СБОРНИК ПОСВЯЩЕН

60-летию КАФЕДРЫ

«КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ»

МГТУ имени Н.Э. БАУМАНА

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»

СОВРЕМЕННЫЕ
КОМПЬЮТЕРНЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

Юбилейный
сборник трудов кафедры
«Компьютерные системы и сети»
МГТУ имени Н.Э. Баумана

НИИ Радиоэлектроники и лазерной техники
Москва 2012

УДК 004.2/9(048.2)
ББК 32.973
С56

Современные компьютерные системы и технологии: Юбилейные
С56 сб. трудов каф. «Компьютерные системы и сети», МГТУ им. Н.Э. Баумана - М.: Изд-во НИИ Радиоэлектроники и лазерной техники, 2012. - 200 с.

ISBN 978-5-4384-0011-0

Настоящий сборник трудов кафедры "Компьютерные системы и сети" является юбилейным и посвящен 60-летию кафедры. Он составлен по материалам двух ежегодных научно-технических конференций аспирантов, студентов и молодых ученых "Современные информационные системы и технологии", состоявшихся в ноябре 2011 г. и апреле 2012 г., а также по материалам научных трудов преподавателей кафедры.

В сборник включены статьи по широкому спектру направлений информатики и вычислительной техники, отвечающим научным направлениям работы аспирантов, студентов, молодых ученых и преподавателей кафедры. Публикуемые материалы могут быть полезными читателям, специализирующимся в области компьютерных систем, сетей и технологий.

Статьи печатаются в авторской редакции.

УДК 004.2/9(048.2)
ББК 32.973

ISBN 978-5-4384-0011-0

УДК 004.827

НОВЫЙ ПОДХОД К МАШИННОМУ ТВОРЧЕСТВУ НА ОСНОВЕ ОШИБОК РАСПОЗНАВАНИЯ ОБРАЗОВ

К.М. Гречищев, В.Э. Подольский, Е.К. Пугачёв

Творчество человека является малоизученным, несмотря на то, что специалисты многих направлений современной науки: биологи, психологии, кибернетики, педагогики, социологии и даже математики – предпринимают попытки для осознания этого механизма и поиска его источников. В частности, классическая кибернетика тяготеет к перебору вариантов [1] и механизмам ассоциативной памяти [2]. Одна из областей кибернетики (и информатики) – искусственный интеллект – ставит перед собой в качестве ключевой задачи создание механизма машинного творчества. Причём результаты работы этого механизма должны в максимальной степени быть похожими на результаты работы человека-творца (художника, скульптора, архитектора и т.д.). На данный момент ни один из предложенных подходов к машинному творчеству не показал сколь угодно удовлетворительного результата. Поэтому в данной работе предлагается новый подход к реализации механизма машинного творчества при помощи учёта ошибок распознавания и использования средств современной математической теории динамического хаоса.

Для понимания машинного творчества, определимся с понятием творчества как такового. «Творчество – деятельность, порождающая нечто качественно новое и отличающееся неповторимостью, оригинальностью и общественно-исторической уникальностью» [3]. Но согласно тому же источнику «творчество специфично для человека». Поэтому в применении к ИИ и вычислительным машинам говорят не о творчестве, а о машинном творчестве и artificial creativity [4].

Машинное творчество – тот творческий процесс, который может быть осуществлён полностью средствами вычислительной техники, причём он может использовать как механизмы, заложенные в человеческом мозге, так и совершенно новые средства.

Основные требования к машинному творчеству были сформулированы Ньюэллом, Шоу и Саймоном [5]:

- результат, представленный творческой программой должен быть новым и полезным;
- результат требует отказа от ранее принятых идей;
- результат получен под воздействием высокой мотивации и настойчивости;
- результат получен от прояснения ранее расплывчатого вопроса.

На данный момент систем и подходов, полностью удовлетворяющих представленным критериям, практически не существует.

Работы в области машинного творчества были начаты давно, с момента зарождения искусственного интеллекта. Интересные детерминистские и стохастические подходы к синтезу музыкальных произведений на вычислительной машине представлены Зариповым [6]. Ньюэллом и Саймоном была создана система GPS (General Problem Solver – с англ. Общий решатель задач) [7], которая удовлетворяла некоторым критериям, представленным выше. Некоторые учёные попытались связать машинное творчество с математикой. Построение изображений при помощи самоподобных структур – фракталов – считается ещё одной ветвью в машинном творчестве [8]. Недавно Шмидтом и Липсоном была предпринята попытка формализации научного творчества: была создана программа выявления общих закономерностей (формул) физических и иных естественных процессов на базе генетических алгоритмов [9]. В Российской ассоциации искусственного интеллекта проблемой машинного творчества занимаются В.Л. Стефанюк, В.Э. Карпов, И.Б. Фоминых, а также некоторые другие учёные [10]. В отечественном сегменте области машинного творчества перспективной считается реализация аппарата Теории решения изобратательских задач (ТРИЗ) на вычислительной машине [11]. Исследования в этом направлении ведутся, но результаты [12] довольно сомнительны с позиции вышеозначенных критериев.

Ни один из представленных выше подходов не приближает учёных к окончательному решению проблемы машинного творчества. Редко предлагаются методы машинного творчества, включающие в себя одновременно несколько принципов. В настоящей статье представлен новый механизм машинного творчества: антропо-математический принцип моделирования ошибок распознавания и построения на их основе новых

образов, а также оценки новизны таких образов при помощи современного математического аппарата.

Ввиду явной несостоятельности существующих подходов, авторами настоящей статьи предлагается в некоторой степени универсальный (в рамках искусственного интеллекта) подход к машинному творчеству. Чтобы полнее объяснить идею подхода, рассмотрим небольшой бытовой пример.

Как известно, система визуального распознавания образов человека (глаза-зрительный нерв-затылочная часть коры головного мозга) иногда даёт известные сбои. Если некто находится в ярко освещённой естественным и искусственным светом комнате и изолирует её полностью от внешнего освещения, то с высокой долей вероятности в процессе адаптации глаз испытуемого к смене темнового режима у человека возникнут короткие по продолжительности галлюцинации. То есть он будет видеть то, чего на самом деле нет. Это происходит из-за того, что некоторые палочки и колбочки (визуальные рецепторы на сетчатке глаза) сохранили предыдущее световое воздействие [13]. Из-за различий в скорости реакции разных рецепторов зрения происходит возникновение световых пятен перед глазами. Часто мозг старается достроить эти световые пятна до некоторых образов, виденных ранее. А если такое световое пятно не похоже в точности ни на один из виденных ранее образов, то происходит комбинирование разных образов в один. В результате такого процесса человек может увидеть то, чего никогда в жизни не встречал, но то, что напоминает некоторые виденные им ранее образы. Этот случай можно отнести к ошибке системы распознавания образов человека.

Основная идея: формализация и построение творческого процесса как процесса обработки ошибки распознавания образа и доведения её до некоторого другого цельного образа с последующей оценкой его оригинальности и осуществимости.

Схему реализации механизма машинного творчества в обобщённом виде можно представить так, как показано на рисунке 1.

В соответствии с представленным рисунком распознаётся некоторый объект посредством первичного распознавателя с некоторой ошибкой. В результате этого получаем ошибочный образ (его наличие обуславливается неполнотой информации об объекте или несовершенством распознавателя). Ошибочный образ далее интерпретируется как самостоятельный объект

реального мира и при помощи комбинации вторичных распознавателей даёт на выходе новый образ, полученный при комбинировании образов от каждого вторичного распознавателя, причём эти образы получены на основе рапознавания образа на выходе первичного распознавателя. Получается, что первичный образ дополняется до некоторого цельного образа, который может одновременно принадлежать нескольким группам (это относится к задачам нечёткой кластеризации [14]).

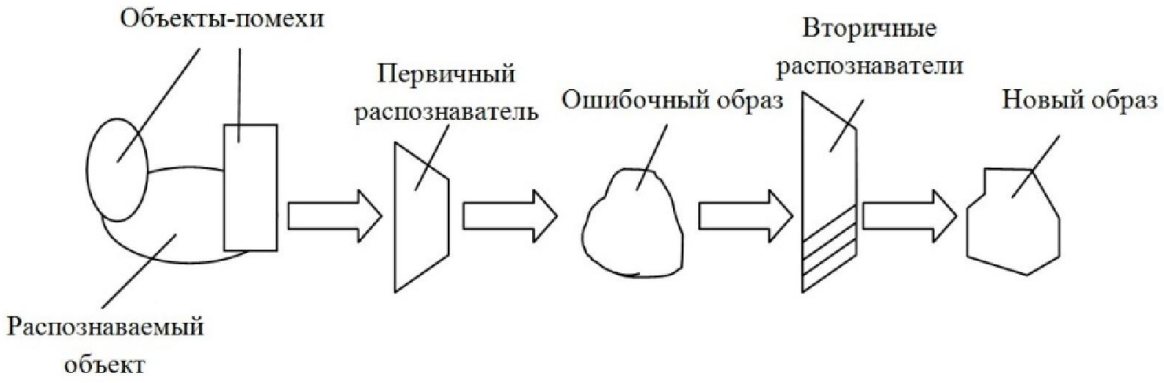


Рисунок 1. Абстрактная схема реализации механизма машинного творчества на основе ошибочного распознавания

Оригинальным будем называть тот образ $I_{нов}$, полученный в результате исполнения вышеописанной процедуры, минимальное расстояние от которого d_{min} до другого реально существующего образа I_i определённой группы (число образов в ней равно n) в некотором метрическом пространстве превосходит на величину максимального расстояния d_{max} между двумя образами I_i и I_j, принадлежащими одной группе. Итак, образ считается оригинальным, если:

$$d_{\min}(I_{нов}, I_i) > d_{\max}(I_j, I_k), \quad i = \overline{1:n};\ j = \overline{1:n};\ k = \overline{1:n} \quad (1)$$

Легко заметить, что (1) ялвяется геометрической интерпретацией задачи поиска выбросов. Задача поиска выбросов заключается в нахождении таких точек данных, которые сильнее всего отличаются от уже сгруппированных по некоторому признаку в классы и кластеры. Проблемной точкой этого простого подхода является определение функции d – степени различия образов. В геометрической интерпретации её можно принять за расстояние. Но в целом эта функция будет зависеть от рассматриваемой проблемы и механизма её решения. Рассмотрим эту функцию на примере

нейросетей. Также приведём основу подхода с применением математической теории хаоса [15].

Концепция искусственных нейронных сетей (ИНС) была разработана на основе принципов функционирования головного мозга человека (с позиции нейрофизиологии). К общим чертам ИНС и НС головного мозга человека можно отнести способность к обобщению, обучаемость, способность корректно обрабатывать информацию при её высокой зашумленности и другие [16]. Обладающие подобными свойствами нейросетевые алгоритмы могут быть использованы в задачах, связанных с проблемой машинного творчества.

В [17] рассматривалась проблематика распознавания символов многослойным персептроном. Причинами ошибочной работы нейронной сети в процессе распознавания являются недостаточная или, напротив, чрезмерная обученность нейронной сети, слишком высокий уровень шума во входных данных или подача на входы нейронной сети образа, эталон которого отсутствовал в обучающей выборке. В [17] предложен критерий, используя который можно выявить ошибку при распознавании или низкую сиепень уверенности нейронной сети.

Остановимся подробнее на проблеме обучения сети. При высокой степени обученности сети теряется одно из важнейших ее свойств: способность к обобщению. Так, сеть просто "запомнит" эталонные образы. В этом случае при незначительном отличии распознаваемого образа от оригинала сеть не будет уверена в своём решении. Если же сеть, наоборот, недостаточно обучена, то она может оказаться неспособной различить два полностью несовпадающих эталона.

Рассмотрим пример применения трехслойного персептрона применительно к задаче машинного творчества. ИНС обучается повторять на своих выходах входной сигнал, причём на входы сети подаётся изображения символов размером *32* на *32* пикселя. Число нейронов во входном и выходном слоях равно *1024 (32 X 32)*. Эталонное множество состоит из символов русского алфавита. При подаче на входы ИНС эталонного образа на выходе получают образ, незначительно отличающийся от входного. Очень важно не «переобучить» сеть.

После обучения на вход нейронной сети подаётся изображение неизвестного ей символа (например, цифры или символа латинского

алфавита). Сеть, обладая способностью к обобщению, производит попытку отнести новое изображения к одному из известных эталонов. В первом случае, если в эталонном множестве имелось изображение похожего символа (например, «*E*» для «*F*», или «*И*» для «*N*», «*P*» для «*R*»), то изображение на выходе нейронной сети будет соответствовать эталонному с некоторой степенью искажения. В противном случае (например, для изображений цифр) сеть не сможет соотнести изображение ни с одним из эталонов, и изображение на ее выходах будет состоять из элементов тех символов, которые более всего похожи на входной. Именно здесь и проявляется механизм машинного творчества на основе ошибок распознавания.

Проводя параллель с вышесказанным, можно заметить, что первый случай аналогичен попытке человеческого мозга достроить неизвестные световые пятна до некоторых образов, виденных ранее. Во втором случае сеть генерирует изображение, состоящее из элементов ранее изученных эталонов.

Интересный эффект наблюдается, если на $i+1$ итерации подать на входы нейронной сети выходные изображения *i*-ой итерации. При подаче на первой итерации неизвестного образа возможны два случая.

В первом из них через *N* итераций на выходах сети будет получено изображение одного эталонного образца практически без искажений. При этом степень искажения выходного изображения будет уменьшаться с ростом *N*. В другом случае сеть не сможет достроить новое изображение до эталонного. На выходах сети будет формировать образ, состоящий из элементов эталонов. С ростом *N* отличия сигналов на выходах на $i+1$ и *i*-ой итерации становятся всё менее значительными. Можно утверждать, что выходное изображение сходится к изображению некоторого не представленного ни в одном алфавите символа, то есть символа, который человеку не известен.

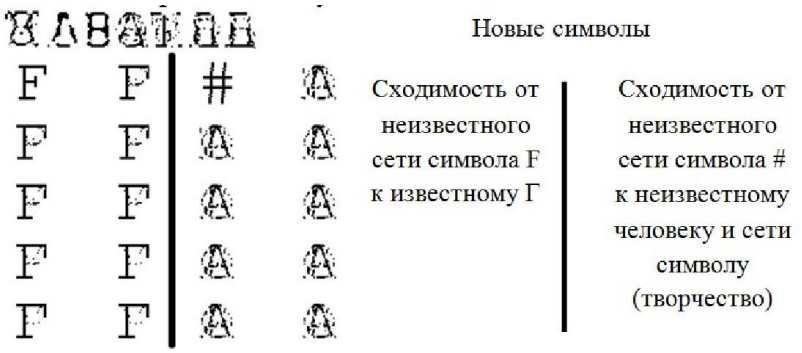


Рисунок 2. Результаты использования натренированной на символах русского алфавита ИНС применительно к неизвестным ей символам латинского алфавита и спецсимволам

Как уже было упомянуто, хаотическая динамика также может быть применена в контексте рассматриваемого подхода.

Предположим, что мы обладаем образом I (обобщая, считаем I многопризнаковым метрическим пространством образов) некоторого реально существующего объекта O, т.е. при помощи некоторого отображения образ был поставлен в соответствие реальному объекту: $\psi : O \to I$ (это может быть сделано как человеком, так и при помощи некоторого алгоритма распознавания образов). Для наших целей также принимаем, что образ I воспроизводит объект O с необходимой точностью. После получения образа I внесём в него некоторый малый шум ε, в результате чего получим образ $I' = I + \varepsilon$, причём ε может быть настолько мало, что $|\varepsilon| = |I' - I| \approx 10^{-20} \div 10^{-10}$. Вообще величина шума зависит от того, насколько быстро должен дать результат процесс, который будет описан в дальнейшем. Теперь если будет найдена некоторая функция $I_{t+1} = f(I_t)$, обладающая существенной зависимостью от начальных условий, то можно будет сконструировать такой математический алгоритм, который способен оценивать степень оригинальности образа I' и его производных во времени. Рассмотрим одну из теорем хаотической динамики, чтобы наиболее полно описать предлагаемый подход.

Пусть (X, d) – метрическое пространство, содержащее бесконечное множество точек. Если отображение $f : X \to X$ непрерывно и транзитивно, а периодические точки f плотны в X, то f обладает существенной зависимостью от начальных условий [19]. Воспользуемся этой теоремой для

составления алгоритма поиска функции *f*, обладающей существенной зависимостью от начальных условий.

Рассмотрим два близких начальных условия: $x_1(0) = x_0$ и $x_2(0) = x_0 + \Delta(0)$. Представим, что со временем траектории $x_1(t)$ и $x_2(t)$ изменяются так, как представлено на рисунке 3. К моменту времени *t* разница между двумя траекториями составляет $\Delta(t) = x_2(t) - x_1(t)$. Если $|\Delta(t)|$ растёт экспоненциально для типичной ориентации вектора $\Delta(0)$, тогда говорят, что система выражает сильную зависимость от начальных условий. Само условие такой зависимости может быть записано в виде:

$$\frac{|\Delta(t)|}{|\Delta(0)|} \sim e^{ht}, h > 0 \quad (2)$$

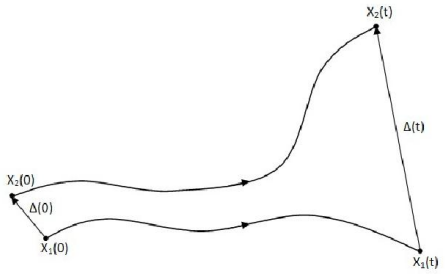


Рисунок 3. Пример сильной зависимости функции от начальных условий

Алгоритм, представленный на рисунке 4, позволяет оценить степень оригинальности нового образа, полученного применением шума к исходному. Оценка производится за счёт анализа динамики разницы между траекториями представления образов (исходного и ошибочного) при применении к ним некоторой функции.

Алгоритм довольно прост. Конечно, в данной области необходимы дальнейшие исследования ввиду того, что проблема оценки оригинальности того или иного объекта, образа сама по себе является сложной и нерешённой задачей. По сути всё машинное творчество может быть сведено к двум задачам: синтез образа, оценка оригинальности образа. Дополнительные критерии, представленные Ньэллом и остальными также могут быть задействованы по мере надобности.

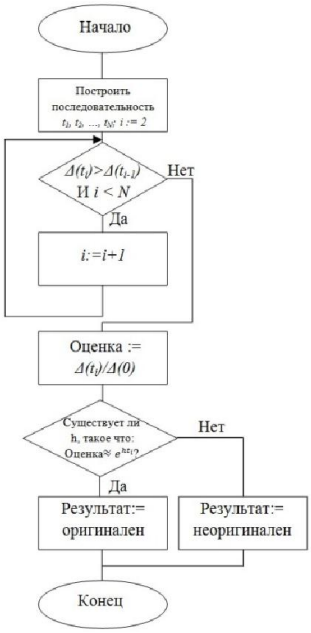


Рисунок 4. Вариант алгоритма оценки оригинальности образа

Конечно, по окончании процесса получения нового образа и определения его оригинальности, необходимо проверить полученный образ (его части) на существование в некоторой библиотеке образов. Также возможна проверка непосредственно по применимости, если полученный образ не обнаружен в библиотеке (т.е. полностью соответствует второму критерию Ньюэлла, Шоу и Саймона).

В настоящей работе была предпринята попытка рассмотреть механизм машинного творчества на основе ошибочных образов, а также интеграции в данную проблему математической теории хаоса. Если первое направление более тяготеет к этапу синтеза нового образа, то второе – к этапу оценки синтезированного образа (на новизну). Выполнение иных проверок по критериальной системе Ньюэлла-Шоу-Саймона представляется на данном этапе работ проблематичным ввиду крайне невысокой возможности их автоматизации. Особый интерес с научной точки зрения представляет собой дальнейшая проработка возможностей применения аппарата математической теории хаоса в динамических системах (какими и являются системы машинного творчества – творческие агенты) не только для оценки, но и непосредственно для синтеза новых образов. Например, требует рассмотрения идея новых (творческих) образов как неких аттракторов.

Применение машинного творчества (как такового, так и представленного в работе подхода) покрывает много направлений: синтез

рифм, написание картин, создание образов новых персонажей для литераторов, синтез рассказов, творческое решение изобретательских задач и т.д.

Итак, в настоящей работе рассмотрена идея нового подхода к актуальной проблеме машинного творчества. Сочетание новейших методов математики с классическими подходами искусственного интеллекта, используемыми при рассмотрении их стандартного функционала с новой позиции (например, ошибки классификации), может подвинуть нас ещё на шаг ближе к разгадке тайны машинного и человеческого творчества.

Литература:

1. *Стобо Дж.* Язык программирования Пролог: Пер. с англ. – М.: Радио и связь, 1993. – 368 с.

2. *Кохонен Т.* Ассоциативные запоминающие устройства. – М.: МИР, 1982. – 384 с.

3. Советский энциклопедический словарь / Гл. ред. А.М. Прохоров. – М.: Советская энциклопедия, 1987. – 1600 с.

4. Интернет-ресурс http://www.thinkartificial.org.

5. *Newell, A., Shaw, J. G., and Simon, H. A.* The process of creative thinking / H. E. Gruber, G. Terrell and M. Wertheimer (Eds.), Contemporary Approaches to Creative Thinking. – New York: Atherton, 1963. – pp. 63 – 119.

6. *Зарипов Р.* Машинный поиск вариантов при моделировании творческого процесса. – М.: Наука. Главная редакция физико-математической литературы, 1983. – 232 с.

7. *Newell, A., Shaw, J.C., Simon, H.A.* Report on a general problem-solving program / Proceedings of the International Conference on Information Processing – USA, 1959. – pp. 256–264.

8. *Федер Е.* Фракталы: Пер. с англ. – М.: Мир, 1991. – 254 с.

9. *Schmidt M., Lipson H.* Distilling Free-Form Natural Laws from Experimental Data / Science, Vol. 324, no. 5923, 2009. – pp. 81 - 85.

10. Интернет-ресурс http://raai.org .

11. *Альтшуллер, Г.* Найти идею. Введение в ТРИЗ – теорию решения изобретательских задач. – М.: Альпина Паблишер, 2011. – 410 с.

12. *Митрофанов В.В.* Машина открытий. Компьютерная программа. Свидетельство об официальной регистрации программы для ЭВМ № 2000610103.

13. *Чупров А., Кудрявцева Ю.* Анатомия и физиология органа зрения. – Киров: КГМА, 2007. – 107 с.

14. *Вятченин Д.* Нечёткие методы автоматической классификации: Монография. – Мн.: УП «Технопринт», 2004. – 219 с.

15. *Ott, E.* Chaos in dynamical systems. – UK: Cambridge University Press, 1994. – P. 386.

16. *Хайкин С.* Нейронные сети: полный курс, 2ое издание: Пер. с англ.– М.: Издательский дом «Вильямс», 2006.– 1104 с.

17. *Гречищев К.М., Самарев Р.С.* Особенности моделей представления изображений для нейросетевого распознавания символов. - М.: Эликс+, 2010.

18. *Кроновер Р.* Фракталы и хаос в динамических системах. Основы теории. – М.: Постмаркет, 2000. – 352 с.

УДК 681.3.07

АНАЛИЗ ТИПОВ АТРИБУТОВ ИНФОРМАЦИИ ТАБЛИЧНОГО ВИДА

Мин Тхет Тин, А.В. Брешенков, Д.Ю. Гудзенко

В информации табличного вид (ИТВ) типы элементов в пределах любого столбца таблицы могут различаться. Такие столбцы недопустимо использовать в реляционных таблицах (РТ) баз данных (БД). Ниже описаны программные средства, которые позволяют разработчику выявить наиболее вероятный тип для каждого столбца ИТВ и принять правильное решение.

Все столбцы ИТВ имеют тип строковый. Поэтому задача состоит в том, чтобы проанализировать в каждом столбце все его элементы и по внешнему виду элементов определить его тип. Предложен следующий подход.

Определяется не тип элементов, а наоборот непринадлежность к типу. Формируются данные о том, сколько и какие элементы столбца не соответствуют каждому из базовых типов. После этого нетрудно сделать вывод о том, какого типа столбец. Могут создаться неочевидные ситуации. Тогда решение принимает разработчик, после чего вид элементов он приводит в соответствии с выбранным типом, а затем назначает выбранный тип в целевой таблице. При этом рассматриваются базовые типы - дата-время, числовой и строковый.

Сначала кратко рассмотрим, что сделано по этому поводу, а потом как это сделано.

На рисунке 1 приведен фрагмент интерфейса для анализа типов ИТВ.

Рисунок 1. Фрагмент интерфейса для анализа типов ИТВ

После нажатия на соответствующую кнопку сформируется сообщение о том, сколько полей не являются датами для анализируемого столбца (рис. 2). В данном случае это “фамилия”

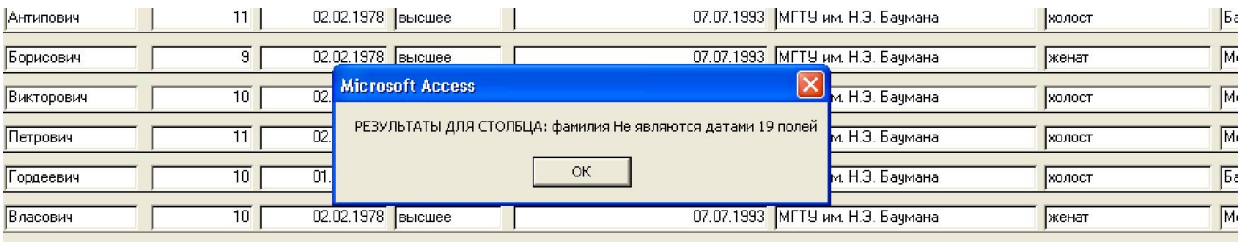


Рисунок 2. Сообщение о том, сколько полей не являются датами

Затем сформируется сообщение о том, в каких записях фамилия не является датой. Это показано на рисунке 3.

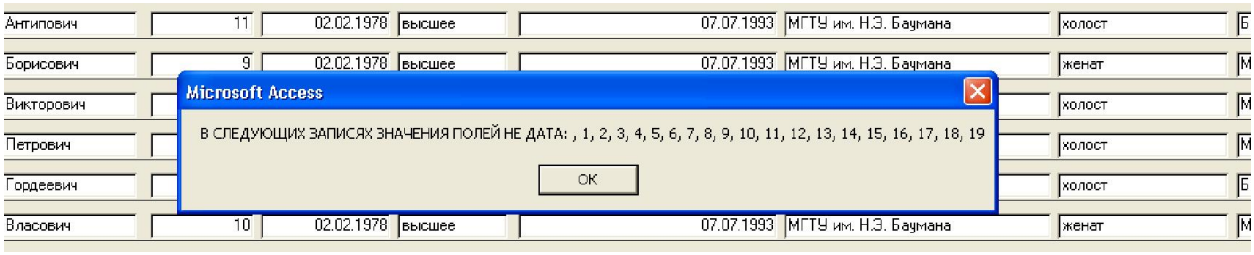


Рисунок 3. Сообщение о том, в каких записях фамилия не является датой

Далее сформируется сообщение о том, сколько полей не являются числами для анализируемого столбца (Рисунок 4).

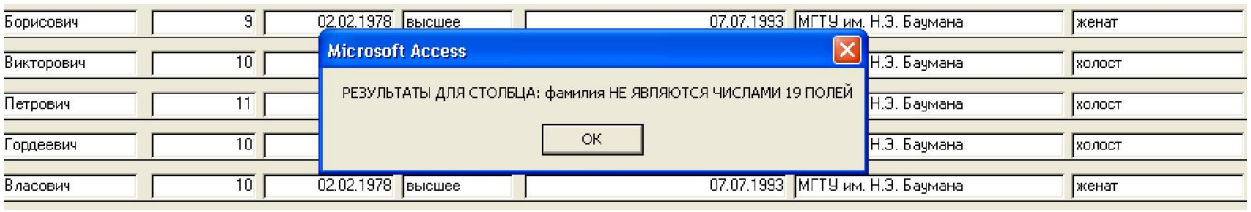


Рисунок 4. Сообщение о том, сколько полей не являются числами для анализируемого столбца

Затем сформируется сообщение о том, в каких записях фамилия не является числом. Это показано на рисунке 5.

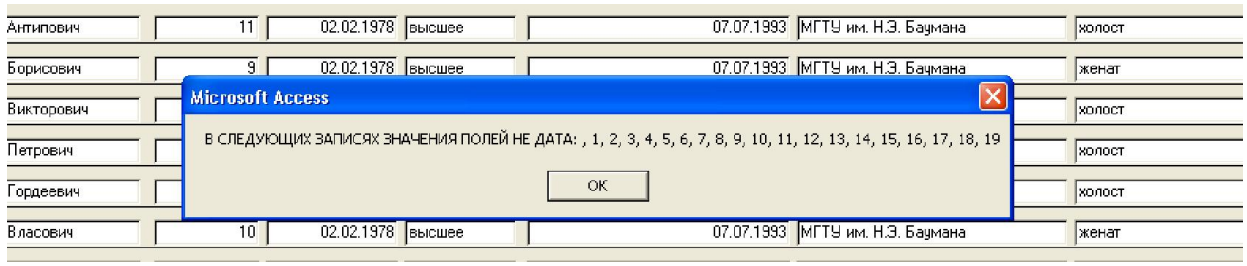


Рисунок 5. Сообщение о том, в каких записях фамилия не является числом

Учитывая то, что в таблице всего 19 полей, и все они не являются не датами не числами, легко сделать вывод о том, что столбец “фамилия” строкового типа

Рассмотрим противоположный случай. Анализируется 5-й столбец с датами. В процессе анализа этого столбца сформируется сообщение о том, сколько полей не являются датами для анализируемого столбца (Рисунок 6).

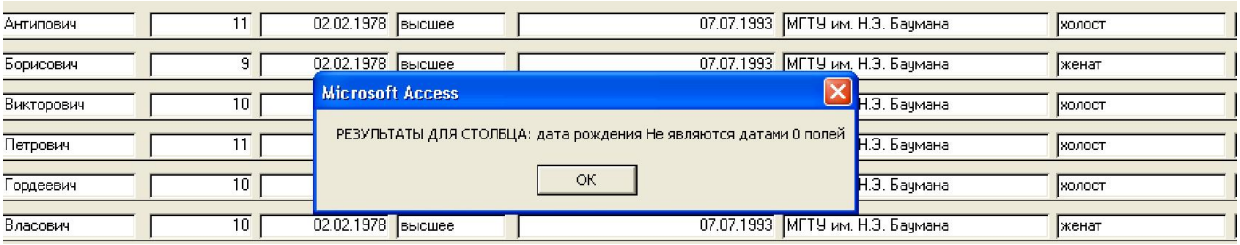


Рисунок 6. Сообщение о том, сколько полей не являются датами для анализируемого столбца

Затем сформируется сообщение о том, в каких записях фамилия не является датой. Это показано на рисунке 7.

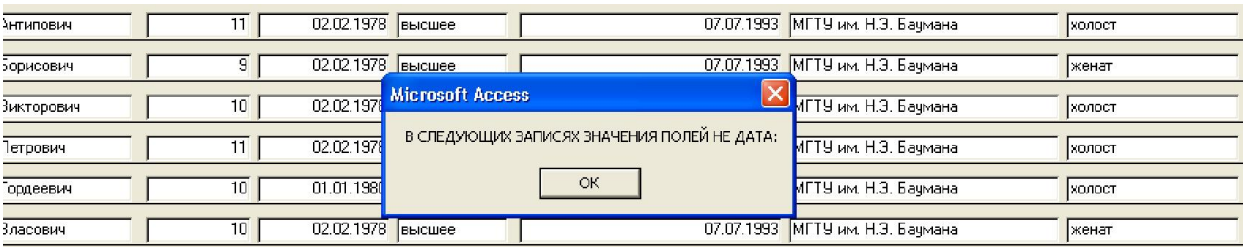


Рисунок 7. Сообщение о том, в каких записях поля не является датой

Легко сделать вывод о том, что столбец “дата рождения” имеет тип дата.

Рассмотрим ситуацию менее очевидную. Например, в столбце “дата рождения” в некоторых полях указана информация следующего вида: 13.10.55 года. Строго говоря, такая запись не удовлетворяет типу “дата”для формата даты используемого по умолчанию. При анализе такой ситуации сформируется сообщение, представленное на рисунке 8.

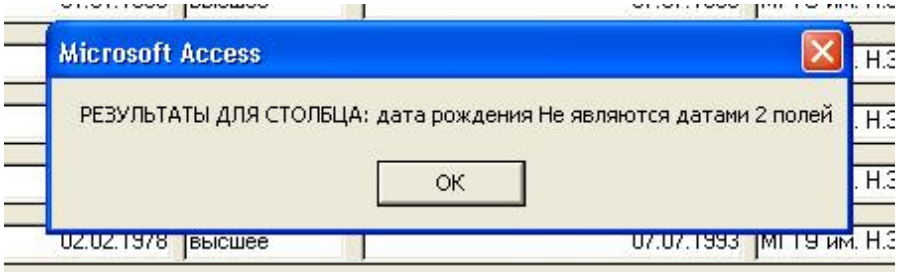


Рисунок 8. Сообщение о несоответствии дате 2-х полей

Далее сформируется сообщение (рисунок 9).

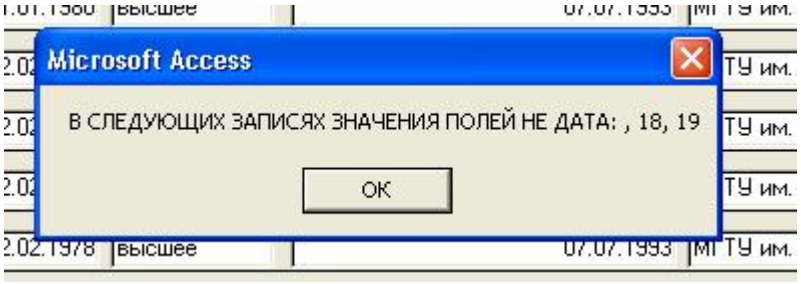


Рисунок 9. Сообщение о номерах записей, где поле “дата рождения” не соответствует типу “дата”

В соответствии с полученными сообщениями разработчик может принять решение о том, что поля в записях 18 и 19 необходимо исправить.

Событию “нажатие кнопки” “Анализ типов элементов столбцов” поставлена в соответствие процедура, фрагмент которой приведен ниже. Для удобства разъяснения основных приемов операторы пронумерованы.

```
1.  Private Sub Кнопка0_Click()
2.  On Error GoTo nd
3.  cnd = 0
4.  Dim mdb As Database  ' mdb объявляется как база данных
5.  Dim rst As Recordset ' rst объявляется как массив записей
6.  Dim a(10) As Variant ' Массив заголовков исходной таблицы
7.  Dim kp As String
8.  Dim np As String
9.  Set mdb = CurrentDb  ' mdb назначается текущая база данных
10. Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из таблицы t1
11. DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается форма t1 для таблицы t1
12. DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
13. kp = ""
14. np = ""
15. n = 0 ' счетчик имен столбцов исходной таблицы
16. For Each fld In rst.Fields ' в цикле перебираются все имена таблицы для их запоминания
17. ' MsgBox fld.Name
18. n = n + 1
19. a(n) = fld.Name
```

```
20.  Next fld
21.  ' DoCmd.Close ' закрывается форма t1
22.  DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
23.  DoCmd.GoToRecord acForm, "t2", acLast
24.  DoCmd.GoToRecord , , acFirst
25.  c = Forms!t2!cpol ' запоминается число полей
26.  cnd = 0
27.  For i = 1 To c
28.  p11 = Forms!t2!p1
29.  v = FormatDateTime(p11)
30.  GoTo d
31.  nd: cnd = cnd + 1
32.  np = np & ", " & i
33.  Resume Next
34.  GoTo d
35.  d: DoCmd.GoToRecord , , acNext
36.  Next i
37.  DoCmd.Close acForm, "t2"
38.  MsgBox ("РЕЗУЛЬТАТЫ ДЛЯ СТОЛБЦА: " & a(1) & " Не
являются датами " & cnd & " полей")
39.  MsgBox ("В СЛЕДУЮЩИХ ЗАПИСЯХ ЗНАЧЕНИЯ ПОЛЕЙ НЕ
ДАТА: " & np)
40.  DoCmd.Close acForm, "t1"
41.  funnnunber (0)
42.  End Sub
```

Все построено на выявлении ошибок преобразования типа, фиксирования количества этих ошибок, номеров строк и имен столбцов. Для этого используется оператор 2 (On Error GoTo nd). Ошибка преобразования может возникнуть в операторе 29 v = FormatDateTime(p11). Здесь осуществляется попытка преобразования типа текущего поля в тип “дата”. Если преобразование невозможно, то возникает ошибка и переход на метку nd (см. предыдущий оператор). В операторе 31 увеличивается счетчик не дат. В операторе 32 запоминается номер строки, в которой анализируемое поле не является дата. В операторе 33 выполняется сброс ошибки. В

операторе 34 выполняется переход к дальнейшей обработке записей. Выбирается следующая запись и начинается следующая итерация цикла перебора записей.

Если ошибки преобразования не возникло после выполнения оператора 29, то осуществляется переход к дальнейшей обработке записей. Выбирается следующая запись и начинается следующая итерация цикла перебора записей.

Таким образом, в результате работы процедуры будут выявлены все поля анализируемого столбца, которые не являются датой и сохранятся номера соответствующих записей.

После окончания цикла закрывается вспомогательная таблица “t2” и выводятся сообщения о числе полей не являющимися датами и сообщение о номерах соответствующих строк. Это видно из соответствующих рисунков.

Но поля проверены только на соответствие типу “дата”. Проверку на соответствие типу “число” в данном контексте выполнить невозможно, т.к. неизвестно по какому поводу возникла ситуация On Error.

Для выхода из этой ситуации написана функция, которая вызывается в операторе 42 (funnnunber (0))

Ниже представлена эта функция.

```
1.  Private Function funnnunber(r) As Integer
2.  On Error GoTo nd
3.  cnd = 0
4.  Dim mdb As Database  ' mdb объявляется как база данных
5.  Dim rst As Recordset ' rst объявляется как массив записей
6.  Dim a(10) As Variant ' Массив заголовков исходной таблицы
7.  Dim kp As String
8.  Dim np As String
9.  Set mdb = CurrentDb  ' mdb назначается текущая база данных
10. Set rst = mdb.OpenRecordset("t1") ' rst назначается массив записей из таблицы t1
11. DoCmd.OpenForm "t1", acNormal, "", "", acEdit, acNormal ' открывается форма t1 для таблицы t1
12. DoCmd.GoToRecord , , acFirst ' переход на первую запись формы
13. kp = ""
14. np = ""
```

```
15.  n = 0 ' счетчик имен столбцов исходной таблицы
16.  For Each fld In rst.Fields ' в цикле перебираются все имена
таблицы для их запоминания
17.  ' MsgBox fld.Name
18.  n = n + 1
19.  a(n) = fld.Name
20.  Next fld
21.  ' DoCmd.Close ' закрывается форма t1
22.  DoCmd.OpenForm "t2", acNormal, "", "", acEdit, acNormal
23.  DoCmd.GoToRecord acForm, "t2", acLast
24.  DoCmd.GoToRecord , , acFirst
25.  c = Forms!t2!cpol ' запоминается число полей
26.  cnd = 0
27.  For i = 1 To c
28.  p11 = Forms!t2!p1
29.  v = FormatNumber(p11)
30.  GoTo d
31.  nd: cnd = cnd + 1
32.  np = np & ", " & i
33.  Resume Next
34.  GoTo d
35.  d: DoCmd.GoToRecord , , acNext
36.  Next i
43.  DoCmd.Close acForm, "t2"
44.  MsgBox ("РЕЗУЛЬТАТЫ ДЛЯ СТОЛБЦА: " & a(1) & " НЕ
ЯВЛЯЮТСЯ ЧИСЛАМИ " & cnd & " ПОЛЕЙ")
45.  MsgBox ("В СЛЕДУЮЩИХ ЗАПИСЯХ ЗНАЧЕНИЯ ПОЛЕЙ НЕ
ЧИСЛА: " & np)
46.  DoCmd.Close acForm, "t1"
47.  End Function
```

По сути, эта функция практически не отличается от вызывающей процедуры. Главное отличие в операторе 29 (v = FormatNumber(p11)). В этом операторе осуществляется попытка преобразования текущего поля в число. Далее работа программы выполняется аналогично. Конечно, функция формирует соответствующие ей сообщения.

Литература:

1. *Брешенков А.В., Балдин А.В.* Анализ проблемы проектирования реляционных баз данных на основе использования информации табличного вида и разработка модели методики проектирования. М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. -150 с.

2. *Брешенков А.В.* Методы решения задач проектирования реляционных баз данных на основе использования существующей информации табличного вида. М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. -150 с.

УДК 004.588:004.773

РАЗРАБОТКА ЭЛЕКТРОННОГО КУРСА В СИСТЕМЕ MOODLE

В.Э. Подольский, А.Ю. Попов

В современной системе высшего профессионального образования всё более важное значение приобретает дистанционное обучение, которое базируется на возможности самостоятельной работы по освоению изучаемого материала и интерактивном взаимодействии обучаемых и преподавателя. Дистанционный характер обучения позволяет расширить арсенал средств, применяемых в образовательной деятельности до таких сложных форм, как: видеолекции, онлайн-семинары, автоматизированные онлайн-тесты и другие. Зарубежный опыт (аналоги используются в Стэнфордском университете, Массачусетском технологическом институте и ряде других) показал, что создание таких систем в дополнение к существующему аудиторному обучению очень полезно. Для применения в рамках университета хорошо подходит бесплатная свободно распространяемая модульная система поддержки обучения – СПО Moodle [1,2,3]. В данной статье приведена процедура создания системы поддержки обучения с использованием СПО Moodle на примере вводного теоретического курса дисциплины «Электроника», преподаваемой на кафедре «Компьютерные системы и сети» в МГТУ имени Н. Э. Баумана.

Общая структура курса

СПО Moodle обеспечивает возможность построения учебных онлайн-ресурсов различного назначения благодаря возможности подключения большого количества модулей[4]: информационных и идентификационных служб, словарей, вики, модулей взаимодействия с внешними ресурсами, проведения вебинаров, тестирования, оценки качества работ, выявления плагиата, оплаты обучения и пр. Таким образом, в зависимости от потребностей учебного процесса, состав система поддержки обучения может варьироваться. На рисунке 1 представлена организационная схема курса «Электроника».

Рисунок 1. Организационная схема электронного курса "Электроника"

В СПО Moodle предусмотрены несколько вариантов организации курсов: в соответствии с календарным планом; в виде блочной структуры; в виде форума; в виде SCORM структуры. Для реализации курса «Электроника» было решено использовать блочную структуру: в виде трёх блоков-тем, каждая из которых соответствует одному из семестров курса. Разметка главной страницы курса осуществлена при помощи ресурсов «Примечание» (горизонтальные линии использованы для отделения логически различающихся частей, информационные блоки – для улучшения ориентировки студентов): их можно легко добавить, отредактировать и в

режиме редактирования переместить в любое место страницы, как и любой другой ресурс или блок. Основная информация о курсе (созданная в теме 0) содержит в себе организационную часть, общую для всего курса.

В блок **«Общая часть»** вынесено четыре субблока. Субблоки «Обсуждения» и «Опросы», обозначенные овалами на рисунке, содержат в себе форумы, чат и опросы о функционировании электронного курса соответственно. Эти субблоки предназначены для осуществления обратной связи со студентами.

Обратная связь: форумы, чаты, опросы

Для реализации обратной связи студентов с преподавателем в субблок *«Обсуждения»* добавлены три форума («Добавить элемент курса…» -> «Форум»): новостной форум (для того, чтобы преподаватели могли оперативно сообщать студентам о намечающихся контрольных, переносах занятий или появлении вакансий) с отключённой возможностью комментирования; форум группы (с установленным параметром «Отдельные группы» в настройках элемента) для того, чтобы каждая группа могла иметь свой форум с возможностью обсуждения предмета курса и оказания взаимопомощи по нему с возможностью комментирования и оценки сообщений; форум для преподавателей курса, невидимый для студентов (чтобы только преподаватели могли общаться в этом закрытом форуме и обмениваться информацией по предмету курса). Также в субблок добавлен чат («Добавить элемент курса…» -> «Чат») с той целью, чтобы преподаватели могли устраивать предэкзаменационные и предзачётные консультационные сессии для студентов, на которых можно обсудить любые спорные моменты, не выезжая специально для этого в вуз.

Субблок *«Опросы»* содержит опросы («Добавить элемент курса…» -> «Опрос»), цель которых заключается в изучении мнения студентов относительно курса, а также пожеланий по его улучшению. Каждый опрос имеет ряд параметров. При его создании необходимо задать название для идентификации его среди других опросов, описание (чтобы студенты могли понять цель опроса), предел количества попыток (для опросов мнения нет необходимости) по каждому варианту ответа. Можно задать большое число вариантов ответа, но рекомендуется не более 7-10. Также можно разграничить опрос по группам (групповой метод), задать время его

доступности для ответа и возможность доступа к результатам опроса. Возможно ограничить возможность изменения ответа студента на опрос одной попыткой, но, как правило, в этом нет необходимости. Варианты ответов вводятся вручную.

