраняется и возвращается клиенту, что отображено на рис. 8. После возвращения агента клиентское приложение может отобразить изменившегося агента. Кроме этого платформа обладает возможностью портировать агента на различные хосты, являющиеся серверами приложений. На какой хост будет портирован агент, определяется платформой во время вы-

полнения агента в соответствии со следующими критериями: качество сетевого соединения, производительность хоста, загруженность хоста, надежность хоста.

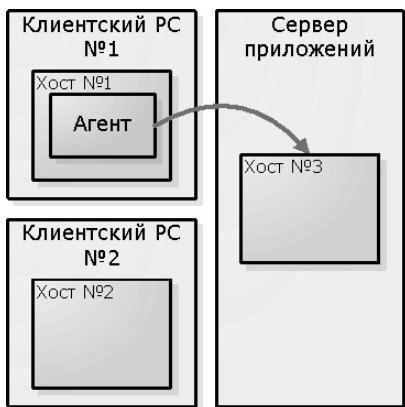


Рис. 6

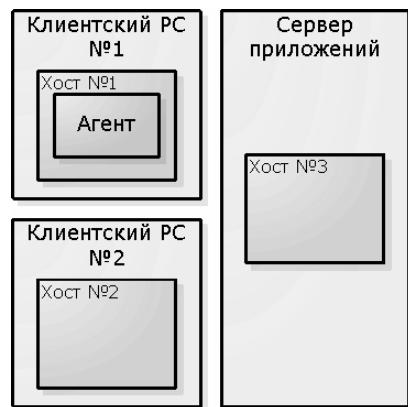


Рис. 7

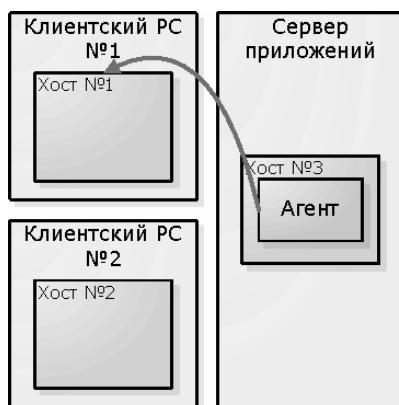


Рис. 8

Итак, подчеркнем наиболее значимые отличия реализации бизнес-приложений на основе многоагентной платформы от обычной и выделим преимущества или недостатки.

- 1) *Отсутствие связи между отдельными звеньями (частями) приложения.*
- 2) При разработке приложения на базе многоагентной платформы от разработчика скрыта логика передачи агентов между различными звеньями приложения. Достоинствами в этом случае являются устойчивость к сбоям и равномерная загруженность серверов. Многоагентная платформа обладает статистическими данными о надежности хостов, которые и учитываются при выборе хоста для портирования агента. Так же при принятии решения при выборе хоста для портирования используется такой критерий, как загруженность хоста.
- 3) *Реализация всей логики работы с сущностью в одном типе.*
- 4) Достоинствами здесь является удобство разработки и отладки. Недостатком является отсутствие явного разделения функциональности агента.

- 5) *Простота и скорость разработки.*
- 6) Многоагентная платформа представляет собой ещё один более высокий уровень абстракции, используемый при разработке бизнес-приложений. Преимуществом является готовая реализация типовой функциональности, такой как, например, сохранение объектов, управление объектами. Недостатком может являться снижение быстродействия при использовании этой функциональности.
- 7) **Сдача вычислительных мощностей в аренду.**
- 8) При портировании агента осуществляется перенос не только данных (свойств агента), но и кода (код агента). Это позволяет выполнять код заказчика на мощностях исполнителя. При этом многоагентная платформа обладает инфраструктурой, необходимой как для заказчика (управление агентами), так и для исполнителя (оптимальное распределение агентов между хостами, система безопасности).

МЕТОД АДАПТАЦИИ ДЛЯ ЗАДАЧ БЕЗУСЛОВНОЙ МИНИМИЗАЦИИ СРЕДНИХ ПОТЕРЬ

А.В. Большаков, О.А. Кузенков

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

В статье предлагается новый метод для решения задач безусловной минимизации средних потерь, доказывается его сходимость с вероятностью 1.

Постановка задачи

Пусть конечная управляемая система имеет управляющие воздействия v_i , $i = \overline{1, n}$. Вероятности выбора воздействия v_i на j -м шаге – $x_i(j)$. При выборе i -го управляющего воздействия функция штрафа $a(v_i) = a_i$ является случайной величиной с математическим ожиданием $Ma_i = M_i$, не зависящим от времени, и с постоянной дисперсией.

Пусть $\xi_i(j)$ – индикатор выбора воздействия v_i на j -м шаге.

Введем функции:

$$b_i(j) = \begin{cases} a_i(j), & \text{если } \xi_i(j) = 1; \\ 0, & \text{если } \xi_i(j) = 0; \end{cases}$$

$$H_i(j) = \left(\sum_{k=1}^j b_i(k) \right) \cdot \left(\sum_{k=1}^j \xi_i(k) \right)^{-1} \text{ при } \sum_{k=1}^j \xi_i(k) \neq 0,$$

неопределенность 0/0 доопределим единицей. Величины $H_i(j)$ имеют смысл среднего значения величин a_i , полученных к j -му шагу. Их можно рассматривать как промежуточные оценки эффективности воздействий v_i .

Пусть δ – некоторая малая положительная константа, $\delta < 1/\sqrt{n}$, тогда определим следующие равенства:

$$\delta_j = \delta(j+n)^{-1/2},$$

$$\alpha_i(j) = \frac{\exp(\sum_{k=1}^j H_i(k)(1-n\delta_k))}{\exp(\sum_{k=1}^j H_1(k)(1-n\delta_k)) + \dots + \exp(\sum_{k=1}^j H_n(k)(1-n\delta_k))}, \quad i = \overline{1, n},$$

$$x_i(j) = \alpha_i(j)(1-n\delta_j) + \delta_j. \quad (1)$$

В начальный момент времени величины $x_i(0)$ удовлетворяют неравенствам

$$\delta_0 \leq x_i(0) \leq 1 - \delta_0(n-1), \quad i = \overline{1, n}.$$

Для алгоритма адаптации (1) справедливы следующие утверждения:

Утверждение 1. В данном алгоритме вероятность выбора управляющего воздействия v_i любое конечное число раз, начиная с любого фиксированного шага, стремится к нулю при количестве шагов, стремящемся к бесконечности.

Утверждение 2. В данном алгоритме вероятность P_∞^k выбора управляющего воздействия v_i на бесконечной серии испытаний любое конечное число раз k равна 0 ($P_\infty^k = 0$).

Утверждение 3. В данном алгоритме число выбора каждого управляющего воздействия v_i при бесконечном числе шагов бесконечно с вероятностью 1.

Утверждение 4. Если $M_k > M_i$ для всех $i \neq k$, то для данного алгоритма справедливо $x_k(j) \rightarrow 1$ при $j \rightarrow \infty$ с вероятностью 1.

Проводились сравнения с существующими алгоритмами на модельных примерах. Было выявлено, что в ряде случаев эффективность предложенного в статье алгоритма превосходит известные аналоги.

Список литературы

1. Кузенков О.А. Математическое моделирование процесса выбора оптимальной стратегии: проблема качества // Математическое моделирование и оптимальное управление / Сб. научных трудов под редакцией Р.Г. Стронгина. Н. Новгород, 1994.
2. Кузенков О.А. Математическое моделирование процесса выбора для биологических систем // Математическое моделирование и оптимальное управление / Сб. научных трудов под редакцией Р.Г. Стронгина. Н. Новгород, 1996.
3. Кузенков О.А. О системе оценок объективного критерия // Математическое моделирование и оптимальное управление / Сб. научных трудов под редакцией Р.Г. Стронгина. Н. Новгород, 1998.

АСПЕКТЫ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ПОДДЕРЖКИ ПРОЦЕССОВ ПРОЕКТИРОВАНИЯ

К.В. Безрук

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматривается система интеллектуальной поддержки процессов проектирования. Данная система построена как оболочка экспертной системы продукционного типа.

Знания в системе представляются в виде продукционных правил. Антецеденты правил представляют собой набор предикатов, объединенных операцией конъюнкции.

Примером предиката является отношение «=». Использование предикатов позволяет описывать все антецеденты общим образом, а также просто и быстро добавлять в реализацию программной системы новые предикаты в случае необходимости. В программной системе существует механизм добавления предикатов в виде дополнительных модулей — динамических библиотек Windows (DLL), исключающий необходимость перекомпиляции программной системы.

Консеквент каждого правила состоит из набора элементарных операций. Все операции можно разделить на две группы. Первая группа операций — это операции присвоения значений переменным вывода. Вторая группа операций — это вызов определенной процедуры, которая должна осуществлять проектное действие. Процедура получает в качестве входных параметров текущие значения переменных вывода и возвращает результат своей работы, который сохраняется в одну из переменных вывода. Доступные процедуры добавляются в систему с помощью библиотек DLL.

Алгоритм вывода в продукционных системах разбивается на 3 этапа: выбор правил кандидатов на срабатывание; разрешение конфликта; срабатывание выбранного правила. Каждый из этапов реализуется в виде отдельного класса, представляющего собой шаблон СТРАТЕГИЯ. Это позволяет, комбинируя различные подходы на каждом из этапов алгоритма вывода, получать различные стратегии логического вывода. В общем случае если есть M методов выбора кандидатов, N методов разрешения конфликта, то можно сконструировать $M \times N$ различных алгоритмов вывода. Выбор конкретной структуры алгоритма вывода является прерогативой пользователя.

Этап срабатывания правил однозначно определяется содержимым консеквента правила. В процессе срабатывания правил изменяется состояние процесса вывода, определяемого значениями переменных вывода.

Состояние процесса вывода описывается двумя векторами переменных. Первый вектор — это набор логических переменных, характеризующих и управляющих процессом вывода. Второй вектор переменных — это переменные, описывающие текущее состояние самого объекта или процесса проектирования. Этот вектор переменных позволяет интеллектуальной системе выступать в роли координирующего звена на любом этапе проектирования, аккумулируя в себе все данные о процессе проектирования, полученные от пользователя или от различных систем автоматизации проектирования, и передавая эти данные из одних программных средств САПР к другим.