Сторонние ресурсы

Субблок *«Ссылки»*, обозначенный прямоугольником со скруглёнными краями на рисунке (что означает ресурсы не интерактивного или слабо-интерактивного характера), содержит в себе несколько ссылок на сторонние сайты, где представлена информация по электронике (отечественные и сайт M.I.T.-OCW). Ссылки создаются при помощи последовательности действий «Добавить ресурс…» -> «Ссылка на файл или веб-страницу». В появившемся окне необходимо заполнить поля с названием и описанием ссылки, а также привести в поле «Размещение» саму ссылку. В разделе «Окно» рекомендуется выставить опцию «Окно» в положение «Новое окно» в выпадающем списке для удобства студентов. Если по ссылке размещён файл, то следует установить флажок «Принудительное скачивание», чтобы не раздражать пользователя лишними соглашениями на скачивание. В разделе «Окно» можно установить параметры нового окна, если был выбран соответствующий пункт в выпадающем списке.

Субблок *«Полезные ресурсы»* является слабо-интерактивным и содержит в себе по два элемента каждого из двух типов: база данных и глоссарий.

Глоссарии

Глоссарии являются важной частью любого теоретического вводного курса, так как они предоставляют студентам возможность привыкнуть к терминологии, используемой в изучаемой области, понять её. Такой глоссарий («Основные термины, законы и приборы») создан при помощи выбора «Добавить элемент курса…» -> «Глоссарий». Для любого глоссария необходимо задать название (осмысленное, чтобы студенты могли сориентироваться), описание и число записей на страницу (оптимально 10 – 15). «Формат отображения» установим в «Простой, вроде словаря», что соответствует сути аскетичного, но полезного словаря с основными терминами по курсу. Так как курс «Электроника» является базовым для

многих других курсов кафедры «Компьютерные системы и сети», и эти курсы используют терминологию данной дисциплины, то логично установить флажок «Этот глоссарий глобальный?» для того, чтобы другие курсы тоже могли иметь к нему доступ. Если планируется постепенно знакомить студентов с новыми терминами или как-то их организовать по темам, то стоит установить «Тип глоссария» в «Главный глоссарий», тогда право на редактирование его принадлежит только преподавателям и в него могут быть импортированы записи из глоссариев с установленной во «Вторичный глоссарий» опцией типа. Во вторичные глоссарии студенты могут добавлять записи. Преподаватели и студенты могут оценивать добавленные студентами определения (опция «Разрешить оценивать записи?»), что позволяет стимулировать активную работу по расширению знаний терминологии. Также в глоссарии организованы категории для улучшения организации терминов и определений (щелчок по глоссарию, вкладка «Обзор по категориям», кнопка «Редактировать категории»).

Вторичный глоссарий «Ответы на часто задаваемые вопросы» неглобального типа также очень важен для базового теоретического курса. У преподавателей за время их деятельности накапливается немало однотипных вопросов от студентов, в том числе вопросов организационного характера. Ответы на такие вопросы хорошо бы вынести в отдельное место – файл, а лучше – в специально оформленный глоссарий. СПО Moodle позволяет легко организовать такой глоссарий: достаточно выбрать в настройках нового глоссария в опции «Формат отображения» пункт «ЧаВо», тогда перед понятием (вопросом студента) автоматически добавляется слово «Вопрос», а перед определением (ответом преподавателя) слово «Ответ». Такая структура позволяет студентам легче разобраться с интересующими их вопросами. Ответы и вопросы лучше строить в понятной студентам, немного разговорной форме, так как это способствует улучшению восприятия материала.

Базы данных

Базы данных («Добавить элемент курса…» -> «База данных») позволяют хранить материалы (файлы проектов, статьи и пр.) организованно, а пользователям – добавлять записи в базу и оценивать добавленное другими пользователями. Для базы данных обязательно задать

название и введение. После того, как база данных создана, следует кликнуть на её название на странице курса. В появившейся странице необходимо перейти на вкладку «Поля», где можно добавить необходимые для записи поля. В базе данных «Книги и статьи» (в курсе «Электроника» используется для хранения и поиска статей и книг) поля: «Название книги / статьи» (текстовое поле), «Автор» (текстовое поле), «Дата публикации» (Дата), «Ссылка на оригинал» (ссылка), «Материал (книга / статья)» (файл – собственно сама книга или статья по теме курса), «Описание» (текстовая область). Название периодического издания для статьи лучше вынести в поле «Описание». После того, как поля при помощи выпадающего списка были добавлены («Создать новое поле») и отредактированы (название поля, описание и формат информации), следует отформатировать отображение записей в базе. Для этого необходимо перейти на вкладку «Шаблоны».

По умолчанию все добавленные на предыдущем этапе поля будут отображены в шаблоне отображения в правой части страницы. Эта правая часть является текстовым редактором наподобие MS Word или OpenOffice Writer, поэтому можно применить те же методы для форматирования шаблона, что и для обычных документов. Если создаётся новое поле, то добавить его в шаблон можно посредством двойного щелчка мыши по этому полю (или действию) в области «Доступные тэги», расположенной слева на вкладке «Шаблоны». Можно и вручную добавить тэг попросту заключив его наименование в двойные квадратные скобки (например, [[Описание]]). В шаблоне одной записи (виды шаблонов приведены сразу под вкладками при включенной вкладке «Шаблон») рекомендуется оставить все возможные поля, а в шаблоне списка убрать поле «Описание», так как оно, как правило, занимает много места в списке и выглядит неаккуратно. В шаблоне списка можно также добавить заголовок и нижний колонтитул (например, предназначение базы данных и соглашение об использовании материалов базы соответственно). База данных со статьями и книгами сослужит хорошую службу для тех студентов, кто особенно заинтересован в предмете курса. База данных «Примеры схем по курсу «Электроника»» содержит различные электронные схемы в виде файлов с расширением ewb, то есть студенты смогут изучить работу схем при помощи средства Electronics Workbench. Для каждой записи представлены следующие поля: «Название схемы» (текстовое поле – общепринятое в электронике название схемы,

например мост Вина), «Автор схемы» (текстовое поле – человек, придумавший схему), «Краткое описание» (текстовая область – описание функционирования схемы на естественном языке), «Схема» (файл формата ewb), «Дата добавления» (дата), «Миниатюра» (картинка – рисунок схемы), «Источники» (текстовая область – источники информации, откуда взята данная схема). На рисунке 2а приведено окно настроек глоссария, а на рисунке 2б – окно редактирования шаблона в базе данных.

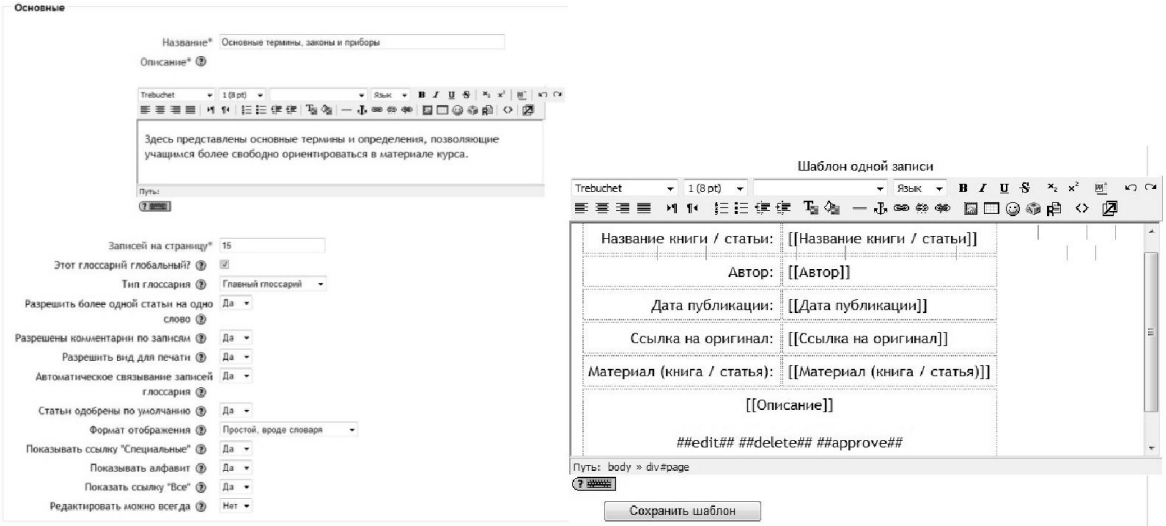


Рисунок 2. а) Настройки глоссария; б) Шаблон одной записи в БД

Каталоги

Для каждого курса предусмотрена возможность создания своей системы каталогов для хранения необходимых частей курса. На рисунке 3 приведён пример древа каталогов для курса «Электроника». Можно создавать новые каталоги (в том числе вложенные), загружать файлы в созданные каталоги. Также можно перемещать файлы и каталоги в другие каталоги, архивировать файлы и папки и, конечно, удалять всё это (эти опции доступны в выпадающем меню каталога).

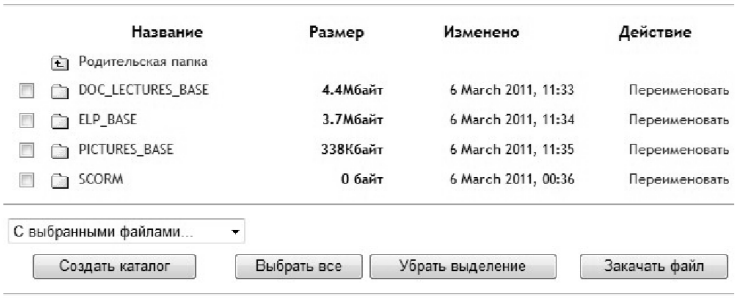


Рисунок 3. Система каталогов электронного курса

Разграничение доступа и группы

Чтобы облегчить обращение к отдельным учебным группам в курсе могут быть созданы соответствующие группы (так, в курсе «Электроника» имеется по три группы на каждый учебный семестр курса). Для этого необходимо в панели «Управление» на главной странице курса выбрать пункт «Группы». На открывшейся странице слева могут быть созданы или удалены группы (при нажатии на соответствующие кнопки). Чтобы добавить студентов в группу, необходимо выбрать одну из групп в левом поле и нажать кнопку «Добавить/удалить участников». На открывшейся странице в правом поле выбирается профиль пользователя и при нажатии на кнопку «Добавить» он добавляется в группу. Для того, чтобы добавить пользователя в группу, он должен быть зарегистрирован в системе.

Разграничение на группы в курсе позволяет обращаться к каждой группе по отдельности. Также при таком разделении упрощается администрирование, и появляется возможность разделения форумов и ресурсов для доступа каждой группы (т.е. члены одной группы имеют доступ к определённым ресурсам, а члены другой – к другим ресурсам).

Пакеты scorm и онлайн-тесты

Основу любого курса должны составлять теоретические сведения. СПО Moodle предоставляет возможность вставлять в курс теорию не только в виде обычных лекций электронного формата (doc, djvu, pdf и пр.), но и использовать для этого формат SCORM [5, 6]. Пакет SCORM может быть создан при помощи свободно распространяемой программы eXe. Такой пакет может включать в себя: лекции, организованные по разделам, главам и пунктам; галереи фотографий; тесты, учитывающиеся в СПО Moodle; приложения на Java; возможность обратной связи с преподавателем; цели курса; список литературы и пр. Создание пакета SCORM (и его собрата AICC) представляет собой предмет самостоятельного интереса и не рассматривается здесь в деталях. В курсе «Электроника» предусмотрены пакеты SCORM.

Для проверки знаний студентов можно использовать средства SCORM, а также создание тестов в курсе. Тест создаётся в режиме редактирования: из выпадающего списка «Добавить элемент курса…» выбирается строка «Тест». В открывшейся странице нужно ввести название теста и его краткое

описание. Интерес представляют настройки комментариев к оценкам. При помощи ввода комментариев за каждую оценку можно указать тестирующемуся, что ему следует повторить и проработать. Также задать можно метод оценивания (например, при методе «Высшая оценка» выбирается наивысший балл из всех попыток). Чтобы добавить в тест вопросы, необходимо нажать на название созданного теста на главной странице курса и перейти на вкладку «Редактировать» (здесь же можно задать последовательность вопросов и баллы за них), а затем на подвкладку «Вопросы». На этой подвкладке можно выбрать тип создаваемого вопроса из выпадающего списка «Создать новый вопрос». Процесс создание вопросов довольно прозрачен и здесь не рассматривается.

Платные курсы

В СПО Moodle имеются модули, позволяющие осуществлять автоматизированную оплату дистанционного курса. Для этого в панели администрирование необходимо перейти в раздел «Курсы» -> «Подписка». В этом разделе можно выбрать вариант подписки студентов на все курсы системы (в том числе платные, например: PayPal). В настройках любого курса можно выбрать метод подписки («Подписка» -> «Метод записи»), в том числе платный – PayPal. После сохранения платного метода подписки в разделе «Доступность» настроек появится поле «Стоимость», где можно установить цену подписки на данный курс. Если одновременно с тем установлено кодовое слово, то часть учащихся сможет подписаться бесплатно. Для получения оплаты за пользование услугами курсов должен быть зарегистрирован и соответствующим образом настроен аккаунт в системе PayPal. Платное предоставление образовательных услуг в области дистанционного обучения очень развито на Западе (особенно в США). Некоторые компании (как, например, Prometric) стараются попасть и в российский сегмент рынка образовательных услуг. Отметим, что образовательные услуги предоставляются не только студентам, но и сотрудникам фирм, а также в качестве дополнительного образования. Также интернет-портал Intuit использует немного другую идею в области платного дистанционного обучения: продажа готовых к инсталляции и работе упомянутых ранее пакетов SCORM.

Заключение

СПО Moodle обладает простым и дружественным интерфейсом. Создать и наполнить курс в ней не составит труда. Богатый инструментарий по управлению учётными записями пользователей, разграничению доступа и оценки качества помогает преподавателю сэкономить свои силы и время. Преимущества от организации курсов по преподаваемым дисциплинам на СПО Moodle есть и у студентов. Материалы становятся более интерактивными, становится понятной организационная структура дисциплины. Также многие положительно оценивают возможность проходить тесты, не выходя из дома (например, время от занятий теперь не тратится на проведение контрольных мероприятий). Сбор домашних заданий становится проще, ликвидируются очереди в кабинет преподавателя (ведь можно получить ответ на интересующий вопрос от преподавателя по сети).

Несмотря на все успехи дистанционного образования, остается нерешенной проблема контроля уровня знаний, в том числе проблема проверки знаний при «недобросовестном дистанционном обучении». Данная проблема решается в настоящий момент с применением технологии «менторства» и увеличения баз вопросов, что существенно усложняет подготовку к контрольным мероприятиям.

Литература:

1. Learning Management System - http://en.wikipedia.org/wiki/Learning_management_system
2. *Ryann K. Ellis*, A Field Guide to Learning Management Systems – 7 p.
3. Сайт Moodle - http://moodle.org
4. *Cole, J.* Using Moodle, 2nd Edition / J. Cole, H. Foster. – O'Reilly Media, 2007. – 284 p.
5. SCORM - http://en.wikipedia.org/wiki/SCORM
6. Сайт Conform 2 SCORM - http://www.conform2scorm.com/

УДК 004.021, 534.222

ОБЗОР МЕТОДОВ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ ПЕРВИЧНОЙ ОБРАБОТКИ ГИДРОЛОКАЦИОННЫХ СИГНАЛОВ

В.В. Ходин, А.С. Романовский

Гидролокация – это определение места нахождения подводного объекта либо по звуковым сигналам, испускаемым самим объектом (пассивная гидролокация), либо по отражению или рассеянию от объекта специально излучённого звукового сигнала (активная гидролокация). Объектами могут быть надводный корабль, подводная лодка, косяк рыб, скала на дне и прочее.

При пассивной гидролокации (шумопеленговании) направление на источник звука определяют, исследуя пространственную структуру звукового поля, создаваемого источником. При этом используются различные методы пеленгования: максимальный, нулевой, фазовый, корреляционный.

При активной гидролокации используется отражённый или рассеянный объектом сигнал, поэтому в активной локации создаётся мощное направленное излучение акустических импульсов с заполнением несущей частотой. Наряду с разрешающей способностью по расстоянию, основной характеристикой гидролокаторов является дальность обнаружения, зависящая от мощности излучаемого звука, уровня акустических помех и условий распространения звука в водной среде. Выбор частоты заполнения зависит от назначения гидролокатора.

Шумовое поле в морях и океанах создается совокупным воздействием многочисленных источников звука, различным образом влияющим на уровень и спектры шумов. Шумы делятся на:

1. Динамические шумы, возникновение которых связано с динамикой морских волн, турбулентных потоков в воде и атмосфере, с шумом прибоя, подводным шумом дождя, кавитацией и другими физическими причинами.

2. Биологические шумы, обусловленные жизнедеятельностью представителей морской фауны.

3. Сейсмические шумы, связанные с тектонической и вулканической деятельностью, образованием волн цунами.

4. Подледные шумы, характерные для арктических районов и связанные с образованием и динамикой ледового покрова, взаимодействием его неровностей с ветром и течениями.

5. Технические шумы, генерация которых связана с деятельностью человека, в том числе шумы судоходных трасс или шумы технических сооружений, расположенных вблизи береговой линии (порты, гавани) и на дне и др.

Гидроакустика – раздел акустики, в котором изучаются характеристики звуковых полей в реальной водной среде для целей подводной локации, связи и других задач.

В приемном тракте гидролокатора осуществляется предварительная, первичная и вторичная обработка гидролокационных сигналов. Первичная обработка сигнала обеспечивает оптимальное обнаружение в рамках принятого критерия, т.е. обеспечивает установление наличия в поступающем на вход гидролокатора процессе, представляющего собой смесь сигнала и помех, наличие полезного сигнала. Самыми распространенными являются три критерия выбора решений.

Критерий минимума среднего риска (байесовский) – это критерий, при котором оптимальной является система, которая обеспечивает наименьшую среднюю вероятность ошибки, наибольшую среднюю вероятность правильного приёма. Также в критерии учитываются последствия, к которым приводят ошибки разного рода. Ошибки можно выразить весовыми, стоимостными коэффициентами, которые приписываются к каждому из ошибочных решений – потери. Правильному приёму приписываются либо отрицательные или нулевые значения потерь.

Критерий Неймана-Пирсона. Одним из существенных недостатков байесовского правила обнаружения сигналов является большое количество априорной информации о потерях и вероятностях состоянии объекта, которая должна быть в распоряжении наблюдателя. Этот недостаток наиболее отчетливо проявляется при анализе гидролокационных задач обнаружения цели, когда указать априорные вероятности наличия цели в заданной области пространства и потери за счет ложной тревоги или пропуска цели оказывается весьма затруднительным. Поэтому в подобных

задачах вместо байесовского критерия обычно используется критерий Неймана-Пирсона. Согласно этому критерию выбирается такое правило обнаружения, которое обеспечивает минимальную величину вероятности пропуска сигнала (максимальную вероятность правильного обнаружения) при условии, что вероятность ложной тревоги не превышает заданной величины.

Критерий последовательного анализа Вальда. При последовательной выборке наблюдения производятся по одному (или, более общим образом, группами) и анализируются в ходе самого эксперимента с тем, чтобы на каждом этапе решить, требуются ли ещё наблюдения (решение о продолжении эксперимента) или наблюдений уже достаточно (решение об остановке эксперимента). То есть число производимых наблюдений (момент остановки наблюдений) не фиксируется заранее, а выбирается по ходу наблюдений в зависимости от значений поступающих данных. В задаче различения (по результатам независимых наблюдений) двух простых гипотез этот критерий дает значительный выигрыш в среднем числе производимых наблюдений по сравнению с другими критериями.

Таким образом одной из задач гидролокации является выделение полезного сигнала на фоне окружающих шумов и помех.

В реальных активных гидролокационных системах почти не применяется режим однократного наблюдения; в них осуществляются непрерывное зондирование среды и анализ принимаемых сигналов, так что практически всегда имеются данные ряда последовательных наблюдений, которые обрабатываются с помощью цифровых устройств.

При совместной обработке последовательности импульсов в активных гидролокационных системах с целью определения дальности до цели и доплеровского смещения отраженного сигнала широко используются современная теория оценок и возможности цифровых схем. Существенно, что для прослеживания трассы цели приходится сглаживать последовательность оценок дальности и доплеровского смещения, получаемых на выходе коррелятора. Это сглаживание должно производиться с учетом динамики движения цели, в том числе взаимосвязи между доплеровским смещением и последовательностью дальностных оценок, а также любых известных ограничений, касающихся цели, таких, как максимальные значения скорости или ускорения. Чаще всего для сгла-

живания используются алгоритмы, основанные на методе наименьших квадратов. Так, простейшим из них является полиномиальная аппроксимация с теми или иными дополнительными ограничениями. Один из наиболее сложных методов основан на моделировании движения цели, которая рассматривается как динамическая система, находящаяся под воздействием неизвестных возбуждающих сил и шума, вызывающего искажение дальностно-доплеровских измерений.

Система первичной или пространственно-временной обработки сигналов (СПВО), подключенная к выходам каналов системы предварительной обработки сигналов (СПО), решает задачи максимизации отношения сигнал/помеха, порогового обнаружения полезных сигналов и определения угловых координат их источников с последующей выдачей информации в систему вторичной обработки. Процессоры СПВО выполняют две основные функции: пространственную фильтрацию сигналов (формирование диаграммы направленности (ДН) антенной решетки), временную обработку сигналов (спектральный анализ и оценивание).

Для выделения полезного сигнала на фоне шумов целесообразно применять адаптивные методы, которые обеспечивают значительно большее отношение сигнал/шум по сравнению с обычными методами. Понятие «адаптивный» означает способность системы изменять свои параметры в зависимости от изменения окружающих условий с целью поддержания своей эффективности. Обладающие гибкостью адаптивные алгоритмы и структуры используются в тех ситуациях, когда возможности жестких алгоритмов и структур уже исчерпаны.

Использование адаптивных методов обработки сигналов в корабельных ГАС приобрело актуальность в связи с прогрессом в снижении внешнего акустического поля кораблей-целей, прежде всего подводных лодок. Кроме того, снижение корабельной помехи, которая в течение длительного времени была основной помехой в работе ГАС, привело к тому, что в результирующей помехе стали доминировать компоненты, обусловленные динамическими шумами океана и шумами судоходства. При работе корабельных ГАС в прибрежных районах стало необходимым также считаться с помехами, порожденными индустриальной деятельностью на морском шельфе.

При работе активной ГАС основной помехой служит реверберационная помеха. Первоначально задача борьбы с реверберацией решалась только в приемном тракте с помощью схем автоматической регулировки усиления. Позднее методы адаптации были распространены и на передающий тракт. При этом было использовано то обстоятельство, что при данной ДН антенны интенсивность реверберационной помехи зависит от несущей частоты, длительности и мощности зондирующего импульса. С учетом этого в зарубежных гидролокаторах внедрено устройство адаптации. Сначала с помощью входящего в состав ГАС построителя зон гидроакустического обнаружения рассчитываются лучевые картины распространения звука и с учетом требуемой дальности обнаружения определяются все основные параметры зондирующего импульса: несущая частота, вид модуляции, длительность и мощность. Задача решается с помощью экспертной системы, оператору остается лишь установить рекомендованные параметры на пульте ГАС.

Задачу обнаружения малошумных подводных объектов современными ГАС, имеющими большую дальность действия, также приходится решать на фоне многочисленных помех от локальных источников (мешающих целей). Задача решается методом автоматической режекции (при помощи адаптивных режекторных фильтров) таких помех в процессе пространственной обработки сигналов. Наиболее распространен метод формирования «проколов»-нулей в ДН антенны в направлениях на мешающие цели с помощью адаптивных антенных решеток.

Адаптивная антенная решетка (ААР) представляет собой сочетание собственно антенной решетки (АР) и связанного с ней цифрового сигнального процессора, работающего в реальном времени. Процессор определяет основные характеристики ААР, а АР играет роль многоканального датчика информации. В качестве первого шага была решена задача подавления помехи от локальной мешающей цели методом формирования нуля ДН (прокола) в направлении на мешающую цель.

Для выделения полезного сигнала может применяться метод когерентного вычитание мешающего сигнала, который реализуется на выходе веера сформированных ДН. Одна из ДН (оценивающий канал) ориентируется на источник локальной помехи. Производится вычисление коэффициентов корреляции между сигналом с выхода оценивающего канала

и процессами на выходе других ДН. Производится вычитание помехи из этих процессов с весами, пропорциональными значениям вычисляемых коэффициентов корреляции. Этим способом может быть подавлено несколько сильных мешающих помех. Два рассмотренных выше метода относятся к методам когерентной режекции мешающих целей и достаточно просты в реализации.

В современных ААР реализуются более сложные алгоритмы, обеспечивающие автоматическое управление весовыми коэффициентами, которые придаются сигналам с отдельных элементов АР (или подрешеток) с целью формирования требуемой ДН.

Алгоритм Уидроу-Гриффитса позволяет выделить один сигнал от цели на фоне других сигналов, рассматриваемых как помехи. Предполагается, что направление прихода сигнала на АР и спектральная плотность его мощности либо известны, либо оцениваются заранее с достаточной точностью, а подобная информация о помехе отсутствует.

Широкое развитие получила группа адаптивных методов, предложенных Кейпоном, Шмидтом, Джонсоном и основанных на математических операциях с матрицей взаимной спектральной плотности N-мерного (по числу элементов или подрешеток АР) входного процесса.

Следует отметить, что алгоритмы типа Шмидта, Джонсона, основанные на разложении ковариационной матрицы по собственным числам и последующем разделении собственных чисел на сигнальные и помеховые, продолжают находиться в стадии исследований. Одна из особенностей таких методов — отсутствие традиционной процедуры формирования ДН. Важное достоинство алгоритмов, использующих оценку ковариационной матрицы, заключается в повышенной угловой разрешающей способности по отношению к сильным сигналам. С помощью этих алгоритмов могут быть разрешены плоские волны, приходящие к АР под углами, различающимися менее чем на ширину ДН.

Важным направлением в теории и технике обработки сигналов, изображений и временных рядов является вейвлет-преобразование (ВП), которое хорошо приспособлено для изучения структуры неоднородных процессов. Вейвлеты представляют собой особые функции в виде коротких волн (всплесков) с нулевым интегральным значением и с локализацией по оси независимой переменной, способных к сдвигу по этой оси и

масштабированию (растяжению/сжатию). Обычные промодулированные импульсы, посланные в среду, имеют при высоких частотах слишком большую длительность для того, чтобы отличать импульсы, отраженные от тонких, прилегающих друг к другу слоев с различной плотностью или от различных объектов сходной структуры. Поэтому вместо посыла импульсов одинаковой длины, посылают на высоких частотах короткие волновые образования, которые получают масштабированием. Такие сигналы и называют вейвлетами.

Любой из наиболее часто используемых типов вейвлетов порождает полную ортогональную систему функций. В случае вейвлет-анализа (декомпозиции) процесса (сигнала) в связи с изменением масштаба вейвлеты способны выявить различие в характеристиках процесса на различных шкалах, а посредством сдвига можно проанализировать свойства процесса в различных точках на всем исследуемом интервале. Именно благодаря свойству полноты этой системы, можно осуществить восстановление (реконструкцию или синтез) процесса посредством обратного ВП. ВП также может выделять и характеризовать прерывистые непериодические явления с помощью процедур рассмотрения объектов в различных масштабах.

Можно реконструировать только часть сигнала или выделить вклад определенного масштаба. Если вейвлет-коэффициенты подвержены случайным ошибкам, они будут действовать на реконструируемый сигнал локально вблизи положения возмущения, а преобразование Фурье распространяет ошибки по всему восстанавливаемому сигналу.

Именно благодаря выявлению локальных особенностей сигнала, принципиально отсутствующему у преобразования Фурье, ВП нашло широкое применение для анализа тонкой структуры сигналов и изображений, для их сжатия и очистки от шума, что важно и полезно в гидроакустике.

Скрытая марковская модель (СММ) – статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами. Задачей ставится определение неизвестных параметров на основе наблюдаемых. Полученные параметры могут быть использованы в дальнейшем анализе, например, для распознавания образов. СММ – расширение марковской модели, когда наблюдения являются некоторой вероятностной функцией состояния. Таким образом, СММ

представляет собой дважды стохастический процесс, состоящий из пары случайных процессов, один из которых является основным и ненаблюдаемым, то есть скрытым. Единственной возможностью является суждение об этом процессе с помощью другого случайного процесса или множества таких процессов, которые дают последовательность наблюдений.

Самыми простыми СММ являются эргодические или полностью связанные модели, когда каждое состояние модели может быть получено за один шаг из любого другого состояния. В некоторых приложениях (например, для задач распознавания звуковых сигналов) оказывается, что наблюдаемым свойствам моделируемого сигнала лучше соответствуют другие типы СММ. Одна из подобных моделей – это лево-правая модель или модель Бакиса, последовательность состояний которой обладает тем свойством, что с увеличением времени индекс состояния также увеличивается, или остается неизменным, иными словами, состояния переходят из одного в другое слева направо.

В этих моделях наблюдения можно считать дискретными символами, выбираемыми из конечного алфавита, а задание текущих значения этих символов выполняется с помощью дискретных функций плотности вероятности. Проблема, возникающая при таком подходе, заключается в том, что в некоторых задачах наблюдения являются непрерывными сигналами или векторами. Хотя такие непрерывные сигналы можно квантовать с помощью кодовых книг и других средств, подобное квантование может иногда приводить к серьезным искажениям исходного сигнала. Этого недостатка лишены СММ с непрерывными плотностями наблюдения. На такие модели наложен ряд ограничений, чтобы гарантировать состоятельность процедуры повторного оценивания функции плотности вероятности.

Таким образом, наиболее перспективными методами повышения эффективности алгоритмов первичной обработки гидролокационных сигналов с точки зрения распознавания полезных сигналов на фоне шумов и улучшение отношения сигнал/шум в принятых сигналах будут методы, основанные на адаптивных алгоритмах, ВП и СММ.

Литература:

1. *Рожин Ф.В., Тонаканов О.С.* Общая гидроакустика. – М.: Изд-во Моск. ун-та, 1988. – 160 с.: ил.

2. *Бурдик В.С.* Анализ гидроакустических систем: Пер. с англ. – СПб: Судостроение, 1988. – 392 с.: ил.

3. *Горбань И.И.* Обработка гидроакустических сигналов в сложных динамических условиях. – Киев: Наукова думка, 2008. – 275 с.: ил.

4. *Уидроу Б., Стирнз С.* Адаптивная обработка сигналов: Пер. с англ. – М.: Радио и связь, 1989. – 440 с.: ил.

5. *Монзинго Р.А., Миллер Т.У.* Адаптивные антенные решетки: введение в теорию: Пер. с англ. – М.: Радио и связь, 1986. – 448 с.: ил.

6. *Григорьев Л.Н.* Цифровое формирование диаграммы направленности в фазированных антенных решетках. – М.: Радиотехника, 2010. – 144 с.: ил.

7. *Яковлев А. Н.* Введение в вейвлет-преобразование: Учеб. пособие. – Новосибирск: Изд-во НГТУ, 2003. – 104с.: ил.

8. *Моттль В.В., Мучник И.Б.* Скрытые марковские модели в структурном анализе сигналов. – М.: Изд-во ФИЗМАТЛИТ, 1999. – 352 с.: ил.

УДК 004.3+519.6

ВЫБОР СПОСОБА ОПРЕДЕЛЕНИЯ ВЕРШИН, СМЕЖНЫХ НЕКОТОРОМУ МНОЖЕСТВУ

О. А. Парменова, В. А.Овчинников

В последовательных алгоритмах, например компоновки схем или размещения микросхем, на каждом шаге необходимо определять множество вершин, смежных вершинам формируемого множества. Моделью схемы в этих задачах является гиперграф $H = (X, U)$ [1]. Обозначим формируемое множество вершин X_i и множество им смежных – $X^{\text{см}} = F_1(X_i)$. В ходе работы алгоритма, например разрезания гиперграфа, $|X_i| = i = 1, 2, \ldots, n_i$, где n_i – количество микросхем в формируемой подсхеме.

Гиперграф представлен в виде $H = (X, U, \Gamma X, \Gamma U)$, где $\Gamma X = \{\Gamma x_t / x_t \in X\}$, $\Gamma U = \{\Gamma u_j / u_j \in U\}$, $\Gamma_1 x_i$ – множество рёбер, инцидентных вершине x_i, и Γu_j– множество вершин, инцидентных ребру u_j. Рассмотрим два способа определения $F_1(X_i)$ и оценим вычислительную сложность их реализации.

Первый способ. На основании определения понятия «смежность» $X_i^{\text{см}} = \Gamma(\Gamma(X_i))$. Перед включением в X_i очередной вершины определяем:

$$U_i = \Gamma(X_i) = \cup\, \Gamma x_j,\ x_j \in X_i.$$

Обозначим $a_j = |U_j'| / |U_j|$, где $U_j' \subset U_j$ – множество рёбер, инцидентных вершине x_j и не инцидентных вершинам множества $X_i \setminus x_j$. Значение $0 \le a_j \le 1$, где $a_j = 0$, если $\Gamma x_j \subseteq \Gamma(X_i \setminus x_j)$, и $a_j = 1$, если вершина x_j не смежна ни одной из вершин множества $X_i \setminus x_j$. Считая $a_j = a_k = \ldots = a$ $(0<a<1)$ и $\rho = |\Gamma x_j|$, получим:

$$|U_i| = \rho\cdot[1 + a\cdot(j - 1)], \text{ где } j = 2, |X_i|, \text{ и}$$

$$i = |X_i|$$

$$N_i' = \rho^2\cdot\sum_{j=2} [1 + a\cdot(j - 2)].$$

Здесь N_i' – количество операций сравнения рёбер множеств Γx_j, необходимых для получения множества U_i на i-м шаге алгоритма.

Суммарное количество операций сравнения, выполняемых при получении всех множеств U_i, будет равно:

$$N_\Sigma' = \rho^2 \sum_{i=2}^{n_i} \cdot \sum_{j=2}^{i} [1 + a\cdot(j - 2)],$$

Далее определяем $X_i^{\text{см}} = \Gamma(U_i) = \cup\, \Gamma u_k,\ u_k \in U_i$. Положив $c_k = |X_k'| / |X_k|$, где $X_k' \subset X_k$ – множество вершин, инцидентных ребру u_k и не инцидентных рёбрам множества $U_i \setminus u_k$. Значение $0 \le c_k \le 1$, где $c_k = 0$, если $\Gamma u_k \subseteq \Gamma(U_i \setminus u_k)$, и $c_k = 1$, если ребро u_k не смежно ни одному из рёбер множества $U_i \setminus u_k$. Считая $c_k = \ldots = c$ $(0 < c < 1)$ и $|\Gamma u_k| = A$, получим:

$$N_i'' = A^2\cdot\sum_{k=2}^{|U_i|} [1 + c\cdot(k - 2)],$$

где N_i'' – количество операций сравнения вершин множеств Γu_k.

Суммарное количество операций сравнения вершин множеств Γu_k, необходимых для получения всех множеств $X_i^{\text{см}}$, будет:

$$N''_{\Sigma} = A^2 \sum_{i=2}^{n_i} \cdot \sum_{k=2}^{|U_i|} [1 + c\cdot(k-2)],$$

Второй способ. Определяем сначала $F_1X = \{F_1x_t \;/\; x_t \in X\}$. Каждый образ $F_1x_t = \cup\, \Gamma_2 u_k$, $u_k \in \Gamma_1 x_t$. Для его получения необходимо выполнить:

$$\rho = |\,\Gamma_1 x_t|$$

$$M'_t = A^2 \cdot \sum_{k=2} [1 + c\cdot(k-2)]$$

операций сравнения, а мощность каждого множества F_1x_t равна

$$|\,F_1x_t\,| = A\cdot[1 + c\cdot(\rho - 1)].$$

Суммарное количество операций сравнения, необходимых для получения F_1X, будет:

$$M'_{\Sigma} = A^2 \cdot \sum_{k=2}^{\rho} [1 + c\cdot(k-2)] \cdot n.$$

Затем определяем множество $X_i^{\text{см}} = F_1(X_i) = \cup\, F_1x_j$, $x_j \in X_i$. Положив $d_l = |F_1x_l'| \,/\, |F_1x_l|$, где $F_1x_l' \subseteq F_1x_l$ – множество вершин, смежных вершине $x_l \in X_i$ и не принадлежащих вершинам множества $X_i^{\text{см}} \setminus x_l$. Значение $0 \le d_l \le 1$, где $d_l = 0$, если $F_1x_l \subseteq F_1(X_i / x_l)$, и $d_l = 1$, если $F_1x_l \cap F_1(X_i / x_l) = \varnothing$. Считая $d_l = d = constant$ $(0 < d < 1)$, получим суммарное количество операций сравнения:

$$M''_{\Sigma} = A^2\,[1 + c\cdot(\rho - 1)]^2 \sum_{i=2}^{n_i} \cdot \sum_{l=2}^{i} [1 + d\cdot(l-2)].$$

Асимптотическая оценка M'_{Σ} равна $O(n)$, а N'_{Σ}, N''_{Σ} и M''_{Σ} – $O(n^3)$, так как $n_i = n \,/\, L$, $n = |\,X\,|$, $L = constant$ – количество частей, на которые разрезается гиперграф.

Поскольку множество X_i формируется последовательным включением вершин, множество $X_i^{\text{см}}$ целесообразно определять по рекуррентным формулам. Получим оценки количества операций сравнения для обоих способов.

При первом способе*:*

– множество рёбер $U_i = \Gamma(X_i)$ находим как Γx_i, $i = 1, |X_i|$;

– множество $X_i^{\text{см}}$ определяем как $X_i^{\text{см}} = X_{i-1}{}^{\text{см}} \cup \{\cup \Gamma u_j, u_j \in U_i\}$

Теперь суммарное количество операций сравнения, необходимых для получения первым способом всех множеств $X_i^{\text{см}}$, будет:

$$N_\Sigma = \rho^2 \cdot \sum_{i=2}^{n_i} [1 + a \cdot (i-2)],$$

При втором способе:

– $F_1X = \{F_1x_t / x_t \in X\}$ определяем так же, как и выше;

– множество $X_i^{\text{см}}$ находим как $X_i^{\text{см}} = X_{i-1}{}^{\text{см}} \cup F_1x_i$, $i = 1, |X_i|$.

Суммарное количество операций сравнения, необходимых для получения вторым способом всех множеств $X_i^{\text{см}}$, будет:

$$M_\Sigma = A^2 \cdot \sum_{k=2}^{\rho} [1 + c \cdot (k-2)] \cdot n. + A^2 [1 + c \cdot (\rho - 1)]^2 \cdot \sum_{i=2}^{n_i} [1 + d \cdot (i-2)].$$

Выбор способа определения $X_i^{\text{см}}$ выполним для варианта использования рекуррентных формул. Из полученных выражений видно, что $N_\Sigma = f_1(n_i^2)$ и $M_\Sigma = f_2(n_i) + f_3(n_i^2)$. Асимптотическая оценка для обоих равна $O(n^2)$. Поскольку асимптотическая оценка первого члена M_Σ равна $O(n)$ и параметры a и d имеют один порядок, достаточно оценить соотношение ρ^2 и A^2 $[1 + c \cdot (\rho - 1)]^2$. Нетрудно видеть, что второй способ будет предпочтительнее первого, если выполняется соотношение:

$$\rho > A\,[1 + c \cdot (\rho - 1)] \text{ или } c \cdot < (\rho - A) / A(\rho - 1).$$

Для схем соединения элементов средств ЭВТ параметр $A_{max} = k_{\text{н}} - 1$, где $k_{\text{н}}$ – нагрузочная способность элемента, а ρ – количество цепей, подключённых к элементу. Реальное значение A для большинства элементов в схеме $1 \le A \le 4$. При автоматизированном проектировании средств ЭВТ параметр ρ определяется степенью интеграции элементов библиотеки функциональных ячеек и может принимать значение от двух до шестнадцати и более. Например, при $A = 2$, $\rho = 16$ и $c \cdot = 0{,}4$ указанное соотношение выполняется.

Литература:

1. *Овчинников В.А.* Алгоритмизация комбинаторно-оптимизационных задач при проектировании ЭВМ и систем: Учеб. для вузов. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. – 288 с.

УДК 004.052.4

ПРОБЛЕМЫ ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ТЕХНИЧЕСКИХ СИСТЕМ

В.С. Буренков, С.Р. Иванов

Проблема обеспечения правильности программных и аппаратных систем на этапе их проектирования имеет первостепенное значение. Верификация является одним из основных методов ее решения. Верификация – подтверждение на основе представления объективных свидетельств того, что установленные требования были выполнены. Наряду с верификацией рассматривается валидация – подтверждение на основе представления объективных свидетельств того, что требования, предназначенные для конкретного использования или применения, выполнены. Таким образом, верификация – проверка того, что продукт удовлетворяет сформулированным требованиям, а валидация – проверка того, что разработано именно то, что требуется заказчику.

Широко используемыми при валидации методами являются моделирование и тестирование. Моделированию подвергается абстрактная схема или модель системы, а тестирование применяется непосредственно к самому продукту. В случае электронных схем моделированию подвергается проект разрабатываемой схемы (Verilog-описание), тогда как тестированию подвергается сама электронная схема.

Несмотря на то, что моделирование и тестирование зарекомендовали себя как эффективные методы на ранних стадиях отладки, когда разрабатываемая система еще содержит большое количество ошибок, результативность этих методов быстро снижается, по мере исправления найденных ошибок [1]. При этом возрастает время, требуемое для

обнаружения все более тонких ошибок. Применение указанных методов осложняется тем, что никто не может с уверенностью сказать, когда их возможности уже исчерпаны, или хотя бы оценить, сколько ошибок может все еще содержаться в системе.

Необходимы методы верификации, позволяющие проверить выполнение требований спецификации на всех возможных траекториях функционирования системы.

В области верификации в течение нескольких десятилетий прилагались серьезные усилия по разработке теории, алгоритмов и техники верификации. В настоящее время эти методы разрабатываются в трех основных направлениях:

- дедуктивная верификация;
- проверка эквивалентности;
- model checking (проверка модели).

Дедуктивная верификация обычно подразумевает применение аксиом и правил вывода для доказательства правильности функционирования системы. Эта весьма сложная процедура не может быть полностью автоматизирована, она требует участия человека, действующего на основе предположений и догадок, использующего интуицию при построении инвариантов и нетривиальном выборе альтернатив. Преимущество дедуктивной верификации состоит в том, что она может быть применена к системам с бесконечным числом состояний.

Проверка эквивалентности связана с разработкой формальных моделей взаимодействующих процессов, выражением в рамках таких моделей как спецификации, так и реализации, и проверкой эквивалентности формально определенных моделей поведения**.**

Метод model checking связан с формальной проверкой выполнения на модели реализации свойств поведения, специфицированных на языке формальной логики. Метод model checking может быть полностью автоматизирован. По этой причине он предпочтительнее дедуктивной верификации в тех случаях, когда может быть использован. Однако всегда будут приложения, для полной верификации которых необходимо проводить доказательство теорем. В связи с этим исследования ведутся и в направлении, предусматривающем такую интеграцию дедуктивной

верификации и метода model checking, чтобы фрагменты системы, имеющие конечное число состояний, проверялись автоматически.

Главным недостатком верификации является то, что проверяется не реальная система, а ее абстрактная модель. Поскольку такая модель создается человеком, она может быть неадекватной, не сохранять существенных черт исходной системы, и в этом случае проверка спецификации для модели не выявит ошибок, имеющихся в реальной системе. С другой стороны, в модели могут появиться свойства, которых в реальной системе не существует.

Общепризнано, что на основании как моделирования и тестирования, так и верификации по отдельности, нельзя быть полностью уверенными в правильности разрабатываемых систем. Поэтому эти подходы можно считать взаимно дополняющими.

Одной из главных проблем, возникающих при применении метода model checking, является огромное число состояний моделей реальных систем. Верифицируемые системы обычно параллельны, и число состояний моделей таких систем растет экспоненциально с ростом числа компонентов. Этот эффект, названный «взрывом числа состояний», ограничивает применение обычных алгоритмов верификации model checking. Ранние системы верификации, реализующие алгоритмы model checking, могли работать с системами переходов, состоящими из 10^4–10^5 состояний [1].

Одним из подходов к решению проблемы «взрыва числа состояний» является символьная верификация. Символьные, или неявные алгоритмы, используют эффективный метод представления дискретных данных – множеств и отношений – в виде булевых функций в форме бинарных решающих диаграмм (BDDs, Binary Decision Diagrams) – канонических представлений булевых функций, зачастую являющихся существенно более компактными, чем представления стандартными способами, – и позволяют увеличить число состояний верифицируемых систем до астрономических значений. Ранние алгоритмы символьной верификации могли применяться к системам более чем с 10^{20} состояниями, дальнейшие их модификации позволяют работать с моделями, содержащими более чем 10^{120} состояний [2, 3].

Алгоритмы проверки модели для темпоральной логики CTL (логика ветвящегося времени, Computational Tree Logic) (как явные, так и неявные)

для каждой подформулы заданной темпоральной формулы вычисляют множество состояний модели (структуры Крипке), на которых эта подформула выполняется. Эти алгоритмы выполняют преобразования множеств состояний структуры Крипке итеративно до тех пор, пока множество, к которому эти преобразования применяются, не перестает изменяться, то есть вычисляют неподвижные точки преобразования. Можно показать, что для каждой темпоральной формулы CTL (EFφ, AFφ, E(φ_1Uφ_2), A(φ_1Uφ_2), EGφ, AGφ, где A, E – кванторы пути, F, G – темпоральные операторы, φ – формула пути) множество состояний структуры Крипке, на которых выполняется данная формула, является наименьшей или наибольшей неподвижной точкой некоторого монотонного оператора [4]. Теорема Тарского дает алгоритмы вычисления наибольших и наименьших неподвижных точек таких операторов. Эти вычисления могут быть реализованы либо явными алгоритмами, работающими с каждым элементом обрабатываемых множеств, либо неявными (символьными) алгоритмами, работающими с булевыми функциями в форме BDD, представляющими эти множества.