Поскольку значения, присваиваемые переменным вывода, могут иметь самую разнообразную природу, то был создан механизм универсального представления значений. Поскольку в основе проектирования системы лежало свойство открытости, то случае необходимости введения в систему новых типов значений, которые являются необходимыми для настройки интеллектуальной системы на определенный этап проектирования или упрощают описание правил проектирования для данного этапа, они могут быть добавлены путем создания и программной реализации дополнительных модулей. Новый тип данных, вводимый в систему, должен указать, какие предикаты могут быть к нему применены. Типы данных реализуются с использованием шаблонов ПРОТОТИП и ФАБРИЧНЫЙ МЕТОД.

Во всех случаях для создания конкретных экземпляров расширяемых типов используются классы, реализующие шаблон АБСТРАКТНАЯ ФАБРИКА.

Интеллектуальная система должна обладать возможностями приобретения и генерации новых знаний. Процесс приобретения новых знаний реализуется через специально созданный пользовательский интерфейс. Генерация новых знаний осуществляется в автоматическом фоновом режиме. В основе этого процесса алгебра исчисления предикатов первого порядка. Как алгоритмы генерации используются различные модификации метода резолюции, а также методы вывода на графе связей и на графике дизъюнктов. Использование алгоритмов вывода на графике связей или дизъюнктов позволяет говорить о возможности увеличения эффективности и сокращения времени работы алгоритма в случае использования методов параллельного программирования. Запуск процесса генерации новых знаний может быть инициирован пользователем либо происходит автоматически в случае долгого нахождения системы в состоянии покоя.

Для генерации новых знаний важным является формирование новых теорем, которые потом будут отображены в соответствующие правила. Формирование данных теорем возможно двумя способами.

Первый способ предполагает, что пользователь формулирует некоторое выражение. Интеллектуальная система проверяет, может ли данное выражение являться теоремой, основываясь на текущем состоянии базы знаний. Если выдвинутое пользователем утверждение выводимо в теории, описываемой текущей базой знаний, то утверждение добавляется как теорема в базу правил. Если утверждение противоречит текущей теории, то пользователь информируется об этом. Пользователь также получает информацию, с каким правилом утверждение вступило в противоречие. Есть два пути — отказалось от утверждения или удалить правило, с которым было противоречие, и проверить выводимость утверждения опять. Этот выбор делается пользователем. Если утверждение невыводимо, но и не вступает в противоречие с текущей теорией, то пользователь может признать данное утверждение истинным, и оно будет добавлено в базу знаний.

Интеллектуальная система непрерывно следит за частотой использования правил и цепочек правил. Если правило не используется в течение долгого времени – то оно становится кандидатом на удаление. Если некоторая цепочка правил срабатывает постоянно, то система формирует новое правило, которое заменит данную длинную цепочку. В любой момент можно получить статистику использования правил и статистику цепочек. Это упрощает инженеру по знаниям анализ базы знаний и соответственно делает более понятным, какие новые правила следует добавить, а какие удалить.

Рассмотренная структура программной системы позволяет говорить о том, что она обладает свойством открытости.

Программная реализация системы выполнена на языке C++ в среде Microsoft Visual Studio. Программная реализация использует возможности динамических библиотек Microsoft Windows, что делает ее легко расширяемой и настраиваемой на различные этапы проектирования.

МЕТОДИКА СИНТЕЗА ЛИНЕЙНЫХ КОМБИНАЦИОННЫХ СХЕМ В НЕЙРОСЕТЕВОМ БАЗИСЕ

П.Ю. Белокрылов

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

Развитие современных вычислительных систем в направлении цифровых архитектур параллельного действия, а также нейроархитектур, реализующих существенный параллелизм вычислений в нейросетевом логическом базисе, ставит в разряд наиболее актуальных проблемы формализации задач под выбранную архитектуру вычислительной системы.

Постоянное совершенствование архитектуры цифровых вычислительных систем, усложнение их конструктивно-технологической базы от интегральной и функциональной микроэлектроники, использование цифровых модулей в реализациях нейрокомпьютеров предъявляют особые требования к методам и технологиям их проектирования. Развитие этих методов, в свою очередь, способствует появлению и внедрению в вычислительную практику новых архитектурных решений, прослеживается положительная обратная связь, определяющая высокие темпы прогресса в данной предметной области.

Традиционные методы синтеза комбинационных схем предполагают переход от таблицы истинности к СДНФ реализуемых функций, их оптимизацию по определенным критериям и реализацию в заданном логическом базисе (на основе программируемой логической матрицы или матричной логики, в виде логического блока табличного типа, в виде универсального логического модуля на основе мультиплексоров, в малом логическом базисе — на вентильном уровне).

Синтез комбинационной схемы в малом логическом базисе (SLC – Simple Logic Cells) предоставляет широкие возможности для поиска наиболее рациональных реше-

ний. Однако применение традиционных методов минимизации функций алгебры логики и алгоритмов их реализации носит эвристический характер и порождает обширное множество вариантов схем, среди которых не всегда оказываются лучшие.

Методика

Предлагается подход, который состоит в сведении проблемы SLC-синтеза комбинационной логической схемы к синтезу эквивалентной бинарной (или биполярной) нейронной сети с последующим покрытием ее цифровыми элементами.

Синтезируемая комбинационная схема представляется в виде искусственной нейронной сети (ИНС), составными элементами которой являются спецпроцессоры двух видов:

- искусственные нейроны (или просто нейроны), преобразующие поступающие на их входы сигналы в единственный выходной сигнал в соответствии с заданной активационной функцией нейрона;
- связи между нейронами, реализующие межнейронные взаимодействия в виде бинарных (принимающих значения 0 или 1) или биполярных (принимающих значения 1 или -1) сигналов, умножаемых на синаптические веса связей.

За основу синтезируемой сети берется многослойный персептрон со всевозможными межнейронными связями, не нарушающими его свойства прямонаправленности (каждый нейрон предшествующих слоев связан со всеми нейронами последующих слоев и никакими другими). Число входов персептрана равно числу аргументов $x = (x_1, x_2, \dots, x_N)$ реализуемых логических функций, а число выходов — числу реализуемых функций $y = (y_1, y_2, \dots, y_M)$. Количество внутренних (скрытых) слоев и число нейронов в них задается эвристически (желательно минимально необходимое).

Процесс синтеза нейронной сети организуется через обучение персептрана с помощью эволюционно-генетических алгоритмов оптимизации [2]. При этом роль множества обучающих шаблонов $P = \langle X^q, Y^q \rangle, q = \overline{1, Q}$ выполняет исходная таблица истинности, т.е функциональная таблица автомата ($X^q = (X_1^q, X_2^q, \dots, X_N^q)$ — q -й вектор известных значений аргументов реализуемых функций, $Y^q = (Y_1^q, Y_2^q, \dots, Y_M^q)$ — вектор соответствующих значений функций, Q — мощность обучающего множества, т.е число строк таблицы истинности, равное 2^N для полностью определенных функций).

В процессе обучения векторы $X^q, q = \overline{1, Q}$, из множества P последовательно подаются на вход персептрана, и для каждого из них оценивается ошибка между фактическим y^q и желаемым Y^q откликом (выходом) сети $e_q = \frac{1}{2} \|y^q - Y^q\|^2, q = \overline{1, Q}$. Затем

определяется общая ошибка $e = \sum_{q=1}^Q e_q$, на основании которой алгоритм обучения осуществляет модификацию значений настроенных параметров сети, направленную на уменьше-

ние ошибки. Обучение продолжается до тех пор, пока сеть не приобретет способность выполнять требуемый тип преобразования, заданный набором шаблонов P .

Из известных результатов моделирования цифровых преобразований в нейросетевом базисе следует упомянуть работу В. Уидроу [3], в которой предложена модель сети с нейронными аналогами цифровых элементов И и ИЛИ. При этом класс моделируемых логических схем оказывается существенно ограниченным из-за неполноты логического базиса (отсутствия нейромодели инвертора).

В данной работе полнота базиса И, ИЛИ, НЕ обеспечивается следующим образом.

Предполагается, что вес синаптической связи выхода j -го нейрона с входом i -го нейрона w_{ji} может принимать значения $\{0, 1, -1\}$. Значение 0 говорит об отсутствии данной межнейронной связи. Значение 1 свидетельствует о наличии обычной связи с весом, равным 1. Значение -1 означает межнейронную связь через инвертор. Для адекватного функционирования нейромодели обычные бинарные сигналы $\{0, 1\}$ заменены соответственно на $\{-1, 1\}$.

В рамках предложенной модели i -й нейрон, представляющий логический элемент И, имеет активационную функцию следующего вида:

$$y_i = f(\sum_j w_{ji}x_j - w_{i0}) = \begin{cases} 1, & \text{если } \sum_j w_{ji}x_j \geq w_{i0} \\ -1, & \text{если } \sum_j w_{ji}x_j < w_{i0} \end{cases}, \quad (1)$$

где y_i — сигнал на выходе i -го нейрона; x_j — сигнал на выходе j -го нейрона, воздействующего на i -й нейрон ($x_j = y_j$); $w_{i0} = \sum_j |w_{ji}|$ — порог активационной функции i -го нейрона.

Для i -го нейрона, представляющего логический элемент ИЛИ, активационная функция имеет вид:

$$y_i = \begin{cases} 1, & \text{если } \max_j w_{ji}x_j = 1 \\ -1, & \text{если } \max_j w_{ji}x_j \neq 1 \end{cases}. \quad (2)$$

Для описания эволюционно-генетической модели нейронной сети использована схема непосредственного кодирования, специфицирующая генными кодами каждую синаптическую связь исходной избыточной нейронной сети (0 означает разрыв связи, 1 — наличие непосредственной связи, 2 — связь через инвертор) и вид активационной функции каждого нейрона (0 отмечает «мертвый» нейрон, 1 — нейрон И, 2 — нейрон ИЛИ). При этом генетический код особи эволюционирующей популяции представлен в виде двух составляющих. Первая составляющая ставит в соответствие каждому слову нейронов участки хромосомы, определяющие его связи со всеми последующими слоями. Вторая фиксирует типы активационных функций для каждого нейрона скрытых слоев и выходного слоя в порядке возрастания их номеров сверху вниз и слева направо, начиная с нейронов входного слоя [4].

Предложенная структура хромосомного набора предполагает возможность применения оригинальных схем генетических операторов репродукции и отбора решений, учитывающих специфику рассматриваемой предметной области. К таковым следует

отнести многоточечный кроссовер с заданными вероятностями распределения точек разрыва по слоям, оператор *изоляции «мертвых» нейронов*, оператор выделения локальных популяций по критерию *ярусности*, оператор миграции по вышеозначенному критерию и другие.

В ходе эволюции персептрона избыточной структуры последовательно получаются все допустимые в ее рамках эквивалентные варианты бинарных сетей, реализующие исходную таблицу истинности (для них ошибка e на выходе должна быть нулевой). Среди них возможно присутствие идентичных структур, построенных на различных наборах нейроэлементов (число таких структур напрямую зависит от степени избыточности исходного персептрона). Из них для дальнейшего рассмотрения оставляется только одна.

После выполнения очередных операций кроссинговера и мутации в структуре сети могут появляться нейроны с разорванными связями по всем входам и выходам (значения синаптических весов всех входных и выходных связей равны 0). Также могут возникать ситуации, когда генетические операторы порождают «мертвые» нейроны (активационная функция такого нейрона помечена нулевым значением генного кода в хромосоме). В таких случаях фиксируется факт *отмирания* нейронов, и значения весов их входных и выходных связей принимаются равными 0.

Наряду с критерием обеспечения требуемой функциональности устройства, к нему могут также предъявляться требования по критериям аппаратной сложности и быстродействию. С точки зрения быстродействия предпочтение отдается схемам с минимальным числом ступеней преобразования входных сигналов (схемам минимальной *ярусности*, представляемым нейронными сетями с минимальным количеством слоев). Аппаратно более простыми, как правило, являются многоярусные схемы. Получаемые в ходе эволюции эквивалентные биполярные сети после тривиальной процедуры покрытия их элементами цифрового базиса подвергаются верификации и введению в них вспомогательных элементов (например, элементов задержки сигналов), обеспечивающих их работоспособность. Следует заметить, что процедуры верификации и введения вспомогательных элементов могут быть выполнены в нейросетевом логическом базисе (до покрытия сети цифровыми элементами). После выполнения процедур покрытия и верификации полученных схемных решений они могут быть оценены по различным критериям качества (быстродействию, аппаратной сложности и другим) с целью выбора из них конкретного оптимально-компромиссного варианта.

Важно отметить, что предложенный подход, отличаясь высокой степенью формализации и унификации, позволяет осуществлять структурный синтез в базисе других логических функций, обладающих свойством полноты. Для этого достаточно сменить виды активационных функций нейронов и, возможно, взять за основу не биполярную, а бинарную нейронную сеть. При этом эволюционно-генетическая модель, схема ее кодирования могут выглядеть значительно проще, чем для базиса И, ИЛИ, НЕ.

Учитывая потенциальную способность предложенного подхода генерировать все возможные схемные решения в заданном логическом базисе, его можно успешно использовать для решения проблемы декомпозиции схем сложной комбинационной логики, интеграция которой на одном кристалле в силу определенных причин оказывается невозможной. Для этого следует рассматривать исходную избыточную структуру нейронной сети как предельную в смысле аппаратной сложности и осуществлять ее обучение по распределенному критерию, оценивающему ошибку отклонения на выходе по каждой из реализуемых функций в отдельности. В случае получения приемлемого решения, обеспечивающего нулевую ошибку по всем выходам, проблема декомпозиции

отпадает. В противном случае нереализованные логические функции могут стать предметом рассмотрения в рамках отдельной нейронной сети, возможно гораздо менее сложной.

Выводы

Таким образом, предложенный подход к решению проблемы синтеза схем произвольной комбинационной логики в сочетании с классическими методами синтеза цифровых автоматов на абстрактном и структурном уровнях имеет высокую степень формализации и унификации, а также потенциальную способность генерировать все возможные схемные решения в любом логическом базисе.

Список литературы

1. Глушков В.М. Синтез цифровых автоматов.— М.: Физматгиз, 1962.— 476 с.
2. Батищев Д.И. Генетические алгоритмы решения экстремальных задач / Под ред. Я.Е. Львовича: Учеб. пособие. – Воронеж, 1995. – 60 с.
3. Widrow B., Hoff M. Adaptive switching circuits // Proc.IRE WESCON Convention Record, 1960. – Р. 107–115.
4. Басалин П.Д., Белокрылов П.Ю. Синтез комбинационных логических схем в нейросетевом базисе. Вестник ВГАВТ. Межвузовская серия Моделирование и оптимизация сложных систем. Вып. 1. 2006.
5. Угрюмов Е.П. Цифровая схемотехника. – СПб.: БХВ-Петербург, 2001. – 528 с.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МАРШРУТИЗАТОРА ПАКЕТОВ НА ОСНОВЕ ЛОГИКО-АЛГЕБРАИЧЕСКОЙ МОДЕЛИ

В.Г. Вантеев, С.А. Зинкин

Пензенский государственный университет

Задача

Описать метод программной реализации модели маршрутизатора пакетов, представленной ингибиторной временной сетью Петри.

Краткое описание проведенного исследования

В качестве базового формализма, пригодного для решения поставленной задачи, выбран формализм алгоритмически эволюционирующих многоосновных алгебраических систем, позволяющий организовать согласованную работу сети алгоритмических модулей. Для согласования и координации модулей предлагается использовать фундаментальные концепции систем искусственного интеллекта: каузальные (причинно-следственные) и темпоральные (временные) отношения, предметные отношения и функции.

Общая методология проектирования основана на использовании новых парадигм сетевых информационных технологий и моделей согласования и координации объектов

и процессов через общее пространство функций и предикатов. В процессе проектирования используются логико-алгебраические модели систем распределенной и параллельной обработки информации.

Сеть абстрактных машин (СеAM), или эволюционирующая многоосновная алгебраическая система (ЭМАС), определяется следующим набором:

$$N = (A_1, \dots, A_n, F_C, F_{Ob}, P_C, P_{Ob}, U_F, U_P, M, \Omega_{Top}, \Omega_{Cop}, L, \Omega_{Lop}, \Phi, B, \Omega_B, \mu_{update}, \mu_{test}, \mu_{block}, \mu_B, \mu_{MB}, \mu_L, C),$$

где

A_1, \dots, A_n – непустые непересекающиеся множества (основы);

$A = \{A_1, \dots, A_n\}$ – множество основ;

$\Sigma = F_C \cup F_{Ob} \cup P_C \cup P_{Ob} = F \cup P$ – текущая сигнатура;

F_C – множество «каузальных» функций;

F_{Ob} – множество «объектных» функций;

P_C – множество «каузальных» предикатов;

P_{Ob} – множество «объектных» предикатов;

U_F – множество элементарных обновлений функций из Σ ;

U_P – множество элементарных обновлений предикатов из Σ ;

$U = U_F \cup U_P$ – система образующих алгебры модулей;

M – множество модулей СеAM, реализующих преобразования (обновления) текущей сигнатуры Σ ;

Ω_{Top} – система темпоральных операций, принимающих значения в множестве модулей M ;

Ω_{Cop} – система дополнительных операций («управляющих конструкций» СеAM), принимающих значения в множестве модулей M ;

L – множество логических условий;

Ω_{Lop} – система логических операций, принимающих значения в множестве логических условий L ;

Φ – множество атомарных формул (атомов) – система образующих алгебры логических условий;

B – множество блоков СеAM;

Ω_B – множество темпоральных операций, выполняемых в блоках модулей СеAM;

$\mu_{update} : M \rightarrow \mathbf{P}(\Sigma)$ – отображение, сопоставляющее каждому модулю подмножество функций и предикатов, которые им модифицируются, \mathbf{P} – символ булевана;

$\mu_{test} : M \rightarrow \mathbf{P}(\Sigma)$ – отображение, сопоставляющее каждому модулю подмножество функций и предикатов, которые им проверяются;

$\mu_{block} : M \rightarrow \mathbf{P}(\Sigma)$ – отображение, сопоставляющее каждому модулю подмножество функций и предикатов, с которыми оперирует данный модуль, причем

$$(\forall m \in M)(\mu_{block}(m) = \mu_{update}(m) \cup \mu_{test}(m)),$$

(в общем случае для корректной работы сети функции и предикаты, с которыми оперирует конкретный модуль, должны быть временно заблокированы в целях недопущения нежелательных взаимоблокировок конкурирующих модулей);

$\mu_B : B \rightarrow \mathbf{P}(U_F \cup U_P)$ – отображение, ставящее в соответствие каждому блоку подмножество совместимых элементарных обновлений сигнатуры Σ ;

$\mu_{MB} : M \rightarrow \mathbf{P}(B)$ – отображение, сопоставляющее каждому модулю СeAM подмножество реализуемых им блоков;

$\mu_L : M \rightarrow \mathbf{P}(L)$ – отображение, сопоставляющее каждому модулю подмножество проверяемых им логических условий;

C – отношение каузации, представленное областью истинности одноименного предиката. Этот предикат определяется следующим образом:

$$p_{inter}(\mu_{update}(m_i), \mu_{test}(m_j)) \supset C(m_i, m_j),$$

где

$$p_{inter}(\mu_{update}(m_i), \mu_{test}(m_j)) = \begin{cases} \text{true, если } \mu_{update}(m_i) \cap \mu_{test}(m_j) \neq \emptyset \\ \text{false, если } \mu_{update}(m_i) \cap \mu_{test}(m_j) = \emptyset \end{cases}$$

Известно, что изучение алгебраических систем позволяет сочетать как алгебраические, так и логические понятия и методы. При этом рассматривается язык $L(F, P)$, алфавит которого содержит счетное число переменных x_1, x_2, \dots , функциональные символы из F , предикатные символы из P , символ равенства, логические связки, кванторы всеобщности и существования. Формулы и термы составляются по известным в многосортном исчислении предикатов первого порядка [6–9] правилам. Отличительной особенностью предлагаемого формализма является использование термов вида

$$(\tilde{\exists}!x \in X)p(x), (\tilde{\exists}!!x \in X)p(x), (\tilde{\forall}x \in X)p(x), (\tilde{\forall}!!x \in X)p(x) \quad (1)$$

в α -выражениях модулей СeAM (то есть в записи формул логических условий α -дизъюнкции и α -итерации). Назовем данные операторы квантифицированными операторами выбора картежей из областей истинности n -арных ($n \geq 1$) предикатов. Например, в случае применения введенных операторов к бинарным предикатам соответствующие термы запишем в следующем виде:

$$\begin{aligned} &(\tilde{\exists}!x \in X, y \in Y)q(x, y), (\tilde{\exists}!!x \in X, y \in Y)q(x, y), (\tilde{\forall}x \in X, y \in Y)q(x, y), \\ &(\tilde{\forall}!!x \in X, y \in Y)q(x, y). \end{aligned} \quad (2)$$

Рассмотрим группу термов (1). Результатом выполнения оператора $\tilde{\exists}!$ является единственное значение предметной переменной x , выбранное произвольным образом из области истинности унарного предиката p . Оператор $\tilde{\exists}!!$ выбирает значение x при ус-

ловии, что оно является единственным в области истинности унарного предиката p .