Сложность BDD зависит от порядка переменных, и хотя для большинства булевых функций представление в BDD линейно или полиномиально при любом порядке переменных, для некоторых функций число вершин в представляющих их BDD может существенно различаться при разных порядках переменных. Более того, найдены классы булевых функций, для которых представление в BDD экспоненциально для любых порядков переменных.

К проверке выполнимости формул темпоральной логики LTL (логика линейного времени Linear Time Logic) наиболее практичным является подход, основанный на теории автоматов. Если конечным образом задать язык, содержащий все цепочки, возможные в произвольной структуре Крипке M, а также язык, содержащий все цепочки, не удовлетворяющие заданной формуле φ логики LTL, и построить пересечение этих языков, проверив его на пустоту, то можно доказать, что φ не выполняется на M, и найти контрпример – те вычисления M, которые удовлетворяют формуле $\overline{\varphi}$.

Цепочки, характеризующие вычисления реагирующих систем (класса информационных систем, основной функцией которых является

поддержание взаимодействия с окружением), являются бесконечными и называются ω-словами. Множества ω-слов называются ω-языками. Некоторые ω-языки можно задать конечной моделью, которая называется автоматом Бюхи. Теоретико-автоматный подход к верификации реагирующих систем состоит в том, что строится автомат Бюхи, задающий все траектории анализируемой системы, другой автомат Бюхи, задающий все неправильные траектории, и анализируется синхронная композиция этих автоматов [4].

Построение автомата Бюхи, допускающего все цепочки, удовлетворяющие заданной LTL-формуле (отрицанию формулы, представляющей проверяемое требование), является одним из самых трудоемких этапов в теоретико-автоматном подходе к верификации. Существует несколько подходов к процедуре такой трансляции, все они концептуально сложны и обычно проводятся в два шага. Первый шаг – построение автомата Бюхи, гарантированно удовлетворяющего поставленным требованиям. На этом шаге обычно получается автомат, число состояний которого экспоненциально относительно длины (числа подформул) рассматриваемой формулы. Для того чтобы сделать эффективными следующие этапы процесса верификации, обычно выполняется второй шаг, который заключается в минимизации полученного автомата. К настоящему времени неизвестны эффективные общие процедуры получения по формуле LTL минимального автомата Бюхи; наилучшие известные методы основаны на использовании эвристик для такой минимизации [4].

Наибольшее распространение среди систем верификации, основанных на теоретико-автоматном подходе, получила система Spin. В системе Spin используется несколько приемов, позволяющих бороться с проблемой «взрыва числа состояний» и увеличить размер тех систем, для которых возможен исчерпывающий анализ.

В Spin не производится полного преобразования системы процессов в структуру Крипке: проверка выполнения свойств системы производится «на лету» (“on-the-fly”), только для части построенной системы переходов, которая постепенно достраивается в ограниченной области памяти.

Spin использует хэш-таблицу для хранения векторов состояний (данных, хранимых для каждого состояния), которые уже были исследованы.

На основании информации из вектора состояния хэш-функция вычисляет индекс элемента массива, в котором хранятся указатели на векторы состояний. В случае хэш-конфликтов (ситуаций, когда разные векторы состояний отображаются хэш-функцией в один и тот же слот хэш-таблицы) используются связные списки [5]. Важно, чтобы было как можно меньше хэш-конфликтов, чтобы исключить поиск в списках. Чем меньше размер вектора состояния, тем большее число элементов может поместиться в хэш-таблицу заданного размера. Чем больше число элементов в хэш-таблице, тем менее вероятно возникновение хэш-конфликта. При проведении верификации в Spin можно изменять размер используемой хэш-таблицы, добиваясь повышения эффективности верификации.

Spin позволяет использовать метод, определенным образом кодирующий вектор состояния, и тем самым позволяющий уменьшить его размер.

Векторы состояний могут быть сохранены без использования хэш-таблицы, а с применением представления, схожего с бинарными решающими диаграммами, – минимального автомата. В данном случае необходимый системе объем памяти может быть существенно снижен, однако время проведения верификации может увеличиться в несколько раз.

Одним из самых важных методов оптимизации, реализованных в Spin, является редукция частичных порядков, состоящая в том, что при верификации часть всех возможных путей (всех полных порядков частично упорядоченного множества операторов параллельной системы) отбрасывается. Например, интерливинг (перекрытие) операторов, работающих с локальными данными в каждом процессе, можно моделировать только одной последовательностью их выполнения: все остальные перестановки выполнения этих операторов будут эквивалентны с точки зрения любого проверяемого свойства. Этот прием позволяет системе Spin существенно снижать сложность проверяемых моделей параллельных систем [4].

Одним из направлений применения метода model checking является стоящая перед автором задача – верификация протоколов когерентности памяти. Методы поиска ошибок в устройствах, реализующих такие протоколы, основанные на симуляции со случайными данными, являются неэффективными. Ошибки могут проявиться лишь при возникновении

длинных последовательностей событий, таких как кэш-промахи и прием сообщений различными частями системы. Поскольку число таких последовательностей является комбинаторным, вероятность их возникновения во время моделирования со случайными данными резко уменьшается при увеличении их длины.

В [6] говорится о том, что системы, основанные на неявном представлении состояний модели (то есть использующие BDD, например, NuSMV), менее надежны и эффективны при верификации протоколов когерентности, чем системы с явным представлением (например, Spin). На то приводится ряд причин. Во-первых, скорость и необходимый объем памяти при использовании явного представления не сильно зависит от деталей структур данных, используемых в протоколах, в то время как BDD, например, не очень эффективно представляют FIFO. Во-вторых, верификация с явным представлением состояний может использовать преимущество редукции за счет симметрии, в то время как такая редукция в системах с неявным представлением может только уменьшить количество доказываемых случаев, но не сложность самих доказательств.

В [6, 7] сообщается, что существующие средства формальной верификации, основанные на алгоритмах model checking (Murphi, Spin), применяются для верификации промышленных протоколов когерентности памяти, и хорошо себя зарекомендовали. Однако при этом имеется ограничение на число узлов процессорной системы, отраженных в модели. В [6] этот параметр ограничивается значением 4.

В различных источниках – в частности, [6, 8], – поднимается проблема параметризованной верификации – доказательства корректности протокола при любых значениях числа процессоров в системе (либо каких-то других параметров). Для решения этой проблемы возможно использование некой комбинации методов model checking, композиционной верификации, абстракции и симметрии. Однако, например, при использовании техники абстракции возникает вопрос, всякое ли свойство, выполнимое на абстрактной модели, будет соблюдаться и для реальной системы. Это означает, что возможно появление ложных контрпримеров, не являющихся контрпримерами для реальной системы, и в этом случае нужно решать не только вопрос о том, как проводить абстракцию, но и как бороться с

ложными контпримерами, а также насколько возможно и необходимо автоматизировать эти процедуры.

Рассмотрим основные идеи методов композиционной верификации, абстракции и симметрии.

Многие конечные системы переходов образованы из большого числа процессов, работающих параллельно. Спецификации таких систем часто могут быть разбиты на отдельные свойства, описывающие поведение небольших фрагментов системы. Тогда (в рамках подхода, называемого композиционной верификацией) возможна следующая стратегия: проверить каждое локальное свойство, используя только тот фрагмент системы, который описывается этим свойством. Если мы сможем прийти к заключению, что система удовлетворяет всем локальным требованиям, и если мы знаем, что конъюнкция локальных свойств влечет выполнимость всей спецификации, то мы сможем прийти к выводу о том, что вся система в целом также удовлетворяет этой спецификации.

Одним из основных подходов в рамках метода абстракции является абстракция данных, которая применяется к описаниям системы на высшем ее уровне, еще до того как построена ее модель. Тем самым удается избежать построения нередуцированной модели, которая может оказаться слишком большой, чтобы поместиться в память. Абстракция данных предусматривает поиск отображения реальных значений данных, используемых в системе, в небольшое множество абстрактных значений данных. Распространив это отображение на состояния и переходы, можно построить абстрактную систему, которая симулирует исходную, но обычно имеет гораздо меньший размер. Из-за такого сокращения размера абстрактную систему зачастую удается верифицировать гораздо легче, чем исходную.

В основу методов редукции, использующих симметрию, положена идея о том, что проявление симметрии в параллельных системах с конечным числом состояний влечет за собой существование нетривиальной группы перестановок, которая сохраняет как разметку состояний, так и отношение переходов. Такие группы можно использовать для определения отношений эквивалентности в пространстве состояний систем. Модель, порожденная этим отношением, часто имеет меньший размер, нежели исходная модель. Кроме того, она бисимуляционно эквивалентна исходной модели. Поэтому

ее можно использовать для верификации любого свойства исходной модели, выраженного CTL*-формулой [1].

Помимо верификации самих протоколов когерентности памяти, важна верификация средств, реализующих эти протоколы. Для этого необходимы тесты (на языке ассемблера), которые можно получить на основании результатов верификации формальной модели протокола. В [9] приводится пример такой комбинации формальных и неформальных методов: на основании данных, получаемых при формальной верификации, строятся воздействия, являющиеся входными воздействиями для системы, моделирующей соответствующую Verilog-реализацию.

Предполагается применить формальную верификацию с использованием инструмента Spin к протоколу когерентности памяти системы на кристалле «Эльбрус-2S». Стандартная конфигурация системы на кристалле «Эльбрус-2S» представляет собой четыре четырехъядерных процессорных модуля, связанных друг с другом высокоскоростными каналами. Когерентность доступа нескольких процессоров в общую память поддерживается посредством запросов проверки когерентности, рассылаемых специальным устройством – системным коммутатором – в процессоры системы. Кэш-память второго уровня реализует протокол когерентности MOSI (от названия основных состояний кэш-строки – Modified, Owned, Shared, Invalid). Описание протокола планируется выполнить на языке Promela (входном языке пакета Spin), который предоставляет возможность пользователю строить модели параллельных систем для последующей проверки в их поведении аспектов координации и взаимодействия параллельных процессов. Для этого необходимо абстрагироваться от совокупности множества устройств, реализующих протокол, и представить протокол в виде набора параллельных процессов, решив проблемы их взаимодействия и синхронизации. На основании представлений о том, как должен работать протокол когерентности памяти и что он должен обеспечить, нужно сформулировать требования к протоколу сначала на естественном языке, а затем привести их к формальному виду с помощью формул темпоральной логики. Имея модель и набор требований к ней, можно провести формальную верификацию. На основании результатов верификации предполагается, помимо заключений о корректности протокола

(и его модели), получить тесты для Verilog-модели системы (и/или C++-модели), и исследовать их поведение.

Литература:

1. *E. M. Clarke, O. Grumberg, D. Peled.* Model Checking. // MIT Press, 1999 – 314 pp.

2. *J.R. Burch, E.M. Clarke, K.L. McMillan*. Symbolic Model Checking: 10^{20} States and Beyond, 1990.

3. *J.R. Burch, E.M. Clarke, D.E. Long, K.C. McMillian, D.L. Dill*. Symbolic Model Checking for Sequential Circuit Verification, 1993.

4. *Карпов Ю.Г.* MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010 – 560 с.

5. *Mordechai Ben-Ari*. Principles of the Spin Model Checker. – Springer, 2008 – 232 pp.

6. *C. Chou, P. K. Mannava, S. Park.* A Simple Method for Parameterized Verification of Cache Coherence Protocols, 2004.

7. *J. Harrison.* Formal Methods at Intel – An Overview. // Second NASA Formal Methods Symposium, 2010.

8. *K. L. McMillan*. Parameterized Verification of the FLASH Cache Coherence Protocol by Compositional Model Checking, 2001.

9. *D. Abts, S. Scott, D. Lilja.* So Many States, So Little Time: Verifying Memory Coherence in the Cray X1, 2003.

УДК 004.052.42

ВЕРИФИКАЦИЯ ТЕХНИЧЕСКИХ СИСТЕМ МЕТОДОМ ПРОВЕРКИ МОДЕЛИ

В.С. Буренков, С.Р. Иванов

Формальная верификация – приемы и методы формального доказательства (или опровержения) того, что модель верифицируемой системы удовлетворяет заданной формальной спецификации.

Для верификации технических систем свойства их поведения должны быть выражены формально логическими утверждениями, которые обеспечат простую, лаконичную и недвусмысленную их запись. Обычная логика высказываний является неадекватной для формулировки утверждений о поведении технических систем, то есть об изменении их состояний во времени. Для спецификации таких свойств необходимы логические утверждения, истинность которых зависит от времени.

Темпоральные логики – логики, в которых истинностное значение логических формул зависит от момента времени, в котором вычисляются значения этих формул.

Наиболее часто рассматривают три темпоральных логики:

- расширенная темпоральная логика ветвящегося времени (CTL*, Extended Computational Tree Logic);

- темпоральная логика линейного времени (LTL, Linear Time Logic);

- темпоральная логика ветвящегося времени (CTL, Computational Tree Logic).

LTL и CTL являются подмножествами CTL*; именно эти логики используются, в основном, в области верификации технических систем. Обе логики интерпретируются (т.е. их формулы принимают истинное или ложное значение) на структуре Крипке – недетерминированной конечной системе переходов, удобной для представления динамики поведений дискретных систем. LTL рассматривает все вычисления структуры Крипке как множество бесконечных траекторий поведения системы. Формулы этой логики построены из атомарных утверждений, связанных логическими операциями и темпоральными операторами. CTL рассматривает возможные варианты развития вычислений в каждом состоянии бесконечных траекторий. Обе логики могут быть использованы для формулировки утверждений о поведении технических систем. Существуют свойства, выразимые в одной логике и невыразимые в другой.

Формулы темпоральной логики оказались наиболее удобным средством для задания требуемых свойств поведения реагирующих систем (reactive systems) – класса информационных систем, основной функцией которых является поддержание взаимодействия с окружением, а не преобразование информации [1]. Останов таких систем обычно связан с поломкой или коллизией (блокировкой) и является ошибкой. Операционные

системы, протоколы коммуникации, планировщики, контроллеры, параллельные взаимодействующие программы, системы логического управления, драйверы – примеры реагирующих систем. Верификация реагирующей системы не может быть сведена к проверке отношения между входом и выходом после прихода системы в заключительное состояние: такие системы имеют начальное состояние, но не имеют заключительного состояния, они вообще не должны останавливаться.

Model checking (проверка модели) – совокупность моделей, приемов и алгоритмов, позволяющих проверить, что формула темпоральной логики, выражающая некоторое свойство поведения динамической системы во времени, выполняется (является истинной) на модели системы с конечным числом состояний (структуре Крипке).

Метод верификации model checking реализует следующие шаги:

1. для верифицируемой системы строится адекватная модель Крипке. Варианты поведения реальной системы представляются разверткой (деревом вычислений) построенной структуры Крипке;

2. с помощью переменных и параметров верифицируемой системы выражаются интересующие разработчика атомарные предикаты структуры Крипке – логические выражения, которые могут принимать значения «истина» или «ложь» в каждом состоянии системы;

3. проверяемое свойство выражается формулой φ темпоральной логики с использованием атомарных утверждений, темпоральных операторов и кванторов пути;

4. с помощью полностью автоматизированной процедуры проверяется истинность утверждения, что φ истинна для построенной структуры Крипке.

Формальная модель системы, как правило, значительно проще самой проверяемой системы, это абстракция, в которой должны быть отражены наиболее существенные характеристики системы. Существуют инструменты верификации (как коммерческие, так и распространяемые свободно), реализующие алгоритмы model checking для LTL (например, Spin), CTL (например, SMV, NuSMV). Пакеты верификации автоматизируют все этапы верификации, в том числе они освобождают пользователя и от необходимости вручную строить структуру Крипке верифицируемой системы. Модель верифицируемой системы описывается на входном языке

пакета. С помощью определенных алгоритмов пакет верификации автоматически строит структуру Крипке каждого из компонентных модулей заданной системы, строит их композицию и автоматически выполняет процесс верификации – проверку выполнимости заданной темпоральной формулы на структуре Крипке. Если формула не выполняется, пакет может выдать контрпример, показывающий поведение системы, нарушающее проверяемое свойство.

Алгоритмы проверки модели осуществляют вычисление множества состояний структуры Крипке, на котором выполняется заданная формула темпоральной логики. Увеличение числа компонентов верифицируемой системы или увеличение числа переменных экспоненциально увеличивает число состояний структуры Крипке. Этот эффект, названный «взрывом числа состояний», ограничивает применение обычных алгоритмов верификации model checking. Ранние системы верификации, реализующие алгоритмы model checking, могли работать с системами переходов, состоящими из 10^4–10^5 состояний [2].

Разработанные неявные, или символьные алгоритмы, использующие эффективный метод представления дискретных данных – множеств и отношений – в виде булевых функций в форме бинарных решающих диаграмм (BDDs, Binary Decision Diagrams) – канонических представлений булевых функций, зачастую являющихся существенно более компактными, чем представления стандартными способами, – позволяют увеличить число состояний верифицируемых систем до астрономических значений. Для логических схем и программ разработаны очень эффективные системы символьной верификации, что привело к возможности использования алгоритмов верификации model checking для реальных, разрабатываемых промышленностью систем и сделало верификацию этапом технологии разработки промышленных систем. Ранние алгоритмы символьной верификации могли применяться к системам более чем с 10^{20} состояниями, дальнейшие их модификации позволяют работать с моделями, содержащими более чем 10^{120} состояний [3, 4].

Обоснованием алгоритмов проверки свойств, выраженных формулами темпоральной логики CTL, является определение этих формул с помощью неподвижных точек (наибольших и наименьших) операторов на множестве состояний структуры Крипке. Теорема Тарского дает алгоритмы вычисления

наибольших и наименьших неподвижных точек таких операторов. Эти вычисления могут быть реализованы либо явными алгоритмами, работающими с каждым элементом обрабатываемых множеств, либо неявными (символьными) алгоритмами, работающими с булевыми функциями в форме BDD, представляющими эти множества.

К решению проблемы «взрыва числа состояний» существуют и другие подходы, помимо символьных алгоритмов: композициональная верификация, редукция частичных порядков, верификация «на лету». Хотя ни одна из разработанных техник не является универсально применимой, на основе каждой из них построены инструментальные системы верификации.

Одним из направлений применения метода model checking является стоящая перед автором задача – верификация протоколов когерентности памяти. Методы поиска ошибок в устройствах, реализующих такие протоколы, основанные на симуляции со случайными данными, являются неэффективными. Ошибки могут проявиться лишь при возникновении длинных последовательностей событий, таких как кэш-промахи и прием сообщений различными частями системы. Поскольку число таких последовательностей является комбинаторным, вероятность их возникновения во время симуляции со случайными данными резко уменьшается при увеличении их длины. В литературе приводятся примеры верификации протоколов когерентности, например верификации символьными методами с использованием инструмента SMV протоколов Gigamax [5, 6] и Futurebus+ [2, 7]. При верификации протоколов возникает ряд проблем. Нужно разрешить вопросы, связанные с адекватностью реальной системе создаваемой модели протокола. Необходимо убедиться в корректности реализации верифицируемого протокола. Для сложных протоколов, метод model checking может быть успешно применен только для малого числа кэш-памятей в системе. Однако в принципе протоколы спроектированы для работы с любым числом кэш-памятей, поэтому поиск и применение методов доказательства корректности протоколов при наличии любого числа кэшей также могут быть осуществлены.

Литература:

1. *Карпов Ю.Г.* MODEL CHECKING. Верификация параллельных и распределенных программных систем. – СПб.: БХВ-Петербург, 2010 – 560 с.

2. *E. M. Clarke, O. Grumberg, D. Peled*. Model Checking. // MIT Press, 1999 – 314 pp.

3. *J.R. Burch, E.M. Clarke, K.L. McMillan*. Symbolic Model Checking: 10^{20} States and Beyond, 1990.

4. *J.R. Burch, E.M. Clarke, D.E. Long, K.C. McMillian, D.L. Dill*. Symbolic Model Checking for Sequential Circuit Verification, 1993.

5. *K.L. McMillan*. Symbolic Model Checking: An Approach to the State Explosion Problem, Ph.D. Dissertation. // Carnegie Mellon University, 1992.

6. *K.L. McMillan, J. Schwalbe*. Formal Verification of the Gigamax Cache Consistency Protocol, 1997.

7. *E. M. Clarke, O. Grumberg, H. Hiraishi, S. Jha, D.E. Long, K.L. McMillan, L.A. Ness.* Verification of the Futurebus + Cache Coherence Protocol, 1993.

УДК 004. 318

КОНФИГУРИРОВАНИЕ, НАСТРОЙКА И ПРОГРАММИРОВАНИЕ МОДУЛЕЙ

Н.И. Ермохина, С.В. Ибрагимов, В.Я. Хартов

СИСТЕМЫ НА КРИСТАЛЛЕ PSOC

В настоящее время возрастающие требования к объему и энергопотреблению электронной аппаратуры стимулируют дальнейшее повышение степени интеграции электронных устройств. Как правило, электронное устройство можно разделить на две части: цифровую и аналоговую. Цифровая часть может состоять из микропроцессора (или микроконтроллера) и дополнительных аппаратных блоков, программная реализация функций которых может оказаться нежелательна. Аналоговая часть содержит устройства и схемы взаимодействия с аналоговыми датчиками.

Программируемые системы на кристалле (PSoC) представляют собой аналогово-цифровые интегральные микросхемы, позволяющие реализовывать все части проекта на системном уровне интеграции. PSoC

включают в себя процессорное ядро, аналоговые и цифровые конфигурируемые блоки и программируемую матрицу межсоединений и позволяют обойтись минимумом внешних элементов при разработке изделия, увеличивая его технологичность и значительно сокращая стоимость и время разработки. Широкие возможности конфигурирования PSoC позволяют легко адаптировать проектное решение на их основе к поставленным требованиям. Компания Cypress, известный производитель PSoC, предоставляет три линейки систем в зависимости от функциональных возможностей и процессорного ядра:

1. PSoC 1, рекомендуемая для замены 8-битных микроконтроллеров с 8-битным ядром M8,
2. PSoC 3 с ядром 8051 и тактовой частотой 67МГц,
3. PSoC 5, высокопроизводительная с ядром ARM Cortex-M3.

Архитектура PSoC (на примере PSOC 5)

Функционально PSOC можно разделить на следующие подсистемы.

Подсистема центрального процессора. Состоит из процессорного ядра ARM, контроллера прерываний, контроллера прямого доступа к памяти и контроллера кэш-памяти.

Подсистема программирования и отладки. Реализует возможности внутрисхемного программирования и отладки с помощью интерфейсов JTAG и SWD.

Подсистема памяти. В состав PSOC входит три вида памяти: статическая память с произвольным доступом, флэш-память программ и перепрограммируемое ПЗУ для хранения параметров, настроек и т.д.

Цифровая подсистема. Состоит из матрицы универсальных цифровых блоков, массива программируемых таймеров/счетчиков и блока аппаратной поддержки USB 2.0. Каждый универсальный цифровой блок соединен с системной шиной посредством пары регистров.

Аналоговая подсистема. Состоит из аналогово-цифровых и цифрово-аналоговых преобразователей, блока цифровой фильтрации, операционных усилителей, компараторов и универсальных аналоговых блоков. Универсальный аналоговый блок строится на основе операционного усилителя по технологии переключаемых конденсаторов и, в зависимости от конфигурации цепей обратной связи, может выполнять роль усилителя с программируемым коэффициентом усиления, трансимпедансного усилителя,

устройства выборки/хранения и т.д.

Общесистемные ресурсы. Программируемое дерево синхронизации, часы точного времени, система управления энергосбережением, подсистема ввода/вывода и т.д.

Для взаимодействия с внешними устройствами применяется многофункциональные конфигурируемые модули ввода/вывода, связанные с каждым выводом микросхемы. Благодаря этому, режим работы линий ввода/вывода может меняться в процессе работы устройства, обеспечивая как аналоговый, так и цифровой ввод/вывод.

Для связи компонентов PSOC между собой с модулями ввода/вывода применяются две программируемые матрицы межсоединений, учитывающие специфические требования, предъявляемые к передаче аналоговых и цифровых сигналов.

Блок цифровой фильтрации реализован на основе ядра цифрового сигнального процессора и, используя механизм прямого доступа к памяти, позволяет осуществлять фильтрацию оцифрованных сигналов без нагрузки на ЦПУ.

Средства разработки и отладки для PSOC

Для разработки и отладки проектов используют две среды: PSoC Designer (для линейки PSoC 1) и PsoC Creator (для линеек PSoC 3 и 5). Обе среды предоставляют возможность максимально простого конфигурирования и программирования PSOC. Реализация проекта на PSOC предполагает использование отдельных компонентов, как специфических для конкретной задачи, так и библиотечных компонентов общего назначения. Такая модульная архитектура значительно упрощает процесс проектирования и повышает уровень повторного использования кода.

Компонент может включать в себя аппаратную реализацию компонента на основе конфигурируемых ресурсов PSOC, шаблоны программного кода (API) для взаимодействия с аппаратной частью, набор параметров, влияющих на реализацию или функциональные возможности и программу на языке Си для упрощения задания параметров. При компиляции проекта, для каждого экземпляра каждого компонента проекта по шаблонам, созданным разработчиком компонента, генерируются файлы со всем необходимым кодом для взаимодействия с компонентом.

Аппаратная часть компонента реализуется либо с помощью встроенного схемного редактора, либо описывается на языке Verilog. Совместное описание аппаратной и программной части компонента и возможность использовать компонента при построении других компонентов позволяет пользователю рассматривать компонент как "черный ящик", абстрагировавшись от деталей его внутреннего устройства, используя объектно-ориентированный подход при проектировании.

При компиляции проводится синтез сгенерированного HDL описания проекта и анализ полученной реализации для определения временных параметров. Отчеты синтезатора и результаты анализа временных параметров доступны пользователю в удобном для чтения формате.

Модуль отладки и трассировки обеспечивает возможность внутрисхемной отладки через интерфейсы JTAG и SWD. Отладчик уровня исходного теста, включенный в среду разработки, имеет широкие функциональные возможности, в частности чтение и изменение содержимого памяти, пошаговое исполнение кода, установку точек останова и т.д.

Примеры конфигурирования и настройки модулей системы

Пример 1 (платформа PSoC 3 – кристалл CY8C3866AXI-040, отладочный набор CY8CKIT-030). Продемонстрируем генерацию синусоиды, используя ЦАП и прямой доступ к памяти. Осуществим мигание светодиода с частотой, пропорциональной напряжению, задаваемому пользователем с помощью потенциометра и преобразуемому АЦП в 8-битовый код, а также выведем это значение на LCD-дисплей. Схема аппаратной части тестового проекта приведена на рисунке 1.

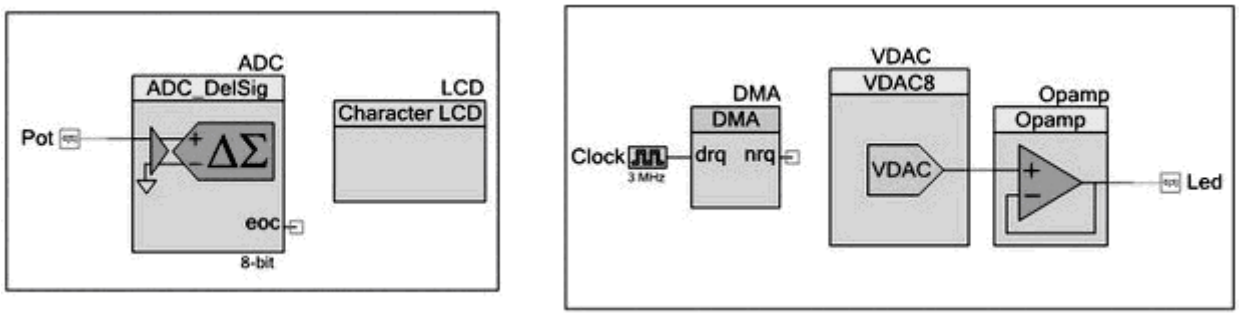
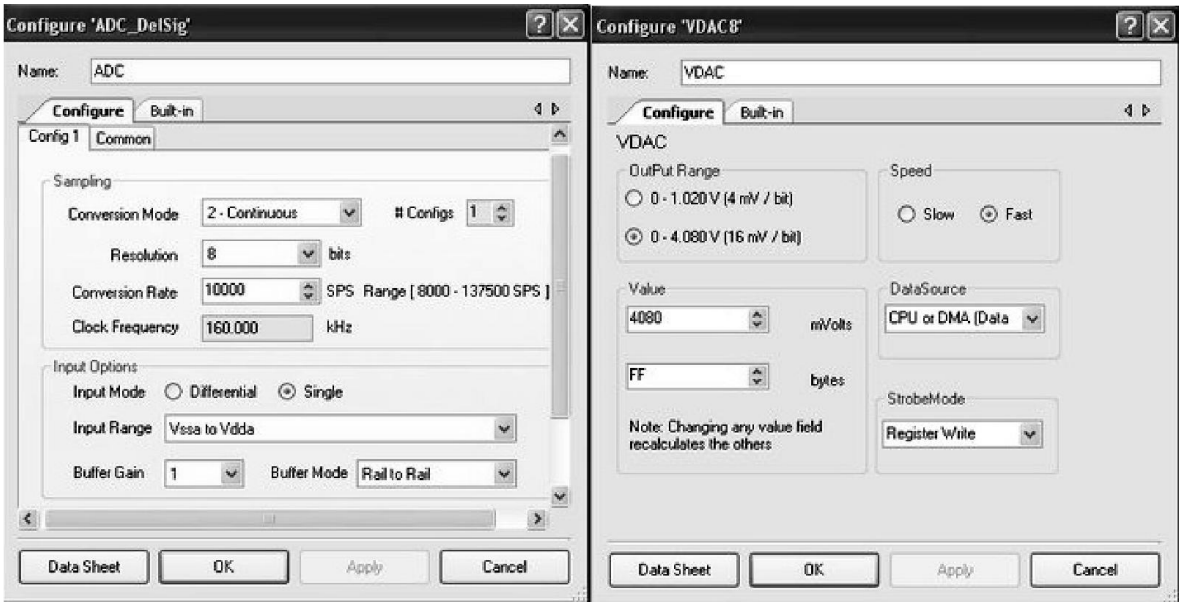


Рисунок 1. Аппаратная часть примера 1

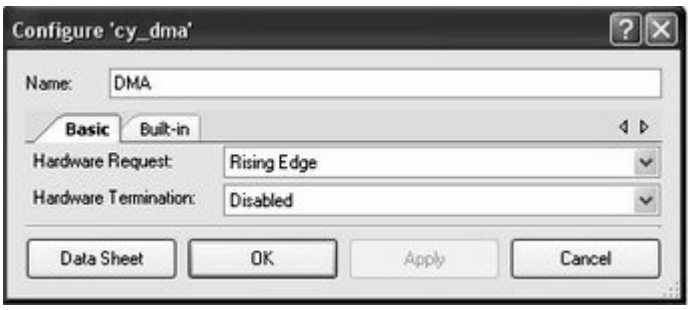
В программе (см. листинг программы) задаем значения для синусоидальной волны в интервале от 0x3D до 0x9F (так как это две

критичные точки для светодиода – когда он не горит и когда находится в режиме насыщения). Считываемое значение напряжения, заданное потенциометром, поступает на сигма-дельта АЦП (на рис.1 видно, что вывод потенциометра, обозначенный меткой Pot, непосредственно связан с входом АЦП). Полученное значение записывается в переменную voltageRawCount, значение которой выводится на дисплей. В зависимости от этой переменной расчитывается делитель частоты и уже с новой частотой тактирования происходит обращение компонентом прямого доступа к памяти (DMA) к памяти данных, где хранятся значения напряжений синусоидальной волны. Считанные данные заносятся в регистр данных, откуда сразу поступают на ЦАП, который в аналоговом виде выводит их на светодиод (Led) с частотой, пропорциональной значению от АЦП.

Так как значения напряжений, задаваемых потенциометром, должны лежать в интервале 0–255, то необходимо настроить компонент сигма-дельта АЦП и установить параметр разрешения 8 бит (остальные параметры автоматически подстроятся под данное изменение). Компонент DMA должен быть настроен на срабатывание по перепаду из 0 в 1 (Rising Edge). Для компонента ЦАП: источник данных – DMA, Register Write – любые данные, записываемые в регистры данных и сразу пересылаемые в ЦАП. Настройка компонентов приведена на рисунке 2.



а) б)



в)

Рисунок 2. Настройка компонентов сигма-дельта АЦП (а), ЦАП (б) и DMA(в)

Листинг программного кода тестового примера:

```
#include <device.h>
/* Переменные LCD-модуля */
#define ROW_0                0
#define COLUMN_0             0
#define COLUMN_9             9
#define COLUMN_10            10
#define COLUMN_11            11
/* Очистка десятых и сотых */
#define CLEAR_TENS_HUNDREDS     "    "
/* Очистка сотых */
#define CLEAR_HUNDREDS          "   "
/* Значения минимального и максимального уровней напряжения */
#define MIN_COUNT            0
#define MAX_COUNT            0xFF
/* Настройка DMA */
#define DMA_BYTES_PER_BURST     1
#define DMA_REQUEST_PER_BURST   1
#define DMA_SRC_BASE         (CYDEV_SRAM_BASE)    /* Адрес
источника */
#define DMA_DST_BASE         (CYDEV_PERIPH_BASE)  /* Адрес
назначения */
/* Переменные для DMA */
uint8 DMA_Chan;
uint8 DMA_TD[1];
```

```
/* Таблица значений напряжений для DMA, посылаемых в ЦАП */
const uint8 voltageWave[] =
{ 0x6D,0x6F,0x71,0x73,0x75,0x77,0x79,0x7B,0x7D,0x7F,0x81,0x83,0x85,0x87,
0x89,0x8B,0x8D,0x8F,0x91,0x93,0x95,0x97,0x99,0x9B,0x9C,0x9D,0x9D,0x9E,
0x9E,
0x9F,0x9F,0x9F,0x9E,0x9E,0x9E,0x9C,0x9C,0x9B,0x99,0x97,0x95,0x93,0x91,0
x8F,
0x8D,0x8B,0x89,0x87,0x85,0x83,0x81,0x7F,0x7D,0x7B,0x79,0x77,0x75,0x73,0
x71,
0x6F,0x6D,0x6B,0x69,0x67,0x65,0x63,0x61,0x5F,0x5D,0x5B,0x59,0x57,0x55,0
x53,
0x51,0x4F,0x4D,0x4B,0x49,0x47,0x45,0x43,0x41,0x40,0x40,0x3F,0x3F,0x3D,0
x3D,
0x3D,0x3D,0x3D,0x3D,0x3F,0x41,0x43,0x45,0x47,0x49,0x4B,0x4D,0x4F,0x51,
0x53,
0x55,0x57,0x59,0x5B,0x5D,0x5F,0x61,0x63,0x65,0x67,0x69,0x6B};
void main()
{ int16 voltageRawCount;
  /* Разрешаем работу АЦП, ЖК-модуля, ЦАП, ОУ */
  ADC_Start();
  LCD_Start();
  VDAC_Start();
  Opamp_Start();
  /* Устанавливаем курсор ЖК-дисплея на строку 0, колонку 0 */
  LCD_Position(ROW_0, COLUMN_0);
  /* Выводим на ЖК-дисплей */
  LCD_PrintString("V Count: ");
  /* Конфигурация DMA */
  DMA_Chan =
DMA_DmaInitialize(DMA_BYTES_PER_BURST,DMA_REQUEST_PER_BUR
ST,
  HI16(DMA_SRC_BASE),HI16(DMA_DST_BASE)); /* Инициализировать
канал */
  DMA_TD[0] = CyDmaTdAllocate();  /* Выделить дескриптор транзакций */
```

```
CyDmaTdSetConfiguration(DMA_TD[0],116,DMA_INVALID_TD,TD_INC_SR
C_ADR);
  /* Настроить дескриптор транзакций */
  CyDmaTdSetAddress(DMA_TD[0],LO16((uint32)voltageWave),
  LO16((uint32)VDAC_Data_PTR));  /* Задать адрес источника и назначения
*/
  CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
  CyDmaChEnable(DMA_Chan, 1); /* Разрешить работу канала */
  Clock_Start();       /* Разрешить работу генератора синхросигнала */
  ADC_StartConvert();  /* Начать преобразование */
  while(1)
  /* Ждем окончания преобразования */
  {ADC_IsEndConversion(ADC_WAIT_FOR_RESULT);
   voltageRawCount = ADC_GetResult16();   /* Получаем результат */
   /* Минимальная граница */
   if (voltageRawCount < MIN_COUNT)
   {voltageRawCount = MIN_COUNT;
   }
   /* Максимальная граница */
   else if(voltageRawCount > MAX_COUNT)
   {voltageRawCount = MAX_COUNT;
   }
   else {}
   /* Настройка делителя частоты. Светодиод мигает в зависимости от
величины
   voltageRawCount. При частоте синхронизации 3 МГц минимальный
делитель (при
   напряжении 0В) равен 1000 для мигания с максимальной скоростью,
   наибольший делитель – 52200 для мигания с минимальной скоростью. */
   Clock_SetDivider(((uint32)999*260+voltageRawCount)/ (260-
voltageRawCount));
   /* Передвигаем курсор на LCD-дисплее */
   LCD_Position(ROW_0, COLUMN_9);
   LCD_PrintNumber(voltageRawCount);
   if (voltageRawCount < 10)
```

```
{/* Передвигаем курсор на LCD-модуле */
LCD_Position(ROW_0,COLUMN_10);
/* Удаляем предыдущие значения переменных */
LCD_PrintString(CLEAR_TENS_HUNDREDS);
}
else if (voltageRawCount < 100)
{/* Передвигаем курсор на LCD-модуле */
LCD_Position(ROW_0,COLUMN_11);
LCD_PrintString(CLEAR_HUNDREDS);
}
else
{ }   }   }
```

Пример 2 (платформа PSoC 5 – кристалл CY8C5588AXI-ES1, отладочный набор CY8CKIT-014). Разработаем частотный манипулятор, используя его для передачи сообщения по протоколу UART. Схема аппаратной части проекта приведена на рисунке 3.

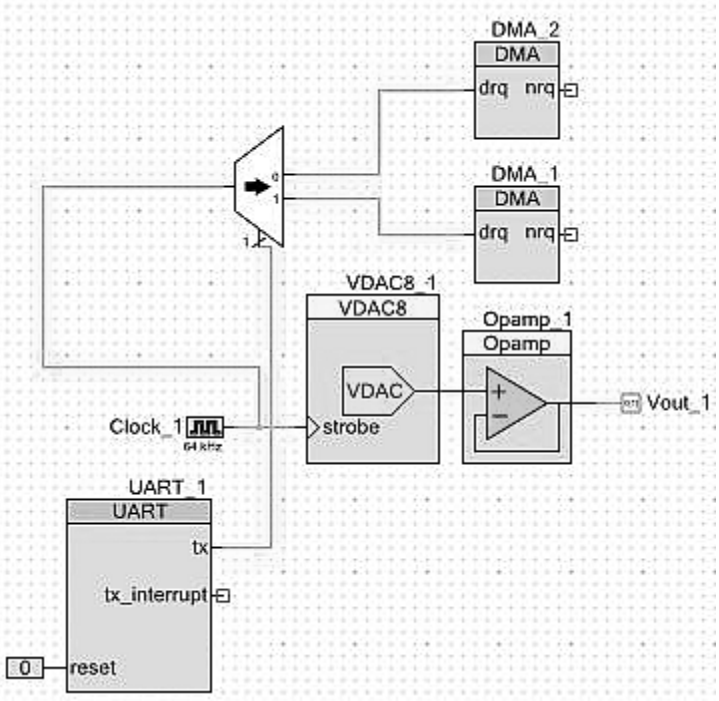


Рисунок 3. Схема аппаратной части примера 2

Частотный манипулятор реализуем на основе ЦАП с разрешением восемь бит и двух независимых каналов DMA. Выбор используемого канала DMA осуществляется с помощью цифрового мультиплексора. Оба канала DMA настроены на передачу входного кода из массива, расположенного во

Flash-памяти, в регистр данных ЦАП, формируя на выходе последнего нужную временную диаграмму. Таким образом, ресурсы центрального процессора не тратятся на генерацию сигнала. Входной сигнал манипулятора снимается с выхода стандартного компонента UART, настроенного на передачу. Сигнал синхронизации Clock_1 с частотой 64 кГц используется для генерации выходных частот, вычисляемых по формуле:

$$f_{вых} = \frac{f_{clk}}{N},$$

где $f вых$ – частота выходного сигнала,

$f clk$ – частота синхросигнала,

N – количество отсчетов в одном периоде выходного сигнала.

Листинг программного кода проекта:

```
#include <device.h>
#include <VDAC8_1.h>
#include <Clock_1.h>
#include <Opamp_1.h>
#include <DMA_1_dma.h>
#include <DMA_2_dma.h>
#include <UART_1.h>
#define TABLE_LENGTH 32 //Количество отсчетов первой частоты.
#define TABLE_LENGTH1 64        //Количество отсчетов второй частоты.
#define DMA_BYTES_PER_BURST 1      //Передача 1 байта за транзакцию
#define DMA_REQUEST_PER_BURST 1
//Осуществление транзакции только по запросу
CYCODE const uint8 sineTable[TABLE_LENGTH]
={128,152,176,198,218,234,245,253,255,253,245,234,218,198,176,152,127,103,7
9,57,37,21,10,2,0,2,10,21,37,57,79,103};
CYCODE const uint8 sineTable1[TABLE_LENGTH1]
={128,140,152,165,176,188,198,208,218,226,234,240,245,250,253,254,255,254,2
53,250,245,240,234,226,218,208,198,188,176,165,152,140,127,115,103,90,79,67,
57,47,37,29,21,15,10,5,2,1,0,1,2,5,10,15,21,29,37,47,57,67,79,90,103,115};
#define DMA_SRC_BASE (&sineTable[0]) //Адрес источника первого канала
#define DMA_SRC_BASE1 (&sineTable1[0])//Адрес источника второго
канала
```

```
#define DMA_DST_BASE (CYDEV_PERIPH_BASE)  //Адрес назначения
void main()
{
uint8 DMA_Chan;
uint8 DMA_TD[1];
uint8 DMA_Chan1;
uint8 DMA_TD1[1];
    CyGlobalIntEnable;  //Разрешить прерывания
    Clock_1_Start();    //Разрешить работу генератора синхросигнала
    VDAC8_1_Start();    //Включить ЦАП
    Opamp_1_Start();    //Включить операционный усилитель
//Инициализировать первый канал
DMA_Chan = DMA_1_DmaInitialize(DMA_BYTES_PER_BURST,
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE),
HI16(DMA_DST_BASE));
    DMA_TD[0] = CyDmaTdAllocate();//Выделить дескриптор транзакции
CyDmaTdSetConfiguration(DMA_TD[0], TABLE_LENGTH, DMA_TD[0],
TD_INC_SRC_ADR); //Настроить дескриптор транзакции
CyDmaTdSetAddress(DMA_TD[0],LO16((uint32)sineTable),
LO16((uint32)VDAC8_1_Data_PTR));//Задать адреса источника и назначения
    CyDmaChSetInitialTd(DMA_Chan, DMA_TD[0]);
    CyDmaChEnable(DMA_Chan,1);        //Разрешить работу канала
//Инициализировать второй канал
DMA_Chan1 = DMA_2_DmaInitialize(DMA_BYTES_PER_BURST,
DMA_REQUEST_PER_BURST, HI16(DMA_SRC_BASE1),
HI16(DMA_DST_BASE));
    DMA_TD1[0] = CyDmaTdAllocate();//Выделить дескриптор транзакции
    CyDmaTdSetConfiguration(DMA_TD1[0], TABLE_LENGTH1,
DMA_TD1[0],
 TD_INC_SRC_ADR); //Настроить дескриптор транзакции
    CyDmaTdSetAddress(DMA_TD1[0], LO16((uint32)sineTable1),
LO16((uint32)VDAC8_1_Data_PTR));//Задать адреса источника и назначения
    CyDmaChSetInitialTd(DMA_Chan1, DMA_TD1[0]);
    CyDmaChEnable(DMA_Chan1, 1);       //Разрешить работу канала
    UART_1_Start();                    //Разрешить работу блока UART
```

```
    UART_1_PutString("Hello, World!"); //Передать строку
    for(;;) {}                  //бесконечный цикл
}
```

Выводы. Высокая степень интеграции PSoC, разнообразие конструктивных исполнений и сравнительно низкая цена позволяют применять системы на кристалле для разработки самых разных устройств автоматики, медицины, потребительской электроники, связи и др. PSOC младшей серии (1) могут с успехом применяться вместо традиционных 8-битных микроконтроллеров. Благодаря наличию встроенных аналоговых блоков, количество внешних элементов в схеме устройства удается сократить, а возможность аппаратной реализации цифровых функций с повышенными требованиями к быстродействию нивелирует относительно низкую производительность ЦПУ. Наличие сопроцессора цифровой обработки сигналов и высокопроизводительного ядра ARM Cortex-M3 в серии PSoC 5 дает возможность использовать их в задачах радиосвязи и телекоммуникаций, а наличие конфигурируемых цифровых блоков дает возможность использовать PSoC вместо ПЛИС малой емкости для решения задач специфической обработки цифровых сигналов.

Литература:

1. PSoC 3: CY8C38 Family Data sheet
2. CY8CKIT-030 PSoC 3 Development Kit Guide
3. PSOC 5: CY8C55 Family Data Sheet
4. CY8CKIT-014 PSOC 5 First Touch Starter Kit Guide
5. *J. Pardue.* C Programming for microcontrollers. 2005
6. PSoC 5 Architecture Technical Reference Manual

УДК: 519.178

РАСЧЕТ ВРЕМЕНИ ВЫПОЛНЕНИЯ РАБОТЫ ПРОГРАММНЫХ МОДУЛЕЙ В УЗЛАХ ВЫЧИСЛИТЕЛЬНОЙ СЕТИ

Е.И. Ермохина, Н.М. Созинова, Ю.М. Руденко

Часто при создании вычислительной сети возникает вопрос о необходимости учитывать время, затрачиваемое на передачу информации между узлами. В связи с этим перед нами ставиться задача оптимизации времени выполнения работы всей ВС.

Для достижения этой цели необходимо разработать алгоритм распределения вершин информационного графа по процессорам заданной структуры вычислительной сети. В результате этого распределения исходная задача должна решаться за минимально возможное время на ВС. Число процессоров при этом должно быть минимизировано с учетом обеспечения решения задачи за минимальное время.

При этом необходимо проанализировать большое количество условий, учесть множество различных ситуаций, которые возникают при распределении операторов по нитям и нитей по процессорам ВС. Кроме того, необходимы точные исходные данные, такие как времена расчета отдельно взятых операторов, объем передаваемых данных между ними. Необходимо также знать времена передачи данных между процессорами в структуре ВС. Решение задачи создания таких алгоритмов и их последующее применение увеличит быстродействие обработки данных на многопроцессорных системах.

Рассмотрим решение данных проблем на примере граф-схемы, приведенной на рисунке 1.

Конфигурации граф-схем позволяют представить в удобной форме все существующие операторы последовательных языков программирования, а также представлять схемы обмена данными между ВМ.

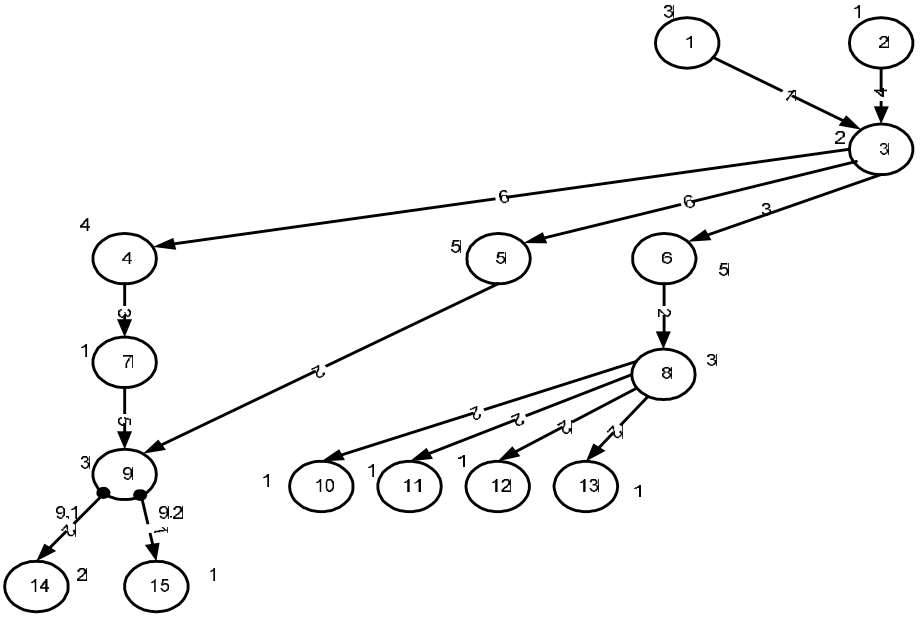


Рисунок 1. Рассматриваемая граф-схема

В данной работе рассматриваются методы отображения вычислительных модулей (ВМ) на структуре вычислительной сети (ВС), типа гиперкуб. Выберем обобщенный трехмерный гиперкуб 2x3x3, представленный на рисунке 2.

Рисунок 2.Обобщенный трехмерный гиперкуб 2x3x3

Поскольку в данной работе проектируется ВС с распределенной памятью, то необходимо учитывать время обмена данными между ВМ. Для этого построим расширенную матрицу следования с указанием весов дуг и вершин данного ИЛГ, приведенную в таблице 1.

Таблица 1.

Расширенная матрица следования.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*P*
1																*3*
2																*1*
3	*1*	*4*														*2*
4			*6*													*4*
5			*6*													*5*
6			*3*													*5*
7				*3*												*1*
8						*2*										*3*
9							*5*									*3*
10								*2*								*1*
11								*2*								*1*
12								*2*								*1*
13								*2*								*1*
14									*2*							*2*
15									*1*							*3*

Используя данные из таблицы 1, вычислим модифицированные веса вершин (*табл*.2) и построим диаграмму ранних сроков окончания выполнения операторов с учетом обмена информации (рисунок 3).

Таблица 2.

Модифицированные веса вершин.

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
T	*3*	*1*	*9*	*19*	*20*	*17*	*23*	*22*	*31*	*25*	*25*	*25*	*25*	*35*	*35*

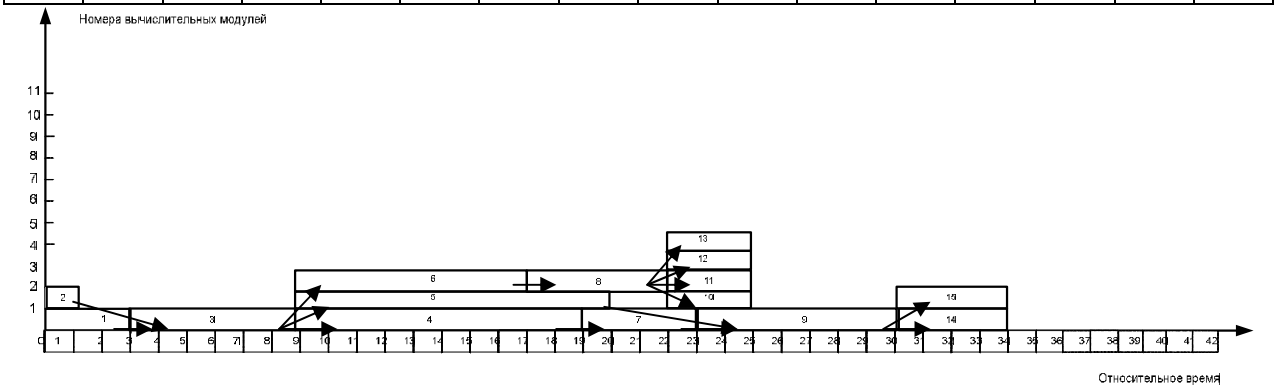


Рисунок 3. Диаграмма ранних сроков окончания выполнения операторов с учетом обмена информации

Каждая строка диаграммы может служить нитью для загрузки процессоров. Таким образом, получается 5 нитей.

T1={1,3,4,7,9,14},T2={2,5,10,15},T3={6,8,11},T4={12},T5={13}

Теперь необходимо преобразовать диаграмму ранних сроков окончания выполнения операторов с учетов времени передачи между узлами. Для этого применим «Алгоритм распределения вычислительных модулей по узлам ВС»:

1. Задана ВС с N вычислительных модулей, нумеруемых как {0,1,…,N-1}. Предполагается, что мощность множества удовлетворяет потребность в количестве ВМ для решения поставленной задачи предполагаемым методом.

2. Количество нитей W. Множество нитей T.

3. Вычислим матрицу расстояний между вычислительными модулями.

$$R = r(i,j)$$

Минимальное расстояние между двумя ВМ равно 1, максимальное – N -2.

4. Для определения показателя близости ВМ определим сумму столбцов матрицы R.

$$St(j) = \sum_{i=1}^{N} r(i,j),$$

j=1,2,…,N.

При j=1 показатель близости наилучший.

5. Упорядочим St(j) в порядке возрастания

6. Построим диаграмму ранних сроков окончания выполнения операторов с указанием связей между операторами, с учетов времени передачи между операторами. Образуем множества несвязных между собой пучков нитей $\{P_s\}$; S={0,1,…,q}.

7. Среди множеств $\{P_s\}$ найдем множество $\{P_z\}$, имеющее нить T_i с максимальным количеством элементов в множестве TS_k (таблице связей к-й нити). Предположим таблица связей имеет S_K элементов.

Примечание: таких множеств нитей может быть несколько, и тогда выбирается любое из них.

8. Составим из связей между нитями множества $\{P_z\}$ массив MS и обнулим все его элементы.

9. Если степень *i*-й вершины вычислительной сети есть V_i, то сравниваем S_K и V_i.

10. Если $S_K \le V_i$, то нить размещается в узле *i,* и переходим к шагу 12, иначе следующий шаг.

11. Если $S_K \le k(V_i - 1),$ то образуется комплексный узел, в котором один вычислитель основной, остальные являются передающими звеньями.

12. Определим с какой нитью из множества {Pz} связана нить T_i . Пусть это будет нить T_{jm}=(max $TS_k \cap$T*j*), T*j*∈T.

13. Нить T_{jm} занимает узел j_m вычислительной сети на минимально возможном расстоянии от узла *i*.

14. Образуем последовательность входящих и исходящих связей *i*-ой нити с T_{jm}, S_1,S_2,…,S_d (1) из множества MS.

15. Пусть связь S_m ,где m∈{1,2,..d}, в нити T_i связывает оператор γ с оператором α нити T_{jm}.

Тогда, если связь входящая, то если $sT(\gamma) \ge fT(\alpha) + r(i,j_m)*\rho A$, *то переходим на шаг* 17, *иначе* $P\gamma = P\gamma + r(i,j_m)*\rho A$.

Если связь исходящая, то если $S_m = 0$, *то* $P\alpha = P\alpha + r(i,j_m)*\rho A$, *иначе если* $sT(\gamma) < fT(\alpha)$, *то* $sT(\gamma) = fT(\alpha)$. Если S_m – входящая связь, то все операторы в нити T_i, начиная с оператора γ, сдвинуты по оси времени вправо на величину r(*i,j*)* ρA. Иначе, аналогично в нити T_{jm} сдвинуты все операторы, начиная с α .

16. Связь S_m=1 в массиве связей всех нитей MS.

17. Из последовательности (1) берем следующую связь для рассмотрения, пусть m=m+1 и обозначим эту связь S_m. Если m≤d, то перейти к шагу 15, иначе шаг 19.

18. $TS_k = TS_k|T_{jm}$, если $TS_k \neq 0$, то переходим к шагу 12, иначе шаг 20.

19. Уменьшим количество нерассмотренных нитей на 1, то есть W=W-1.

Если W=0, то переходим к п.17, иначе выбираем из оставшихся нитей множества {P_z} нить с максимальным числом связей. Пусть эта нить T_i и переходим к шагу 12.

20. Исключить из рассмотрения множества $\{P_z\}$ и переномеровать множество $\{P_s\}$; S={0,1,…,q-1}. Если $\{P_s\} \neq 0$, то перейти к шагу 7, иначе шаг 22.

21. Конец алгоритма.

Используемые обозначения:

Вершина – оператор ИЛГ заданной задачи.

Вес вершины (P) – время расчета вершины на *i*-ом процессоре.

Степень вершины (V_i) – количество узлов, находящиеся от *i*-ой вершины на минимальном расстоянии.

Коэффициент передачи (pA) – время передачи между узлами ВС.

Комплексный узел – транзитный узел ВС.

Время старта вершины (sT) – время старта расчета вершины в существующем разбиении вершин между процессорами.

Время финиша вершины (fT) – время финиша расчета вершины в существующем разбиении вершин между процессорами.

Нить – набор из одной или нескольких вершин, которые последовательно рассчитываются на одном процессоре.

Множество нитей (T) – совокупность всех нитей заданного ИЛГ.

Пучок нитей ($\{P_z\}$) – множество связанных между собой нитей.

Таблица связей к-ой нити (TS_k) – совокупность нитей, связанных с k-ой нитью (имеет S_k элементов).

Массив связей (MS) – упорядоченное множество связей всех нитей одного пучка. Согласно алгоритму построим для него матрицу дистанций (табл.3), которую в дальнейшем будет удобно использовать для размещения нитей решаемой задачи.

Для определения близости ВМ определим сумму столбцов матрицы и упорядочим их в порядке возрастания. Находим нить, содержащую максимальное количество связей с соседними (T1 – 5 связей; T2 – 5 связей; T3 – 4 связи; T4,T5 – 1 связи) - это нить T1. В матрице дистанций определяем строки с минимальными суммами. В данном случае – это 9-я и 11-я строки. Выбираем 9-ю строку. Расположим первую нить в 9-м ВМ. Так как на ближайшем расстоянии к 9-й строке находятся 3-й, 7-й, 10-й, 11-й и 15-й столбцы, нет необходимости организовывать комплексный узел.

Таблица 3.

Матрица дистанций

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	1	1	2	2	3	1	2	2	3	3	4	2	3	3	4	4	5
2	1	0	2	3	1	2	2	1	3	4	2	3	3	2	4	5	3	4
3	1	2	0	1	1	2	2	3	1	2	2	3	3	4	2	3	3	4
4	2	3	1	0	2	1	3	4	2	1	3	2	4	5	3	2	4	3
5	2	1	1	2	0	1	3	2	2	3	1	2	4	3	3	4	2	3
6	3	2	2	1	1	0	4	3	3	2	2	1	5	4	4	3	3	2
7	1	2	2	3	3	4	0	1	1	2	2	3	1	2	2	3	3	4
8	2	1	3	4	2	3	1	0	2	3	1	2	2	1	3	4	2	3
9	2	3	1	2	2	3	1	2	0	1	1	2	2	3	1	2	2	3
10	3	4	2	1	3	2	2	3	1	0	2	1	3	4	2	1	3	2
11	3	2	2	3	1	2	2	1	1	2	0	1	3	2	2	3	1	2
12	4	3	3	2	2	1	3	2	2	1	1	0	4	3	3	2	2	1
13	2	3	3	4	4	5	1	2	2	3	3	4	0	1	1	2	2	3
14	3	2	4	5	3	4	2	1	3	4	2	3	1	0	2	3	1	2
15	3	4	2	3	3	4	2	3	1	2	2	3	1	2	0	1	1	2
16	4	5	3	2	4	3	3	4	2	1	3	2	2	3	1	0	2	1
17	4	3	3	4	2	3	3	2	2	3	1	2	2	1	1	2	0	1
18	5	4	4	3	3	2	4	3	3	2	2	1	3	2	2	1	1	0
	4	4	3	4	3	4	3	3	3	3	3	3	4	4	3	4	3	4
	5	5	9	5	9	5	9	9	3	9	3	9	5	5	9	5	9	5

Теперь необходимо преобразовать диаграмму ранних сроков окончания выполнения операторов с учетов времени передачи между узлами. В результате применения алгоритма для поставленной задачи получим диаграмму, изображенную на рисунке 4.

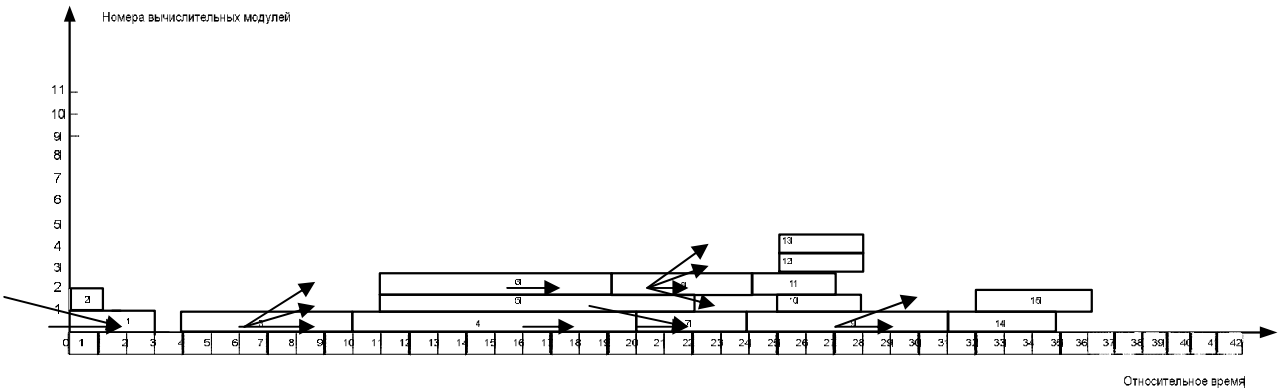


Рисунок 4. Диаграмма ранних сроков окончания выполнения операторов с учетом времен передачи между процессорами

На диаграмме видно, что для решения поставленной задачи за минимальное время необходимо 5 процессоров.

Таким образом, «Алгоритм распределения ВМ по узлам ВС» позволяет оптимизировать время и число процессоров для решения поставленных задач. Данный алгоритм применим для различных топологий ВС, не зависимо от количества операторов, то есть он является универсальным.

Недостатком данного алгоритма можно считать его высокую вычислительную сложность, так как фактически при получении распределения анализируются все возможные варианты размещения вершины на все процессорах. При обработке широко распространены рекурсивные алгоритмы, которые имеют экспоненциальную вычислительную сложность и потому плохо масштабируются.

Литература:

1. *Руденко Ю.М., Волкова Е.А.* Вычислительные системы. Архитектура и составляющие ее компоненты. Представление параллельных алгоритмов и оптимизация времени их решения. Изд. МГТУ им. Баумана, Москва, 2010г. 211 с.: ил.

УДК 004.05

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПЛАТФОРМ DOTPROJECT И MOODLE ДЛЯ РЕАЛИЗАЦИИ СИСТЕМЫ МОНИТОРИНГА ВЫПОЛНЕНИЯ ДИПЛОМНОГО ПРОЕКТА

А.А. Костырко, Е.В. Смирнова

1. Введение

Электронный документооборот является одним из самых явных показателей качественной работы организации, будь то бизнес-предприятие или образовательное учреждение. Хранение и обработка документации в электронном виде повышает надежность работы в целом, обеспечивает удобный мониторинг выполнения задач, позволяет быстро передавать информацию между сотрудниками. В настоящее время большое влияние

уделяется электронному документообороту в бизнес – структурах [1, 2]. На эту темы написано множество программных продуктов, самые заметные из них: DotProject и Web2Project [3-5]. На мой взгляд, эта тема не менее востребована в образовательной отрасли, хотя внимание к «оцифровке» образовательного процесса довольно ограничено [6,7].

Пожалуй, дипломная работа – одна из самых больших по объему документации работ за все время обучения в ВУЗе. Каждую главу документации необходимо утверждать у руководителя, а руководителю в свою очередь необходимо следить за графиком выполнения проекта. Вывод: в образовательной отрасли существует стабильный спрос на надежную систему, позволяющую удобно загружать результаты этапов работы по проекту и оценивать их, а также следить за графиком выполнения работы каждым студентом.

«С нуля» писать такой продукт бессмысленно, ибо уже существует набор платформ, требующий минимальных изменений для удовлетворения приведенным выше требованиям. Все такие средства разделяются на два крупных класса: образовательные и бизнес - продукты. Первые изначально разработаны для использования в сфере образования, что позволяет сохранить логику образовательного процесса и дает удобный механизм оценки. Бизнес – средства полезны, если мы представляем диплом как бизнес – проект, с руководителем и исполнителем. Преимущества использования таких продуктов, прежде всего - в удобном механизме отслеживания сроков.

Несложно выделить конкретные продукты, каждый из которых – лидер в своей сфере. В сфере приложений для ведения бизнес – проектов вне конкуренции система DotProject [3], дающая быстрый и надежный механизм для мониторинга работы по проекту. В области систем образовательного мониторинга безусловное лидерство за системой Moodle [4], позволяющей создавать образовательные курсы, наполнять их заданиями и лекционным материалом и следить за прохождением курса. В данной статье будет детально разобран процесс доработки каждой из этих систем для удовлетворения требованиям системы мониторинга выполнения дипломного проекта, со сравнительным анализом и выбором наилучшего варианта.

2. Система мониторинга на основе DotProject

Как уже было сказано выше, основная сфера применения данного продукта – бизнес. Однако все его функции отвечают нашим требованиям. Использование состоит из следующих этапов.

На первом этапе создается проект (один или несколько), и назначаются исполнители и руководители. Затем исполнитель проекта выполняет соответствующую работу, и докладывает о ней в журнале (Log), с указанием выполненной доли всего проекта (в процентах). Также исполнитель загружает файлы в систему. С журналом (Log) и загруженными файлами может ознакомиться руководитель проекта. Система также имеет сервис личных сообщений, обеспечивающий удобную оперативную связь.

Данная платформа имеет еще ряд и других существенных преимуществ. Прежде всего, необходимо отметить возможность сохранения логики диплома именно как проекта с набором контрольных точек. Наряду с этим, DotProject имеет отлично проработанную структуру заданий и подзаданий в проекте. Эта система позволяет включать в проект задания с фиксированными датами начала и конца, задания, где жестко задана только одна из этих дат, а также задания, где время начала определяется временем полного завершения другого задания. Последний тип заданий позволяет реализовать последовательность выполнения частей дипломного проекта. Также, DotProject дает возможность для разбиения каждого этапа на почти неограниченное количество более мелких этапов, позволяя наиболее полно описать проект. Еще одним плюсом является наглядная демонстрация исполнителю просроченных, текущих, выполненных и ожидающих выполнения заданий различным цветовым обозначением. Студент постоянно видит выполненную им долю всего проекта в процентах, а руководитель – график производительности каждого студента.

Однако нельзя не упомянуть ряд принципиальных недостатков этой платформы. Главный недостаток исходит из основного назначения платформы - обеспечение удобной кооперативной работы по одному проекту. Соответственно, если 20-ти студентам необходимо выполнить дипломные работы, то мы не сможем создать один проект и назначить 20 исполнителей – система будет считать их командой с общим заданием, и если каждый из них отметит выполнение 5-ти процентов общего плана – это приведет к 100%-му выполнению всего проекта. Таким образом, в нашем

случае необходимо будет создавать 20 проектов с одним исполнителем и одним руководителем, одинаковой структурой и формулировкой заданий и подзаданий, и отличающихся только названием. В системе имеется удобный механизм для этого: механизм создания «проекта по шаблону», но неудобство в виде 20-ти однотипных проектов все-таки остается.

Вторым главным недостатком DotProject является полное отсутствие механизма одобрения или неодобрения исполнителем содержания загруженного файла или участка журнала (лога – Log). Основная привилегия руководителя при работе в системе DotProject только одна – возможность редактировать и удалять данные, загруженные исполнителями. Разумеется, реализовать неодобрение несложно: руководителю достаточно послать личное сообщение исполнителю. Но, имея 20 исполнителей, рассылать им всем личные сообщения крайне неудобно. Чтобы исправить это и лучше приспособить систему DotProject к задаче, автором статьи выполнено редактирование одного из файлов платформы. Суть изменений состояла в следующем: исполнитель не мог поставить выполнение задания любого уровня на 100%, доступный ему максимум – 95%. 100%-е выполнение задания может поставить только руководитель, и это означает одобрение. Реализовывается это одной дополнительной строчкой в условном операторе, который проверяет правильность заполнения формы отчета о выполненной доле задания. Такое изменение облегчает механизм одобрения, но все равно остается некоторое логическое несоответствие.

Вследствие этого автор пришел к следующему выводу: система DotProject является мощной системой с рядом серьезных преимуществ, но также и серьезными недостатками. Технические усовершенствования позволяют лишь частично устранить недостатки, но логическое несоответствие все равно остается. Использование данной системы для решения поставленной задачи возможно, но требуются значительные доработки для восстановления логического соответствия.

3. Система мониторинга на основе Moodle

Система мониторинга на основе Moodle имеет ряд существенных отличий от той, что описана выше. Прежде всего – она создана исключительно для усовершенствования учебного процесса и переноса его в Интернет. Изменены соответствующие названия: вместо проектов в системе – учебные курсы, руководство курсами осуществляют преподаватели,

исполнители – студенты. Учебные курсы состоят из разделов, каждый раздел можно наполнять заданиями и лекционным материалом. Студенты выполняют задания, присылая файлы на сервер, откуда преподаватель их скачивает и либо выставляет оценки, либо отправляет на доработку. Файл, отправленный после срока завершения задания, отмечается как «просроченное выполнение».

Очевидным и неоспоримым преимуществом является полное соответствие образовательному направлению поставленной задачи, и отличная проработка всех необходимых механизмов. Присутствуют разные шкалы оценки. Имеется удобный механизм отправления файлов на доработку. Также есть ряд дополнительных функций: сервис новостей, форум и т.д.

Главный недостаток системы Moodle только один: дипломный проект, который представляет собой, по сути, только набор заданий, в этой системе может быть представлен только как учебный курс, что не соответствует действительности. Еще одним минусом является полное отсутствие механизма создания «курса по шаблону». То есть, создав курс «Дипломный проект» один раз, спустя год придется заново прописывать всю структуру для создания нового курса.

Решением этой проблемы повторного создание структуры явилось создание автором статьи дополнительного плагина к системе Moodle, который добавляет специальный формат курса. Автор назвал этот формат MentalDiploma, так как структура проекта построена в соответствии с ментально-структурированным подходом [7]. Для создания показанной структуры в один из файлов системы Moodle был добавлен участок кода, выполняющий добавление необходимых записей в таблицу.

4. Сравнение и выводы

Из описаний двух систем видно, что Dotproject дает лучший инструментарий для составления структуры проекта, но при этом уступает по массе других параметров. В частности, в системе Moodle работа каждого студента по курсу воспринимается как полностью независимая, то есть можно создать один курс и назначить на него столько человек, сколько необходимо, тогда как система DotProject вынуждает придерживаться соотношения «1 человек – 1 проект». Система Moodle потребовала больших дополнительных работ, связанных с добавлением участков кода, однако в

результате автор получил удобную и приспособленную к поставленной задаче систему.

Литература:

1. Внедрение электронного документооборота в коммерческих структурах http://www.business-gazeta.ru/article/24896/21/

2. Электронный документооборот для среднего и малого бизнеса http://www.delo-press.ru/articles.php?n=4945

3. *Документация DotProject* http://docs.dotproject.net/index.php?title=Main_Page

4. *Документация Moodle* http://docs.moodle.org/22/en/Main_page

5. *Документация Web2Project* http://wiki.web2project.net/index.php?title=Main_Page

6. *И.А. Козлов, А.А. Шайхутдинов, И.Д. Маслов* Система оценки качества подготовки студентов на основе компетентностного подхода. / Сб. статей Молодежной научно-технической выставки «Политехника» - 2011. 21-24 ноября 2011 г. МГТУ им. Н.Э. Баумана

7. *Добряков А.А., Балдин А.В., Галямова Е.В.* Мейнфрейм как платформа для создания ментально-структурированной системы комплексной оценки качества подготовки специалистов инженерного профиля (бакалавров, магистров, специалистов) / Сб. докладов конференции День технологий IBM в РГУ нефти и газа имени И.М. Губкина, 2010 - с. 8-25

УДК 004.91

ВЫБОР СИСТЕМЫ ЭЛЕКТРОННОГО ДОКУМЕНТООБОРОТА ДЛЯ РЕШЕНИЯ ЗАДАЧ ИННОВАЦИОННОЙ ДЕЯТЕЛЬНОСТИ НА ПРЕДПРИЯТИИ

Ю.Ю. Митрушенков, А.В. Брешенков, Д.Ю. Гудзенко

В условиях современной стремительно растущей экономике, как России, так и все мира, выживание различных предприятий основывается на принятии тактических и стратегических задач. Зачастую такие задачи

базируются на документах, которые предоставляются руководящим лицам предприятий. Своевременность согласования и получения таких документов составляет основу управления предприятием, помогает принимать вовремя правильные стратегические решения. В современных условиях для повышения качества управления необходимо уделять достаточное внимание совершенствованию работы с документами, так как управленческое решение всегда базируется на информации, носителем которой является документ.

Организация работы с документами влияет на качество реализации бизнес процессов компании. Зачастую успех деятельности предприятия зависит от того насколько качественно настроены процессы работы с документами. Как показывают современные исследования, 85% рабочего времени сотрудников организаций тратится на подготовку, сопровождение, заполнение, копирование и передачу документов. По данным ISO (International Standards Organization), управление и работа с документами становится одним из главных факторов конкурентоспособности любого предприятия. Оно значит особую работу с документами: управление процессами создания, движения документа от одного участника к другому, создание версий документа и управление ими, контроль распространения документов среди сотрудников. Правильно настроенная модель документооборота предприятия улучшает внутреннюю деятельность предприятия, в частности снижается время на поиск требуемого документа. При этом с ростом компаний увеличиваются объемы документов, усложняются схемы согласования документов, увеличивается время ожидания руководителя (актуально, когда руководитель часто бывает в командировках) для подписания документа.

Вследствие вышеизложенного, возникает необходимость использования отработанных методов поиска, обработки и хранения информации (документов) с разработкой совершенно новых приемов, режимов и методик оценки, анализа и оптимизации как внутренних, так и внешних документационных потоков предприятия, используя современные компьютерные технологии.

Руководители компаний, наводя порядок, во внутренних процессах работы с документами находят свои организационные решения различных проблем документооборота, примерно соответствующие уровню задач каждой компании. Зачастую такие решения приводят к еще большему

усложнению процессов согласования документов. Помимо этого часто документы для общей работы размещают в общей папке на сервере компании, что приводит к проблемам отслеживания изменений по документу. Для пересылка документов используют электронную почту, которая несет функцию контроля исполнения по документу. Однако эти частичные меры работают только до определенного момента. Дальше, когда компания ставит перед собой все более сложные задачи и вдобавок растет в размерах, таких средств для хранения информации, обеспечения взаимодействия и контроля выполнения поручений начинает не хватать. Возникают две возможности: либо внедрить в компании классический бумажный документооборот, что выглядит уже как «каменный век», либо внедрить электронную систему. Обычно выбор делается однозначно в пользу второго пути.

В целом с этими задачами справляется целый класс информационных систем которые относятся к ECM (Enterprise Content Management) системами. Зачастую выбор такой системы занимает длительное время и приводит к дополнительным затратам различных ресурсов предприятия. В данной статье описана предполагаемая модель оценки систем электронного документооборота, для последующего выбора. Для начала рассмотрим основные требования, влияющие на решение о выборе системы:

- Система хранения. Если в компании в день обрабатывается порядка более 50-100 документов (что соответствует среднему значению по объему документооборота), необходимо чтобы система, поддерживала иерархическое структуру хранения. При этом обеспечивает быстрый доступ к требуемому документу.
- Возможность создание организационной структуры предприятия с распределением соответствующих прав. Актуально для холдингов и корпораций с большим количеством внутренних компаний, между которыми происходит взаимодействие.
- Возможность создания распределенной базы данных территориально распределенных организаций.
- Наличие возможности массового ввода документов. Актуально при наличии большого бумажного архива, который необходимо перевести в электронный вид

- Возможность создавать свои собственные маршруты согласования документов, не прибегая к помощи разработчиков.
- Возможность интеграции с другими информационными системами и поддержки текущего оборудования предприятия.
- Поддержка всех форматов документов, актуально для компаний которые используют документы определенных форматов (чертежи, проектную документацию, расчеты смет)
- Различные типы поиска информации, включая системы поиска документов по содержимому.
- Требования к безопасности (шифрование, организация доступа, и т. д.). Возможность использования электронной цифровой подписи.
- Возможность веб-доступа.

Как говорилось выше, важной характеристикой документооборота является его объем. Под объемом документооборота понимается количество документов, поступивших в организацию и формируемой ею в течение года. Объем документооборота – важный показатель, используемый в качестве критерия при выборе организационной формы делопроизводства, организации информационно-поисковой системы по документам учреждения, установлении структуры службы делопроизводства, ее штатного состава и другие.

Установление порядка движения документов, или управление документацией организации заключается в создании условий, обеспечивающих хранение необходимой документной информации, ее быстрый поиск и доведение ее до потребителей в установленные сроки и с наименьшими затратами. Таким образом, оно включает в себя организацию документооборота, включая технологию личной работы исполнителей, создание информационно-поисковых систем по документам организации, контроль их исполнения.

Выбор системы электронного документооборота сложный процесс, требующий детальной проработки возможностей системы перед поставленными задачами автоматизации процессов. Предусматривается учет установленных показателей эффективности системы электронного документооборота.

Исходя, из вышеизложенных требований, можно выделить более конкретные желаемые функции СЭД, которые можно включить в модель выбора системы СЭД.

Для реализации данной модели предполагается использовать не самый простой (в связи с необходимостью сравнения каждой системы, и, как следствие, ее приобретение), но довольно наглядный процесс:

Каждому параметру присваивается удельный вес от 0 до 1 согласно частоте и важности использования данного параметра, каждая СЭД оценивается по 5ти бальной системе в зависимости от реализации требуемого параметра. В конце подсчитывается итоговая оценка с учетов веса и принимается решение пользу системы набравшей большую оценку.

В качестве примера приведем расчет выбора системы электронного документооборота из 3х наиболее популярных систем. Далее представлены оценки систем электронного документооборота среди трех выбираемых по их функциональности:

Функция	**Удельный вес использования**	**Lotus**	**Directum**	**Naudoc**
Регистрация документа	0,2	4	5	5
Словари справочники	0,5	5	5	4
Задание маршрута	0,7	4	5	4
Проектирование маршрутов	0,4	4	4	4
Web-доступ	0,3	4	5	4
Защита от сбоев	0,3	3	5	5
Поддержка различных форматов документов	0,6	4	5	5
Взаимодействие с другими системами	0,6	5	5	5
Итоговая оценка		33	39	36
Итоговая оценка с учетом весов		15,2	17,6	16,1

В предполагаемую модель можно добавлять абсолютно любые параметры, которые необходимы в процессе деятельности предприятия. Для реализации такой модели на предприятии необходимо установить все системы и оценить каждую, тем более большинство компаний, осуществляющих внедрение таких систем, могут предоставить тесовую

версию системы. Плюсом является, что такие системы предоставляются бесплатно, по функционалу ничем не ограничены.

В заключение можно добавить, что использование такой модели может помочь выбрать систему электронного документооборота, благодаря которой можно улучшить многие показатели по бизнес процессам компании. И к таким показателям можно отнести:

- Контроль исполнительской дисциплины
- Единый стандарт работы с документацией, позволяющий стандартизовать процессы документооборота предприятия
- Разграниченный доступ к документации предприятия
- Контроль местонахождения документа
- Быстрый поиск требуемого документа или комплекта связанных документов

В итоге выбор остается за руководящими и принимающими решения лицами на предприятии.

Литература:

1. Реинженеринг корпорации: Манифест в бизнесе / *Майкл Хаммер, Джеймс Чамли;* пер. с англ. Ю. Е. Корнилович. М.: Манн, Иванов и Фербер, 2006.

2. Artificial Neural Networks in Real-Life Applications / *Rabunal J. R., Dorrado J.* (Eds.).

Hershey – London – Melbourne – Singapore: Idea Group Publishing, 2006.

3. *Саттон М.Дж.* Корпоративный документооборот. – М.: Азбука, 2002. – 448 с.

УДК 519.766

МОДЕЛИ ПРЕДСТАВЛЕНИЯ ТЕКСТОВ В СИСТЕМАХ ОБРАБОТКИ ДАННЫХ

В. И. Чернов, А. М. Андреев, Г. П. Можаров

Современные системы обработки данных, основываются на двоичной логике и оперируют байтами, представляющих из себя числа. Это значит, что любые данные представляются в виде чисел и текстовые данные − не исключение. Наиболее удобный для человека способ представления текстов − представление в виде набора символов определенного естественного языка. Для такого представления в вычислительных системах каждому символу языка ставиться в соответствие определенное число − код, формируя тем самым ту или иную кодировку (UTF-8, CP1251, KOI8-R и т. п.). Таким образом, в простейшем случае, любой текст представляется в виде последовательности байт или чисел определенной кодировки. Однако, в системах обработки естественно-языковых текстов, таких как поисковые системы или системы автоматической рубрикации, такой способ представления не является оптимальным, так-как обладает большой избыточностью. В таких системах применяют различные модели представления данных, призванные упростить обработку естественно-языковых текстов.

Существует три подхода к построению моделей текстов на естественном языке: эвристический, формальный и смешанный.

При эвристическом подходе, нивелируются особенности формирования семантики языка. Текст на естественном языке рассматривается как некое случайное распределение символов некоторого алфавита или цепочек из них (предложений, слов, слогов и т. п.) и представляется в виде вектора числовых характеристик. Различные способы формирования такого вектора определяют различные эвристические модели представления естественно-языковых текстов. Такой подход, к примеру, лежит в основе n-граммных моделей.

N-граммные модели текста основаны на разбиении исходного текста на подпоследовательности определенной длины − *n*-граммы. Элементами n-грамм могут выступать слова, словосочетания, фонемы или символы. Вектор

характеристик формируется на основе частот вхождения n-грамм в текст. В процессе разбиения или после него возможно применение процедур фильтрации или взвешивания по различным признакам (таким как частотность, вхождение в стоп-словарь и т. п.).

На n-граммных моделях текста основываются вероятностные модели для систем распознания слитной речи. В таких моделях вероятность получения очередной распознаваемой n-граммы, оценивается частотой встречаемости в некотором обучающем корпусе текстов.

В качестве примера рассмотрим триграммную модель представления естественно-языкового текста, нашедшую широкое применение в задачах классификации.

Пусть, документ D состоит из последовательностей символов a_i:

$D = (a_1, a_2, \dots , a_n)$.

Триграмма t_j формируются из этих символов следующим образом:

$t_j = (a_j, a_{j+1}, a_{j+2})$, где $j \in [1, n-2]$.

Каждая триграмма кодируется уникальным номером:

$k_j = M^2 \cdot r(a_j) + M \cdot r(a_{j+1}) + r(a_{j+2})$,

где M – количество символов в алфавите, $r(a_j)$ – номер символа в алфавите.

Документ D представляется в виде вектора частотного распределения триграмм V:

$V = (v_1, v_2, \dots , v_m)$,

где m – количество возможных триграмм в алфавите, v_k – частота вхождения k-ой триграммы в документ.

При формальном подходе на первый план выступает семантика языка и способы ее представления в вычислительных системах. В основе построения таких моделей лежат внутренние особенности языка − правила формирования слов, высказываний и дискурса. Такой подход используется при построении формальных семантических моделей языка, которые применяются для систем автоматического перевода [1], а так-же в системах синтаксического разбора. Такой подход лежал в основе таких моделей как

“Смысл-Текст” И. А. Мельчука [2] и “Математической модели языка” В. А. Тузова.

Рассмотрим подробнее “Математическую модель языка” предложенную В. А. Тузовым [3] [4], где высказывания естественного язык предлагается рассматривать как суперпозицию вычисляемых функций, выражающих смысл. Например, фраза *Сарай горит* может быть преобразован в вызов функции *Горит(Сарай)*. Однако, значение глагола *гореть* во фразе *Глаза горят* имеет иной смысл, а следовательно необходимо вводить еще одну функцию *Горит2(Глаза)*. Такие преобразования из текста в функции автором называется формализацией естественного языка. Таким образом, язык рассматривается как алгебраическая система $(f_1, f_2, \ldots, f_n, \boldsymbol{M})$, где $\boldsymbol{M}$ – множество, имеющая различные названия в разных теориях: глубинная структура, семантическая структура, множество значений (смыслов), структура данных и т. д., $(f_1, f_2, \ldots, f_n)$ – определённым образом образованная конечная система формализующих функций.

При смешанном подходе, учитываются внутренние особенности языка, однако они не являются основой для таких моделей, но выступают в качестве вспомогательной базы при использовании эвристического подхода. Учет языковых особенностей в некоторых задачах позволяют добиться более лучших результатов. Такой подход применяется в моделях документ-термин.

Модели «документ-термин» основаны на представлении текста в виде вектора весов терминов. Различные модели «документ-термин» отличаются методами получения набора терминов и алгоритмами вычисления весов, что обусловлено различными целями их применения.

Есть два источника набора терминов: предварительное ручное задание (применяется в задачах классификации) и автоматическое формирование на основе обрабатываемых текстов (применяется в задачах кластеризации).

Термин – нормализованное слово или словосочетание. Нормализация – процесс приведения в форму удобному для оперирования системой. В общем случае, нормализация может включать процедуры морфологического, синтаксического и семантического анализов, морфологической и лексической нормализации. Нормализация может минимизировать влияние

таких негативных для построения модели свойств естественного языка, как многозначность и синонимичность.

В процессе извлечения терминов из текста возможно применение процедуры фильтрации по различным критериям. В качестве критериев могут служить вхождение в словарь запрещенных терминов, наличие различных морфологических, лексических и др. признаков, выход за диапазонов допустимых частот и т. п.

Для примера рассмотрим матричную модель текстов, которая применяется для задач классификации и кластеризации.

Дано множество документов $\boldsymbol{D}$:

$\boldsymbol{D} = \{ D_1, D_2, \ldots , D_m \}$

и множество терминов этих документов:

$\boldsymbol{T} = \{ t_1, t_2, \ldots , t_n \}$.

Вес термина t_j в документе D_i можно определять по различным мерам, например:

$w_{ij} = t_{ij} \cdot log_{10} (m / n_j)$,

где t_{ij} – количество вхождений термина t_j в документ D_i, m – количество документов во множестве, n_j – количество документов, в которые входит термин t_j.

На основе весов терминов формируется матрица весов терминов $\boldsymbol{W}$:

	t_1	t_2	...	t_n
D_1	w_{11}	w_{12}	...	w_{1n}
D_2	w_{21}	w_{22}	...	w_{2n}
...	...	...	...	...
D_m	w_{m1}	w_{m2}	...	w_{mn}

Таким образом были предложены и рассмотрены принципы, которые лежат в основе построения моделей текстов на естественном языке, показаны принципы построения n-граммных моделей и рассмотрены особенности реализации моделей «документ-термин». Отмечены особенности формирования списка терминов при построения моделей «документ-термин». Приведены математические модели триграмной и матричной моделей представления текстов.

Литература:

1. *Шемакин Ю. И.* Начала компьютерной лингвистики: Учеб. пособие. М.: Иэд-во МГОУ, А/О "Росвузнаука", 1992. ISBN 5-7045-0132-X – 80 с.

2. *Мельчук И.А.* Русский язык в модели «Смысл-Текст» – Москва – Вена: Школа «Языки русской культуры», Венский славистический альманах, 1995. – 682 с.

3. *Тузов В.А.* Компьютерная семантика русского языка. – СПб.: Изд-во СПбГУ, 2003. – 393 с.

4. *Тузов В.А.* Математическая модель языка. – Л.: Изд-во Ленинград, ун-та, 1984. – 176 с.

УДК 681.3.07

РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ ОРГАНИЗАЦИИ ЦЕНТРА СЕРТИФИКАЦИИ

Г.П. Гаджиев, Т.Н. Ничушкина

Асимметричная криптография в настоящее время обретает все большую важность в нашей повседневной работе. Основная причина этого заключается в том, что применение асимметричных алгоритмов является самым простым способом обеспечить приемлемый уровень защиты информации от прослушивания при передаче по открытым каналам. Сфера приложений, которые нуждаются в подобной защите, растет каждый день: текстовые и табличные процессоры, системы документооборота, банковские приложения, ПО для видео и аудио-связи – все больше привычных нам приложений переносятся в сеть.

На данный момент, наиболее популярным алгоритмом асимметричного шифрования является RSA, который, несмотря на существенные недостатки (низкая скорость работы и ненадежность при использовании ключа размером меньше 1024 бит), активно применяется для защиты программного обеспечения и цифровой подписи.

Из-за низкой скорости шифрования RSA (около 30 кбит/с при 512 битном ключе на процессоре Intel Pentium 4 2 ГГц), сообщения шифруются симметричными алгоритмами со случайным (сеансовым) ключом, а с помощью RSA шифруется лишь этот ключ (таким образом, реализуется гибридная криптосистема).

Для решения проблемы обмена открытыми (публичными) ключами были разработаны так называемые цифровые сертификаты, устанавливающие соответствие открытого ключа конкретному объекту (домену, компании, физическому лицу и т.д.). В операционные системы и браузеры изначально встроен набор корневых (самоподписанных) сертификатов доверенных организаций, которыми подписываются сертификаты более низкого уровня – таким образом, выстраивается цепочка доверия из сертификатов. Так, например, при посещении защищенного веб-сайта, браузер запрашивает с сервера сертификат и проверяет всю цепочку, пока не дойдет до корневого (публичный ключ которого установлен в браузер и является доверенным). Лишь после этого браузер начинает сеанс защищенного соединения, уже с использованием полученного с сервера открытого ключа.

Исходя из актуальности описанной проблемы, цель представляемой разработки состояла в создании приложения, предназначенного для организации упрощенного центра сертификации и предоставляющего удаленный программный интерфейс к его функциям. Таким образом, разрабатываемое ПО не предназначено для организации полноценного удостоверяющего центра, однако реализовывает наиболее часто используемые в приложениях функции, такие как создание ключей, корневых и конечных сертификатов, проверка сертификатов и их отзыв.

Реализация подобного приложения возможна в виде локальной библиотеки. Такое решение обеспечило бы большую скорость работы, однако не обладало бы при этом должным уровнем ортогональности и остальными преимуществами сервисной архитектуры. Поэтому, было решено реализовать приложение в виде веб-сервиса, так как такое ПО просто интегрировать в сложные системы (благодаря использованию стандартных протоколов, библиотеки для работы с которыми есть во всех популярных языках).