Оператор $\tilde{\forall}$ позволяет выбрать все значения переменной x из области истинности предиката p . Оператор $\tilde{\forall}!!$ выбирает все значения переменной x из области истинности предиката p при условии, что эта область совпадает с областью определения данного предиката.

Одноименные операторы в термах, заданных выражениями (2), выполняют аналогичные действия над бинарными предикатами. В результате вычисления термов $(\exists!x \in X, y \in Y)q(x, y)$ и $(\exists!!x \in X, y \in Y)q(x, y)$ находятся по одному кортежу, а результатами вычисления термов $(\forall x \in X, y \in Y)q(x, y)$ и $(\forall!!x \in X, y \in Y)q(x, y)$ являются отношения. Результаты вычисления термов обеих групп используются в блоках согласованных обновлений сигнатуры Σ .

Ю. Гуревичем [4, 5] предложено использовать в машинах абстрактных состояний специальные операции – элементарные обновления функций и предикатов. Элементарное правило обновления функции или предиката запишем в виде правила вывода

$$\frac{t_1, t_2, \dots, t_k, t_{k+1}}{s(t_1, t_2, \dots, t_k) \leftarrow t_{k+1}},$$

где t_1, t_2, \dots, t_k – термы различных сортов, в том числе термы, определенные выражениями (1), s – функциональный или предикатный символ. В случае, если s – функциональный символ, t_{k+1} – суть терм любого сорта, а если s – предикатный символ, то t_{k+1} – булево выражение.

В отличие от модели Ю. Гуревича, мы допускаем использование в блоках согласованных обновлений операций реляционной алгебры, рассматривая каждое алгебраическое выражение как функцию, которая отображает множество отношений в одно отношение (отношения определены областями истинности одноименных предикатов из Σ). Операторы реляционной алгебры в нашем случае являются сокращающими символами для обозначения множества согласованных обновлений сигнатуры Σ .

В отличие от обычных, используемых в многоосновном исчислении предикатов, термов, значения термов с квантифицированными операторами выбора $\tilde{\exists}!$, $\tilde{\exists}!!$, $\tilde{\forall}$, $\tilde{\forall}!!$ не всегда могут быть вычислены, то есть не удается осуществить выбор ни одного кортежа из области истинности некоторого предиката. В этом случае будем считать, что данный оператор выполняет «тождественное» обновление R^E , то есть он не вносит ни одного изменения в сигнатуру Σ . Дадим попутно еще одно определение модуля, как выражения в алгебре модулей, все вхождения переменных в котором связаны обычными кванторами или квантифицированными операторами выбора.

Модель маршрутизатора пакетов, представленная ингибиторной временной сетью Петри [2], может быть представлена рядом СeAM-выражений на основе рассмотренного выше формализма. Ниже приведен список некоторых из них:

$$t_7 = [p_7(a_7)](\{p_7(a_7) \leftarrow \text{false}, p_8(a_8) \leftarrow \text{true}, p_9(a_9) \leftarrow \text{true}\} \vee R^E);$$

$$t_8 = [p_9(a_9)](\{p_9(a_9) \leftarrow \text{false}, p_{10}(a_{10}) \leftarrow \text{true}, p_{11}(a_{11}) \leftarrow \text{true}, \\ p_{12}(a_{12}) \leftarrow \text{true}, p_{13}(a_{13}) \leftarrow \text{true}, p_{38}(a_{38}) \leftarrow \text{true}\} \vee R^E);$$

$$t_9 = [p_{10}(a_{10}) \& p_1(a_1)](\{p_{10}(a_{10}) \leftarrow \text{false}, p_1(a_1) \leftarrow \text{false}, p_{14}(a_{14}) \leftarrow \text{true}, \\ p_{30}(a_{30}) \leftarrow \text{true}\} \vee R^E);$$

$$t_{17} = [p_{10}(a_{10}) \& \neg p_1(a_1)](\{p_{10}(a_{10}) \leftarrow \text{false}, p_1(a_1) \leftarrow \text{false}, \\ p_{18}(a_{18}) \leftarrow \text{true}\} \vee R^E);$$

$$t_{19} = [p_{20}(a_{20})](\{p_{20}(a_{20}) \leftarrow \text{false}, p_{19}(a_{19}) \leftarrow \text{true}\} \vee R^E);$$

$$t_{29} = [p_{38}(a_{38}) \& p_{33}(a_{33}) \& \neg p_{31}(a_{31})](\{p_{38}(a_{38}) \leftarrow \text{false}, p_{33}(a_{33}) \leftarrow \text{false}, \\ p_{31}(a_{31}) \leftarrow \text{false}, p_{36}(a_{36}) \leftarrow \text{true}\} \vee R^E);$$

$$t_{37} = [p_8(a_8) \& p_{36}(a_{36})](\{p_8(a_8) \leftarrow \text{false}, p_{36}(a_{36}) \leftarrow \text{false}, \\ p_1(a_1) \leftarrow \text{true}\} \vee R^E).$$

Здесь t – модули, представляющие переходы рассматриваемой модели; $p(a)$ – функции предикатов, реализующие позиции модели. Данные СеAM-выражения легко представить в виде исполняемого кода на любом языке программирования, например, C++:

```
If(p7 == 1)
{
    p7 = 0;
    p8 = 1;
    p9 = 1;
}
```

Заключение

В данной статье рассмотрен метод программной реализации модели маршрутизатора пакетов, представленной ингибиторной временной сетью Петри. Предлагаемый формализм является универсальным инструментом и позволяет реализовывать на его основе новые интегрированные (сетевые и информационные) технологии проектирования.

Список литературы

1. Зинкин С.А. Интеграция сетевых и информационных технологий на основе парадигм искусственного интеллекта // Новые информационные технологии и системы: Труды VII Международной конференции, Пенза, Пенз. гос. ун-т, 2006. – С. 108–117.
2. Ильин В. П., Смирнов М. И. Модели процессов управления в вычислительных сетях на базе временных ингибиторных сетей Петри // Автоматика и вычислительная техника. – 1989. – № 5. – С. 66–69.

ИСПОЛЬЗОВАНИЕ ГРАФОВЫХ МОДЕЛЕЙ ДАННЫХ В ЗАДАЧЕ ДОСТУПА К ДАННЫМ ХРАНИЛИЩ СУЩЕСТВЕННО РАЗЛИЧНОГО ФОРМАТА

И.Н. Крылова, А.В. Линёв

Нижегородский госуниверситет им. Н.И. Лобачевского

Введение

К настоящему моменту создано и находится в эксплуатации множество геоинформационных систем. Некоторые из них разработаны на основе широко распространенных пакетов, другие являются оригинальными разработками, при этом большинство систем используют оригинальные модели описания и форматы хранения геоинформационных данных. Это достаточно часто приводит к ситуации, когда имеется описание одной и той же области в различных форматах, и необходимо получать данные одновременно из нескольких источников.

В НИИ ПМК проводятся исследования проблемы использования данных, рассредоточенных по депозитариям (хранилищам) существенно различного формата. Предлагаемая методология создания программных комплексов доступа к данным депозитариев существенно различной структуры, архитектура данного комплекса и характеристики модулей описаны в [1, 2]. Отличительной особенностью подхода является ориентация на выделение в обрабатываемых хранилищах объектов предметной области и их моделирование посредством классификационных графов [4]. В данной работе описывается один из возможных алгоритмов установления соответствия между двумя моделями в виде классификационных графов и реализация переноса данных.

Архитектура программного комплекса

Программный комплекс доступа к данным депозитариев существенно различной структуры должен предоставлять целевому приложению интерфейс использования данных в некотором заранее выбранном формате. При этом данные могут размещаться в нескольких различных хранилищах, разработка которых выполнялась различными разработчиками с использованием различных средств и форматов. Соответственно, до начала работы целевого приложения необходимо определить множество данных, которое может быть ему предоставлено, и способ их формирования; при работе приложения – необходимо обеспечить доступ к данным. Рассмотрение некоторых типов хранилищ: реляционная база данных, электронная карта в формате интегрального файла, векторная база данных в формате DXF, проект MapInfo – показало, что

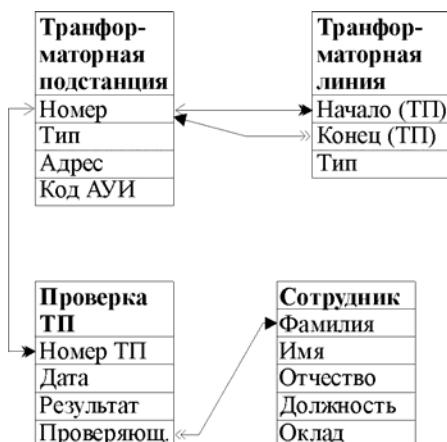


Рис. 1. ER-модель с дополнительными направлениями уточнения

в конкретных случаях для них возможно выделение множества объектов предметной области и описание их в виде классификационного графа [5]. На рис. 1–2 представлен пример построения классификационного графа по ER-модели.