О сервисной архитектуре

Организация сервисной архитектуры веб-приложений уходит корнями в идеологию UNIX-систем. Главные ее принципы сохраняют тут свою силу:

Создание программ, которые делают что-то одно и делают это хорошо.

Создание программ, которые работают вместе.

Создание программ, которые поддерживают текстовые (универсальные) интерфейсы.

Ярким примером сервис-ориентированной архитектуры может служить инфраструктура компании Amazon. Их базы данных разбиты на небольшие части и для каждой из них построен отдельный интерфейс, выполненный в виде сервиса, который является единственным способом получить доступ к данным. Приложение, генерирующее страницы для Amazon.com, является агрегатором, собирающим данные с десятков сервисов и отправляющим их конечным пользователям.

Такая архитектура дает необходимый уровень изоляции для построения множества независимых программных компонентов. Подобно тому, как опытные unix-пользователи используют комбинации нескольких простых утилит для выполнения своих задач, инженеры Amazon используют комбинации простых сервисов для организации новых программных продуктов (которые, в свою очередь, могут быть как сервисами, так и агрегаторами).

Таким образом, оформление рассматриваемого в данной работе приложения для работы с ключами и сертификатами в виде сервиса (т.е. определение для него программного интерфейса и ограничение доступа к данным только через него) вполне обосновано. Такое исполнение, помимо простой интеграции в программные комплексы, позволит, в случае необходимости, выполнить перенос кодовой базы на другой язык или технологию, не затрагивая вышестоящие сервисы и агрегаторы.

Выбор технологий и процесс разработки

Для разработки сервиса была выбрана связка из языка программирования Ruby и программного каркаса Ruby on Rails.

Выразительный синтаксис Ruby сочетается с его способностями по созданию DSL (предметно-ориентированных языков программирования) и объектной ориентированностью (в частности, существует возможность изменения базовых типов в процессе выполнения программы, чем

пользуется Rails, добавляя полезные в веб-разработке вспомогательные методы к стандартным типам данных). В свою очередь, Ruby on Rails предоставляет приложению шаблон проектирования MVC (model-view-controller, модель-представление-контроллер, разделение модели данных приложения, пользовательского интерфейса и взаимодействия с пользователем на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные). Кроме того, Rails предоставляет объектный интерфейс к языку запросов SQL, позволяющий абстрагироваться от конкретной базы данных использованием объектов-моделей, сохраняемых затем в базе.

Выполнение операций с ключами и сертификатами производится с помощью открытого криптографического пакета OpenSSL, надежность которого подтверждается использованием в большинстве веб-серверов, а также во многих бизнес приложениях и ПО для государственных учреждений.

Доступ к сервису и работа с ним определяется архитектурным стилем REST (Representational State Transfer, передача состояния представления). Термин был введен и определен в 2000 году Роем Филдингом, одним из основных авторов спецификации протокола HTTP версий 1.0 и 1.1. Основным понятием REST является ресурс — источник конкретной информации. Каждый ресурс определяется ссылкой с глобальным идентификатором (в случае HTTP это URI). Компоненты сети обмениваются представлениями этих ресурсов через стандартизованный интерфейс, таким образом, взаимодействие с ресурсами возможно указанием двух параметров: идентификатор ресурса и требуемое действие.

Пример ресурса (ключ) и методов работы с ним:

Ресурс	GET	PUT	POST	DELETE
URI коллекции, пример: http://example.com/keys/	Извлечение URI всех элементов		Создание нового элемента, операция также возвращает URI созданного элемента	

Ресурс	GET	PUT	POST	DELETE
URI элемента, пример: http://example.com/keys/42	Извлечение представления адресованного элемента (вывод его содержимого)	Обновление адресованного элемента коллекции (смена пароля ключа)		Удаление адресованного элемента коллекции.

Одной из целей разработки было сохранение простоты сервиса. В частности в нем не реализуется аутентификация и авторизация пользователей (и вообще ведение пользовательской базы и разграничение доступа), эта задача должна быть возложена на веб-сервер, либо на другой сервис. Несмотря на это, закрытые ключи надежно защищены шифром AES-256, который не даст возможности воспользоваться ключами без знания их пароля в случае компрометации базы данных.

В процессе реализации были применены принципы TDD (Test Driven Development, разработка через тестирование), в соответствии с которыми выполняется разбиение процесса разработки на очень короткие циклы. Сначала пишется тест, покрывающий желаемое изменение и только затем код, позволяющий пройти этот тест. После этого кодовая база компонента может изменяться без боязни нарушить функционирование всего приложения: система тестирования запускает набор тестов после каждого сохранения файла и сигнализирует в случае нарушения его выполнения. По ходу разработки приложение также запускалось в среде Heroku Cedar Stack, максимально приближенной к стандартным условиям работы Rails-приложений.

Для сохранения изменений кодовой базы использовалась распределенная система управления версиями Git, данные в которой сохранялись после каждой итерации разработки, что дает возможность сравнивать файлы различных версий и вести параллельно несколько веток разработки (например, стабильную и новую версии) и синхронизировать их при необходимости.

Результаты разработки и перспективы развития

Разработанное приложение способно выполнять возложенные на него функции и может быть использовано для создания ключей и цепочки доверенных сертификатов небольшими организациями и командами

разработчиков. В планах по дальнейшей разработке — внедрение очереди задач (для выполнения ресурсоемких запросов) и создание вспомогательного пользовательского ПО на Backbone.js (программный каркас на JavaScript для разработки одностраничных клиентских веб-приложений).

Литература:

1. *Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swami Sivasubramanian, Peter Vosshall, Werner Vogels*, “Dynamo: Amazon's Highly Available Key-Value Store”. WA, 2007.

2. *Andrew Hunt and David Thomas*. The Pragmatic Programmer. Addison-Wesley, 1999. ISBN: 020161622X.

3. *Dave Thomas, Chad Fowler, Andy Hunt*. “Programming Ruby 1.9”. The Pragmatic Bookshelf, 2011. ISBN: 1-934356-08-1.

5. *Sam Ruby, Dave Thomas, David Heinemeier Hansson*. “Agile Web Development with Rails”. The Pragmatic Bookshelf, 2011. ISBN: 978-1-934356-54-8.

6. “Unix philosophy”. Wikipedia EN, http://en.wikipedia.org.

7. Amazon Web Services Blog, http://aws.typepad.com.

УДК 004.91

АННОТИРОВАНИЕ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ КЛАСТЕРИЗАЦИИ

К.М. Гречищев, Р.С. Самарев

Введение

Большие объемы информации сегодня хранятся в формате электронных документов. При этом возникает необходимость анализировать информацию по какой-либо тематике. В этой ситуации зачастую бывает недостаточно поиска по содержимому, поскольку сложно указать точную поисковую фразу, встречающуюся в имеющихся документах документе.

В таких случаях полезно предоставить пользователю краткое содержание (аннотацию) того или иного документа или их группы, по которому пользователь может понять его приблизительное содержимое. Эти краткие описания можно использовать и для поиска.

Аннотация должна отражать тематику документа и содержать наиболее значимые его положения. Задачу автоматизированного построения аннотаций можно решать различными методами. В данной работе рассматривается способ построения аннотации для документа или групп документов, полученных на основе предварительной кластеризации по критерию близости содержимого.

Основной целью данной работы был поиск способа получения аннотации для групп документов с неизвестной тематикой. Такие группы могут например формировать источники новостных сообщений. Ключевой особенностью таких групп является наличие подгрупп, содержащих документы с общей темой, причём они могут содержать как описание различных аспектов общей темы (новости о политике, о языках программирования, о компьютерном оборудовании), так и представлять описание одного и того же события (проведена встреча с представителями какой-то делегации, выпущен новый процессор и пр.). Построение аннотации по группе документов, имеющих различные темы неизбежно приведёт к ошибкам определения наиболее значимых текстов. Поэтому был предложен способ построения аннотаций после этапа кластеров схожих документов с тем, чтобы проводить аннотирование уже в рамках однородных по содержанию групп.

На рисунке 1 представлен процесс проведения аннотирования документов с использованием методов кластеризации.

Исходные документы подвергаются преобразованию в векторную модель представления, после чего над ними возможно проведение кластеризации. Для каждого из полученных кластеров проводится процедура выделения наиболее значимых предложений, из которых и формируется аннотация.

Таким образом, для проведения аннотирования над документами неизвестной тематики необходимо:

- определить модель векторного представления документа для выполнения кластеризации;

- определить алгоритмы кластеризации и метрики определения схожести документов;
- определить метрику для вычисления наиболее значимых предложений;
- определить критерии оценки качества проведения аннотирования.

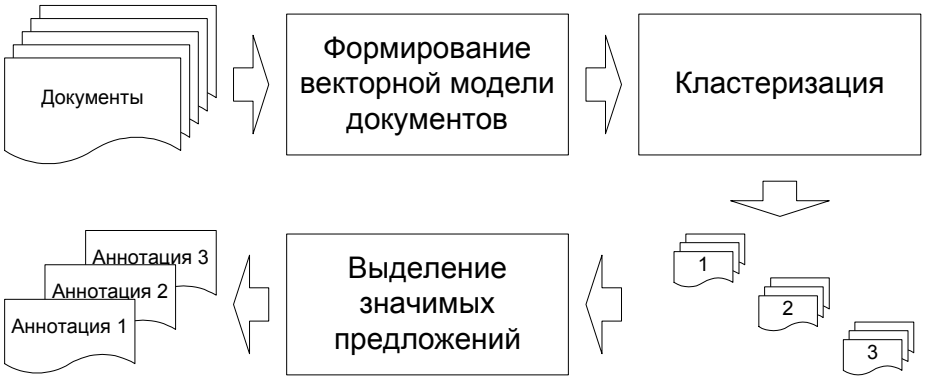


Рисунок 1. Процесс построения аннотаций с использованием кластеризации

Решение указанных проблем и будет рассмотрено в этой статье.

Модели представления документов

Первым этапом обработки документов является кластеризация, которая является задачей разбиения заданной выборки объектов на подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Для выполнения кластеризации необходимо получить формальное описание предложения. Наиболее распространенной моделью представления текста является модель термин-документ. Другими словами, необходимо каждое предложение представить в виде вектора $A_i = (\alpha_{i,1} \quad \alpha_{i,2} \quad \dots \quad \alpha_{i,n})$, где n - число термов во всех предложениях. Данная модель применяется как на этапе кластеризации предложений, так и на этапе аннотирования.

Существуют различные разновидности модели термин-документ[1]. Основными являются модель TF и TF*IDF, которые использовались в данной работе.

В случае модели TF коэффициент $\alpha_{i,j}$ равен

$$\alpha_{i,j} = \frac{t_{i,j}}{\sum_{k=0}^{n} t_{i,k}} \quad (1).$$

Выражение $\sum_{k=0}^{n} t_{i,k}$ – есть общее число термов в данном предложении. Таким образом, можно утверждать, что $\alpha_{i,j}$ представляет собой частоту встречаемости терма j в i-ом предложении.

Модель TF*IDF предполагает наличие в формуле для $\alpha_{i,j}$ дополнительного логарифмического множителя, понижающего все тех термов, которые встречаются в большом числе документов:

$$\alpha_{i,j} = TF \cdot IDF = \frac{t_{i,j}}{\sum_{k=0}^{n} t_{i,k}} \cdot \log\left(\frac{m}{n_j}\right) \quad (2),$$

где: m – общее число документов предложений, а n_j – число предложений, содержащих j-ый терм.

Для построения модели представления необходимо сформировать множество термов $T = \{\tau_1 \quad \tau_2 \quad \ldots \quad \tau_n\}$. При этом возникает потребность в использовании средства морфологического анализа для приведения слов к начальной форме. В качестве такого средства может быть использован продукт mystem от компании Yandex, который проводит разбиение документа на слова, определяет их начальную форму, а также имеет возможность вывода грамматической информации о терме. Однако исходный код mystem закрыт, а для некоммерческого использования предоставляется только консольная версия приложения. В качестве альтернативы было выбрано программной обеспечение от рабочей группы Aot.ru, распространяемое под лицензией LGPL.

В качестве меры сходства документов при кластеризации использовалась метрика на основе косинуса угла между векторами (Cosine similarity), которая по результатам эксперимента оказалась лучше, чем мера расстояния по Евклиду:

$$d(a,b) = 1 - \cos(\theta) = 1 - \frac{a \cdot b}{\|a\| \cdot \|b\|} = 1 - \frac{\sum_{i=1}^{m} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{m} (a_i)^2} \cdot \sqrt{\sum_{i=1}^{m} (b_i)^2}} \quad (3),$$

где: m – размерность векторов, а θ – угол между векторами.

Данная формула может быть существенно упрощена, если выполнить предварительную нормализацию векторов. В этом случае, угол между ними не измениться, а знаменатель дроби станет равным 1.

Алгоритмы кластеризации

В данной работе, основными алгоритмами, используемыми для кластеризации документов, были кластеризация методом К-средних (*k*-means clustering) и алгоритм “Canopy clustering”. В качестве инструментального средства использовалась библиотека Apache Mahout [2,3].

Алгоритм К-средних являлся основным алгоритмом кластеризации. Его недостатком является необходимость предварительно указать число кластеров и их начальные центры. Для решения этой проблемы предварительно проводилась кластеризация документов методом “Canopy clustering”, который требует задания двух пороговых параметров T1 и T2[4]. Из числа сформированных кластеров выбирались самые крупные кластеры, количество которых не превышало заданного. Эти кластеры и использовались в качестве инициализирующих.

Оценка применимости данных алгоритмов и качества получаемых результатов выполнена в работе [1].

Построение аннотации

Результатом кластеризации является разбиение предложений на кластеры. Основная идея автоматизированного построения аннотации на основе результатов кластеризации заключается в том, чтобы выбрать из каждого кластера фиксированное число предложений, сумма весов терминов которых имеет наибольший вес.

Вес предложения в простейшем случае может быть вычислен как сумма весов всех термов, которые содержаться в этом предложении. Однако при таком подходе в результирующей выборке будут преобладать не предложения, содержащие ключевые для документа термины, а наиболее длинные предложения, что было подтверждено экспериментально. Следовательно, для устранения этого недостатка необходимо разделить полученный вес на число термов в документе.

Альтернативные формы представления содержимого документов

В ряде случаев нет необходимости формировать полноценную аннотацию документа, ограничившись лишь получением списка ключевых

слов документа. Самым простым решением данной задачи является построение модели представления документа в виде вектора, не разбивая его на предложения. Затем из этого вектора выбираются компоненты с наибольшим весом. Соответствующие им термы можно считать ключевыми словами для этого документа. При таком подходе никак не учитывается структура документа.

Пример:

У мэра Москвы Юрия Лужкова был более мягкий вариант ухода со своего поста, сообщила журналистам пресс-секретарь президента РФ Наталья Тимакова.

"Вчера Лужков вернулся из отпуска. Сегодня Дмитрий Медведев подписал указ. Выводы можете делать сами", - отметила Тимакова.

Ранее 28 сентября Медведев подписал указ об отставке Юрия Лужкова с формулировкой" в связи с утратой доверия президента". 27 сентября, сразу после выхода из отпуска, Лужков заявлял, что подавать в отставку по собственному желанию не намерен.

Ключевые слова: ***Лужков***, ***мэр***, *вариант, Медведев, Москва, президент,* ***отставка***, *пресс-секретарь,* ***доверие***, *Юрий, глава.*

Более точные результаты дает метод, при котором выполняется разбиение документа на предложения и их кластеризация. Затем из каждого кластера выбирается фиксированное число предложений с наибольшим весом. Из каждого предложения выбирается несколько ключевых слов по тому же принципу.

Пример:

Ключевые слова: ***Лужков***, *вариант,* ***мэр***, ***отставка***, ***указ***, *пресс-секретарь, мягкий,* ***утрата***, *формулировка,* ***доверие***, *Медведев.*

Оценка качества полученных аннотаций

В качестве формальной метрики, применимой для оценки качества аннотаций и переводов, можно использовать одну из метрики семейства Rouge [5]. Для получения оценки необходимо сравнить автоматически сгенерированную аннотацию с одной или несколькими аннотациями по данной предметной области, сделанными экспертами.

Чаще всего применяют одну из метрик Rouge-N, где N - размер n-граммы, типовые значения которого от 1 до 3.

В этом случае, оценка вычисляется по формуле:

$$ROUGE - N = \frac{\sum_{M_i} count\left(Ngram(A) \cap Ngram(M_i)\right)}{\sum_{M_i} count\left(Ngram(M_i)\right)} \quad (4),$$

где: A – оцениваемая обзорная аннотация, M_i – i-ая ручные аннотация, Ngram(D) – множество всех n-грамм из лемм соответствующего документа D [7].

Полученные результаты

Для проверки приведенного выше способа построения аннотаций был поставлен следующий эксперимент. Из интернета было загружены новостные документы экономической тематики. Кластеризуемые документы были автоматически разбиты на 4 кластера. Из каждого кластера было выбрано предложение с наибольшим весом.

В результате генерировалась аннотация из 4 предложений. Экспертом была построена аннотация из 4 предложений, отражающих осиновые экономические события. Для оценки качества была использована методика, приведенная в [6]. Аннотация эксперта сравнивалась с 2 аннотируемыми документами, с искусственно построенными из заголовков и предложений аннотациями, а также с аннотациями, построенными системой. Результаты оценок приведены в таблице 1.

Таблица 1. Оценка качества аннотации

Эталонная аннотация	Rouge-1	Rouge-2	Rouge-3
Первый аннотируемый документ	0,204878	0,158585	0,063916
Последний аннотируемый документ	0,277311	0,214090	0,064588
Заголовки документов	0,684681	0,374516	0,167073
Первые предложения 4 документов	0,559220	0,351005	0,152273
Автоматическая аннотация	0,771160	0,467065	0,202663

Во время экспериментов была замечена особенность метрики Rouge-N, заключающаяся в том, что в случаях, когда эталонная аннотация существенно короче автоматически сгенерированной, оценка бывает завышена. Это связано с тем, что в знаменателе формулы (4) учитывается лишь число n-грамм в эталонном документе.

Заключение

Проведенные эксперименты показали практическую применимость использованного метода, однако качество аннотации, построенной на основе выделения ключевых предложений позволяет лишь ограниченно понять суть описываемого документа. Критерии оценки качества на основе сравнения с эталонной выборкой, подготовленной экспертами, существенно ограничивает автоматическую оценку полученных аннотаций.

Литература:

1. *Гречищев К.М., Самарев Р.С.* Некоторые современные модели и алгоритмы кластеризации текстов на естественном языке

2. *S. Owen, R. Anil и др*. Mahout in Action. - Shelter Island: Manning Publications Co., 2012, 387 с.

3. Интернет-ресурс: http://mahout.apache.org/

4. Интернет-ресурс: https://cwiki.apache.org/MAHOUT/canopy-clustering.html

5. Интернет-ресурс: http://berouge.com/default.aspx

6. *Лукашевич Н.В., Добров Б.В.* Автоматическое аннотирование новостных кластеров на основе тематического представления.–

[http://www.dialog-21.ru/digests/dialog2009/materials/html/46.htm]

УДК 05.13.00

ХРАНИЛИЩА ДАННЫХ И ИХ ПРОЕКТИРОВАНИЕ С ПОМОЩЬЮ CA ERWIN

Н.С. Зо, А.В. Брешенков, Д.Ю. Гудзенко

Корпоративные системы управления предприятием, созданные на основе реляционных СУБД, как правило, эффективно решают задачи учета, контроля и хранения данных. Однако в силу своей специфики, реляционная структура не позволяет решать задачи анализа имеющейся информации с требуемой производительностью. Особенно остро эта проблема стоит в гетерогенных информационных средах, когда в центральном офисе

организации и в филиалах эксплуатируются СУБД различных производителей (рисунок 1).

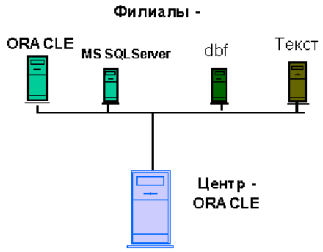


Рисунок 1. Гетерогенная информационная среда

Такая ситуация часто возникает либо в результате слияния компаний, когда компания превращается в филиал более крупной, но при этом нерентабельно перестраивать исторически сложившуюся информационную инфраструктуру, либо вследствие неудовлетворительного управления, когда филиалы не придерживаются корпоративного стандарта и внедряют собственные информационные системы. Одной из основных задач, решаемых в корпоративных информационных системах, является предоставление аналитической информации необходимой для принятия решений. Для поддержки принятия решения необходим не один заранее подготовленный отчет, а серия разнообразных отчетов, причем менеджер не всегда представляет, какой именно отчет понадобится ему в следующие полчаса. Например, при анализе продаж по компании оказывается, что в феврале текущего года произошел спад. Чтобы выяснить причины спада, необходимо просмотреть отчет о продажах в регионах. Отчет о продажах в регионах показывает, что спад произошел, видимо, по причине неудовлетворительной работы одного из филиалов, следовательно, необходим отчет о работе данного филиала и т.д. и т.п. Организовать выполнение таких отчетов в гетерогенной среде крайне сложно. Для эффективного анализа данных в этом случае необходимо объединять в одном запросе данные из разнородных источников. В настоящее время существуют мониторы транзакций и генераторы отчетов (например, Crystal Reports), обладающие такой функциональностью, однако производительность таких систем не может быть высокой. В процессе анализа данные, необходимые для принятия решений должны поступать к потребителю в режиме реального времени. Если же данные собираются из разных источников, то, во-первых, отчет готовится недопустимо медленно,

во-вторых, другие приложения, работающие с реляционными СУБД во время выполнения отчета скорее всего будут заметно замедляться.

Решением проблемы производительности является создание специализированной базы данных – хранилища данных (Data Warehouse), предназначенной исключительно для обработки и анализа информации (рисунок 2).

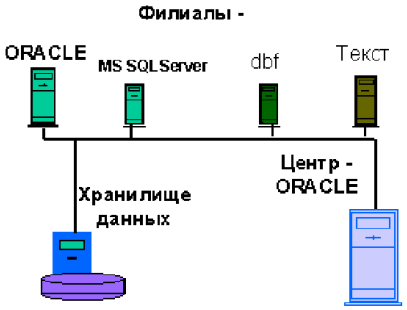


Рисунок 2. Пример гетерогенной информационной системы, включающей хранилище данных

Хранилища данных позволяют разгрузить оперативные базы данных, и тем самым, позволяют пользователям более эффективно и быстро извлекать необходимую информацию. Они могут быть включены в общую корпоративную сеть, по которой в хранилище по заранее определенному расписанию, как правило, в период наименьшей загрузки сети и серверов копируется накопленная за день или за неделю информация. Поскольку данные меняются редко, то к хранилищу данных не предъявляются жесткие требования, которые обычно предъявляются к обычным базам данных - отсутствие аномалий при выполнении операций обновления или удаления и избыточности хранения информации. По этой причине может сложиться неверное представление, что проектировать хранилище проще, чем базы данных, предназначенные для оперативной обработки информации. На самом деле, проектирование хранилища данных является весьма сложной задачей.

- Данные в хранилище должны регулярно пополняться. Требуется тщательно документировать правила пополнения и резервного копирования данных.
- Поскольку отчет будет создавать конечный пользователь, должны быть упрощены требования к запросам с целью исключения тех

запросов, которые могли бы требовать множественных утверждений SQL в традиционных реляционных СУБД.

- Обработка запросов к хранилищу должна быть проведена с высокой производительностью, желательно в реальном масштабе времени, поэтому должна быть обеспечена поддержка сложных запросов SQL, которые требуют последовательной обработки тысяч или миллионов записей.

Очевидно, что для решения этой задачи необходимо использовать специальные инструментальные средства. Одним из таких инструментов является Erwin ERX- CASE-средство фирмы Computer Associates International, Inc.

Erwin ERX является незаменимым инструментом для проектирования хранилищ данных по нескольким причинам:

1. Хотя реализовать хранилище данных можно на любом сервере БД, существуют специализированные сервера, специально предназначенные для поддержки хранилищ данных. Erwin поддерживает генерацию схемы БД для двух таких серверов – Teradata и Red Brick.

2. Как было указано выше, при проектировании хранилища необходимо создавать подробные спецификации для всех источников данных, в том числе самых разных типов. Erwin поддерживает на физическом уровне прямое и обратное проектирование объектов более чем для 21 типа БД, поэтому является идеальным CASE-средством для работы с гетерогенными информационными системами.

3. Для эффективного проектирования хранилищ данных ERwin использует размерную (Dimensional) модель. Dimensional - методология проектирования, специально предназначенная для разработки хранилищ данных.

Рассмотрим основные особенности техники моделирования хранилищ данных с помощью Erwin.

Поддержка методологии Dimensional

Нормализация данных в реляционных СУБД приводит к созданию множества связанных между собой таблиц. В результате, выполнение сложных запросов неизбежно приводит к объединению многих таблиц, что существенно увеличивает время отклика. Проектирование хранилища данных подразумевает создание денормализованной структуры данных

(допускается избыточность данных и возможность возникновения аномалий при манипулировании данными), ориентированной в первую очередь на высокую производительность при выполнении аналитических запросов. Нормализация делает модель хранилища слишком сложной, затрудняет ее понимание и ухудшает эффективность выполнения запроса.

ERwin поддерживает методологию моделирования хранилищ благодаря использованию специальной нотации для физической модели – Dimensional. Наиболее простой способ перейти к нотации Dimensional в ERwin - при создании новой модели (меню File / New) в диалоге ERwin Teamplate Selection выбрать из списка предлагаемых шаблонов DIMENSION. В шаблоне DIMENSION сделаны все необходимые для поддержки нотации размерного моделирования настройки, которые, впрочем, можно установить вручную.

Моделирование Dimensional сходно с моделированием связей и сущностей для реляционной модели, но отличаются целями. Реляционная модель акцентируется на целостности и эффективности ввода данных. Размерная (Dimensional) модель ориентирована в первую очередь на выполнение сложных запросов к БД.

В размерном моделировании принят стандарт модели, называемый **схемой звезда (star schema),** которая обеспечивает высокую скорость выполнения запроса посредством денормализации и разделения данных. Невозможно создать универсальную денормализованную структуру данных, обеспечивающую высокую производительность при выполнении любого аналитического запроса. Поэтому схема звезда строится так, чтобы обеспечить наивысшую производительность при выполнении одного самого важного запроса, либо для группы похожих запросов.

Схема звезда обычно содержит одну большую таблицу, называемую таблицей факта (*fact table)*, помещенную в центр, и окружающие ее меньшие таблицы, называемые таблицами размерности (*dimensional table)*, соединенные с таблицей факта в виде звезды радиальными связями. В этих связях таблицы размерности являются родительскими, таблица факта - дочерней. Схема звезда может иметь также консольные таблицы*(outrigger table)*, присоединенные к таблице размерности. Консольные таблицы являются родительскими, таблицы размерности - дочерними.

В размерной модели ERwin обозначает иконкой роль таблицы в схеме звезда (рисунок 3)

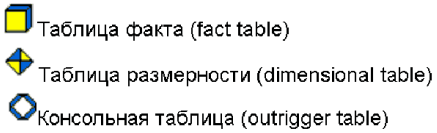


Рисунок 3. Обозначения таблиц в схеме “звезда”

Прежде чем создать базу данных со схемой типа звезда, необходимо проанализировать бизнес-правила предметной области с целью выяснения центрального вопроса, ответ на который наиболее важен. Все прочие вопросы должны быть объединены вокруг этого основного вопроса и моделирование должно начинаться с него. Данные, необходимые для ответа на этот вопрос, должны быть помещены в центральную таблицу модели - таблицу факта. На рисунке 4 приведен фрагмент учебной модели, входящей в поставку ERwin. Модель представляет собой хранилище данных фирмы, занимающейся продажей видеокассет. Например, необходимо создавать отчеты об общей сумме дохода от продаж за период, или по типу фильмов, или по рынкам фильмов. В таком случае следует разрабатывать модель так, чтобы каждая запись в таблице факта представляла общую сумму продаж, сумму для каждого клиента за определенный период времени и для каждого рынка. В примере таблица факта содержит суммарные данные о продажах (“REVENUE”), а таблицы размерности содержат данные о заказчике и заказах (“CUSTOMER”), продуктах (“MOVIE”), рынках (“MARKET”) и периодах времени (“TIME”).

Таблица факта является центральной таблицей в схеме звезда. Она может состоять из миллионов строк и содержать суммирующие или фактические данные, которые могут помочь ответить на требуемые вопросы. Она соединяет данные, которые хранились бы во многих таблицах традиционных реляционных базах данных. Таблица факта и таблицы размерности связаны идентифицирующими связями, при этом первичные ключи таблицы размерности мигрируют в таблицу факта в качестве внешних ключей. В размерной модели направления связей явно не показываются – они определяются типом таблиц. Первичный ключ таблицы факта целиком состоит из первичных ключей всех таблиц размерности. В примере (таблица

факта “REVENUE”) первичный ключ составлен из четырех внешних ключей: movie_key, market_key, customer_key и time_key.

Таблицы размерности имеют меньшее количество строк, чем таблицы факта и содержат описательную информацию. Эти таблицы позволяют пользователю быстро переходить от таблицы факта к дополнительной информации.

В примере на рисунке 4 таблица “REVENUE” – таблица факта; “CUSTOMER”, “TIME”, “MOVIE” и “MARKET” – таблицы размерности, которые позволяют быстро извлекать информацию о том, кто и когда сделал покупку, какой продавец и на какую сумму продал и какие именно товары были проданы.

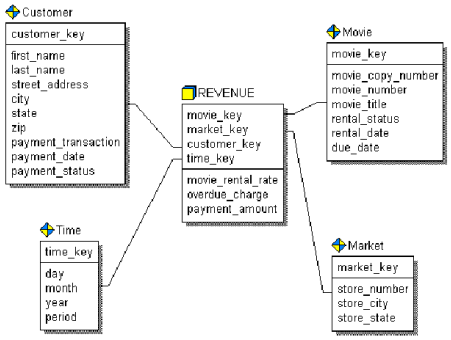


Рисунок 4. Схема звезда

ERwin поддерживает использование вторичных таблиц размерности, называемых консольными (outrigger) таблицами, хотя они не требуются для схемы звезда. Консольные таблицы могут быть связаны только таблицами размерности, причем консольная таблица в этой связи родительская, а таблица размерности - дочерняя. Связь может быть идентифицирующей или неидентифицирующей. Консольная таблица не может быть связана таблицей факта. Она используется для нормализации данных в таблицах размерности. Нормализация данных полезна при моделировании реляционной структуры, но она уменьшает эффективность выполнения запросов к хранилищу данных. В размерной модели главной целью является обеспечение высокой эффективности просмотра данных и выполнения сложных запросов. Схема снежинка обычно препятствует эффективности, потому что требует объединения многих таблиц для построения результирующего набора данных, что увеличивает время выполнения запроса. Поэтому при

проектировании не следует злоупотреблять созданием множества консольных таблиц. Когда консольные таблицы используются в размерной модели для нормализации каждой таблицы размерности, модель называется снежинка.

В диалоге описания свойств таблицы Table Editor имеется закладка Dimensional, в которой задаются специфические свойства таблицы в размерной модели (рисунок 5).

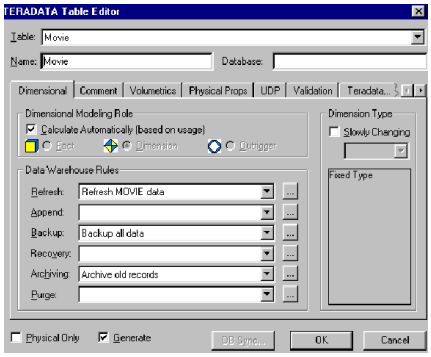


Рисунок 5. Закладка Dimensional диалога Table Editor

Роль таблицы в схеме (Dimensional Modeling Role). По умолчанию Erwin автоматически определяет роль таблицы на основании созданных связей (таблица факта, размерности или консольная). Таблица без связей определяется как таблица размерности, таблица факта не может быть родительской в связи, таблица размерности может быть родительской по отношению к таблице факта, консольная таблица может быть родительской по отношению к таблице размерности. При ручном назначении роли таблицы ERwin автоматически проверяет корректность размерной модели и выдает диалог с предупреждающим сообщением в случае следующих нарушений синтаксиса:

- Таблица факта не является в связи дочерней;
- Консольная таблица не является в связи родительской;
- Установлена идентифицирующая связь между консольной таблицей и таблицей факта.

Тип таблицы размерности (Dimension Type). Каждая таблица размерности может содержать неизменяемые, либо редко изменяемые данные (slowly changing dimensions). Поскольку хранилище данных имеет ненормализованную структуру, редактирование таблиц размерности может привести к коллизиям. Для того, чтобы избежать противоречий при

хранении данных, ERwin позволяет задать тип редко изменяемых данных, который отличается способом редактирования данных:

1. Перезаписывание старых данных новыми. При этом старые данные теряются.

2. Создание новой записи в таблице размерности с новыми данными и временем изменения. В этом случае сохраняются старые данные и можно проследить историю изменения редактируемых данных, но необходимо генерировать ключ для ссылки на старые данные.

3. Запись новых данных в дополнительном поле той же самой записи. В этом случае сохраняется первоначальное и последнее новое значение. Все промежуточные данные теряются.

Правила хранения данных (Data Warehouse Rules). Для каждой таблицы можно задать шесть типов правил манипулирования данными: обновление (Refresh), дополнение (Append), резервное копирование (Backup), восстановление (Recovery), архивирование (Archiving) и очистка (Purge). Для задания правила следует выбрать имя правила из соответствующего списка выбора. Каждое правило должно быть предварительно описано в диалоге Data Warehouse Rule Editor (меню Edit / Data Warehouse Rule). Для каждого правила должно быть задано имя, тип, определение. Например, определение правила дополнения данных может включать частоту и время дополнения (ежедневно, в конце рабочего дня), продолжительность операции и т.д. Связать правила с определенной таблицей можно с помощью диалога Table Editor.

Создание спецификаций для источников данных

При проектировании хранилища данных важно определить источник данных (для каждой колонки), метод, которым исходные данные извлекаются, преобразовываются и фильтруются прежде, чем они импортируются в хранилище данных. Хранилище данных может объединять информацию из текстовых файлов и многих баз данных, как реляционных (в том числе других БД на платформе Informix), так и нереляционных в единую систему поддержки принятия решений. Чтобы поддерживать регулярные обновления и проверки качества данных, необходимо знать источник для каждой колонки в хранилище данных. Для документирования информации об источниках данных используется редактор Data Warehouse Source Editor (рисунок 6.)

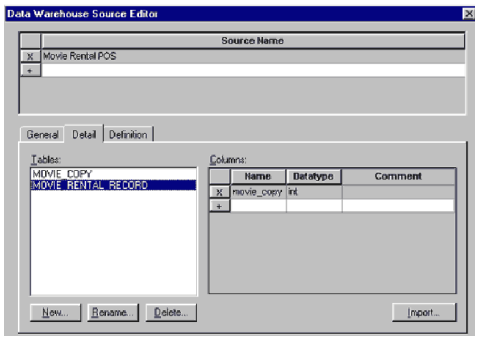


Рисунок 6. Диалог Data Warehouse Source Editor

Источник данных может быть описан вручную в диалоге Data Warehouse Source Editor, либо импортирован. В качестве источника при импорте могут быть использованы другие модели Erwin, хранящиеся в файле, SQL – скрипты, модели, хранящиеся в репозитории ModelMart, либо системные каталоги СУБД, на основе которых в результате обратного проектирования могут быть созданы модели Erwin.

Литература:

1. *Маклаков С.В.* BPwin и ERwin. CASE-средства разработки информационных систем. – М.: ДИАЛОГ-МИФИ, 1999. – 256 с.

2. Goetz Graefe. Query Evaluation Techniques for Large Databases // ACM Computing Surveys. - 1993. - Vol. 25, № 2. - P. 73-170.

3. *Mishra P., Eich M.H.* Join Processing in relational databases. // ACM Computing Surveys. - 1992. - Vol. 24, № 1.

УДК 004.91

СПОСОБ ОРГАНИЗАЦИИ МЕТАПОИСКА ИНФОРМАЦИИ НА ПРЕДПРИЯТИИ

И.В. Лабун, Р.С. Самарев

Информационная инфраструктура предприятия включает в себя определенное количество информационных систем – система документооборота, система управления проектами, телефонный справочник и пр. Информация, которая требуется пользователю в процессе решения его

должностных задач, может находиться во всех этих системах. Не всегда известно и то, где конкретно необходимо искать нужную информацию. Для упрощения доступа к информации необходимо применить поисковую систему. Однако сложность поисковой системы для предприятия, использующего множество различных информационных систем, заключается в том, что каждая из них в отдельности может предоставлять ограниченный доступ пользователями в соответствии с их правами доступа к информации. Это делает невозможным применение централизованной поисковой системы, подобной google или yandex, поскольку в случае индексации всех данных, будут утеряны права доступа к информации, а также понижается надёжность хранения и оперативное прекращение доступа к информации определённых должностных лиц. Выходом из сложившейся системы является создание метапоисковой системы, которая использует интерфейсы доступа существующих информационных систем с тем, чтобы решение задачи разграничения доступа не было вынесено за пределы этих систем.

Помимо задачи доступа к разнородным источникам, серьезной задачей метапоисковой системы является помощь пользователю в поиске информации путем интеллектуальной обработки поисковых запросов и выдачи ранжированного списка документов, найденных во всех (или в части) информационных системах, который наиболее полно соответствует поисковому запросу. Метапоисковая система автоматизирует этот процесс, экономя время пользователя и улучшая релевантность полученных данных. Интеграция метапоисковой системы предприятия с поисковыми движками интернета, такими как Google, Yahoo, Yandex, позволяет создать единую точку входа в поисковые системы и обеспечить поиск не только в рамках информационных систем предприятия, но и увеличить релевантность получаемых данных.

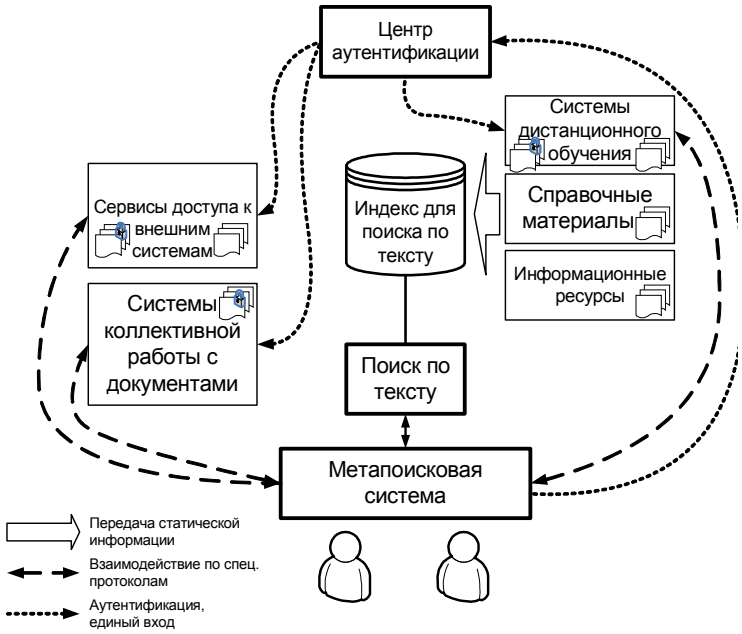


Рисунок 1. Схема потоков информации предприятия

На рисунке 1 представлен пример организации поиска информации в информационных системах предприятия. Знаком  отмечены информационные системы, доступ к информации в которых имеет ограничения на права доступа.

Метапоисковая система получает запрос пользователя и перенаправляет его к так называемым поисковым движкам нижнего уровня (реализуются конкретными информационными системами или представляют собой автономные средства индексации и поиска), возможно предварительно изменяя его формат, затем получает поисковые результаты, преобразует их в единый формат и выполняет их слияние в один ранжированный список [5]. В общем случае поисковые движки нижнего уровня гетерогенны, т. е. имеют различные структуры выходных данных, разные алгоритмы ранжирования и т.п. В связи с этим основные проблемы проектирования метапоисковых систем — унификация форматов результатов поиска нескольких поисковых систем нижнего уровня и ранжирование объединенного списка результатов поиска.

Краткий обзор существующих метапоисковых систем

На данный момент в сети Интернет существует множество сайтов, предоставляющих услуги по метапоиску своим пользователям. В зарубежном сегменте Сети самые популярные — это Webcrawler, Dogpile,

Yippy (бывш. Clusty) и пр. [3] В рунете же метапоисковые системы представлены не столь широко и наиболее известный из них Nigma.ru.

Помимо отображения непосредственно результатов поиска, т. е. тех результатов, которые были получены от поисковых движков нижнего уровня, метапоисковые движки могут предоставлять средства, облегчающие навигацию по списку найденных документов.

Движок Yippy разбивает множество результатов поиска на группы — т. н. «облака» (clouds), которые формируются при помощи алгоритмов кластеризации [4]. Манипулируя «облаком» из панели навигации, пользователь может существенно сузить область поиска. Также этот поисковик предоставляет возможность фильтрации найденной информации по источнику, домену (.com, .ru, .org и пр.). Metacrawler же для этих целей предоставляет пользователю список похожих запросов.

Метапоисковый сервис Nigma.ru более «интеллектуально» обрабатывает поисковые запросы и данные, получаемые от поисковых движков нижнего уровня. Кроме непосредственно ранжированного списка результатов поиска Nigma.ru, также как и поисковая система Yippy, выдает пользователю список групп, на которые разделено множество поисковых результатов (секция «фильтр»). Выбирая одну или несколько групп, можно сократить множество релевантных результатов поиска, что облегчает нахождение нужной информации. Данный метапоисковый движок предоставляет краткую справку об объекте поиска, источником которой предположительно являются проиндексированные Nigma страницы информационного ресурса Wikipedia. Nigma.ru изменяет представление области поиска в зависимости от поискового запроса. Например, при вводе в поисковую строку уравнения математической функции, перед всеми результатами поиска выводится график этой функции. Представление отдельного пункта в списке найденных страниц так же варьируется в зависимости от типа найденного документа (торрент, книга в интернет-магазине, изображение и пр.).

Особенности построения метапоисковых систем

При всем многообразии метапоисковых систем, есть две объединяющие их всех вещи. Первая – любая метапоисковая система обращается к нескольким (в общем случае гетерогенным) источникам информации. Вторая – при разработке любой метапоисковой системы остро

стоит задача разработки алгоритма слияния данных, полученных от низлежащих поисковых движков.

Тот факт, что источники информации гетерогенны подразумевает наличие в составе системы как минимум двух слоев (или модулей) на нижнем уровне (рисунок 2), которые бы создавали интерфейс доступа к поисковым движкам более низкого уровня для вышестоящей части системы.

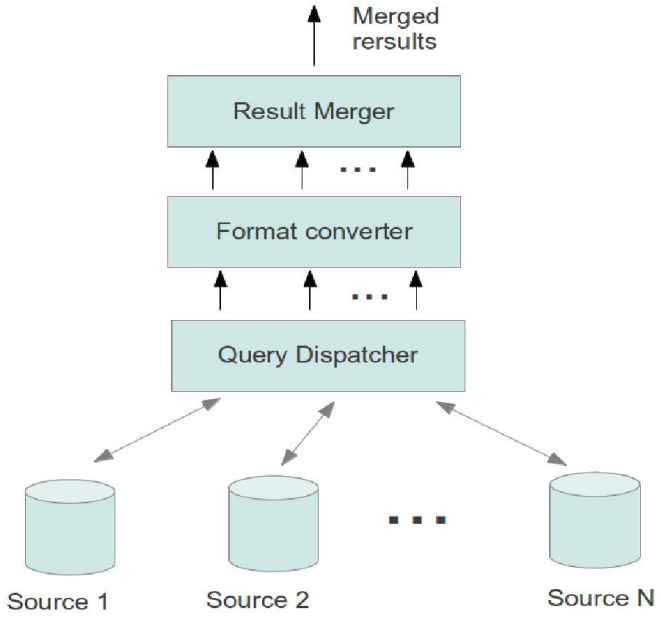


Рисунок 2. Структурная схема метапоисковой системы

Еще одним аспектом доступа к информации на нижнем уровне является аутентификация от имени пользователя, которых запрашивает информацию у метапоисковой системы, однако в данной работе этот аспект не рассматривается.

Первый слой (Query Dispatchers) предназначен для нивелирования различий в протоколах доступа к источникам информации [5]. В современных информационных системах широко распространены протоколы передачи информации HTTP, FTP, SMTP. Некоторые протоколы (например HTTP) используют несколько методов доступа к данным (GET, POST, PUT и пр.). Поэтому очень важно обеспечить для вышестоящих модулей унифицированный доступ к информационным ресурсам. Архитектура системы метапоиска должна быть спроектирована таким образом, чтобы обеспечить возможность подключения модулей и для сбора информации с источников по другим протоколам.

Второй слой (Format Converter) предназначен для унификации форматов данных, получаемых от нижестоящих поисковых движков. В веб-приложениях (используемых как в сети Интернет, так и во внутренних сетях) применяются три основных формата данных, которые используются для передачи информации между клиентом и сервером: HTML, XML и JSON. Эти форматы позволяют хранить и передавать данные различной структуры. Данный слой преобразует данные, полученные от поисковых движков, во внутренний формат, который позволяет единым образом обрабатывать эту информацию.