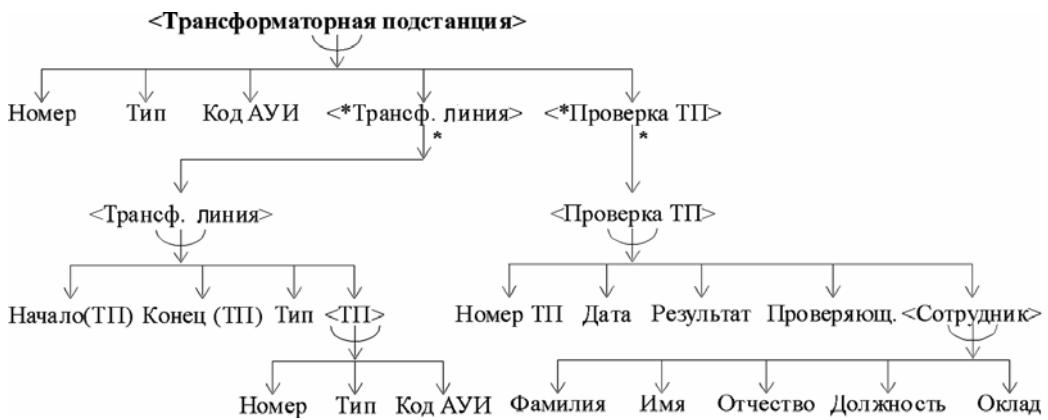


Рис. 2. Построенный классификационный граф

Для каждого типа хранилища разрабатывается методология создания его объектной модели – необходимо выделить все типы объектов хранилища и определить структуру объектов каждого типа. В результате работы программных блоков создания объектных моделей мы получим описание множества классификационных графов, которыми представляются структуры конкретных типов объектов. Архитектура соответствующей части программного комплекса представлена на рис. 3.

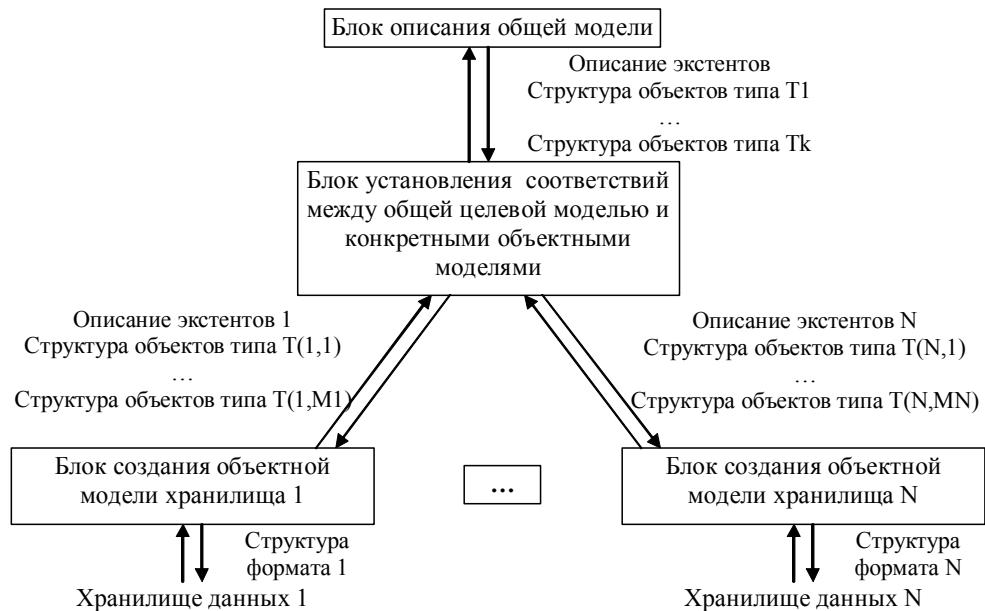


Рис. 3. Архитектура подсистемы установления соответствий между моделями

После установления соответствия между моделями можно получить доступ к конкретным экземплярам объектов. Архитектура данной подсистемы представлена на

рис. 4. Структура схожа со структурой подсистемы работы с моделями, и ее блоки используют результаты работы соответствующих им блоков предыдущей подсистемы.

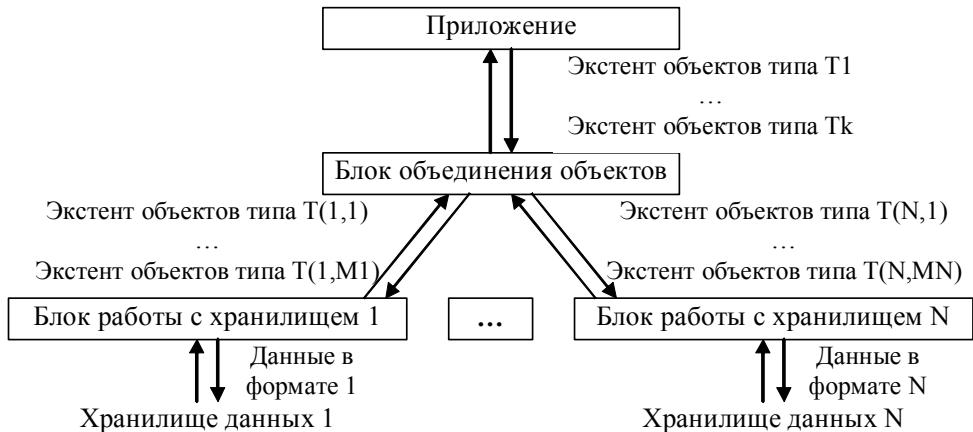


Рис. 4. Архитектура подсистемы доступа к данным

В результате работы подсистемы целевому приложению предоставляется доступ к данным в виде, определенном при описании «общей» модели на этапе установления связей между моделями.

Модели и экземпляры объектов

Описание структуры объектов приводится в виде классификационного графа [4], в котором присутствуют только мультиплективные, итеративные и элементарные вершины. Соответственно, экземпляры объектов представляют собой дерева. В случае если для какой-либо элементарной вершины в хранилище не было значения, она все равно создается со специальным значением «нет значения» (это необходимо для сохранения целостности структуры объекта). Пример классификационного графа, описывающего структуру объекта, и дерева, представляющего собой экземпляр объекта, приведен на рис. 5.

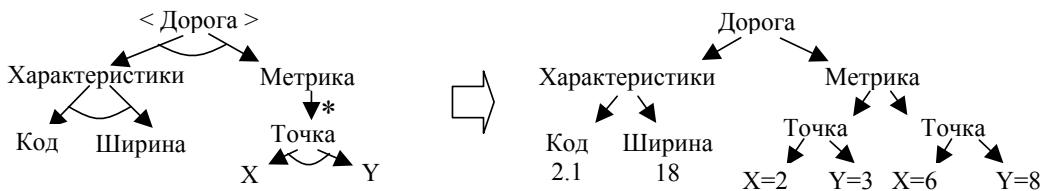


Рис. 5. Модель объекта и экземпляр объекта

Архитектура программного комплекса предполагает совместное использование нескольких хранилищ и моделей в виде классификационных графов. При этом данные на пути от каждого хранилища до целевого приложения должны последовательно пред-

ставляться в рамках нескольких моделей. Для обеспечения выполнения последовательности преобразований достаточно решить следующую задачу: пусть имеются два классификационных графа, описывающих исходную и конечную структуру объекта; необходимо определить способ установления соответствия между ними и установить алгоритм переноса данных.

Установление соответствия и перенос данных

Предлагается следующее решение, которое также позволяет объединять несколько экземпляров. Для его описания необходимо определить следующие понятия.

Контекст – поддерево экземпляра объекта, в котором в каждом входящем в него итеративном кусте выбран один потомок и для каждого итеративного куста сохранен номер потомка. Для каждого набора выбранных элементарных вершин существует множество контекстов, содержащих выбранные вершины и всех их родителей вплоть до корневой вершины. Множество контекстов, полученное при выборе всех элементарных вершин, мы будем называть полным множеством контекстов.

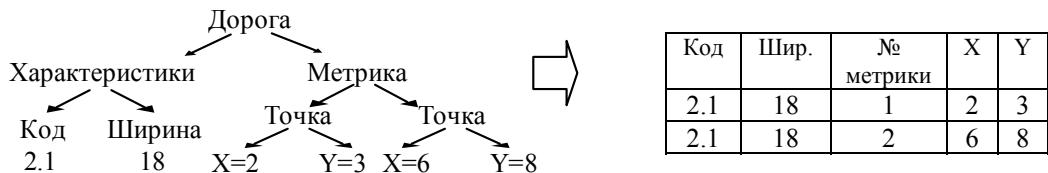


Рис. 6. Пример полного множества контекстов экземпляра

Ключ экземпляра объекта – поддерево экземпляра объекта, уникально его идентифицирующее. Ключ итеративного куста – поддерево итеративного куста, уникально идентифицирующее ветви экземпляра объекта.

Ключ экземпляра объекта используется при поиске, выборке и объединении экземпляров. Ключ итеративного куста используется при построении экземпляра из множества контекстов. Например, в качестве ключа итеративного куста может использоваться номер потомка – это позволит ветвям куста в результирующем экземпляре разместиться в том же порядке, что и в исходном.

На этапе установления соответствия выполняются следующие шаги:

1. Задается используемая последовательность контекстов исходного экземпляра.

В простейшем случае для этого достаточно указать множество элементарных вершин, участвующих в преобразовании. В более сложных ситуациях можно использовать несколько последовательностей.

2. Указывается способ формирования результирующего контекста.

На основании каждого исходного контекста формируется один результирующий полный контекст (т.е. контекст, содержащий все элементарные и итеративные вершины результирующей модели). В простейшем случае для элементарных и итеративных вершин из исходного контекста указываются элементарные и итеративные вершины модели получателя.