Слой Result Merger выполняет слияние результатов поиска в единый ранжированный список. Проблема ранжирования результатов является одной из основных, так как очень важно предоставить пользователю документы в порядке релевантности его запросу. Более широко этот вопрос освещен в работе [6].

Ряд поисковых движков предоставляет дополнительные сведения о весе каждого пункта в списке найденных ресурсов в соответствии с релевантностью запросу. Однако эта характеристика доступна не везде и существует проблема согласования весов, полученных из различных поисковых систем. Поэтому невозможно производить ранжирование результатов поиска, основываясь только на весе, которые вычисляет поисковая система для каждого документа.

Для ранжирования результатов необходимо пересчитать вес соответствия запросу для каждого найденного документа по единому методу вычисления. Пересчет происходит в две стадии. Первая стадия называется Database Selection [5], на этом этапе метапоисковая система выясняет насколько каждый поисковый движок релевантен данному запросу путем присуждения каждому движку очков. На второй стадии происходит непосредственно вычисление ранга для каждого найденного документа [5,6].

Существует множество алгоритмов вычисления ранга в процессе слияния результатов поиска. В ходе дипломного проектирования был выбран способ (рисунок 3), который на стадии Database Selection использует алгоритм TopD [6] (Top Document). Этот алгоритм лучше всего подходит для решаемой задачи, т.к. он не основывается на поисковых очках документов, которые не всегда есть возможность получить от поискового движка.

Данный алгоритм начисляет очки конкретному поисковому движку в зависимости от значения меры схожести между поисковым запросом и первым документом в выдаче. Интуитивно понятно, что документ, стоящий на первом месте в выдаче наиболее релевантен поисковому запросу и отражает общую релевантность поискового движка.

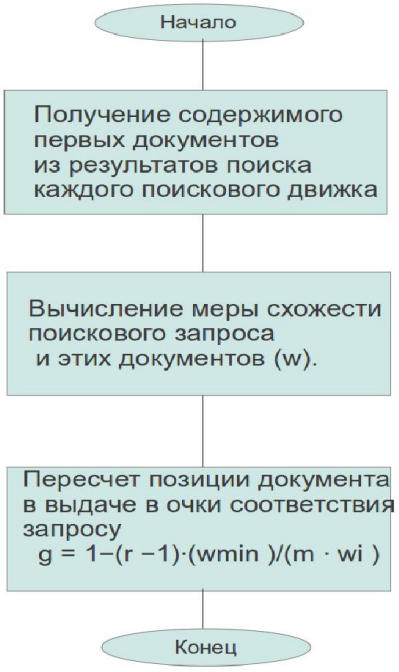


Рисунок 3. Схема алгоритма ранжирования результатов поиска

На стадии Database Selection для вычисления очков релевантности поисковых движков удобно использовать библиотеку Lucene [2]. Эта библиотека предназначена для полнотекстового поиска. Для вычисления очков релевантности сначала получается содержимое первых документов всех поисковых выдач, затем создается индекс в оперативной памяти ЭВМ (из соображений безопасности) и проводится полнотекстовый поиск по поисковому запросу, в процессе которого и вычисляются меры схожести запроса и найденных документов.

Также эту библиотеку планируется использовать для оценки качества алгоритма слияния результатов поиска. В данном случае план эксперимента следующий:

1. Проводится поиск по тестовому поисковому запросу.
2. Получается содержимое всех найденных документов.
3. С помощью библиотеки Lucene создается индекс этих документов.

4. Средствами Lucene проводится поиск по индексу.

5. Сравниваются результаты работы алгоритма слияния и библиотеки Lucene.

Заключение

Рассмотренные проблемы построения информационных систем являются актуальными и не решенными в полной мере. Разработка указанной метапоисковой системы предприятия позволит существенно улучшить удобство доступа к информации и повысить эффективность работы сотрудников.

В настоящее время разработка метапоисковой системы не завершена. Дополнительным требованием к работе является необходимость оценки качества слияния результатов, которая на данный момент также не получена. Решение данных проблем планируется завершить в ближайшее время.

Литература:

1. *К. Маннинг, П.Рагхаван, Х.Шютце.* Введение в информационный поиск.: Пер. с англ. - М.: ООО «И.Д. Вильямс», 2011, 528 с.

2. Интернет-ресурс: http://lucene.apache.org/

3. Интернет-ресурс: http://www.wikipedia.org/

4.Интернет-ресурс: http://www.readwriteweb.com/archives/overview_of_clu.php

5. *W. Meng, C. Yu, K. Liu*. Building Efficient and Effective Metasearch Engines

6. *Y. Lu, W. Meng, L. Shu, C. Yu, K. Liu*. Evaluation of Result Merging Strategies for Metasearch Engines

УДК 159.9;004.7;004.4

ПРОФЕССИОНАЛЬНАЯ ПРОБЛЕМА ПРОГРАММИСТОВ В ОБЛАСТИ РАЗРАБОТКИ ИНТЕРНЕТ-СИСТЕМ

И.Д. Маслов, В.В. Сюзев

Бытует мнение, что программисты, то есть люди, которые создают программы для различных устройств – "ненормальные" люди. Термин "ненормальные" каждый понимает по-своему: кто-то связывает ненормальность с формой эгоизма, кто-то с повышенной самооценкой, а кто-то судит этим словом по тому, как человек общается с внешним миром. Безусловно, внутренние качества человека непременно связаны с тем, как человек взаимодействует с окружающей средой, но всё же, понимание термина "ненормальный" в каждом случае различно.

Почему в обществе сложился стереотип программиста с такими не лучшими качествами, причем стереотип столь актуальной в наше время профессии? Позвольте обратиться к уже проведённому социологическому исследованию, которое имеет непосредственное отношение к затронутой теме.

Исследование было выполнено Сарой Саррафи Заде и Кируниссой Бегум на факультете по изучению питания Университета города Майсур (Индия). Испытуемыми были 91 инженер-программист, работающие в фирме, занимающейся разработкой программного обеспечения, расположенной в Майсуре.

Согласно Заде и Бегум, не менее чем 20.9% несчастных инженеров имеют сильную бессонницу и еще 35.2% имеют эту проблему в "мягкой" степени. Меньше половины программистов спят нормально, по сравнению с 77% общего населения.

Согласно сопроводительному документу, умственное и психическое здоровье у людей, страдающих бессонницей значительно хуже, нежели у нормально спящих участников. Связь между бессонницей и плохим состоянием особенно сильна в области умственного здоровья.

Заде и Бегум делают вывод: "Программы по управлению стилем жизни, которые включают гигиену сна и заботу о нем, должны быть

внедрены в качестве политики предприятий в сфере информационных технологий".

Я хочу обратить ваше внимание на следующие особенности: на основе приведенного социологического исследования можно сказать, что стереотип профессии программиста сложен не самым благоприятным образом по отношению к человеческим качествам того же программиста, что безусловно откладывает свой нехороший отпечаток на качестве взаимодействия с окружающим миром, то есть с другими людьми. Казалось бы: "Ну и что?", возможно именно у вас и возник подобный вопрос, в таком случае позвольте задать вам встречный: "Считаете ли вы себя независимым и полностью автономным человеком?". Другими словами: "Можете ли вы полностью изолировать себя от всех окружающих вас живых существ и при этом сохранить свой человеческий облик?" Интуиция и здравый смысл подсказывают мне, что в этом вы полностью согласитесь со мной и назовёте себя зависимым и лишь частично автономным существом. Но если вдруг на миг вы подумали, что зависимость от окружающего мира – это плохо, то поспешу прояснить вам ситуацию и тем самым переубедить: на протяжении всего того времени, что существует человечество, всегда происходило взаимодействие между людьми, и если бы не взаимодействие, то скорее всего человечество прекратило свое существование. Точнее говоря: контакт человека с окружающим миром – это необходимое свойство человека, без которого человек просто-напросто не мог бы существовать в силу своих физиологических и духовных качеств.

В этом ключе хотелось бы поподробнее рассмотреть следующий вопрос: "Влияют ли профессиональные навыки программиста на его человеческие качества общения с окружающим миром?". Возможно, было бы не совсем корректно давать однозначные ответы такого плана как: "Да, действительно, профессия на всех влияет одинаково, и все в равной степени будут страдать от её воздействия", хотя бы потому, что человек – существо непредсказуемое и до конца не изученное, причем, вполне возможно, что никогда и не будет до конца изученным. Но выявить для себя некоторую закономерность в этом вопросе среди некоторого множества специалистов – считаю вполне уместным и даже необходимым.

Начну с того, что из вышеприведённого исследования можно извлечь следующую информацию: больше половины программистов имеют

проблемы со сном. Дальше можно сказать, что, как правило, проблемы со сном связаны с переносом трудностей, другими словом – стресса, значит большинство программистов подвержено стрессу. Стресс, в свою очередь, возникает из-за какого-либо события, например, из-за нерешенной задачи, которую нельзя или не получается забыть, и как следствие она начинает донимать человека. Вдобавок к этому, проанализировав свою профессиональную деятельность, я сделал вывод о том, что в программисты зачастую сталкиваются с трудностями при решении задач, связанных с оборудованием и программным обеспечением. То есть, программист решает задачу реализации (кодирования) созданных алгоритмов на вычислительной машине, что вызывает в свою очередь стресс у человека, а соответственно и повышает вероятность нарушения сна. Итак, мы пришли к выводу, что программисты подвержены неблагоприятным условиям в силу своей профессиональной деятельности, и на основе этого вывода пойдет дальнейшее размышление.

На сегодняшний день существует огромный выбор средств разработки для программистов, таких как Rational Application Developer, Visual Studio, Delphi, NetBean, WebStorm и т.д., которые создают благоприятные условия для разработки программных продуктов. Но даже при таком обилии программных продуктов, задача перед разработчиками по-прежнему велика – а именно, велика задача кодирования алгоритмов руками программистов, даже при всех тех инструментах, которые стремятся упростить для программиста этап кодирования. В каждом отдельном случае, естественно, объём и сложность программ различны, но в идеале, все варианты оставляют желать лучшего.

Предположим, что программист по профессии – это, в первую очередь, человек, который создает программы. Определим 2 основных вида работ программиста: 1 – создание алгоритмов будущей программы для решения определенной задачи; 2 – создание (написание и отладка) кода программы, реализующей те самые алгоритмы, которые были придуманы программистом при выполнении первого типа работ (в данном случае описание и тестирование программы производится на определённом языке программирования, так называемые – кодирование и тестирование).

В реальной жизни мы (люди на земле) очень часто сталкиваемся с тем, что к программисту поступает задача, и он решают ее, сначала выполняя

работу первого типа, а потом выполняя работу второго типа. Также встречаются разработки, в которых не соблюдается такое разграничение, например, когда этот программист с момента получения задания начинает писать код программы, обдумывая алгоритмы, и в тот же самый момент, подстраивая их под определённый язык программирования. В целях упрощения понимания смысла этой статьи, введу допущение и соответствующие им обозначения: пусть тот промежуток времени, за который ведется разработка программного продукта от момента получения задания до момента выдачи его заказчику, будет называться "время разработки". Определю время разработки, как промежуток времени, за который выполняется 2 этапа: 1– программист занимаются работой 1-го типа; 2 – программист занимаются работой 2-го типа; и введу очевидное, но в то же время и не очень информативное, условие – время, потраченное на первый этап + время потраченное на второй этап, равно времени разработки. Замечу, что при таком условии, под термином "время разработки" я понимаю только то время, которое было затрачено непосредственно на создание программного продукта.

Итак, рассмотрю этапы поподробнее. С первым этапом, думаю, всё ясно: программист разрабатывают алгоритмы, ориентированные на решение поставленной им задачи, в силу своих возможностей. Сразу оговорюсь, что при создании алгоритмов используются не только элементы творчества, но еще используются определённые умения, полученные программистом в процессе своего обучения.

А вот что насчет второго этапа? Использование языка программирования - это ряд дополнительных задач, которые нужно своевременно решать программисту для того, чтобы добиться конечной цели, а именно выдать заказчику конечный программный продукт; такими дополнительными задачами могут быть:

- знакомство с языком (синтаксис, семантика);
- знакомство с библиотеками, необходимыми для работы;
- разрешение проблем совместимости, как аппаратных устройств, так и программных решений (библиотек, служб, программ и т. п.);

- разрешение вопросов, возникших в связи с особенностями выбранного языка программирования при реализации определённых алгоритмов;
- написание кода (набивание клавиш);
- отладка кодированных алгоритмов.

Вот основные вопросы, возникающие перед программистом в тот момент, когда дело доходит до реализации программы на языке программирования.

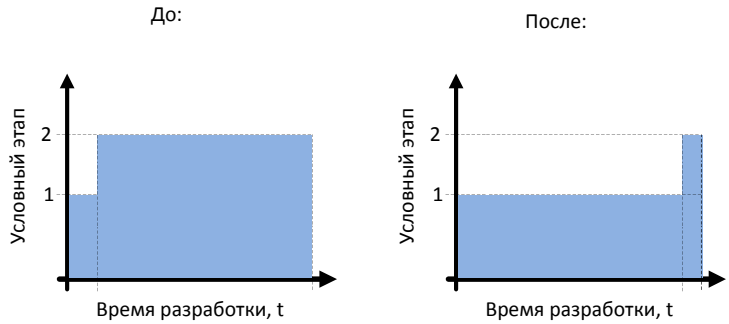
Практика показывает, что 2 этап является менее нагруженным в творческом плане (во многих случаях элемент творчества сведён к минимуму) по сравнению с 1-м этапом. Действительно, данное наблюдение несложно подтвердить, но, к сожалению, при том, что составляющая творчества во 2-м этапе несколько ниже, чем в первом, элемент нетворческой работы выше. Причем, как правило, не просто выше, а значительно выше, и как следствие, время, которое нужно затратить на первый этап, численно меньше, чем время, которое нужно затратить на второй этап.

Под термином "нетворческая работа", я понимаю следующие задачи:

- чтение источников (интернет, газеты, буклеты, книги) определённой направленности;
- разрешение проблем совместимости, связанных с выбранным языком программирования и не имеющих отношения к самим алгоритмам;
- консультации со специалистами из области кодирования;
- печать (написание в электронном формате) кода;
- проверка реализованных алгоритмов.

Так вот с учетом вышеизложенного, хотелось бы задать следующий вопрос: "А можно ли сделать так, чтобы программист избавился от второго этапа?". Иными словами добиться того, чтобы программистов 2-го типа не было. По такому вопросу мой здравый смысл говорит мне, что такое невозможно, точнее говоря – нельзя получить готовый программный продукт, всего лишь придумав алгоритм для этого продукта, хотя может быть когда-нибудь в будущем такое будет возможно. Но тот же самый здравый смысл подсказывает мне кое-что еще: может быть, нельзя уничтожить 2-й этап как таковой, но вот уменьшить его, свести к минимуму,

чтобы не он занимал основную часть времени разработки, а наоборот, чтобы первый этап занимал большую часть времени разработки – это мне кажется возможным. Представлю то, что я сказал в виде графиков:



Таким образом, я хочу сказать, что хотелось бы реализовать программный продукт для программистов, позволяющий получить производительность согласно графику, показанному чуть выше.

Поставлю задачу: получить программное решение (программный продукт), предназначенное для разработчиков (программистов), которое сведёт к минимуму 2-й этап в процессе разработки программных продуктов.

Звучит, на мой взгляд, ясно, но обширно, а именно обширно в плане описания области применения этой «чудо-программы». При осмыслении такой задачи я исхожу из того, что если какой-либо вопрос в задаче не обозначен, то предполагается максимально полное разрешение данного вопроса. В таком случае я, попросту говоря, не могу себе представить конечную сложность такой программы, поскольку в наше время существует огромное количество структур (языков), а как следствие, областей для программирования, и, соответственно, чем больше область применения, тем сложнее программа. Такая ситуация не вызывает симпатию, поэтому имеет смысл сузить область применения этого программного продукта. С учетом того, что на сегодняшний день практически каждый человек на земле контактирует с сетью Интернет, то имеет смысл заниматься решением поставленной задачи в области Интернет-разработок.

Таким образом, получаю более детальную постановку задачи: получить программное решение (программный продукт), предназначенное для разработчиков (программистов) в области Интернет-разработок, которое сведёт к минимуму 2-й этап в процессе разработки программных продуктов.

Литература:

1. Иисследование университета майсура: http://webplanet.ru/news/research/2010/11/25/prg.html

2. О назначении человека; *Бердяев Н. А.*;isbn 978-5-94865-322-8; 2008 г.;701с

3. Практика и теория программирования. В 2 книгах; *Н.А. Винокуров, А. В. Ворожцов*; Физматкнига; ISBN 978-5-89155-182-4, 978-5-89155-180-0; 2008 г.288с.

УДК 681.3.07

НАЗНАЧЕНИЕ ВНЕШНИХ КЛЮЧЕЙ В ЗАПОЛНЕННЫХ РЕЛЯЦИОННЫХ ТАБЛИЦАХ

Мин Тхет Тин, А.В. Брешенков, Д.Ю. Гудзенко

В статье рассмотрены причины необходимости назначения внешних ключей в заполненных реляционных таблицах. Представлен неформальный алгоритм назначения внешних ключей в заполненных реляционных таблицах. Представлен формальный алгоритм назначения внешних ключей в заполненных реляционных таблицах.

Введение

Необходимость назначения внешних ключей в заполненных реляционных таблицах возникает в том случае, если эти ключи разработчиками базы данных (БД) не назначены. Впоследствии БД эксплуатируется и таблицы заполняются соответствующими записями. В конечном итоге из соображений обеспечения нормальных форм, а также необходимости отражения реальных связей между таблицами может потребоваться назначение внешних ключей.

Кроме того, внешние ключи необходимо назначать после преобразования информации табличного типа (ИТВ) в реляционные таблицы, которые являются основными компонентами реляционных БД. В качестве ИТВ могут быть представлены текстовые файлы, электронные таблицы файлы формата doc..

При назначении внешних ключевых полей следует руководствоваться основным требованием к ним – обеспечение ссылочной целостности. Внешние ключи должны совпадать с первичными ключами каких либо таблиц.

Вышесказанное и является предпосылками для разработки алгоритмов. Предлагается сканировать значения первичных ключей каждой таблицы и сравнивать их со значениями всех атрибутов остальных таблиц. Если в остальных таблицах найдутся атрибуты, значения которых совпадают со значениями первичного ключа анализируемой таблицы, то такие атрибуты претендуют на роль внешних ключей. Решение о назначении найденных атрибутов их внешними ключами принимает разработчик.

Неформальные алгоритмы назначения внешних ключей в заполненных таблицах

В соответствии с [1] определение внешнего ключа звучит следующим образом. Пусть имеется отношение R2, значения атрибутов которого равны значениям атрибутов первичного ключа отношения R2. Тогда такие атрибуты являются внешним ключом по отношению к R1.

В соответствии с определением для нахождения внешних ключей необходимо проанализировать каждую таблицу все возможные сочетания пар таблиц БД. Для каждой пары таблиц выполнить поиск неключевых атрибутов одной таблицы, значения которых были бы равны значениям ключевых атрибутов другой таблицы. Если таковые найдутся, то они и будут претендовать на внешний ключ.

Следует обратить внимание на то, что эти ключи ориентированы на формирование связи между таблицами типа “многие к одному”. Они и представляют наибольший интерес для баз данных (БД). Это связано с тем, что связь типа “один к одному” по сути представляет собой связь между частями одной таблицы, а связь типа “многие ко многим” базируется на двух связях типа “многие к одному” между рассматриваемыми таблицами и какой-либо третьей таблицей.

Неформальный алгоритм.

П1. Осуществляется поиск всех возможных сочетаний пар таблиц анализируемой БД.

Пусть имеется табличное пространство T:

T=(T1, ..., Ti, ..., Tm, ..., Tk), $i = \overline{1,k}$, где k – число прикладных таблиц БД.

$MPT = \varnothing$

Ищутся все возможные сочетания пар таблиц и запоминаются в массив MPT

```
Count=0
For i = 1 to k-1
    For j = i+1 to k
        Count = Count + 1
        S = Concat (Ti ,  TJ)
       MPT (Count) = S
    Next i
Next j
```

Таким образом, в массиве MPT сформируются все возможные пары атрибутов, а счетчике Count хранится их количество.

П2. Для каждой пары таблиц из массива MPT выполняется поиск внешнего ключа.

Организуется цикл на Count итераций, в котором перебираются все пары таблиц.

Для каждой пары таблиц выполняются проверки. Причем рассматриваются два случая: когда в качестве таблицы с первичным ключом рассматривается 1-я таблица; когда в качестве таблицы с первичным ключом рассматривается 2-я таблица.

П2.1. Для ключа состоящего из одного атрибута.

$A_i = (e_{i_1} \ldots . e_{i_m})$, где A_i - атрибут, соответствующий первичному ключу 1-й таблицы, e_{i_1} - 1-й элемент домена с атрибутом A_i, e_{i_m} – m-й элемент домена с атрибутом A_i, m – мощность 1- й таблицы.

$A_j^2 = (e_{j_1}^2 \ldots e_{j_f}^2)$ где A_j^2 – атрибут, соответствующий неключевому атрибуту 2-й таблицы, $e_{j_1}^2$ – 1-й элемент домена с атрибутом A_j^2, $e_{j_f}^2$ – f-й элемент домена с атрибутом A_j^2, f – мощность 2-й таблицы. Важно помнить, что число столбцов у 2-й таблицы, как правило, больше одного.

Следует отметить, что m и f в общем случае не равны. Ведь таблицы, как правило, имеют различные мощности.

Организуется цикл на g итераций, где g – число столбцов 2- й таблицы.

Сравниваются значения первичного ключа 1-й таблицы и значения текущего столбца 2-й таблицы:

Count1 = 0

For t = 1 to m

For p = 1 to f

If $e_{i_t} = e^2_{j_p}$ THEN Count1 = Count1 + 1

End p

End t

If Count1 ≥ Min(m, f) Then PRINT (“столбец”, g, “2-й таблицы претендует на внешний ключ”)

Завершается цикл по g.

Если ни для одного столбца таблицы 2 не сформируется сообщение, то во 2-й таблице внешних ключей нет.

Здесь m – мощность 1-й таблицы,

f – мощность 2-й таблицы.

В качестве пояснения можно сказать, что истинность конструкции Count ≥ Min(m, f) в подавляющем случае подтверждает, что найден внешний ключ. Действительно, для каждого значения одной таблицы нашлось равное ему значение другой таблицы.

П2.2. Для ключа, состоящего из двух атрибутов.

Принципиальное отличие от предыдущего алгоритма состоит в том, что во второй таблице рассматриваются все возможные сочетания пар атрибутов 2-й таблицы.

В связи с этим необходимо предварительно сформировать массив всех возможных пар атрибутов 2-й таблицы.

MPA = ∅

Count1 = 0

For i = 1 to f-1

For j = 1 to f

S = Concat(Ai, Aj)

Count1 = Count1 + 1

MPA(Count1) = s

Next j

Next i

Здесь MPA – массив пар атрибутов 2-й таблицы,

f – степень 2-й таблицы,

Ai, Aj – два атрибута 2- й таблицы.

Организуется цикл на Count1 итераций, где Count1– число сочетаний пар атрибутов 2- й таблицы, которые содержатся в массиве MPA, z – параметр цикла.

Сравниваются все пары значений, соответствующих первичному ключу 1-й таблицы, и все пары значений для каждого сочетания атрибутов 2-й таблицы:

Count1 = 0

For t = 1 to m

For p = 1 to f

If Concat(e_{i_t}, e_{k_t}) = ep (MPAz)THEN Count1 = Count1 + 1

End p

End t

If Count ≥ Max(m, f) Then PRINT (“столбцы”, MPAz , “2-й таблицы претендует на внешний ключ”

Завершается цикл по Count1.

Если ни для одного сочетания таблицы 2 не сформируется сообщение, то во 2-й таблице внешних ключей нет.

Здесь

$A_i, A_k = ((e_{i_1}, e_{k_1}) \ldots . (e_{i_m}, e_{k_m})$), где A_i – 1-й атрибут, соответствующий первичному ключу 1-й таблицы, A_k – k-й атрибут, соответствующий первичному ключу 1-й таблицы, e_{i_1} – 1-й элемент домена с атрибутом A_i, e_{k_1} – 1-й элемент домена с атрибутом A_k.

m – мощность 1-й таблицы,

f – мощность 2-й таблицы,

e_{i_t}, e_{k_t} – текущие t-е значения двух элементов первичного ключа 1-й таблицы,

ep (MPAz) – текущие p-е значения двух элементов 2-й таблицы. При этом номера сочетаний определяются z-м элементом массива MPA (массива, в котором содержатся все пары атрибутов 2-й таблицы).

П2.3 и П2.4. Для ключа, состоящего из трех атрибутов и для ключа, состоящего из четырех атрибутов выполняются действия в основном на базе конструкций П2.1. и П2.2. Но из соображений их значительного объема описания не приводятся.

Завершается цикл на Count итераций, в котором перебираются все пары таблиц.

Формальные алгоритмы назначения внешних ключей в заполненных таблицах

П1. Осуществляется поиск всех возможных сочетаний пар таблиц анализируемой БД.

/* T=(T1, …, Ti, …, Tm, …, Tk), $i = \overline{1,k}$, где к – число прикладных таблиц БД */

$MPT = \varnothing$ _ _ Массив всех пар таблиц БД

/* Ищутся все возможные сочетания пар таблиц */

```
 Count=0
For i = 1 to k-1 _ _ k – число таблиц БД
    For j = i+1 to k
        Count = Count + 1_ _ счетчик числа пар таблиц
        S = Concat (Ti ,  TJ)
       MPT (Count) = S
    Next i
Next j
```

П2. Для каждой пары таблиц из массива MPT выполняется поиск внешнего ключа.

/* Перебираются все возможные пары таблиц, сформированные ранее, перебираются все столбцы 1-й таблицы текущей пары, перебираются все столбцы 2-й таблицы текущей пары. Каждый элемент первого текущего столбца сравнивается с каждым элементом второго текущего столбца. Если элементы равны, то они подсчитываются. Если совпавших элементов оказалось больше, чем наибольшая мощность пары таблиц, то столбцы претендуют на ключевые */

```
F = 0 _ _ флажок наличия связанных ключей
Count1 = 0 _ _ счетчик совпадений
For i = 1 to Count  _ _  – число пар таблиц БД
    For l = 1 to C1_ _  – степень 1-й таблицы из i-й пары
        For m= 1 to C2 _ _  – степень 2-й таблицы из i-й пары
            For j= 1 to M1_ _ –  мощность 1-й таблицы из i-й пары
                For k= 1 to M2_ _  –  мощность 2-й таблицы из i-й пары
                    If e_lj = e_mk Then Coun1 = Coun1 + 1
                Next k
            Next j
            If  Coun1 >= Max(M1, M2) Then
            Begin
                Print (“Для пары таблиц ”, i, “столбцы”, l, “ и ”, m)
                Print (“ претендуют на первичный и внешние ключи”)
                Count1 = 0
                F = 1
            End
        Next m
    Next l
Next i
If F = 0 Then  Print (“ внешних ключей не обнаружено ни для одной из таблиц”)
```

Следует обратить внимание на то, что атрибуты, предложенные алгоритмом в качестве связанных атрибутов, могут не удовлетворять разработчика, поэтому необходимо предоставить ему возможность окончательного решения.

Литература:

1. *Дейт К. Дж.* Введение в системы баз данных: Пер. с англ. - М.: Наука, 1980.-464 с.

2. *Брешенков А.В., Балдин А.В.* Анализ проблемы проектирования реляционных баз данных на основе использования информации табличного вида и разработка модели методики проектирования. М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. -150 с.

3. *Брешенков А.В.* Методы решения задач проектирования реляционных баз данных на основе использования существующей информации табличного вида. М.: Изд-во МГТУ им. Н.Э. Баумана, 2007. - 150 с.

УДК 681.322.01

НАДЕЖНЫЕ ЦИКЛИЧЕСКИЕ ТОПОЛОГИИ ДЛЯ БОЛЬШИХ КОМПЬЮТЕРНЫХ СЕТЕЙ

Д.А. Ошуркевич, Г.П. Можаров

Циклическая конфигурация сети процессоров компьютерных систем (КС) отличается простотой организации и управления. Сообщения между элементарными процессорами (ЭП) передаются по связям в одном направлении, пока не поступят на требуемый процессор КС, при этом отпадает необходимость в принятии решения о направлении пересыпки сообщения.

Для сравнительного изучения различных способов организации циклической сети процессоров КС интересны оценки максимальной и средней задержки сообщения, надежности, счетного времени процессоров, а также максимальной пропускной способности. Эти величины взаимозависимы и связаны с топологией КС [1].

Что касается надежности, то циклическая сеть неустойчива к ошибкам, т.к. любой сбой нарушает связь между процессорами КС. Далее, при увеличении количества ЭП в КС сообщениям приходится проходить больше промежуточных ЭП. Так, диаметр однонаправленного цикла из N ЭП составляет $N-1$, а средняя длина пути сообщения – $N/2$. С другой стороны, при увеличении размера сети БЦВС требуется большая общая пропускная способность, с которой элементы связи могут не справиться.

Для повышения отказоустойчивости можно, в частности, соединить все ЭП с центральным ЭП, способным рассылать сообщения по периферийным ЭП, но при этом нарушается однородность сети и повышается ее зависимость от живучести центрального ЭП. Другой подход к

проблеме предусматривает введение дополнительных связей между ЭП. Помимо повышения отказоустойчивости это уменьшает время передачи сообщений и разгружает каналы связи, повышая пропускную способность КС. Рассмотрим топологии двойного цикла, для которых исследуются проблемы живучести и отказоустойчивости к сбоям КС. В рассматриваемых КС N узлов (ЭП) соединены в основной однонаправленный цикл и, кроме того, каждый узел имеет дополнительную связь для улучшения функционирования и повышения отказоустойчивости. Пример такой сети – двунаправленный полный цикл, выдерживающий одиночные сбои. Аппаратный интерфейс цикла может быть выполнен на базе микропроцессоров и обеспечивать простой алгоритм обхода поврежденного ЭП или связи. Диаметр такой сети составляет $N/2$, а среднее время передачи сообщения – $N/4$ [1, 2].

Другим примером изучаемой архитектуры является цикл, в котором узел x соединяется с узлами $x+1$ и $x-s \pmod N$, т.е. имеет переход на один узел вперед и на s узлов назад. Такая сеть имеет большую пропускную способность и отказоустойчивость и меньшее время передачи сообщений, чем предыдущая. Для большинства значений N оптимальная (или близкая к оптимальной) в смысле функционирования и отказоустойчивости сеть достигается при $s = \lfloor \sqrt{N} \rfloor$. В оптимальной сети между каждой парой узлов существует несколько путей, поэтому система может работать при выходе из строя нескольких элементов.

Для каждого узла существует диаметрально противоположный узел, отстоящий на расстоянии d диаметра сети. Параметр s выбирается таким образом, чтобы минимизировать $d = \max d_{ij}$ для всех i, $i = 0, 1, \ldots, N-1$, где d_{ij} – длина кратчайшего пути между узлами i и j.

Для получения кратчайшего пути между узлами i и j выполняются переходы назад до тех пор, пока это выгодно. Для диаметрально противоположных узлов переходы назад выгодны, пока d больше, чем $(N - bs)$, где b – количество шагов назад. Дополнительно к этому для диаметрально противоположных узлов требуется $s-1$ переход вперед. Таким образом, соотношение между d и s имеет вид:

$$d = N - (d - s + 1)s + (s - 1),$$

откуда

$$d = \left\lfloor \frac{N}{s+1} \right\rfloor + (s-1).$$

Для минимизации d при заданном N подходит, в частности, значение

$$s = \left\lfloor \sqrt{N} \right\rfloor.$$

Средняя длина пути $\bar{\rho}$, который проходит сообщение, может быть определена из уравнения

$$\bar{\rho} = \frac{1}{N} \sum_{i=0}^{N-1} \left[\frac{1}{N-1} \sum_{j=0}^{N-1} d_{ij} \right],$$

где $d_{ij} = 0$. Если КС симметрична, то выражение в квадратных скобках одинаково для всех i, следовательно, средняя длина пути прохождения сообщения составляет

$$\bar{\rho} = \frac{1}{N-1} \sum_{j=0}^{N-1} d_{ij} \text{ при всех } i,$$

а максимальное количество пересылок назад –

$$\rho_{b_{\max}} = \left\lfloor \frac{N}{s+1} \right\rfloor.$$

Если N – полный квадрат, то $\rho_{b_{\max}} = s-1$. Отсюда можно посчитать сумму расстояний от заданного узла до других, составляющую

$$\rho_{\Sigma} = \frac{\left(N - \rho_{b_{\max}} s\right)\left(N - \rho_{b_{\max}} s - 1\right)}{2} + \frac{\rho_{b_{\max}} s}{2}\left(\rho_{b_{\max}} + s\right).$$

Среднее число пересылок $\Pi_{\text{ср}}$ будет определяться выражением

$$\Pi_{\text{ср}} = \rho_{\Sigma} / (N-1).$$

Подставляя вместо $\rho_{b_{\max}}$ и ρ_{Σ} их выражения, можно получить

$$\Pi_{\text{ср}} = \frac{1}{2(N-1)} \left[\left(N - \left\lfloor \frac{N}{s+1} \right\rfloor s \right) \cdot \left(N - \left\lfloor \frac{N}{s+1} \right\rfloor s - 1 \right) + \left\lfloor \frac{N}{s+1} \right\rfloor s \left(\left\lfloor \frac{N}{s+1} \right\rfloor + s \right) \right].$$

Чтобы минимизировать среднее число пересыпок, требуется минимизировать выражение во вторых квадратных скобках. Это достигается при

$$s+1 = \frac{N}{s+1},$$

откуда $s = \left\lfloor \sqrt{N} \right\rfloor$.

В таблице для различных значений N приведены максимальные (левые колонки) и средние (правые колонки) количества пересылок сообщения

для двунаправленного цикла $(s=1)$, для сети при $s=2$ и для КС при $s=\lfloor \sqrt{N} \rfloor$.

Для устранения конфликтных ситуаций, связанных с одновременным поступлением в интерфейс нескольких сообщений от разных источников, каждый интерфейс должен содержать буферный регистр поступающих сообщений. Среднее время задержки сообщения в этом регистре примерно пропорционально расстоянию между источником и приемником. При отсутствии конфликтов среднее время задержки $\overline{T}$ характеризуется выражением

$$\overline{T} = \rho_{\Sigma} + \overline{\rho} \times T_{\text{буф}},$$

где ρ_{Σ} – задержка передачи, $T_{\text{буф}}$ – задержка в буфере на запоминание и проверку шапки сообщения.

Если сообщение буферизуется в каждом промежуточном узле, то средняя задержка $\overline{T}'$ составляет

$$\overline{T}' = \overline{\rho} \times \left(\rho_{\Sigma} + T_{\text{буф}}\right).$$

Дать точную оценку задержки в промежуточных узлах трудно, однако видно, что время задержки пропорционально количеству промежуточных пересылок, т.к. среднее количество конфликтующих сообщений пропорционально количеству процессоров, через которые они проходят [1, 3].

Для изучения задержек в сетях двойного цикла принимаются, что возникающие сообщения составляют пуассоновский поток, а длины сообщений, времена задержки и обработки заголовков сообщений распределены экспоненциально. Для каждого звена (связи между соседними узлами) создается своя очередь с достаточно большим буфером во избежание переполнения.

Для анализа скорости пересылки сообщений приняты следующие предположения: сообщения образуют пуассоновский поток со средней скоростью γ; все звенья равнозначны; КС работает на пределе пропускной способности.

Предполагается также, что для передачи сообщений используются кратчайшие маршруты между узлами. Если таких маршрутов несколько, то сообщения передаются поровну по каждому из маршрутов. Среднее число переходов $\Pi_{\text{ср}}$ вычисляется как среднее число кратчайших маршрутов от

одного узла ко всем другим. Общий объем пересылок характеризуется соотношением

$$\rho_{f_{\max}} + \rho_{b_{\max}} = \Pi_{\text{ср}} \times \gamma / N,$$

где $\rho_{b_{\max}}$ – вклад в пересылку сообщений переходов назад, а $\rho_{f_{\max}}$ – вперед. Посредством перераспределения сообщений по нескольким кратчайшим маршрутам равной длины возможен баланс передачи сообщений в обоих направлениях, при этом

$$\rho_{f_{\max}} = \rho_{b_{\max}} = \frac{1}{2} \cdot \Pi_{\text{ср}} \times \gamma / N.$$

Если приравнять объем пересылок к пропускной способности системы, можно получить:

$$\left(\rho_{f_{\max}} + \rho_{b_{\max}}\right) N = 2N\mu,$$

где μ – пропускная способность звена. Посредством арифметических преобразований и подстановки в результирующую формулу выражения для $\Pi_{\text{ср}}$, имеем:

$$\gamma_{ST} = \frac{2N(N-1)\mu}{\frac{1}{2}\left[\left(N - \left\lfloor \frac{N}{s+1} \right\rfloor s\left(N - \left\lfloor \frac{N}{s+1} \right\rfloor s - 1\right) + \left\lfloor \frac{N}{s+1} \right\rfloor s\left(\left\lfloor \frac{N}{s+1} \right\rfloor + s\right)\right)\right]},$$

где γ_{ST} максимально, если средняя длина пути сообщения минимальна. Максимум γ_{ST} достигается при $s = \left\lfloor \sqrt{N} \right\rfloor$. Для двунаправленного цикла эта величина составляет

$$\gamma_{ST} = \frac{2N(N-1)}{\left\lfloor \frac{N^2}{4} \right\rfloor},$$

а для двойного цикла при $s = 2$

$$\gamma_{ST} = \frac{4N(N-1)\mu}{\left[\left(N - \left\lfloor \frac{N}{3} \right\rfloor 2\left(N - \left\lfloor \frac{N}{3} \right\rfloor 2 - 1\right) + \left\lfloor \frac{N}{3} \right\rfloor 2\left(\left\lfloor \frac{N}{3} \right\rfloor + 2\right)\right)\right]}.$$

Так как γ_{ST} обратно пропорционально $\Pi_{\text{ср}}$ и в рассматриваемой топологии $\Pi_{\text{ср}}$ почти минимально, то производительность для данной топологии оптимальна.

В особом случае, если N – полный квадрат, то $d = 2\left(\sqrt{N} - 1\right)$, средняя длина пути сообщения будет равна $\bar{\rho} = \dfrac{N}{\left(\sqrt{N} + 1\right)}$, откуда $\gamma_{ST} = \dfrac{2\left(N - 1\right)}{\left(\sqrt{N} - 1\right)}\mu$.

При выходе элемента сети из строя максимальная и средняя длина пути сообщения увеличивается, однако и в этом случае параметры оптимального цикла лучше параметров других рассматриваемых структур.

В оптимальной сетевой архитектуре используется алгоритм распределенной рассылки, в соответствии с которым сообщение передается по назначению, обходя неисправные узлы и связи, на основе лишь локальной информации при выборе маршрута. Локальные соседи – это узлы, связанные друг с другом короткими прямыми связями, а отдаленные соседи – это узлы, связанные длинными обратными связями. Кратчайший путь между источником и приемником может состоять как из длинных, так и коротких переходов. Локальный путь состоит только из коротких, а путь высокого уровня – только из длинных переходов [3].

При отсутствии неисправных узлов узел, получив сообщение, читает адрес приемника D из заголовка и пытается найти его среди адресов узлов впереди по основному циклу, содержащихся в его таблице. Если выяснилось, что D – собственный адрес узла, то сообщение пришло по назначению. Если D имеется в просмотренной части таблицы локальных адресов, то сообщение направляется по этому адресу через короткие связи. В остальных случаях сообщение передается отдаленному соседу.

Таблица локальных адресов составляется и модифицируется при помощи распределенного алгоритма. Периодически, но асинхронно каждый узел передает своим (локальному и отдаленному) соседям свою таблицу локальных адресов. На основе таблицы, полученной от соседнего узла, узел модифицирует свою собственную таблицу.

При выходе из строя узла или связи (без ограничения общности – узла) в алгоритме рассылки рассматриваются случаи, когда неисправность возникла на пути высокого уровня или на локальном пути, В первом случае передающий узел, обнаружив неисправность отдаленного соседа (в соответствующей позиции таблицы адресов проставлен признак ∞), пересылает сообщение локальному соседу, который, не найдя адресата в своей таблице, перешлет сообщение по длинной связи, и неисправный узел

будет обойден. Сети двойного цикла обладают более высокой надежностью и устойчивостью к неисправностям КС, чем сети простого цикла, вследствие наличия нескольких путей. Сети двойного цикла обеспечивают несколько альтернативных путей между двумя узлами. Например, для сети из 20 узлов наличие неисправного процессора влечет за собой увеличение диаметра от 9 до 18 в двунаправленном цикле, от 7 до 8 в циклах при $s = 2$ и $s = \lfloor \sqrt{N} \rfloor$.

Влияние неисправных узлов на среднее число пересылок сообщения в сетях двойных циклов относительно мало.

Мерой устойчивости к сбоям является надежность пересылки сообщения между двумя наиболее удаленными друг от друга (противоположными) узлами. Интуитивно, она пропорциональна количеству различных маршрутов между этими узлами и обратно пропорциональна диаметру сети.

При $s = \lfloor \sqrt{N} \rfloor$ диаметр минимизируется, а число альтернативных маршрутов максимизируется. Каждый из кратчайших маршрутов состоит из $\rho_{b_{\max}} = \left\lfloor \frac{N}{s+1} \right\rfloor$ длинных пересыпок назад и $s-1$ коротких пересылок вперед. Число альтернативных маршрутов между противоположными узлами составляет

$$N_R - 1 = \begin{pmatrix} \rho_{b_{\max}} + s - 1 \\ s - 1 \end{pmatrix},$$

откуда

$$N_R = 1 + \begin{pmatrix} \rho_{b_{\max}} + s - 1 \\ s - 1 \end{pmatrix}.$$

При $s = 2$ получаем $N_R = 2 + \lfloor N/3 \rfloor$, а при $s = \lfloor \sqrt{N} \rfloor$ –

$$N_R = 1 + \begin{pmatrix} \left\lfloor \frac{N}{s+1} \right\rfloor + s - 1 \\ s - 1 \end{pmatrix}.$$

Максимум N_R достигается при значениях $s = \lfloor \sqrt{N} \rfloor$.

В сети с двумя неисправными узлами особый случай составляют изолированные узлы, когда оба неисправных узла являются соседями такого узла, выключая его из сети. Вероятность появления отрезанного узла (если известно, что два узла неисправны) составляет

$$N_{DN} = \frac{2N}{N(N-1)} = \frac{2}{N-1}$$

и уменьшается при увеличении размера сети. В случае отрезанного узла в сети оптимального двойного цикла выключается $1/N$ всех возможных маршрутов в сети, в то время как в неоптимальных сетях двойного никла при наличии двух неисправных узлов перестает действовать до половины всех возможных связей, откуда следует, что вероятность разрыва каждой связи из-за двух неисправных узлов в оптимальной циклической сети на два порядка ниже, чем в неоптимальной.

Литература:

1. *Андреев А.М., Можаров Г.П., Сюзев В.В.* Многопроцессорных вычислительные системы: теоретический анализ, математические модели и применение: учеб. пособие / – М.: Изд-во МГТУ им. Н. Э. Баумана, 2011. – 334 с.

2. *Андреев А.М., Можаров Г.П.* Анализ основных параметров компьютерных систем методом спектральной теории графов. Наука и образование: электронное научно-техническое издание, 2011, 10 [электронный ресурс] http://technomag.edu.ru/doc/ 232774.html (77-30569/232774).

3. *Андреев А.М., Березкин Д.В., Можаров Г.П., Свирин Ил.С.* Математическое моделирование надежности компьютерных систем и сетей. Вестник МГТУ им. Баумана. Серия «Приборостроение». Специальный выпуск «Моделирование и идентификация компьютерных систем и сетей». Изд. МГТУ им. Н.Э. Баумана, 2012. – с. 3-46.

УДК 004.272.26

РЕШЕНИЕ ПАРАЛЛЕЛЬНЫХ ЗАДАЧ НА МНОГОПРОЦЕССОРНЫХ СТРУКТУРАХ

Л.В. Мусина, Ю.М. Руденко

В связи с появлением вычислительных систем актуальнейшей проблемой стала разработка параллельных алгоритмов и программ, которые

являются основными компонентами этих систем. Решение этих проблем поставило перед разработчиками много новых проблем. В частности, потребовалось создание новых методов, позволяющих вычислять требуемые результаты, получая промежуточные данные одновременно на нескольких вычислительных модулях. Второй важной проблемой является создание на базе существующих математических методов параллельных алгоритмов решения задач на вычислительных системах, учитывающих их архитектуру. Конечно, список проблем на этом не заканчивается и можно продолжать этот список ещё достаточно долго.

Темой исследования данной статьи являются способы представления параллельных алгоритмов, также методы отображения этих алгоритмов на структурах вычислительных систем.

На последовательном компьютере (в основе его архитектуры и организации процесса функционирования лежит принцип последовательного выполнения отдельных действий) время реализации любого алгоритма пропорционально, главным образом, числу выполняемых операций и почти не зависит от того, как внутренне устроен сам алгоритм. На вычислительных системах параллельной архитектуры время решения задач решающим образом зависит от того, какова внутренняя структура алгоритма, и в каком порядке выполняются его операции. Возможность ускоренной реализации алгоритма на параллельных системах достигается за счет того, что в них имеется достаточно большое число процессоров, которые могут параллельно выполнять операции алгоритма.