3. Задаются ключи для всех итеративных кустов результирующей модели.

Основными шагами этапа преобразования данных являются:

1. Определение последовательности всех контекстов исходного экземпляра.

2. Формирование последовательности контекстов результирующего экземпляра.
3. Последовательное добавление к результирующему экземпляру сформированных контекстов.

Место контекста в результирующем экземпляре определяется посредством анализа ключей итеративных вершин. Если при добавлении контекста выяснилось, что он противоречит уже сформированному дереву, это считается ошибкой (источником этой ошибки, скорее всего, является неверное указание соответствия между типами объектов).

Заключение

Несмотря на существенные различия в структуре хранилищ различных типов, исследования показывают, что популярные форматы хранения геоинформационных данных позволяют привести их к объектному виду и использовать для описания структуры объектов классификационный граф. Это позволяет разработчикам использовать унифицированный и хорошо известный структурный подход при обработке данных.

Предложенный подход к установлению соответствия между типами объектов и переносу данных достаточно прост при использовании в тривиальных случаях, но допускает вариации, существенно расширяющие его возможности.

Список литературы

1. Васин Ю.Г., Енгулатов Ю.И., Кошелев М.В., Линёв А.В. О технологии разработки систем обмена информацией между депозитариями разнородной структуры. Вестник Нижегородского государственного университета. Математическое моделирование и оптимальное управление. Изд-во ННГУ, Нижний Новгород, № 1 (20), 1999, с. 248–256.
2. Линёв А.В. Технология интеграции данных из хранилищ существенно различного формата на основе объектной модели // Тезисы VII Всероссийской конференции «Методы и средства обработки сложной графической информации». 2003. С. 38–39.
3. Линёв А.В. Технологии и инструментальные средства создания тематических электронных карт в информационной среде депозитариев существенно различного формата // Материалы конференции «Математика и кибернетика 2002», 2002. С. 65–67.
4. Кузин С.Г. Ролевой граф в качестве моделей понятия. В кн. Математическое моделирование и оптимальное управление // Вестник Нижегородского государственного университета. Нижний Новгород, 1998, 2 (19). С. 224–235.
5. Использование объектно-ориентированного подхода при интеграции геоинформационных данных, рассредоточенных по нескольким разноформатным источникам // Отчет по теме «Интеграция» НИИ ПМК, 2002.

ИНФОРМАЦИОННАЯ ОБУЧАЮЩАЯ СИСТЕМА «ПАРАМЕТРИЧЕСКОЕ МОДЕЛИРОВАНИЕ В AUTODESK INVENTOR»

А.А. Титов

*Институт радиоэлектроники и информационных систем,
Нижегородский областной ЦНИИТ, кафедра ГИС НГТУ*

Информационная обучающая система «Параметрическое моделирование в Autodesk Inventor» представляет собой теоретический учебный курс по разделам моделирования, а также практические занятия и упражнения, способствующие более глубокому и быстрому усвоению материала.

Autodesk Inventor – это готовое решение для комплекса задач всего цикла конструкторской подготовки производства; оптимальные инструменты для специализированных производств (тонколистовое проектирование, проектирование сварных конструкций, прокладка технологических трубопроводов и гидропневмосистем, разработка электронных устройств и их компоновка в сборке); моделирование нагрузок и оптимизация конструкции изделия; постоянное расширение спектра решаемых задач в области машиностроительных производств.

Система состоит из 6 разделов:

- *Введение* – основные понятия, основные правила.
- *Теория* – теоретический курс для работы с Autodesk Inventor.
- *Упражнения* – виртуальные ролики и 2 полных примера по созданию сборок (10 виртуальных роликов).
- *Тесты* – тесты по пройденному материалу (40 вопросов).
- *Поиск* – поиск по ключевым словам.
- *Ссылки* – интернет-ресурсы.

Для удобства в информационной обучающей системе существуют дополнительные возможности:

- ссылки – ссылки по конкретной тематике на ресурсы в Интернете;
- загрузка файлов упражнений в соответствующий программный продукт, напрямую из обучающей системы;
- печать страниц напрямую из системы.

Объем ИОС «Параметрическое моделирование в Autodesk Inventor» – 400Mb.

Потенциальными пользователями системы являются студенты, обучающиеся по специальностям «Техническое моделирование», преподаватели (курсы повышения квалификации), а также рядовые пользователи.

Весь учебный материал структурирован по темам, главам и параграфам. Все они снабжены графическим материалом и анимированными иллюстрациями, что помогает пользователю более эффективно познавать изучаемый предмет. Дизайнерское решение информационной системы продиктовано темой проекта. В целом выбранная цветовая гамма создает положительный психологический фон, что способствует изучению материала.

Информационная обучающая система «Параметрическое моделирование в Autodesk Inventor» выполнена при использовании html-технологий, языка программирования Java Script, Flash технологии. Выбор именно этих технологий основан на возможности достаточно быстрой и простой модификации системы, добавления новых разделов, обновления уже имеющейся информации. Также данные технологии обеспечивают защиту прав владельцев текстовой и графической информации от несанкционированного копирования.

Кроме технологии Flash, исполненной в пакете Macromedia Flash MX 2004 для создания ИОС, использовались: Adobe Photoshop CS для обработки графики, Corel Graphics Suite 12. Для создания анимационной заставки, виртуальных роликов и практических занятий использовался программный пакет 3D Studio Max 7.0.

Легкий интерфейс, хорошо структурированное, большое по объему информационное наполнение, наглядность представления основных данных обеспечивают успешное использование разработанной ИОС.

ПРОБЛЕМА ОТБОРА В СИСТЕМАХ РАЗНОСТНЫХ УРАВНЕНИЙ НА ЕДИНИЧНОМ СИМПЛЕКСЕ

О.А. Кузенков, Д.В. Капитанов

Нижегородский госуниверситет им. Н.И. Лобачевского

Целью работы является исследование систем разностных уравнений вида

$$\Delta x_i = F_i(x) \Delta t \quad (1)$$

при выполнении условий

$$x_i \geq 0, \quad \sum_{i=1}^n x_i = 1, \quad i = \overline{1, n}. \quad (2)$$

Здесь и далее под x понимается n -мерный вектор $x = (x_1, x_2, \dots, x_n)$, а $\Delta x_i = x_{i,k+1} - x_{i,k}$. Систему (1), при условиях (2) будем называть системой на симплексе. В [1] доказано, что для системы (1) при условиях (2) справедливо представление

$$\Delta x_i = (\Phi_i(x) - x_i \sum_{j=1}^n \Phi_j(x)) \Delta t; \quad i = \overline{1, n}, \quad (3)$$

где функции $\Phi_i(x) = \sum_{j=1}^n x_j F_i(x / \sum_{j=1}^n x_j)$ положительно однородные и неотрицательные. Также в [1] выведена связь решений систем (1) и (3).

Определение 1. Систему (1) при условиях (2) будем называть системой отбора, если найдется такой номер i , что независимо от начальных условий $x_i(0) \neq 0$ выполняются условия: $\lim_{n \rightarrow \infty} x_i(t_0 + n\Delta t) = 1$; $\lim_{n \rightarrow \infty} x_j(t_0 + n\Delta t) = 0$; $i \neq j$.

Наличие процесса отбора можно интерпретировать как самоорганизацию системы, так как вещества или энергия, первоначально хаотически распределенные между всеми ее элементами, постепенно собираются на одном, в известном смысле самом лучшем. Явление отбора в биофизических [2] и экономических системах соответствует естественному или искусственноому отбору. В [1] сформулирован и доказан ряд теорем, предоставляющих необходимые и достаточные условия отбора.

Определение 2. Систему (1), при условиях (2) будем называть системой, близкой к системе отбора, если существует число $\varepsilon > 0$ и найдется номер N , начиная с которого $\forall n > N$, имеет место неравенство: $x_1(t_0 + n\Delta t) > 1 - \varepsilon$.

Сформулирован и доказан ряд теорем, предоставляющих необходимые и достаточные условия, при которых системы (1), (2) и (3), (2) являются близкими к системе отбора. Кроме того, рассмотрена система вида

$$\Delta x_i = \left(\sum_{j=1}^n f_{ij} b_j x_j - x_i \sum_{k=1}^n \sum_{j=1}^n f_{kj} b_j x_j \right) \Delta t; \quad i = \overline{1, n}; \quad (4)$$

$$f_{ij} > 0, \quad \sum_{i=1}^n f_{ij} = 1, \quad f_{ij} = f_{ji}, \quad f_{ii} = f_{jj}, \quad f_{ii} \gg f_{ij}. \quad (5)$$

Доказывается, что система (4), (2), (5) является близкой к системе отбора.

Список литературы

1. Кузенков О.А., Капитанов Д.В. Системы разностных уравнений на единичном симплексе. МКО Сб. научных трудов. Том 2 // Под ред. Г.Ю. Ризниченко. – М.–Ижевск: НИЦ «Регулярная и хаотическая динамика», 2006. С. 39–50
2. Ризниченко Г.Ю. Математические модели в биофизике и экологии. – М.: Институт компьютерных исследований, 2003.

**СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ
ПЕРВОГО ПОРЯДКА С НЕЛОКАЛЬНЫМИ
КРАЕВЫМИ УСЛОВИЯМИ НА СТАНДАРТНОМ СИМПЛЕКСЕ**

О.А. Кузенков, О.З. Дырдин

Нижегородский госуниверситет им. Н.И. Лобачевского

Рассматриваются системы дифференциальных уравнений в частных производных первого порядка:

$$\frac{\partial z_i}{\partial t} + \sum_{s=1}^k a_s(x, t) \frac{\partial z_i}{\partial x_s} = \Phi_i(x, t, z), \quad i = \overline{1, n}, \quad (1)$$

относительно функций $z_i(x, t)$. Заданные функции $a_s(x, t)$, $s=1, \dots, k$, определенные для любого момента времени $t > 0$, обеспечивают единственность монотонно возрастающего решения

$$\chi = \chi(x^0, t^0, t) \equiv (\chi_1(x^0, t^0, t), \dots, \chi_k(x^0, t^0, t)),$$

начально-краевой задачи для уравнения характеристик

$$\begin{aligned} \frac{\partial \chi}{\partial t} &= a_s(\chi, t), \\ \chi_s(x^0, t^0; t) &= x_s^0, \quad \chi_s(0, t; t) = 0 \quad s = \overline{1, k}. \end{aligned}$$

Заданные в системе уравнений (1) функции $\Phi_i(x, t, z)$ (вообще говоря, нелинейные) определены в некоторой области Π изменения переменных (x, t, z) . В этой области функции $\Phi_i(x, t, z)$ вместе с производными по z непрерывны по совокупности переменных.

Для системы (1) в области $\Omega = \{x = (x_1, \dots, x_k) : x_s \in [0, l_s], s=1 \dots k\}$ (l_s – конечные вещественные постоянные) ставится начальное условие

$$z(x, 0) = \zeta(x), \quad x \in \Omega, \quad (2)$$

и граничное условие

$$z_i(x_{q0}^*, t) = \int_0^{x_q^*(x_{ql}^*, 0; t)} b_q(x_q^*, t) z_i(x_q^*, t) dx_q^*, \quad (3)$$

где $x_{q0}^* = (x_1, \dots, x_{q-1}, 0, x_{q+1}, \dots, x_k)$ и $x_{ql}^* = (x_1, \dots, x_{q-1}, l_q, x_{q+1}, \dots, x_k)$.

Закон согласования краевых и начальных условий имеет вид:

$$\zeta_i(x_1, \dots, x_{q-1}, 0, x_{q+1}, \dots, x_k, 0) = \int_0^{x_q^*(x_{ql}^*, 0; 0)} b_q(x_q^*, 0) z_i(x_q^*, 0) dx_q^*.$$

Область определенности G_T решения начально-краевой задачи (1), (2), (3) задается условиями $G_T: 0 < t < T, 0 < x_s < \chi_s(x_1^*, 0; t), s = 1 \dots k$.

В настоящей работе рассматривается случай, когда решение $z = (x_1, \dots, x_n)^T$ в каждой точке (x, t) из области G_T принадлежит стандартному симплексу

$$S = \{(z_1, \dots, z_n) : z_i \geq 0, i = \overline{1, n}, \sum_{i=1}^n z_i = 1\}.$$

Находятся необходимые и достаточные условия принадлежности систем данного класса стандартному симплексу. Определяются формы представления систем дифференциальных уравнений на стандартном симплексе. Также в работе устанавливается форма вспомогательных систем для решения начально-краевых задач для гиперболической системы на симплексе.

СОДЕРЖАНИЕ

Программный комитет конференции	3
<i>Абрамов Е.С.</i> Моделирование эволюции пучка частиц с учетом собственного заряда.....	6
<i>Алтуфьев М.Ю.</i> MS Visual Studio.NET (C#), MS SQL Server 2000, Gentle.NET, MyGeneration и NUnit при использовании разработки через тестирование для создания автоматизированной системы управления деканатами вуза	8
<i>Аппель И.В.</i> Применение алгоритмов комбинаторной оптимизации в САПР сетей интеллектуальных датчиков	12
<i>Ахметзянов А.И.</i> Онлайновая игра для обучения составлению алгоритмов	15
<i>Белов В.Н., Сарайкин А.С.</i> Векторный графический редактор на основе веб-технологий	16
<i>Белов В.Н., Сарайкин А.С.</i> Реализация модели задачи о ханойских башнях с использованием современных технологий создания трехмерной компьютерной графики	19
<i>Белов В.Н., Сарайкин А.С.</i> Система учета деятельности студентов на уровне кафедры ...	20
<i>Бочкин А.В., Казаков В.Г., Федосин С.А.</i> Система автоматизированного комплексного анализа использования программного обеспечения сетевых компьютерных систем Maxapt QuickEye.....	22
<i>Винокуров С.В.</i> Опыт разработки интеллектуальной системы медицинской диагностики на платформе .NET FRAMEWORK	25
<i>Гафнер Е.А., Аппель И.В.</i> Портал сотрудничества в исследовательской работе «взаимодействие».....	29
<i>Генералов К.А.</i> Программная среда для разработки и распараллеливания генетических алгоритмов	31
<i>Герасимов А.М., Колчин П.А., Кудряшов П.П., Фоменков С.А.</i> Автоматическая коррекция дефекта «красных глаз» на цифровых фотографиях	34
<i>Герасимов А.М., Колчин П.А., Фоменков С.А.</i> Использование семантических зависимостей при поиске физических эффектов.....	36
<i>Герасимов А.М., Фоменков С.А.</i> Соционика и взаимодействие человека и компьютера	38
<i>Горбунова А.С., Козинов Е.А., Мееров И. Б., Шишкиов А. В.</i> Сравнение подходов к вычислению интегральной функции распределения многомерной нормальной случайной величины.....	39
<i>Горбунова А.С., Козинов Е.А., Мееров И.Б., Шишкиов А.В.</i> Сравнительный анализ численных методов решения систем стохастических дифференциальных уравнений.....	43
<i>Городецкий Е.С.</i> Система проектирования кинематики пространственных механизмов «mechanics studio .net» на основе managed directx.....	48
<i>Городецкий Е.С., Половинкин А.Н.</i> Сервис Windows communication foundation для организации видеоконференций «говорящих голов»	52
<i>Городецкий Е.С., Сидоров С.В.</i> Использование новейших технологий Microsoft в реализации коммуникационной системы организации и проведения видео-конференций «Virtual Class Room».....	57
<i>Гущин О.П.</i> Особенности внедрения среды обеспечения учебного процесса в высшем учебном заведении.....	61
<i>Денисов В.М., Шуляченко А.В., Шилин А.В.</i> Программное приложение «расписание занятий» для высших учебных заведений и средних профессиональных училищ.....	66

<i>Добряев Д.Н., Гришигин В.А.</i> Программная система конструирования и исследования характеристических методов многоэкстремальной оптимизации	68
<i>Домрачев В.В., Сенин А.В.</i> Разработка системы управления интегрированной средой высокопроизводительных вычислений ННГУ.....	71
<i>Дубровина Е.Н., Казакова Е.А., Майоров А.В., Катыков И.А., Шибанов С.В.</i> Разработка многоуровневых распределенных информационных систем на основе интегрированного семантического представления.....	76
<i>Егошин А.В.</i> Исследование возможности создания самообучаемой нейронной сети для решения задачи прогнозирования финансовых временных рядов	79
<i>Елсаков С.М., Ширяев В.И.</i> Однородные алгоритмы глобальной оптимизации и модели целевых функций	83
<i>Ефимов А.С.</i> Об одном подходе к извлечению нечетких знаний из статистических данных	87
<i>Жарков А.В.</i> Прототип распределенной СУБД реального времени для имитационного моделирования методов управления транзакциями.....	90
<i>Живодеров А.В., Корняков К.В., Мееров И.Б.</i> Алгоритмы размещения элементов смешанных размеров при проектировании СБИС	95
<i>Ильин Г.А., Морозов А.В.</i> Применение технологий Microsoft при автоматизации задач управления контингентом студентов в Астраханском государственном техническом университете	98
<i>Ионов А.А.</i> Приложение для проектирования кинематики механизмов в среде AutoCAD с использованием библиотеки ATL.....	102
<i>Казакова Е.А.</i> Проблемы интеграции разнородных CASE-средств в систему автоматизации проектирования ИС	106
<i>Казаков В.Г., Федосин С.А.</i> Разработка программной файлово-ориентированной системы резервного копирования данных	108
<i>Казакова Е.А., Дзюба Д.П.</i> Инstrumentальные средства для определения и анализа функциональных зависимостей на этапе логического проектирования РБД.....	110
<i>Казакова Е.А., Барышева Ю.С.</i> Информационная система учета текущей успеваемости студентов и подготовки итоговых документов.....	112
<i>Калганова И.С., Калганова Е.С.</i> Автоматизированная система исследования условий жизни населения на примере города Елабуга.....	114
<i>Кирьянов В.А.</i> Построение моделей полуправильных и сферических многогранников с использованием САПР AutoCAD	117
<i>Кондакова А.И., Шустрова Е.А., Малышева Е.Ю.</i> Разработка комплекса программ в среде VB.NET для согласования распределенной базы данных ИС вуза.....	118
<i>Коннова Н.А., Гурьянов Л.В.</i> Автоматизация администрирования данных экологического надзора Пензенской области	121
<i>Конюхова В.М.</i> Конструктор веб-страниц доступа к структурированным данным ..	124
<i>Корняков К.В., Курина Н.В., Мееров И.Б.</i> Анализ сложности алгоритма размещения элементов СБИС инструмента itlDragon	125
<i>Корняков К.В., Сенин А.В., Шишков А.В.</i> Интеграция Microsoft Compute Cluster Server 2003 с системой управления кластерами «Метакластер»	129
<i>Кудаков А.А., Кузенкова Г.В.</i> Реализация среды создания электронного учебника на базе технологии .NET	132
<i>Кудряшов П.П., Фоменков С.А.</i> Быстрое обнаружение лица человека	135
<i>Кудряшов П.П., Фоменков С.А.</i> Предварительная обработка изображений для повышения точности распознавания объектов.....	139
<i>Кузнецов А.И.</i> Универсальный растеризатор	140
<i>Кузнецов А.И.</i> Контроль качества изображений	143