На любой вычислительной технике одновременно могут выполняться только независимые операции. В процессе реализации алгоритма во времени на любой вычислительной системе, последовательной или параллельной, операции этого алгоритма разделяются на группы. Все операции каждой группы независимы и выполняются одновременно, а сами группы реализуются во времени последовательно одна за другой. Это неявно порождает некоторую специальную форму представления алгоритма, называемую параллельной [3], в которой фиксируются как группы операций, так и их последовательность.

Параллельную форму алгоритма можно ввести и как эквивалентный математический объект, не зависящий от вычислительной системы. Входные данные фиксируются, а все операции алгоритма разделяются на группы,

называемые ярусами и обладающие следующими свойствами. Во-первых, в каждом ярусе находятся только независимые операции. Во-вторых, существует такая последовательная нумерация ярусов, что каждая операция из любого яруса использует в качестве аргументов либо результаты выполнения операций из ярусов с меньшими номерами, либо входные данные алгоритма. Число операций в ярусе принято называть шириной яруса, число ярусов в параллельной форме – высотой параллельной формы.

При одних и тех же значениях входных данных между математическими параллельными формами алгоритма и реализациями того же алгоритма на конкретных или гипотетических вычислительных системах с одним или несколькими процессорами существует взаимно однозначное соответствие. Если какая-то параллельная форма отражает реализацию алгоритма на некоторой вычислительной системе, то ширина яруса говорит о числе используемых в каждый момент времени независимых устройств, а высота – о времени реализации алгоритма.

Каждый алгоритм при фиксированных входных данных, в общем случае, имеет много параллельных форм. Наибольший интерес представляют параллельные формы с минимальной высотой, так как именно они показывают, насколько быстро может быть реализован алгоритм, по крайней мере, теоретически.

Достигаемое ускорение при использовании параллельной формы, в среднем, пропорционально числу операций, выполняемых в каждый момент времени. Если оно равно общему числу имеющихся процессоров, то данный алгоритм при заданных входных данных реализуется на используемой вычислительной системе эффективно. В этом случае, возможный ресурс ускорения использован полностью, и более быстрого счета достичь на выбранной системе невозможно ни практически, ни теоретически.

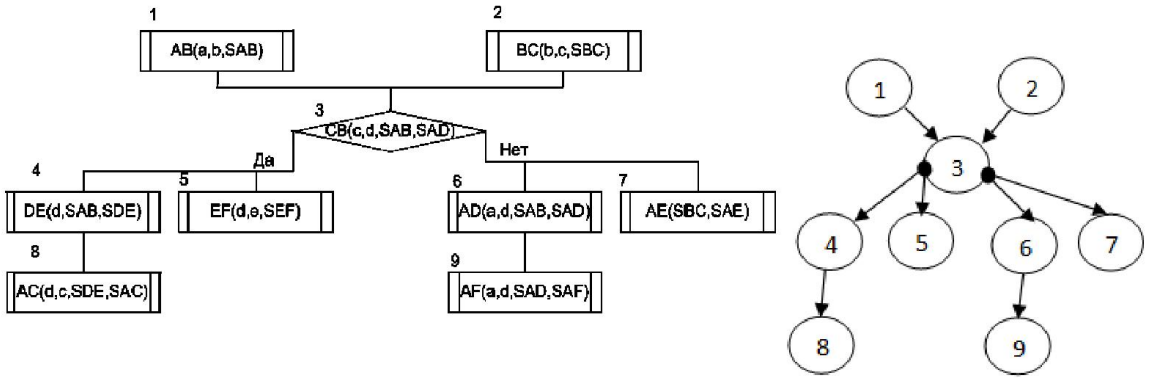
Но если ускорение значительно меньше числа устройств? Предположим, оно меньше средней ширины ярусов параллельной формы реализуемого алгоритма. Это значит, что не очень удачно выбрана схема реализации алгоритма. Изменив ее, можно попытаться более полно использовать имеющийся потенциал параллелизма в алгоритме. Если же ускорение равно средней ширине ярусов параллельной формы, то весь потенциал параллелизма в алгоритме выбран полностью. В таком случае, никаким изменением схемы счета нельзя добиться большего ускорения.

Наконец, вполне возможно, что даже при использовании всех процессоров реальное ускорение не соответствует ожидаемому. Это означает, что при реализации алгоритма приходится осуществлять какие-то передачи данных, требующие длительного времени. Для того чтобы понять причины замедления, необходимо тщательно изучить структуру алгоритма и/или вычислительной системы.

Таким образом, как только возникает необходимость решать какие-то вопросы, связанные с анализом ускорения при решении задачи на вычислительной системе параллельной архитектуры, обязательно требуется получить сведения относительно структуры алгоритма на уровне связей между отдельными операциями. Более того, чаще всего эти сведения приходится сопоставлять со сведениями об архитектуре вычислительной системы.

Выше предполагалось, что все процессоры вычислительной системы работают в синхронном режиме и выполняют любую операцию за одно и то же время. Такое предположение не принципиально по сути, оно позволяет более отчетливо показать смысл введения параллельной формы как очень важных объектов, описывающих структурные свойства и характеристики алгоритмов. Эти объекты удобно задавать ориентированным графом.

Будем считать операции алгоритма вершинами графа. На множестве вершин-операций естественным образом определен частичный порядок, показывающий, какие операции вслед за какими могут выполняться. Этот же порядок позволяет задать дуги графа. Таким образом, будет построен ориентированный ациклический граф, называемый граф-схемой алгоритма [1] (рисунок 1).



а) *б)*

Рисунок 1. ПФ алгоритма *(а)* и соответствующая ей граф-схема *(б)*

Граф-схема алгоритма играет важную роль в изучении параллельных свойств самого алгоритма. Во-первых, она устроена значительно проще самого алгоритма, так как не связана ни с какой атрибутикой, сопровождающей описание алгоритма. Ее можно рассматривать как классический математический объект, и ничто не мешает исследовать ее чисто математическими методами. Во-вторых, по граф-схеме алгоритма очень просто строить параллельные формы.

Таким образом, граф-схема алгоритма является объектом, который может стать основой создания математического аппарата, предназначенного для изучения информационной структуры алгоритмов. Но чтобы этот объект превратился в действенный инструмент, он должен быть представлен в форме, удобной для проведения как теоретических, так и практических расчетов.

Основным инструментом анализа граф-схем алгоритмов служит матрица следования [1], а также различные её модификации. Это транспонированная матрица смежности. Ее использование вместо матриц смежности объясняется удобством размещения и анализа граф-схем.

Если в граф-схеме алгоритма существует связь между операторами α, β, и α – исполнительный блок [1], то в графе существует дуга, исходящая из вершины α и входящая в вершину β. Эту связь будем называть информационной.

Если в граф-схеме алгоритма существует связь между операторами α, β, и α – логический блок [1], то в графе существует дуга, исходящая из вершины α и входящая в вершину β. Эту связь будем называть логической или связью по управлению.

Теперь можно уточнить определение матрицы следования. Это квадратная матрица, количество строк и столбцов в ней совпадает с количеством вершин граф-схемы, i-ой вершине графа ставятся в соответствие i-ые столбец и строка этой матрицы.

Если существует связь по управлению между вершинами j и i, то элемент матрицы равен (i, j) = j.n, если существует информационная связь, то элемент матрицы равен (i, j) = 1. Остальные элементы матрицы следования равны 0 (рисунок 2).

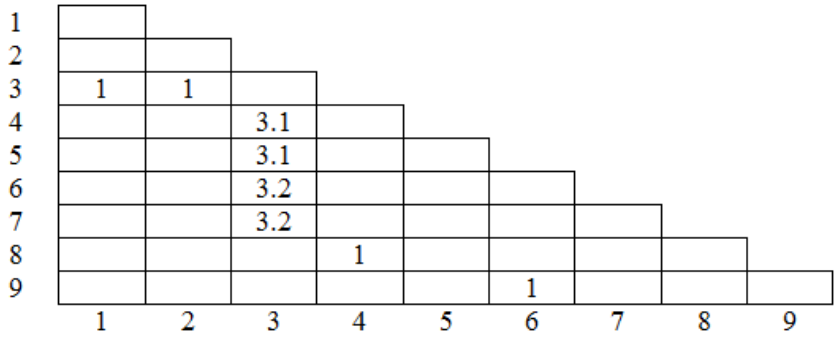


Рисунок 2. Матрица следования для граф-схемы алгоритма, представленной на рис.1*б*

Для реализации различных стратегий выполнения программных модулей параллельно на вычислительной системе, важную роль играют логически несовместимые операторы, поэтому важно создать методы, с помощью которых можно было бы манипулировать матрицами следования, содержащими логические операторы. Среди множества логических операторов выделим логически несовместимые операторы.

Если операторы, принадлежащие различным логическим ветвям, могут выполняться в какой-то из этих ветвей при однократном выполнении алгоритма, то эти операторы называются логически несовместимыми.

Последовательное программирование предполагает выполнение команд одна за другой. Но в программе встречаются команды, независимые по управлению или данным, поэтому их можно выполнять параллельно. Зависимость по управлению, которые проявляются как переходы по условию, представляют главное препятствие параллельному выполнению потому, что эта зависимость должна быть определена прежде, чем будут выполнены все последующие команды.

Для оценки возможности выполнения программных модулей параллельно, в программе, содержащей логические операторы, возможно несколько вариантов построения параллельных вычислений в зависимости от внешних факторов. Будем считать, что при решении параллельной задачи в распоряжении пользователя имеется количество процессоров, которое больше или равно требуемому. В этом случае, можно считать условно, что в алгоритме нет логических путей, и в вычислительную систему загружаются все программные модули, которые возможно потребуются при решении данной программы. Такой подход, за счёт использования дополнительного оборудования, может существенно сократить время решения поставленной

задачи. Эта же тактика может быть оправдана при непрерывном многократном решении задачи, если исходные данные каждый раз меняются и вероятность использования различных логических операторов достаточно высока.

Второй способ заключается в том, что вначале, как в первом случае, загружаются все программные модули, но затем по мере выполнения программы и прохождения логических операторов незадействованные логические ветви отключаются, а освободившиеся вычислительные модули поступают в распоряжение планировщика заданий.

Третий способ – наиболее затратный по времени, заключается в том, что первоначально из алгоритма исключаются все логически несовместимые операторы. Составляется план решения усечённой задачи. Во время решения такой задачи все исключённые ранее программные модули по мере необходимости подключаются планировщиком заданий.

Для реализации этих трёх способов необходимо иметь методы преобразования различных матриц, описывающих граф-схемы. Одним из наиболее сложных является метод построения матриц логической несовместимости операторов.

Решение поставленной задачи заключается, в основном, в нахождении множеств логически несовместимых операторов. Имея эти множества, можно, исключая элементы этих множеств из множества операторов, представляющих рассматриваемую граф-схему, осуществить второй или третий способы распределения операторов по вычислительным модулям. Для определения возможности распараллеливания операторов необходимо также произвести анализ независимости операторов по данным и по управлению [1].

При исследовании граф-схем алгоритмов одними из основных характеристик являются ранние и поздние сроки окончания выполнения операторов. Имея эти величины, можно построить планы выполнения операторов с учётом распределения операторов по вычислительным модулям. Необходимо подчеркнуть, что на основе граф-схемы алгоритма, можно определить:

1. частичную упорядоченность выполнения алгоритма;

2. веса операторов p_j, $j = 1, \ldots, m$ (обычно – времена выполнения процедур);

3. началом отсчёта времени решения задачи является начало выполнения операторов, являющихся входами в алгоритм;

4. путь максимальной длины в граф-схеме (минимальное время, за которое может быть решена данная задача).

Диаграммы выполнения операторов для ранних и поздних сроков – удобный способ наглядного представления многопроцессорной обработки [1]. Структура граф-схемы, при отображении ранних или поздних сроков окончания выполнения операторов в виде диаграмм, порождает возможные способы распределения операторов по вычислительным модулям (рисунок 3).

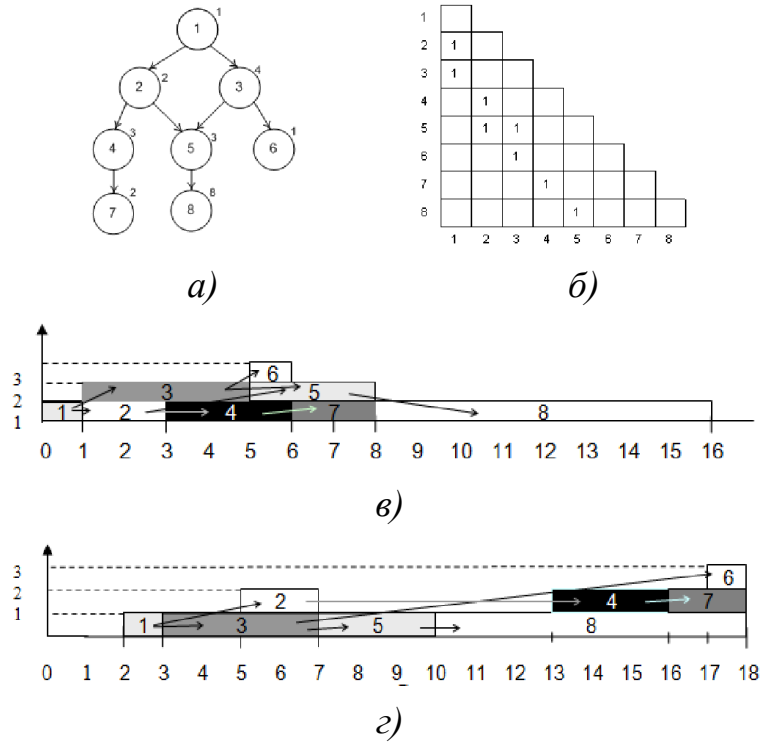


Рисунок 3. Пример граф-схемы *(а)*, ее матрицы следования *(б)*, для которой определены ранние *(в)* и поздние *(г)* сроки окончания выполнения операторов

Каждая строка (ветвь) диаграммы определяет множество операторов, которое целесообразно разместить на одном вычислительном модуле. Если в диаграмме имеется n строк, то целесообразно выбрать n вычислительных модулей для решения задачи, соответствующей данной диаграмме. Для такого способа планирования в случае обеспечения минимального времени решения задачи целесообразно использовать ранние сроки окончания выполнения операторов. Для обеспечения заданного времени решения задачи, превышающего время прохождения пути максимальной длины, необходимо использовать поздние сроки окончания выполнения операторов.

Литература:

1. *Руденко Ю.М., Волкова Е.А.* Вычислительные системы. Учебное пособие - М.: Изд-во МГТУ им. Н.Э. Баумана, 2010. - 211 с.

2. *Корнеев В. В.* Вычислительные системы. - М.: Гелиос АРВ,2004. - 512 с.: ил.

3. *Воеводин В.В.* Вычислительная математика и структура алгоритмов. - М.: Изд-во МГУ, 2006. - 112 с.

4. *Хорошевский В.Г.* Архитектура вычислительных систем. - М.: Изд-во МГТУ им. Баумана, Москва, 2005. - 511 с.: ил.

УДК 004.3 +519.6

ОЦЕНКА ЭФФЕКТИВНОСТИ РАЗЛИЧНЫХ СПОСОБОВ СНИЖЕНИЯ ВРЕМЕНИ ВЫПОЛНЕНИЯ ОПЕРАЦИЙ НАД УЛЬТРАГРАФАМИ

А.Е. Павлов, Г.С. Иванова

В настоящее время аппарат ультраграфов широко применяется для решения различных задач в системах автоматизированного проектирования. Среди них выделяют такие задачи, как: потоковый анализ и синтез, структурная оптимизация, комбинаторный анализ, синтез структур высокого уровня сложности.

Решение перечисленных выше задач имеет высокую вычислительную сложность. Одним из наиболее эффективных методов снижения последней является уменьшение времени выполнения проектных процедур за счёт сокращения времени выполнения основных операций, производимых над ультраграфами.

К таким операциям можно отнести: добавление или удаление вершины, добавление или удаление ребра, стягивание двух ребер, подразбиение ребра, удаление вершины из образов и прообразов инцидентных ей ребер, удаление ребра из образов и прообразов инцидентных ему вершин, формирование части графа и т.д.

Вычислительную сложность операции теоретически можно определить как количество операций сравнения и копирования в зависимости от мощности обрабатываемых множеств. При этом наибольшей вычислительной сложностью формирования множества, как правило, обладают процедуры формирования образов и прообразов относительно предикатов смежности вершин и ребер.

Рассмотрим в качестве примера операцию добавления вершины в ультраграф. За основу возьмем формально-содержательное описание данной процедуры из [1]:

Обозначение операции: $H_{U1}(X_1,U_1) = H_U(X,U) + x_k(U_k^+,U_k^-)$,

где $U_k^+ = \Gamma_1 x_k$ – множество ребер инцидентных вершине x_k;

$U_k^- = \Gamma_2 x_k$ – множество ребер, которым инцидентна вершина x_k.

Порядок выполнения операции:

1. Создаем множества X_1, $\Gamma_1 X_1$ и $\Gamma_2 X_1$.

$X_1 = X \bullet x_k$; $\Gamma_1 X_1 = \Gamma_1 X \bullet \Gamma_1 x_k$; $\Gamma_2 X_1 = \Gamma_2 X \bullet \Gamma_2 x_k$.

2. Копируем множество U под именем U_1.

$U_1 = U$.

3. Определяем множество образов ребер $\Gamma_2 U_1$, занося в него $\Gamma_2 u_j$ из множества $\Gamma_2 U$, если ребро u_j не принадлежит U_k^-, и добавляя вершину x_k в образы тех ребер, которым она инцидентна.

$$\Gamma_2 U_1 = \left\{ \Gamma_2 u_j : u_j \notin \Gamma_2 x_k \vee \left\{ \Gamma_2 u_j \bullet x_k \right\} : u_j \in \Gamma_2 x_k \,/\, u_j \in U_1, \Gamma_2 u_j \in \Gamma_2 U \right\}.$$

4. Формируем множество прообразов ребер $\Gamma_1 U_1$, занося в него $\Gamma_1 u_j$ из множества $\Gamma_1 U$, если ребро u_j не принадлежит U_k^+, и добавляя вершину x_k в образы тех ребер, которые ей инцидентны.

$$\Gamma_1 U_1 = \left\{ \Gamma_1 u_j : u_j \notin \Gamma_1 x_k \vee \left\{ \Gamma_1 u_j \bullet x_k \right\} : u_j \in \Gamma_1 x_k \,/\, u_j \in U_1, \Gamma_1 u_j \in \Gamma_1 U \right\}.$$

5. Создаем множество образов $F_1 X_1$ вершин относительно предиката смежности F_1, занося $F_1 x_i$ из $F_1 X$, если вершина x_k не инцидентна ребрам, которые принадлежат образу $\Gamma_1 x_i$ вершины x_i, и добавляя в $F_1 x_i$ вершину x_k в противном случае. Определяем вершины смежные добавляемой и заносим их в $F_1 X_1$.

$$F_1X_1 = \{\{F_1x_i \in F_1X : \Gamma_1x_i \cap \Gamma_2x_k = \varnothing \vee F_1x_i \bullet x_k : \Gamma_1x_i \cap \Gamma_2x_k \neq \varnothing / x_i \in X,$$
$$\Gamma_1x_i \in \Gamma_1X\} \bullet F_1x_k\},$$

где $F_1x_k = \bigcup_{u_j \in \Gamma_1x_k} \Gamma_2u_j, \Gamma_2u_j \in \Gamma_2U$.

6. Формируем множество прообразов $F_1^{-1}X_1$, занося $F_1^{-1}x_i$ из $F_1^{-1}X$, если вершине x_k не инцидентны ребра прообраза Γ_2x_i, и добавляя в $F_1^{-1}x_i$ вершину x_k в противном случае. Определяем вершины прообраза $F_1^{-1}x_k$ и заносим его в множество $F_1^{-1}X_1$.

$$F_1^{-1}X_1 = \{\{F_1^{-1}x_i \in F_1^{-1}X : \Gamma_2x_i \cap \Gamma_1x_k = \varnothing \vee F_1^{-1}x_i \bullet x_k : \Gamma_2x_i \cap \Gamma_1x_k \neq \varnothing / x_i \in X,$$
$$\Gamma_2x_i \in \Gamma_2X\} \bullet F_1^{-1}x_k\},$$

где $F_1^{-1}x_k = \bigcup_{u_j \in \Gamma_2x_k} \Gamma_1u_j, \Gamma_1u_j \in \Gamma_1U$.

7. Определяем множество образов ребер F_2U_1, копируя в него F_2u_j из F_2U, если ребро u_j не принадлежит U_k^-. В противном случае добавляем в F_2u_j не принадлежащие ему ребра множества U_k^+.

$$F_2U_1 = \{F_2u_j : u_j \notin \Gamma_2x_k \vee F_2u_j \cup U_k^+ : u_j \in \Gamma_2x_k / u_j \in U_1, F_2u_j \in F_2U\}.$$

8. Формируем множество прообразов ребер $F_2^{-1}U_1$, копируя в него $F_2^{-1}u_j$ из $F_2^{-1}U$, если ребро u_j не принадлежит U_k^+. В противном случае добавляем в $F_2^{-1}u_j$ не принадлежащие ему ребра множества U_k^-.

$$F_2^{-1}U_1 = \{F_2^{-1}u_j : u_j \notin \Gamma_1x_k \vee F_2^{-1}u_j \cup U_k^- : u_j \in \Gamma_1x_k / u_j \in U_1, F_2^{-1}u_j \in F_2^{-1}U\}.$$

Можно показать, что асимптотическая оценка вычислительной сложности для данной операции над ультраграфом равна:

- в худшем $O(m^3)$ при $m > n$, если $|U_k^-|, |U_k^+|u|F_2u_j|$ или $|U_k^-|, |U_k^+|u|F_2^{-1}u_j|$ ограничены величиной m, и $O(m^2 \cdot n)$ при $n > m$, если $|U_k^-|u|\Gamma_1x_i|$ или $|U_k^+|u|\Gamma_2x_i|$ ограничены величиной m.

- в лучшем $O(m)$ при $m > n$ или $O(n)$ при $n > m$, если мощности образов и прообразов вершин и ребер ограничены константой.

Цикл решения задачи на графах обычно содержит следующие этапы: выбор метода решения (или разработка нового метода), построение алгоритма, реализация алгоритма. Очевидно, что один и тот же алгоритм можно реализовать несколькими способами. И в каждом случае будет свой

показатель быстродействия. Можно также показать, что время выполнения алгоритма существенно зависит от выбора структур данных, с которыми он оперирует [2].

Таблица 1.

Оценка оптимизирующих преобразований

Вид оптимизирующего преобразования	**Описание оптимизирующего преобразования**	**Вклад в снижение вычислительной сложности**
1. Разбиение цикла на блоки	Разделение пространства итерирования исходного цикла на небольшие блоки меньшего размера с целью размещения в кэше этих блоков для их неоднократного использования	Низкий, так как часто выполняется встроенным компилятором программной среды.
2. Замена операции копирования на присваивание указателя	В участках программы, где множеству присваиваются все элементы другого множества, цикл копирования элементов заменяется присваиванием указателя на структуру, содержащую в себе элементы исходного множества.	Высокий, при большой мощности множества вершин и ребер.
3. Исключение лишних проверок при вычислении выражения $\Gamma_2 x_i \cap \Gamma_1 x_k \neq \varnothing$	Замена полного перебора вершин для проверки пересечения двух множеств на итерационное попарное пересечение элементов двух множеств с последующим сдвигом элемента множества с меньшим номером до первого совпадения элементов.	Существенный, при большой мощности множества вершин и множества ребер.
4. Исключение лишних проверок при вычислении выражения $F_2 u_j \cup U_k^+$	Замена полного перебора вершин для объединения двух множеств на итерационное копирование пары элементов - элемент множества $F_2 u_j$ и элемент множества U_k^+, с последующим сдвигом элементов каждого из множеств.	Существенный, при большой мощности множества вершин и множества ребер.
5. Выбор структуры представления данных множеств	Применение структуры представления данных, позволяющих более эффективно выполнять добавление и удаление элементов во множества образов и прообразов вершин и ребер, а так же выполнять операции объединения, пересечения и дополнения двух множеств.	Может быть существенным при большой мощности множества вершин и множества ребер.

В таблице 1 показано несколько видов оптимизирующих преобразований (в том числе над упорядоченными множествами [4]) и проанализирован их вклад в снижение вычислительной сложности операции добавления вершины в ультраграф.

Для экспериментальной оценки вычислительной сложности операции добавления вершины в ультраграф были выбраны структура односвязного списка и структура двусвязного списка, при этом анализировались оптимизирующие преобразования с 2 по 4. Измерения проводились для ультраграфа с мощностями множества вершин и множества ребер равными 1000. Количество измерений для оценки времени выполнения операции в каждом случае составляло 100 повторений, результаты измерений усреднялись. При этом среднеквадратическое отклонение не превышало 7-10 % от среднего значения.

На рисунке 1 показан вклад в снижение вычислительной сложности разных видов оптимизирующих преобразований для структур односвязного и комбинированного списков. Оценка произведена в процентах с использованием соотношения:

$$Выигрыш\ от\ оптимизации = \frac{t^{неопт} - t^{опт_i}}{t^{неопт}} \cdot 100\%,$$

где $t^{неопт}$ – время выполнения операции без применения оптимизирующих преобразований, $t^{опт_i}$ – время выполнения операции с применением i-го оптимизирующего преобразования.

Как видно из рисунка 1, наибольший эффект дает применение оптимизирующего преобразования 2-го вида, что можно объяснить существенной экономией памяти при использовании указателей и отказе от временных буферов большой емкости. Суммарный выигрыш от применения описанных выше оптимизирующих преобразований составил в ряде случаев от 55 до 90 % от исходного времени выполнения операции добавления вершины в ультраграф.

Эффективность и возможность применения использованных для оценки оптимизирующих преобразований напрямую зависит от выбранной структуры представления данных. Так, например, при векторном представлении ультраграфа невозможно применить преобразования 2-4. При этом для ультраграфов с большим числом связей между вершинами и ребрами, время выполнения операций достаточно близко по значению к

оптимизированным операциям, реализованных на списковых структурах данных. На рисунке 2 проиллюстрирована данная зависимость.

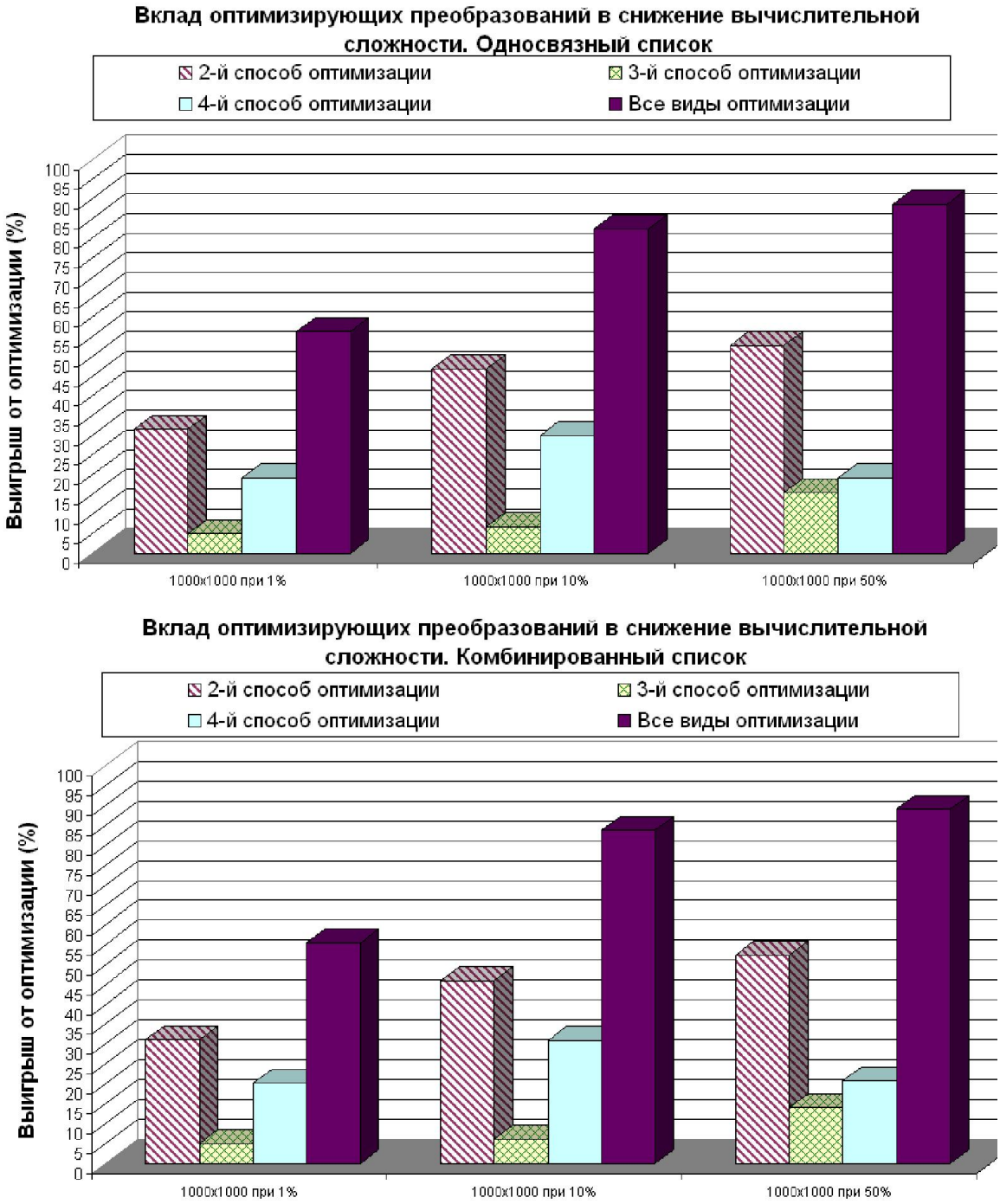


Рисунок 1. Вклад оптимизирующих преобразований в снижение вычислительной сложности операции добавления вершины в ультраграф

Из рисунка 2 видно, что при малой связности ультраграфа желательно применение списковых структур, так как они дают выигрыш по времени выполнения операции. При средней и высокой связности результаты близки

друг другу, хотя наименьшее значение наблюдается при использовании структуры комбинированного списка.

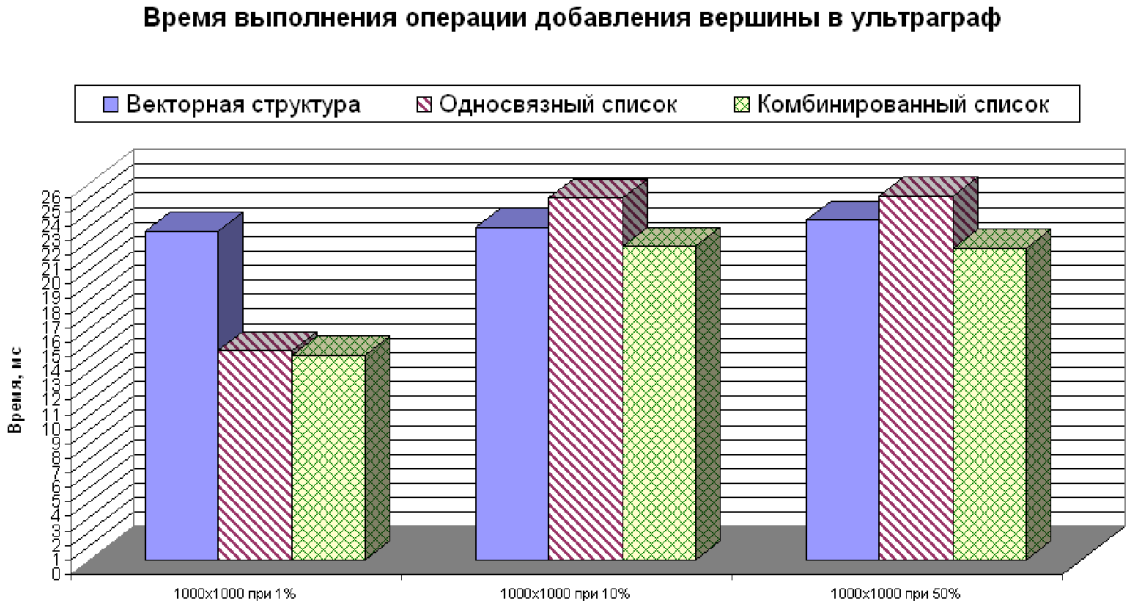


Рисунок 2. Оценка времени выполнения операции добавления вершины в ультраграф в зависимости от структуры представления данных

Из всего вышесказанного следует, что для максимального снижения времени выполнения операций над ультраграфами необходимо разрабатывать и применять такие структуры данных, которые позволят наиболее эффективно оперировать с элементами множеств образов и прообразов относительно вершин и ребер. При этом к операциям над ультраграфами для разработанных структур требуется найти и адаптировать под структуру такие оптимизирующие преобразования, которые могут внести существенный вклад в снижение вычислительной сложности этих операций.

Литература:

1. *Овчинников В.А.* Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем. - Наука и образование. Инженерное образование: Эл. науч. издание. – 2009. – № 10-12.

2. *Иванова Г.С., Пасечников К.А.* Макрогенерация описаний структур данных для системы проектирования алгоритмов решения задач структурного синтеза. - Современные информационные технологии. Сб.

трудов каф, посвященный 175-летию МГТУ им НЭ Баумана - М Эликс+, 2004 - с 69-73.

3. *Овчинников В.А.* Алгоритмизация комбинаторно-оптимизационных задач при проектировании ЭВМ и систем: учебник для вузов. - М.: Изд-во МГТУ им. Н. Э. Баумана, 2001. - 284 с.

4. *Овчинников В.А.* Операции над упорядоченными множествами. - Наука и образование. Инженерное образование: Эл. науч. издание. – 2011. – № 6.

УДК: 004.051

ОЦЕНКА ЭФФЕКТИВНОСТИ ОРГАНИЗАЦИИ МНОГОПОТОЧНЫХ ПРИЛОЖЕНИЙ

Ю. К. Петров, Г.С. Иванова

Начиная с 2004 года в связи с появлением технологии Hyper-threading Technology (HTT), представленной компанией Intel, резко возрастает интерес к разработке многопоточных приложений [1-2]. В основе указанной технологии лежало более рациональное использование ресурсов компьютера за счёт многопоточности, которая обеспечивалась разбивкой одного физического ядра на 2 логических. При использовании этой технологии по подсчётам компании Intel прирост производительности многопоточных приложений должен был составить 15-30% [3]. С появлением серийных многоядерных процессоров производительность многопоточных приложений еще возросла.

К достоинствам многопоточных приложений, помимо снижения времени работы, относят:

- хорошую стабильность времени работы;
- возможность уменьшения времени реакции приложения на команды пользователя за счет выполнения вычислений в фоновом режиме.

Однако многопоточные приложения также:

- предъявляют более высокие требования к аппаратуре компьютера;

- имеют более высокую сложность написания и отладки;
- предполагают использование специальных средств, обеспечивающих синхронизацию параллельных процессов, например мьютексов.

Настоящая работа посвящена оценке эффективности выполнения многопоточных приложений на многоядерных процессорах. Наличие такой оценки позволит более точно определить целесообразность создания более дорогостоящих многопоточных приложений в конкретных ситуациях.

Наибольший эффект многопоточности проявляется при создании фоновых процессов в приложении (особенно для ресурсоёмких фоновых вычислений), а также при распараллеливании сложных независимых вычислений. Оценку эффективности организации многопоточных приложений выполним на следующем примере.

Имеется некая файловая система, в которой есть всего лишь 12 типов файлов. Приложение должно найти все файлы, разделив их по типам, отсортировать результаты в алфавитном порядке и записать в файл. Решение усложняется тем, что пользователь может отказаться от поиска каких-либо типов файлов. При этом рассмотрим худший вариант: пользователь всё-таки выбирает все 12 типов файлов, то есть необходимо запустить 12 потоков или последовательно 12 универсальных функций поиска. Тесты будут проводиться на 2-х машинах: одна с 4мя ядрами (Intel Core 2 Quad Q8400 @2,66GHz), вторая с одним ядром (Intel Atom Z540 @1,86 GHz).

Предварительно оценивая результаты, можно сказать, что на системе с 4-мя ядрами должен однозначно выигрывать многопоточный алгоритм, на системе с одним ядром результаты будут близки, возможно, в некоторых ситуациях линейный алгоритм выиграет. Сложно заранее сказать, где будет более стабильно время работы, однако предположим, что время работы линейного алгоритма можно предсказать более или менее точно.

При первом тесте будем брать определённое число файлов и папок, запускать алгоритм 1 раз и засекать его время работы. Такое исследование является достаточно грубым, однако в дальнейшем для некоторых опытов мы проведём повторные тесты, чтобы определить среднее время работы и сравнить полученные нами результаты с теоретическими.

На рисунке 1 показаны результаты выполнения теста на многоядерном процессоре, по оси абсцисс отложено количество обрабатываемых файлов, по оси ординат – время в миллисекундах.

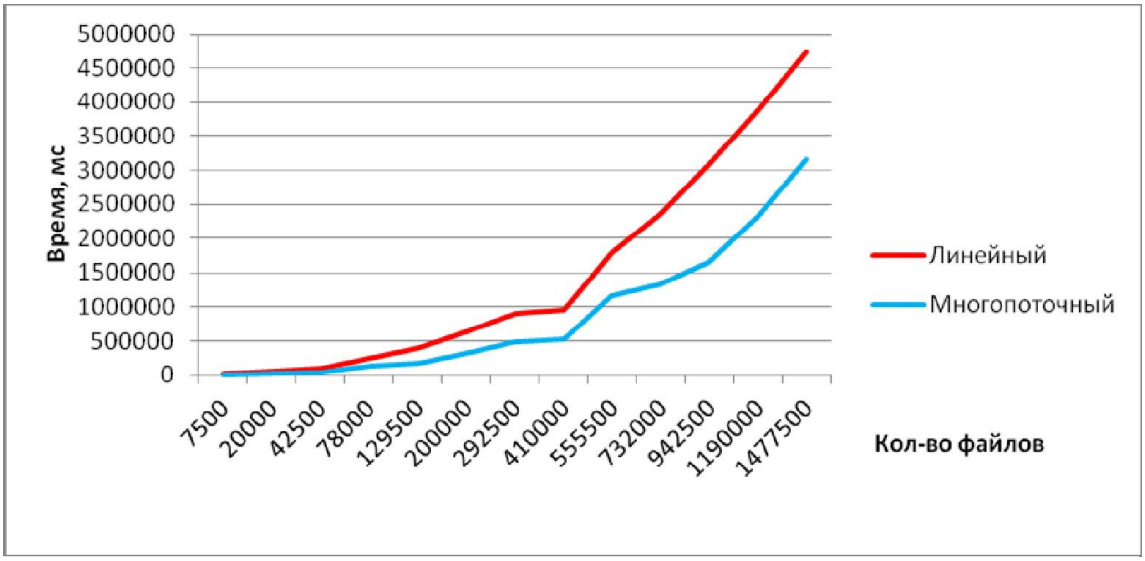


Рисунок 1. Результаты выполнения тестов производительности с линейным и многопоточными алгоритмами на 4-х ядерном процессоре

Как и ожидалось, на малых количествах файлов алгоритмы почти идентичны по скорости работы, при росте количества файлов наблюдаем нарастающий разрыв в производительности в пользу многопоточного алгоритма. Далее сравним время выполнения программ, реализующих те же алгоритмы, на одноядерной машине, из-за достаточно небольшой вычислительной мощности ограничим максимальное количество файлов на уровне 78000, результат представлен на рисунке 2.

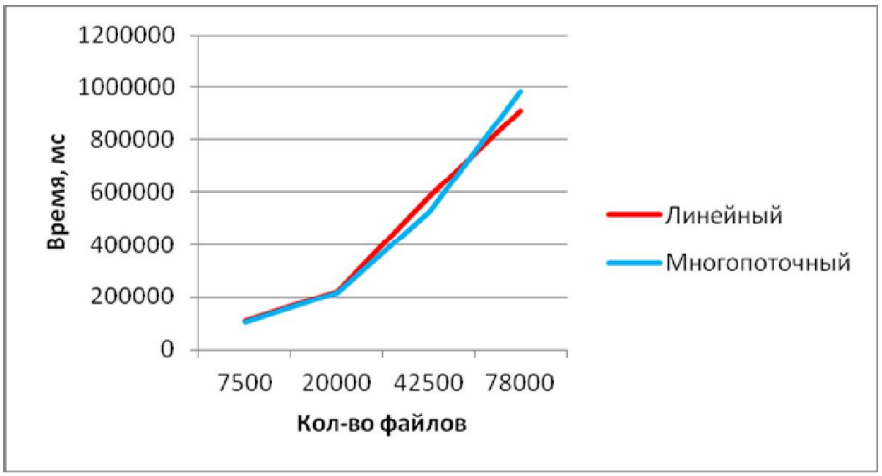


Рисунок 2. Результаты выполнения тестов производительности с линейным и многопоточным алгоритмами на одноядерном процессоре

Ожидаемо получили примерно совпадающие кривые, причем при большом количестве файлов линейный алгоритм даже оказался более производительным.

Чтобы оценить разброс значений, выполним статистическую обработку результатов для 5 измерений (тесты производились на многоядерной машине). Результаты показаны в таблицах 1 и 2, рисунки 3 и 4 иллюстрируют полученные значения.

Таблица 1. Стабильность времени выполнения многопоточного алгоритма

Количество файлов, шт	Время выполнения (запуск 1), мс	Время выполнения (запуск 2), мс	Время выполнения (запуск 3), мс	Время выполнения (запуск 4), мс	Время выполнения (запуск 5), мс	Среднее время работы, мс	Среднее абсолют. отклонение, мс	Среднее относит. отклонение, %
7500	8256	10910	11283	10212	10321	10196,4	776,16	7,6
20000	24501	23715	21689	21909	23371	23037	990,4	4,3
42500	47430	41521	44893	46873	48434	45830,2	2098,56	4,57

Таблица 2. Стабильность времени выполнения линейного алгоритма

Количество файлов, шт	Время выполнения (запуск 1), мс	Время выполнения (запуск 2), мс	Время выполнения (запуск 3), мс	Время выполнения (запуск 4), мс	Время выполнения (запуск 5), мс	Среднее время работы, мс	Среднее абсолют. отклонение, мс	Среднее относит. отклонение, %
7500	15221	21754	21961	21760	21933	20489,8	2143,52	10,4
20000	62586	46741	47088	47051	46886	50070,4	5006,24	10
42500	89511	90787	91773	90915	91693	90935,8	637,76	0,7

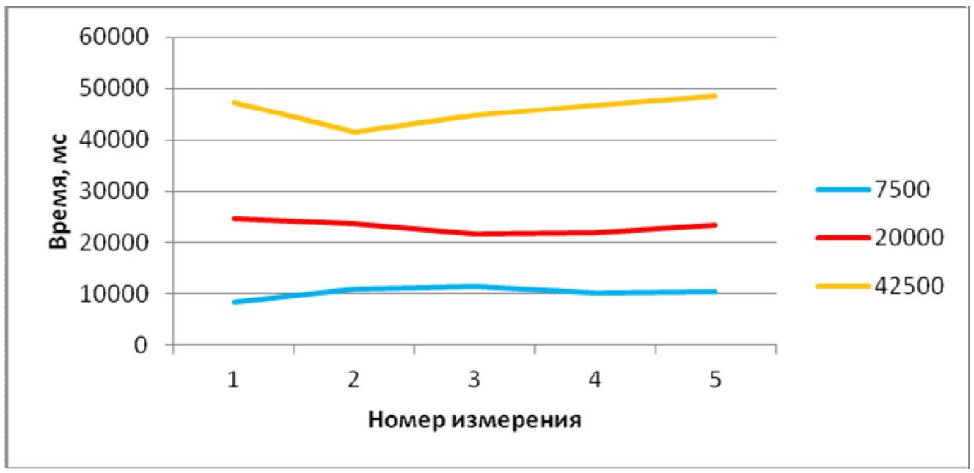


Рисунок 3. Тест стабильности времени выполнения многопоточного алгоритма

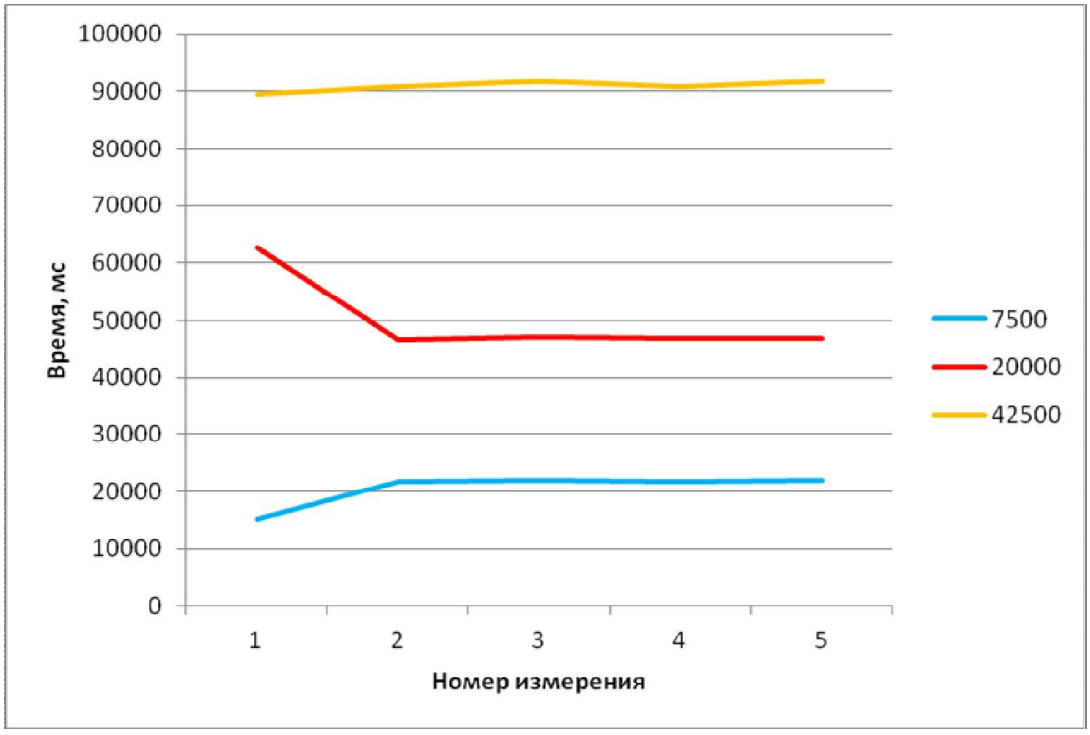


Рисунок 4. Тест стабильности времени выполнения линейного алгоритма

Таким образом, в процессе проведения эксперимента была доказана эффективность применения многопоточных приложений на современных многоядерных процессорах. Прирост производительности для рассмотренного теста составил примерно 40-50 %. Полученные результаты показывают стабильность как линейного, так и многопоточного алгоритмов. Сравнительно высокая стабильность многопоточного алгоритма может быть объяснена погрешностью, обусловленной малым числом измерений, однако данный вопрос требует отдельного, более глубокого изучения.