<i>Кузьмин А.В.</i> . Объемное представление трехмерных объектов.....	148
<i>Лабутин Д.Ю., Лабутина А.А., Ливерко С.В.</i> . Развитие системы Паралаб для изучения и исследования параллельных методов решения сложных вычислительных задач	152
<i>Лабутин Д.Ю., Боголепов Д.К., Алексин А.А.</i> . Реализация приложения распределенной трассировки лучей на базе инструментария Alchemi	155
<i>Ладыгин Д.О., Камаев В.А.</i> . Автоматизированная система поискового конструирования и изобретения новых вибрационных устройств, эффектов и технологий «AlPS Vibro»	159
<i>Маврин В.Г., Макарова И.В.</i> . Совершенствование учета переговоров с потенциальными клиентами с целью увеличения объема продаж фирмы	162
<i>Макарова И.В., Беляев А.И.</i> . Проектирование базы данных рекламаций по отказам гарантийной автомобильной техники КамАЗ в составе системы электронного документооборота предприятия посредством СУБД MS SQL Server 2000	164
<i>Макарова И.В., Буйвол П.А.</i> . Проектирование информационно-аналитической системы «библиотека» посредством СУБД MS SQL Server 2000	169
<i>Майоров А.В., Дубровина Е.Н., Казакова Е.А.</i> . Автоматизированная информационная система сельского муниципального образования на мобильной платформе.....	172
<i>Матвеев З.А.</i> . Проектирование и принципы построения приложений, предназначенные для обработки электронных карт	176
<i>Махмутов Л.Н.</i> . Автоматизация делопроизводства арбитражного управляющего ...	177
<i>Мичасова О.В.</i> . Исследование моделей макроэкономического роста с помощью пакета MatLab	178
<i>Мичасова О.В.</i> . Применение пакетов имитационного моделирования для анализа моделей экономического роста.....	182
<i>Мокшин В.В., Якимов И.М.</i> . Поиск оптимальных решений для исследования и управления предприятием	187
<i>Морёнов О.А., Ефимов А.С.</i> . Применение гибридных систем искусственного интеллекта для решения задач медицинской диагностики.....	189
<i>Нетребский Л.С.</i> . Интеллектуальная информационная система индивидуального комплексного мониторинга развития студентов в течение всего периода обучения.....	192
<i>Одушев Д.Е.</i> . Обзор современных методов корпоративной работы над проектами в строительстве	196
<i>Орлова Ю.А., Заболеева-Зотова А.В.</i> . Подсистема предварительной обработки текста технического задания	198
<i>Павлов Р.А.</i> . Решение обратных задач оптики с помощью искусственных нейронных сетей.....	202
<i>Половинкин А.Н., Сидоров С.В., Суслов А.П.</i> . Библиотеки SharperCV.NET и Intel OpenCV в реализации слежения за лицом человека в видеопотоке	204
<i>Перелюбский А.М., Батищев Д.И.</i> . Алгоритм нахождения оптимального решения задачи коммивояжера с помощью нескольких близких к оптимальному решений .	208
<i>Прилуцкий М.Х., Власов В.С.</i> . Метод комбинированных эвристик построения конвейерных расписаний.....	211
<i>Прилуцкий М.Х., Куликова Е.А.</i> . Многокритериальные задачи распределения ресурсов в иерархических системах	213
<i>Прилуцкий М.Х., Слободской В.В.</i> . Распараллеливание метода ветвей и границ для задачи многоресурсного сетевого планирования	217
<i>Пылин В.В.</i> . Система электронной цифровой подписи на базе эллиптической кривой....	219
<i>Райкин Л.И., Вязанкина М.Н.</i> . 3D-модели в технологиях Autodesk для информационной поддержки жизненного цикла изделий.....	222

Райкин Л.И., Лупанов К.В., Лексиков А.В. Формирование электронного архива информационно-графических работ выпускников технического университета	229
Рубцов М.О. Коррекция нелинейности фото- и видеоизображений	235
Русских И.В. О сериализации объектной модели в библиотеке Colorer.....	239
Рябчиков А.В. Визуализация небольших волн на водной поверхности средствами Microsoft DirectX с использованием шейдеров	243
Рыбаков А.Е., Сидоркина И.Г., Савельев Р.О. Применение модуля решения задач линейного программирования для поиска оптимального пути в графе, определяющего последовательность изучения образовательного курса	248
Рязапов М.Р., Коляда Е.П. Проект: «Расчет кредитных ставок по сельскохозяйственной технике и ее рациональное использование».....	251
Седова Я.А., Морозов А.В. Автоматизация перевода текста на русском языке в дореволюционную орфографию	253
Сидоров С.В., Горюнов Ю.В., Городецкий Е.С. Анимация виртуальной головы человека с использованием Managed DirectX.....	255
Сидоров С.В., Половинкин А.Н., Латышев А.А. Библиотека ConferenceXP как .NET-каркас для приложений, обеспечивающих проведение аудио- и видеоконференций.....	258
Сидоров С.В., Сысоев А.В. Общая схема реализации остановки и возобновления счета индексного метода поиска глобально-оптимальных решений	259
Смирнов Е.В., Егоров Д.В., Иванов П.В. Автоматический анализ изображения.....	263
Соколов А.Ю., Гребенищиков А.В. Высокопроизводительная реализация алгоритма обучения нейронной сети методом обратного распространения ошибки с использованием средств разработки Intel и MS Visual Studio.....	267
Сорокин Е.С., Жерзев С.В. Система контроля версий электронных карт	269
Старостин Н.В., Филимонов А.В. Место задачи компоновки в процессе проектирования БМК.....	271
Степина А.Н. Разработка программной реализации вычисления интегралов методом Монте-Карло.....	275
Субботина Е.В., Сидоров С.В., Гергель В.П. Реализация поддержки страничной памяти в алгоритмах поиска глобально-оптимальных решений	279
Сысоев А.В., Лабутина А.А., Камаев А.М., Коваленко А.А., Сиднев А.А. Об опыте использования технологии Cluster OpenMP для создания параллельных программ	282
Тамбовцев Д.П., Шибанов С.В. Проект автоматизированного информационно-торгового комплекса управляющих систем реального времени.....	288
Ульянов Д.П., Палькин Е.А., Гусятников В.Н. Использование нейросетевых технологий при построении автоматизированной системы банковского кредитного скоринга.....	291
Федченко А.А. Автоматизация проектирования круглых протяжек в среде T-FLEX CAD с использованием технологии СОМ и библиотеки MFC.....	295
Федченко А.А., Прохоров А.А., Чугунов М.В., Полунина И.Н. Учебно-исследовательский проект «параметрическая оптимизация» на базе T-FLEX CAD и MFC.....	297
Филандыш Н.И., Лаптев В.В. Автоматизированная система формирования личных дел абитуриентов	300
Фомин К.В., Голубинский Д.О. Виртуальный лабораторный практикум по механике ..	303
Швецов В.И., Ефремов Я.В. Проблемы архитектуры современных компьютерных систем поддержки жизненного цикла изделий и пути их решения	305

Шлигает А.Ю., Федосенко Ю.С. Комбинированный алгоритм синтеза стратегий двухрежисового обслуживания стационарных объектов в разветвленной рабочей зоне	309
Шумков Д.С., Кузнецов Е.С., Сидоркина И.Г. Исследование индикаторов технического анализа для решения задачи прогнозирования временных рядов	314
Юсов Е.А. Параллельная модель вычислений для отображения больших участков ландшафта на основе прогрессивного квадрдерева и максимального использования графического процессора средствами Microsoft DirectX	316
Афраймович Л.Г., Прилуцкий М.Х. Равномерное перераспределение ресурсов в иерархических системах транспортного типа	320
Вильданов С.А., Конюхова В.М., Махмутов Л.Н. Автоматизированная информационная система управления вузом на базе электронной карты	321
Голубев И.А. Отображение составных большеформатных растровых документов ..	323
Егоров А.А. Мобильный программный картографический комплекс «BigViewer CE»....	325
Жерзев С.В. Синхронизация изображений в формате BIG	326
Жерзев С.В. Сжатие изображений в извещениях мореплавателям	328
Золотых Н.Ю., Лялин С.С. Arageli – библиотека для символьных вычислений. Обзор архитектуры.....	329
Золотых Н.Ю., Лялин С.С. Оптимизация одной модификации алгоритма двойного описания для многогранного конуса	333
Кирличев Ю.С. Актуальные проблемы сжатия видео.....	337
Конюхова В.М., Шулаева Н.А. Использование статистических данных и имитационного моделирования при определении оптимальной конфигурации аппаратного и программного обеспечения СУБД.....	340
Резников М.Б. Синтез оптимальной стратегии обслуживания потока объектов двумя независимыми mobile-процессорами в узловой рабочей зоне.....	341
Паршкова Е.А. Определение сроков реализации проекта с использованием MS Excel и MATLAB.....	343
Тюрина С.А., Чернопятова И.Г., Макарова О.П., Малышева Е.Ю. Разработка лабораторного практикума по созданию элементов экономических ИС с использованием технологии .NET	345
Шогулин Э.В., Андреев В.В. Многоагентная платформа для создания распределенных приложений	347
Большаков А.В., Кузенков О.А. Метод адаптации для задач безусловной минимизации средних потерь	352
Безрук К.В. Аспекты программной реализации интеллектуальной системы поддержки процессов проектирования	354
Белокрылов П.Ю. Методика синтеза линейных комбинационных схем в нейросетевом базисе	356
Вантеев В.Г., Зинкин С.А. Программная реализация маршрутизатора пакетов на основе логико-алгебраической модели	360
Крылова И.Н., Линёв А.В. Использование графовых моделей данных в задаче доступа к данным хранилищ существенно различного формата.....	365
Титов А.А. Информационная обучающая система «параметрическое моделирование в Autodesk Inventor»	370
Кузенков О.А., Капитанов Д.В. Проблема отбора в системах разностных уравнений на единичном симплексе	371
Кузенков О.А., Дырдин О.З. Системы дифференциальных уравнений первого порядка с нелокальными краевыми условиями на стандартном симплексе.....	373

ТЕХНОЛОГИИ MICROSOFT В ТЕОРИИ И ПРАКТИКЕ ПРОГРАММИРОВАНИЯ

*Материалы конференции
(Нижний Новгород, 3–4 апреля 2007 г.)*

Под ред. проф. **Р.Г. Стронгина**

Отв. за выпуск В.П. Гергель

Бумага офсетная. Печать офсетная. Формат 70×108 1/16.
Уч.-изд. л. 37,6. Усл. печ. л. 32. Тир. 300 экз. Зак. 311.

Издательство Нижегородского госуниверситета им. Н.И. Лобачевского.
603950, Н. Новгород, пр. Гагарина, 23.

Типография Нижегородского госуниверситета.
Лиц. ПД № 18-0099 от 04.05.2001.
603000, Н. Новгород, ул. Б. Покровская, 37.