Однако эксперимент также подтвердил, что, несмотря на все достоинства многопоточного подхода, его не целесообразно использовать там, где невозможно обосновать необходимость использования многопоточности [4]. Так, на примере поставленной задачи, в случае, если пользователь хочет искать только 1 тип файлов, многопоточный алгоритм будет менее производительным.

Литература:

1. Введение в технологии параллельного программирования [Электронный ресурс]/Intel. – Режим доступа: http://software.intel.com/ru-ru/articles/writing-parallel-programs-a-multi-language-tutorial-introduction/, свободный. – Яз. рус.

2. Intel Hyper-Threading Technology [Электронный ресурс] .- Режим доступа: http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html, свободный. – Яз. англ.

3. Intel Technology Journal [Электронный журнал]. – Режим доступа: ftp://download.intel.com/technology/itj/2002/volume06issue01/vol6iss1_hyper_threading_technology.pdf, свободный. – Яз. англ.

4. *Макс Шлее*. Qt 4.5. Профессиональное программирование на C++. – СпБ.: Изд-во «БХВ-Петербург», 2010.

УДК 004.021:004.75

ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ УСКОРЕНИЯ МОДЕЛИРОВАНИЯ ФИЗИЧЕСКИХ ПРОЦЕССОВ НА ПРИМЕРЕ БРОУНОВСКОГО ДВИЖЕНИЯ ЧАСТИЦ И НЕЧЁТКАЯ МОДИФИКАЦИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ

В.Э. Подольский

Естественные и технические науки призваны решать довольно сложные задачи. Сложность современных задач заключается в том, что необходимы большие вычислительные мощности для их решения с нужной точностью. Персональные компьютеры (ПК) не могут предоставить достаточно ресурсов для решения таких задач. Так как суперкомпьютеры дорогостоящи и обладают ограниченной способностью к наращиванию, то при работе с ресурсоёмкими задачами применяются вычислительные кластеры и параллельное программирование (для кластеров используется стандарт MPI – Message Passing Interface). В этой работе приведён параллельный подход к моделированию поведения броуновских частиц, тайминги параллельного моделирования. Также показаны два способа уменьшения времени работы параллельного алгоритма.

Суть броуновского движения заключается в том, что любые маленькие частицы, находящиеся во взвешенном состоянии в жидкости или газе, движутся. Движение мелких частиц обусловлено толчками окружающих молекул жидкости или газа. Хотя молекулы жидкости (или газа) ударяют частицы со всех сторон, но всё же их удары не уравновешивают полностью друг друга. Вектор результирующего воздействия на взвешенную частицу со стороны среды может принимать произвольное направление, в результате чего частица начинает движение по направлению данного вектора. При перемене направления результирующей силы частица изменит направление движения. Таким образом, можно зарегистрировать беспорядочное (хаотическое) движение взвешенной частицы. Феномен броуновского движения является непосредственным обоснованием молекулярно-кинетической теории (МКТ), которую мы и используем для построения модели этого движения.

В физике встаёт задача моделирования броуновского движения (применяется для изучения физических и экономических процессов) некоторого числа частиц. Часто среда, в которой моделируется движение, содержит многие миллиарды частиц, поэтому для моделирования приходится применять высокопроизводительные системы. Описать взвешенные частицы и среду можно при помощи следующих макропараметров физической системы: абсолютная температура T (процесс считаем изотермическим), геометрические размеры площадки, на которой рассматривается броуновское движение (длина L и ширина W), молярные массы взвешенных частиц и частиц среды ($M_{взв}$ и $M_{среды}$ соответственно), плотности вещества из взвешенных частиц и среды ($\rho_{взв}$ и $\rho_{среды}$ соответственно). На основе заданных макропараметров можно вычислить все микропараметры частиц для моделирования движения. Также необходимо задать время моделирования $t_{мдл}$, по истечении которого процесс прекращается и выдаются результаты моделирования. Необходимые для инициализации среды и частиц в ней формулы (МКТ) приведены ниже.

$$r = \sqrt[3]{\frac{3 \cdot M}{4\pi\rho \cdot N_A}} \text{ – радиус шарообразной молекулы } (N_A \text{ – число Авогадро}), \tag{1}$$

где $d = 2r$ – её диаметр.

$$N = \frac{\rho \cdot S \cdot d \cdot N_A}{M} \text{ – число частиц вещества на площади } S = L \cdot W. \tag{2}$$

$$< v_{кв} >= \sqrt{\frac{i \cdot R \cdot T}{M}} \text{ – начальная среднеквадратичная скорость молекулы,} \tag{3}$$

где i – число степеней свободы, R – универсальная газовая постоянная.

$$m_0 = \frac{M}{N_A} \text{ – масса молекулы.} \tag{4}$$

Зная число частиц и основные характеристики (начальная скорость, масса, размер) каждой из них, можно приступать к моделированию. Так как скорость величина векторная, а движение хаотическое, то направление вектора скорости определяется при помощи генератора псевдослучайных чисел, генерирующего начальное значение угла наклона вектора скорости к оси абсцисс. Также примем, что при достижении границы участка по некоторой координате частица появляется с другой стороны этого участка, то есть внешние границы квадратного участка «склеены» между собой.

Примем, что молекулы отталкиваются друг от друга, когда расстояние между их центрами (частицы шарообразны) составляет: $D = r_1 + r_2$, то есть сумму радиусов этих молекул. Для сталкивающихся частиц выполняются законы сохранения. Из них и получим проекции скоростей частиц на оси координат:

$$v_{x1} = \frac{(m_{01} - m_{02})v_{x01} + 2m_{02}v_{x02}}{m_{01} + m_{02}},\ v_{y1} = \frac{(m_{01} - m_{02})v_{y01} + 2m_{02}v_{y02}}{m_{01} + m_{02}}; \quad (5)$$

$$v_{x2} = \frac{(m_{02} - m_{01})v_{x02} + 2m_{01}v_{x01}}{m_{01} + m_{02}},\ v_{y2} = \frac{(m_{02} - m_{01})v_{y02} + 2m_{01}v_{y01}}{m_{01} + m_{02}}. \quad (6)$$

На основе формул (1) – (6) мы и строим моделирующую программу. Ниже рассматривается параллельная организация программы.

Для решения описанной выше задачи лучше всего воспользоваться архитектурой кластерных вычислений типа master-slave, где нулевой процесс (master) осуществляет ввод информации, предварительный расчёт начальных условий и инициализацию пространства для моделирования, а также перераспределение частиц между остальными процессорами. После окончания процесса инициализации плоскость, на которой проводится эксперимент, разбивается при помощи заранее заданной сетки на $\boldsymbol{N_{sp}}$ частей. Каждая такая часть соответствует одному из slave-процессов. Таким образом, на каждый slave-процесс приходится моделирование броуновского движения на площади S/N_{sp}. Master-процесс производит пересылку slave-процессу записей о тех молекулах, которые находятся в соответствующей slave-процессору ячейке (пространственное, топологическое распараллеливание). Так как молекулы хаотично движутся и могут перемещаться по всему пространству площадью $\boldsymbol{S}$, то необходимо предусмотреть пересылку информации о молекулах, подошедших вплотную к границе между процессами, которые и разделяет эта граница. По окончании процесса моделирования все slave-процессы пересылают свои блоки информации о молекулах master-процессу. После этого нулевой (master) процесс и выводит полученную информацию в файл для дальнейшего анализа. Отметим, что также можно снимать через равные интервалы времени моделирования показания со slave-процессов, чтобы потом построить траектории движения взвешенных молекул. Описанная параллельная архитектура представлена на рисунке 1.

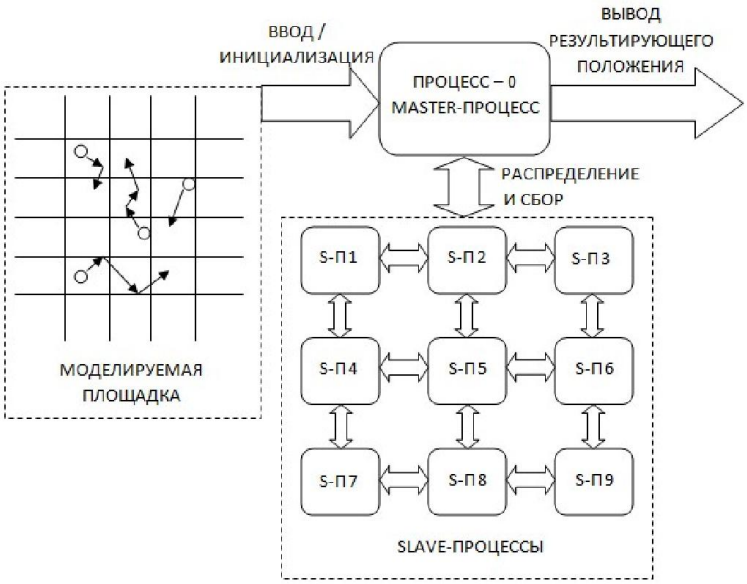


Рисунок 1. Параллельная архитектура, используемая в моделирующей программе (для случая 10 процессов, 9 из которых являются slave-процессами)

При работе со столь объёмной задачей (так, на площадь 100 X 100 нм приходится порядка 12 млн. молекул среды) на кластере необходимо принимать во внимание возможные частые перескоки молекул через границы своей области, ведь каждый такой перескок соответствует пересылке сообщения между процессами, что требует определённого времени. Задача при проектировании параллельной архитектуры алгоритма заключается в том, чтобы уменьшить число пересылок сообщений между процессами, сэкономив таким образом время работы алгоритма. В дальнейшем будут приведены два варианта решения этой задачи. Пока остановимся на результатах последовательного и параллельного моделирования для 15 случаев, приведённых в таблице 1.

Таблица 1. Исходные данные для моделирования

№	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S, $нм^2$	0,01	0,04	0,09	0,16	0,25	0,36	0,49	0,64	0,81	1,00	1,21	1,44	1,69	1,96	2,25
$N_{среды}$, шт	12	28	51	80	115	157	206	260	321	389	463	544	631	724	824

Ввиду полного моделирования взаимодействий как частиц среды со взвешенной частицей, так и частиц среды между собой, моделируемое время было выбрано малым: 1 пкч, то есть 10^{-12} часов. Среда – вода, взвешенная частица – цинк. Температура 273,15 К.

Для того, чтобы понять, как влияет на время моделирования использование параллелизма, проводились эксперименты с последовательной реализацией алгоритма моделирования, а также с его параллельными версиями (на 5, 10, 17 и 26 процессорах). При моделировании работы параллельной версии один из процессоров являлся мастером, а остальные – подчинёнными. Всё пространство, по которому перемещаются молекулы, разбивалось на равновеликие квадратные зоны. По результатам 10 экспериментов для разных размеров зон (и разного числа частиц среды при постоянном наличии лишь одной частицы примеси, см. таблицу 1) были получены результаты по времени, представленные на рисунке 2. На рисунке 3 представлена дисперсия результатов измерений. Отметим, что дисперсия для времени довольно мала.

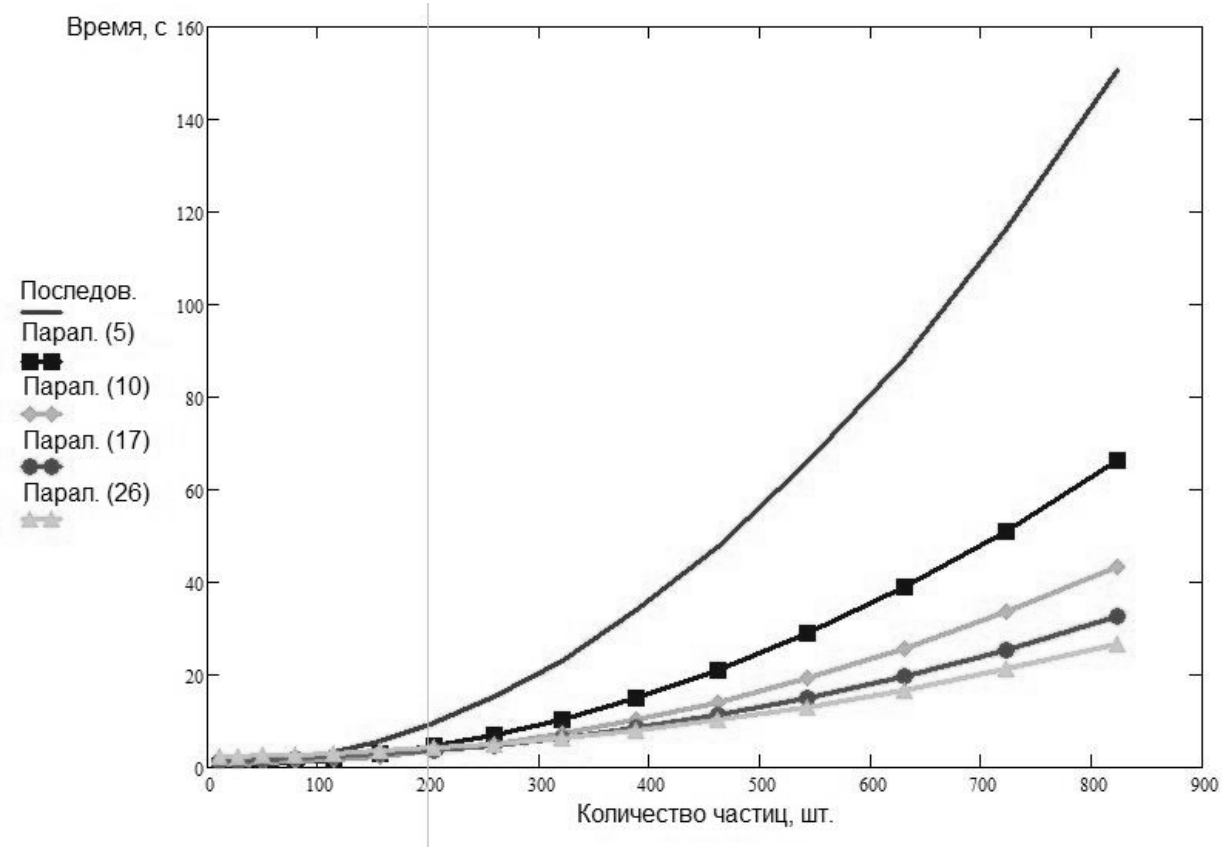


Рисунок 2. Время моделирования (среднее по 10 экспериментам)

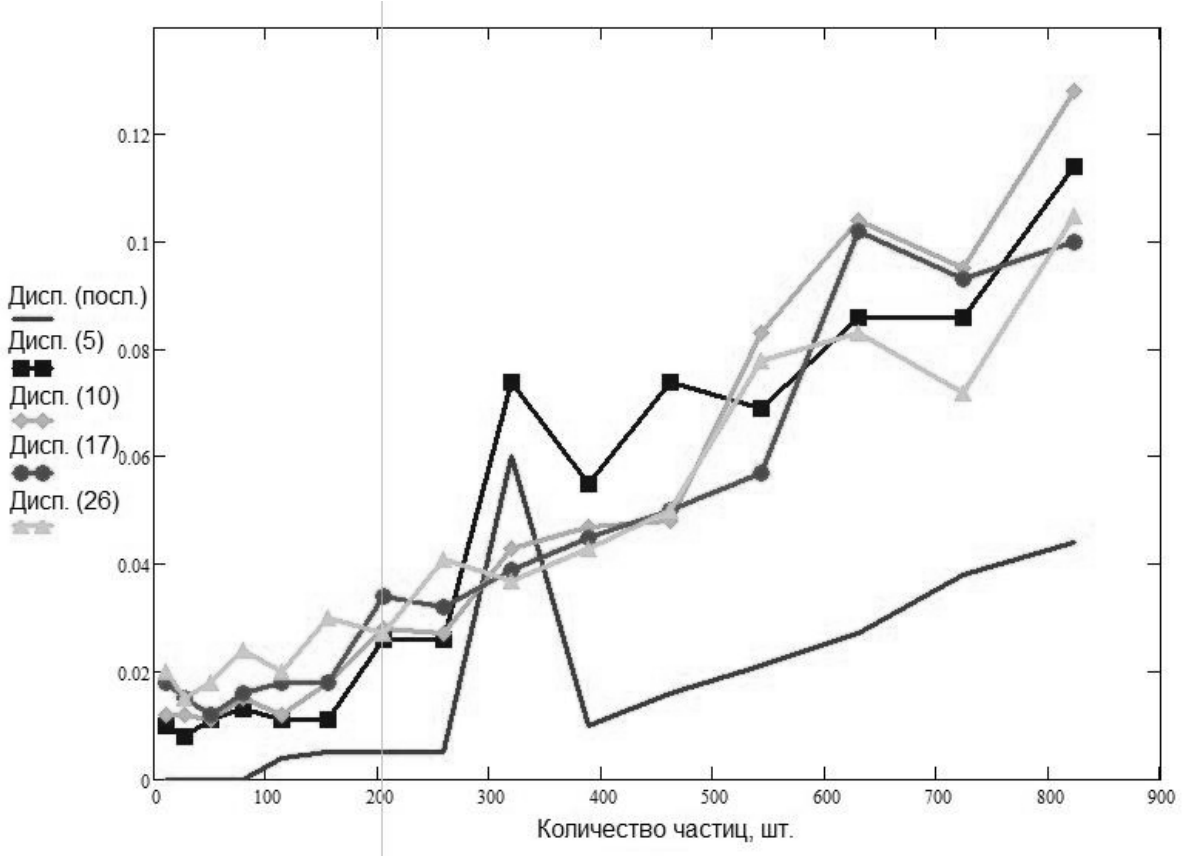


Рисунок 3. Дисперсия времени (по 10 экспериментам)

Как можно наблюдать на графиках, представленных на рисунке 2, с увеличением числа процессоров, участвующих в процедуре моделирования, время обработки падает. Также легко заметить, что графики постепенно уплотняются, сходясь к некоторой кривой. Эта предельная кривая и будет тем постоянным элементом времени, затрачиваемым на пересылку частиц-сообщений между процессорами. Выраженные далее в работе идеи призваны уменьшить эту составляющую, постоянную для данного числа частиц.

На рисунке 4 представлены графики ускорений параллельных версий по отношению к последовательной. При моделировании на большом числе процессоров для большого числа частиц достигается практически шестикратное ускорение. Резонно предположить, что для гораздо большего числа частиц (моделирование проводилось для частиц числом до 800, чтобы сократить временные затраты на эксперименты) ускорение, достижимое при использование большего числа процессоров будет выше.

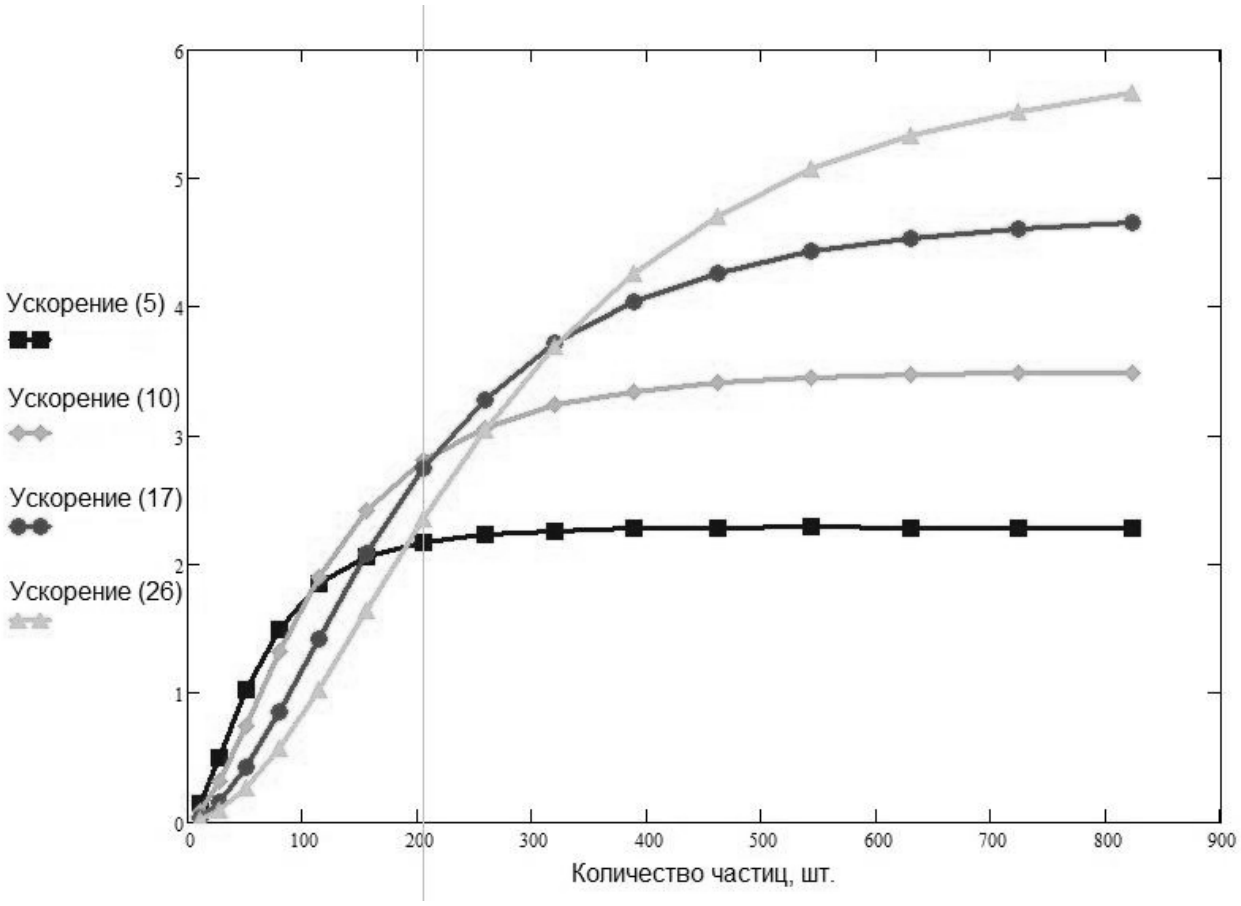


Рисунок 4. Ускорение параллельных версий по сравнению с последовательной

Существует несколько возможностей по дальнейшему ускорению процесса моделирования на высокопроизводительной параллельной аппаратуре за счёт уменьшения времени, затрачиваемого на пересылку сообщений между процессорами. Рассмотрим подробнее пару способов экономии времени работы алгоритма на пересылках:

- применение различных топологий разбиения поверхности;
- использование нечётких границ между ячейками разбиения.

Первый подход довольно прост. Заключается он в том, чтобы использовать при моделировании разбиение поверхности на зоны, отличные от квадратных. Каждой такой зоне, как и ранее, ставится в соответствии свой обрабатывающий процессор. Некоторые варианты таких разбиений приведены на рисунке 5.

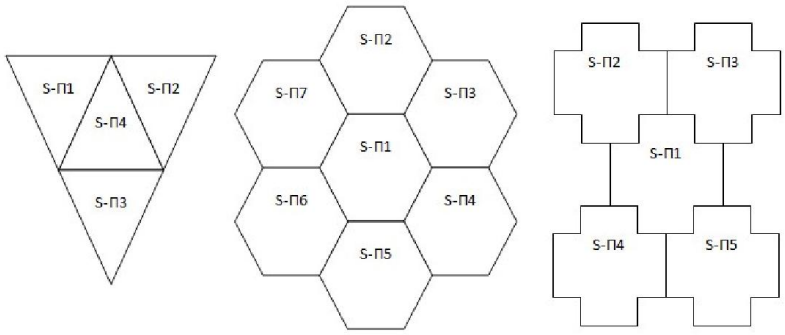


Рисунок 5. Некоторые возможные варианты форм разбиения моделируемой площадки на зоны

Для того чтобы понять, какое из разбиений является наиболее экономичным с точки зрения временных затрат на пересылку между зонами сообщений, необходимо проводить сравнительные эксперименты по времени. Можно предположить, что разбиение на шестиугольники представляется наиболее экономичным по времени ввиду того, что разбиение плоскости на ячейки равной площади и минимального периметра возможно при использовании в качестве таких ячеек шестиугольников (решение задачи Кельвина Томасом Хейлом, 2001 год).

Дополнительного ускорения при параллельном моделировании можно достигнуть путём применения некоторых простейших методик нечёткой логики. Для этого в представление разбиения пространства с частицами на зоны введём третье измерение – степень принадлежности частицы некоторой зоне $I \in [0;1]$. Причём степень принадлежности, очевидно, должна зависеть от скорости частицы и вектора этой скорости. По направлению вектора движения частицы степень принадлежности текущей зоне быстрее убывает, но возрастает у соседней, по направлению которой перемещается частица. В результате построения такой поверхности (иллюстративный пример представлен на рисунке 6) можно будет заранее формировать список частиц на пересылку в соседнюю зону, избегая таким образом многократных постоянных пересылок при попадании частиц на границу. Дополнительный выигрыш во времени будет особенно заметен при малых скоростях движения частиц или при сочетании высоких скоростей и больших размеров ячеек.

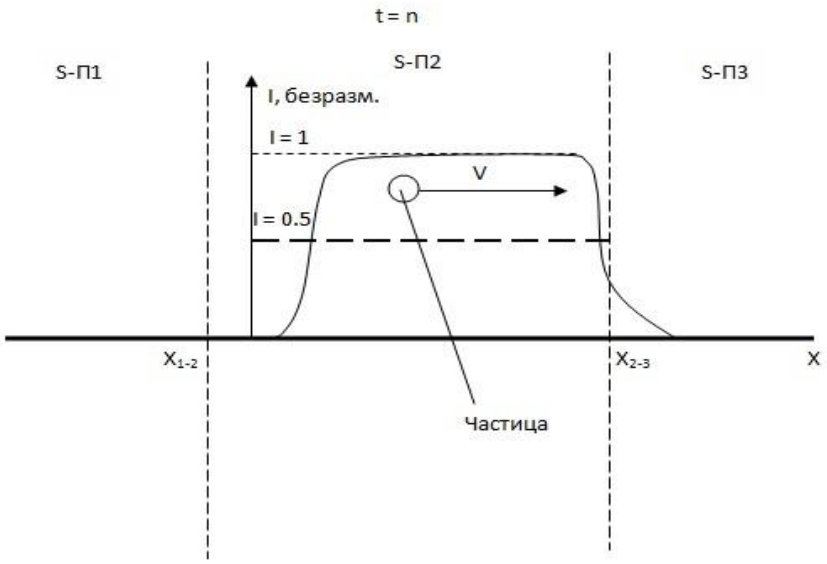


Рисунок 6. Применение методики теории нечётких множеств для убыстрения пересылки частиц-сообщений между процессорами

Например, возможно переслать частицу от процессора *i* процессору *j*, если выполнятся следующее простое условие:

$$I(x_{i-j}) \geq 0.5\,. \tag{7}$$

Сама же функция нечёткой принадлежности может иметь следующую форму:

$$I_j(X_{частицы}) = C \cdot \frac{1}{\left|X_{частицы} - X_{i-j}\right|} \cdot \left[\left|V_{\perp}^{(i-j)}\right|\right]^{\mathrm{sgn}(V_{\perp}^{(i-j)})} \cdot \frac{1}{p_{столк.}(X_{частицы} \pm \varepsilon)}\,. \tag{8}$$

C – коэффициент нормализации (до интервала от 0 до 1), $V_{\perp}^{(i-j)}$ - проекция скорости движения частицы на направление, перпендикулярное границе между текущей зоной i и соседней зоной j, $X_{частицы}$ – вектор координат частицы, X_{i-j} – вектор координат ближайшей к частице точки границы между зонами процессоров i и j, $p_{столк.}$ – вероятность столкновения частицы в области с координатами $X_{частицы} \pm \varepsilon$ с другой частицей, вычисляемая на основе наблюдений за движениями молекул за некоторый интервал времени Δt как $\dfrac{N_{столкновений}(X_{частицы} \pm \varepsilon)}{N_{частиц}(X_{частицы} \pm \varepsilon)}$.

Моделирование требующих больших вычислительных затрат процессов на кластерах и суперкомпьютерах актуально в связи с интенсивным развитием естественных наук. В условиях ограниченности вычислительных мощностей кластеров приходится искать способы

увеличить скорость обработки информации за счёт уменьшения времени, затрачиваемого на пересылки данных между процессорами. Предложенный в настоящей статьей способ распараллеливания и его модификации могут привести к повышению быстродействия при осуществлении вычислительных экспериментов на компьютерных высокопроизводительных системах. Дальнейшее развитие изложенных идей и их практическая реализация помогут добиться ускорения обработки данных параллельными системами.

Литература:

1 *Яворский Б.М.* Справочник по физике для инженеров и студентов вузов / Б.М. Яворский, А.А. Детлаф, А.К. Лебедев. – 8-е изд., перераб. и испр. – М.: ООО «Издательство Оникс»: ООО «Издательство Мир и Образование», 2006. – 1056 с.

2 Элементарный учебник физики: Учеб. Пособие. В 3 т. / Под ред. *Г.С. Ландсберга*: Т. 1. Механика. Теплота. Молекулярная физика. – 13-е изд. – М.: ФИЗМАТЛИТ, 2006. – 608 с.

3 *Корнеев В.Д.* Параллельное программирование в MPI. – 2-е изд., испр. – Новосибирск: Изд-во ИВМиМГ СО РАН, 2002. – 215 с.

4 *Шпаковский Г.И., Серикова Н.В.* Программирование для многопроцессорных систем в стандарте MPI / Г.И. Шпаковский, Н.В. Серикова. – Минск: Изд-во БГУ, 2002. – 323 с.

5 *Кофман А.* Введение в теорию нечётких множеств: Пер. с франц. – М.: Радио и связь, 1982. – 432 с.

6 *IBM*. Tivoli Workload Scheduler LoadLeveler: Using and Administering. – version 3, release 4. Publication No. SA22-7881-06, 2006. – 718 p.

УДК 005

НЕКОТОРЫЕ СОВРЕМЕННЫЕ МОДЕЛИ И АЛГОРИТМЫ КЛАСТЕРИЗАЦИИ ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

К.М. Гречищев, Р.С. Самарев

Введение

Большая часть знаний человечества хранится в форме текстов. Все больше документов переводятся из "твердой" копии в электронный формат либо создаётся непосредственно в электронном формате. В связи с этим возникают задачи автоматической и автоматизированной обработки текстовой информации. На сегодняшний день проблемы кластеризации и классификации документов, построения аннотаций, автоматического реферирования решены лишь частично.

Задача кластеризации играет важную роль при обработке документов. Кластеризация – это задача разбиения заданной выборки объектов на подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались [1]. Количество кластеров обычно неизвестно.

В настоящее время разработан ряд алгоритмов кластеризации, которые могут быть применены и для документов. Однако для этого необходимо формальное представление текстовых данных.

Модели представления текстовой информации

Наиболее распространенной моделью представления текста является модель термин-документ. В этом случае документ представляется как совокупность термов.

Под термом могут пониматься как отдельные слова, так и словосочетания. Множество термов $T = \{\tau_1 \quad \tau_2 \quad ... \quad \tau_n\}$ может быть определено заранее (при использовании словаря) или формироваться. При написании данной статьи, авторы формировали множество термов из начальных форм всех различных слов, входящих во все документы множества $D = \{d_1, d_2, ..., d_m\}$. При этом необходимо было использовать средство морфологического анализа. Кроме того, часть слов отсекалась при помощи стоп-словаря.

Отдельный документ может быть представлен в виде вектора:

$$A_i = \begin{pmatrix} \alpha_{i,1} & \alpha_{i,2} & \dots & \alpha_{i,n} \end{pmatrix},$$

где n – число термов множества T.

Тогда множество документов $D = \{d_1, d_2, ..., d_m\}$ при помощи модели термин-документ можно представить в виде следующей матрицы:

$$A = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \dots & \alpha_{1,n} \\ \alpha_{2,1} & a_{2,2} & \dots & \alpha_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & \alpha_{m,n} \end{pmatrix},$$

где n – число термов множества T, а m - число документов множества D.

В зависимости от способа вычисления коэффициентов $\alpha_{i,j}$ различают следующие виды моделей термин-документ:

1. *Бинарная модель*

При использовании бинарной модели коэффициент $\alpha_{i,j}$ может принимать значения 0 или 1. Значение 1 говорит о наличии данного терма τ_j в документе d_i, 0 – об его отсутствии.

Достоинством этой модели является малая вычислительная сложность, так как не требуется производить сложные вычисления при формировании коэффициент $\alpha_{i,j}$. Коэффициенты $\alpha_{i,j}$ не зависят друг от друга.

Существенным недостатком подобной модели является то, что она не учитывает число термов ни в конкретном документе, ни во всех документах множества. По этой причине на практике обычно применяются более сложные модели.

2. Модель с использованием меры TF (Term Frequency)

Обозначим $t_{i,j}$ число повторений терма τ_j в документе d_i. Тогда коэффициент $\alpha_{i,j}$ можно вычислить следующим образом:

$$\alpha_{i,j} = \frac{t_{i,j}}{\sum_{k=0}^{n} t_{i,k}}.$$

В знаменателе дроби стоит число повторений всех термов в документе d_i.

Важным достоинством модели TF является повышение веса тех термов, которые встречаются в документе чаще всего.

Этот факт одновременно является и недостатком. Например, если некий терм (предположим, предлог) часто встречается во всех документах, то его вес должен быть понижен, так как на основе данного терма невозможно отличить один документ от другого. Следовательно он не имеет значения для кластеризации.

3. Модель с использованием меры TF-IDF (Term Frequency - Inverse Document Frequency).

При использовании этого метода, коэффициент $\alpha_{i,j}$ вычисляется следующим образом:

$$\alpha_{i,j} = TF \cdot IDF = \frac{t_{i,j}}{\sum_{k=0}^{n} t_{i,k}} \cdot \log\left(\frac{m}{n_j}\right), \text{ где}$$

m – общее число документов множества $D = \{d_1, d_2, ..., d_m\}$, а n_j - число документов, содержащих терм τ_j ($m \geq n_j$).

Основное преимущество модели *TF-IDF* - снижение все тех термов, которые встречаются в большом числе документов. Это дает возможность существенно снизить все предлогов, союзов и других служебных частей речи. Однако, служебные части речи можно исключить из множества T, используя словарь стоп-слов. Тем не менее, данное свойство модели не утрачивает ценность. Например, при кластеризации деловых писем схожесть разделов приветствия и заключения не должны служить основанием для помещения документов в один кластер, поэтому вес слов, входящих в эти разделы, должен занижаться.

Модель *TF-IDF* имеет ряд модификаций. Например, модификация Okapi BM25 [2,3,4] на сегодняшний день широко применяется в поисковых системах для поиска информации и ранжирования выдачи [2].

Помимо моделей, основанных на модели термин-документ, существует ряд других[4]: Long Sent, Heavy Sent, Megashingles [5], Lex Rand, Log Shingles.

Средства построения моделей документов

Для построения модели термин-документ необходимо сформировать множество термов T, если оно не было зафиксировано, а также определить вхождения каждого терма τ_j в каждый документ d_i. Эту задачу решают в несколько этапов.

Поданный на вход документ необходимо разбить на отдельные слова. Возможность разбиения текста на отдельные слова предоставляет библиотека Apache Lucene[6]. В составе Lucene имеются средства разбиения текста на отдельные слова, а также различные фильтры для перевода слов в нижний регистр, отсечения слов определенной длины, отбора ключевых слов и так далее. Пакет org.apache.lucene.analysis.ru включает классы для разбиения русскоязычных текстов, например класс RussianAnalyzer.

Следующим этапом построения модели термин-документ является морфологический анализ. В русском языке наблюдается большое разнообразие различных форм одного и того же слова. Для выполнения кластеризации не имеет принципиального значения, в какой форме было употреблено то или иной слово. Поэтому, все формы одного и того же слова должны быть расценены как один и тот же терм.

Отметим, что существует продукт Apache UIMA (Unstructured Information Management Applications), переданный компанией IBM[7]. UIMA предназначен для выделения знаний из большого массива неструктурированной информации и представляет собой конструктор для выполнения семантической обработки текстов, подключая и используя различные модули, называемые аннотаторами. В состав UIMA входит ряд аннотаторов (модулей), позволяющих производить разбиение текста на отдельные слова.

Пристального внимания заслуживает продукт mystem компании Яндекс [8,9], который и был использован в этой работе. Программа mystem производит морфологический анализ текста на русском языке. Для слов, отсутствующих в словаре, порождаются гипотезы. Данное средство проводит анализ текста, выполняет разбиение его на слова, определяет их начальные формы, а также имеет возможность определения частей речи и формы слова.

Мера сходства документов

В качестве меры сходства документов использовалась метрика на основе косинуса угла между векторами (Cosine similarity), которая по результатам эксперимента оказалась лучше, чем мера расстояния по Евклиду:

$$d(a,b) = 1 - \cos(\theta) = 1 - \frac{a \cdot b}{\|a\| \cdot \|b\|} = 1 - \frac{\sum_{i=1}^{m} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{m} (a_i)^2} \cdot \sqrt{\sum_{i=1}^{m} (b_i)^2}},$$

где : m – размерность векторов, а θ – угол между векторами.

Алгоритмы кластеризации

В данной работе, основными алгоритмами, используемыми для кластеризации документов, были кластеризация методом К-средних (*k*-means clustering) и алгоритм “Canopy clustering”. В качестве инструментального средства использовалась библиотека Apache Mahout [10,11].

Алгоритм К-средних являлся основным алгоритмом кластеризации. Его недостатком является необходимость предварительно указать число кластеров и их начальные центры. Для решения этой проблемы предварительно проводилась кластеризация документов методом “Canopy clustering”, который требует задания двух пороговых параметров T1 и T2[13]. Из числа сформированных кластеров выбирались самые крупные кластеры, количество которых не превышало заданного. Эти кластеры и использовались в качестве инициализирующих.

Эксперимент и результаты

Целью эксперимента являлся подбор параметров T1 и T2 < T1, обеспечивающих наилучшее качество кластеризации. Оценка качества выполнялась на основе экспертного разбиения документов на классы. Критериями оценки служили следующие индексы [13]:

1. Индекс Ранда (Rand statistic)

$$Rand = \frac{SS + DD}{SS + DS + SD + DD}$$

2. Индекс Жаккарда (Jaccard index)

$$\text{Jaccard} = \frac{SS + DD}{SS + DS + SD}$$

3.Индекс Фолька и Маллоу (Folkes and Mallows index)

$$FM = \sqrt{\frac{SS}{SS + DS} * \frac{SS}{SS + SD}},$$

где: SS – число пар, элементы которых принадлежат одному классу и одному кластеру; SD – одному кластеру, но разным классам; DS – разным кластерам, но одному классу; DD – разным кластерам и разным классам. Под кластерами понимается результат работы алгоритма, под классами – результат разбиения документов экспертом.

Эксперимент 1 проводился на множестве из 10000 новостных документов, которые были разбиты экспертом на два класса: новости, посвященные программному обеспечению, и новости, посвященные оборудованию. Было проведено 3 опыта, которые отличались соотношением документов двух классов в кластеризуемом множестве (указано в заголовках диаграмм). Результаты для индекса Ранда при различных значениях T1 и T2 представлены на рисунках 1, 2 и 3. Использована модель TF и метрика косинуса угла.

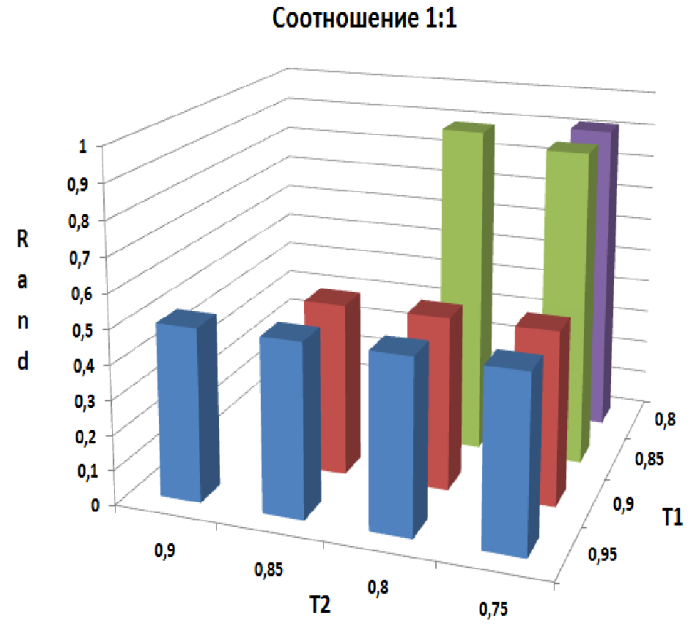


Рисунок 1. Результаты эксперимента №1 при соотношении документов 1:1

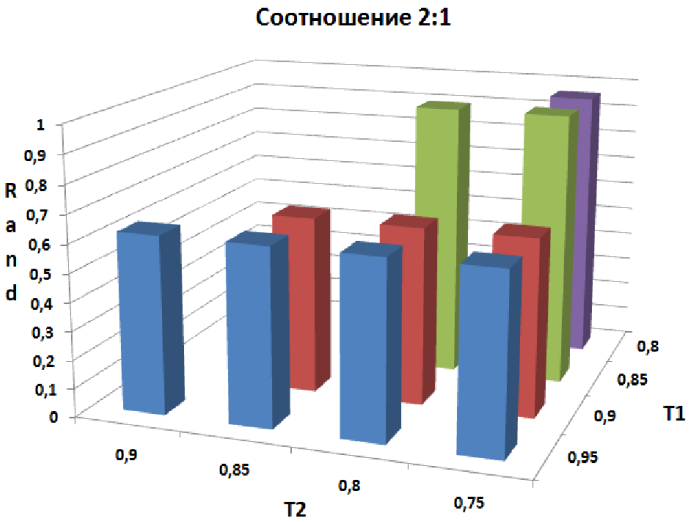


Рисунок 2. Результаты эксперимента №1 при соотношении документов 2:1

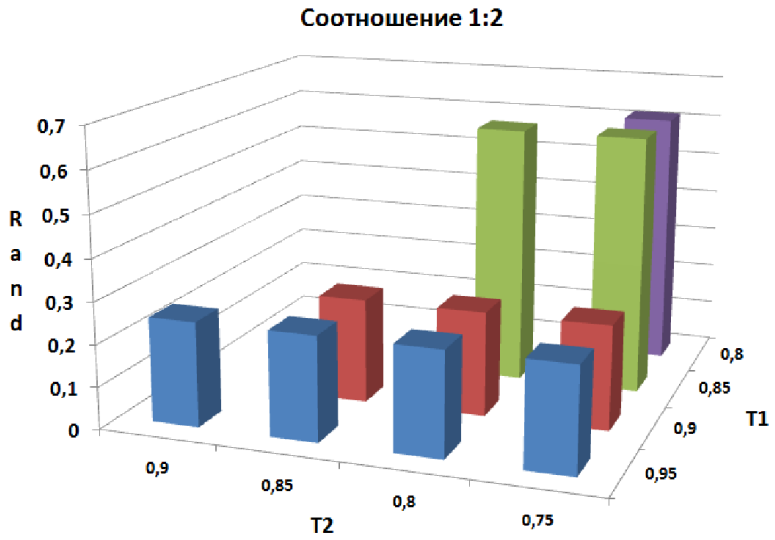


Рисунок 3. Результаты эксперимента №1 при соотношении документов 1:2

Выявлено, что при последовательном использовании двух алгоритмов кластеризации Cannopy и K-средних, наилучшие результаты получены при коэффициентах T1 = 0.85 и T2 = 0.8.

Для проверки полученных результатов был проведен эксперимент № 2, заключавшийся в кластеризации 12500 документов по 5 тематикам: новости политики, экономики, кинематографа, программного и аппаратного обеспечения. Результаты эксперимента приведены на рисунке 4.

Эксперимент подтвердил, что наилучшее качество кластеризации достигается при значениях T1 = 0.85 и T2 = 0.8.

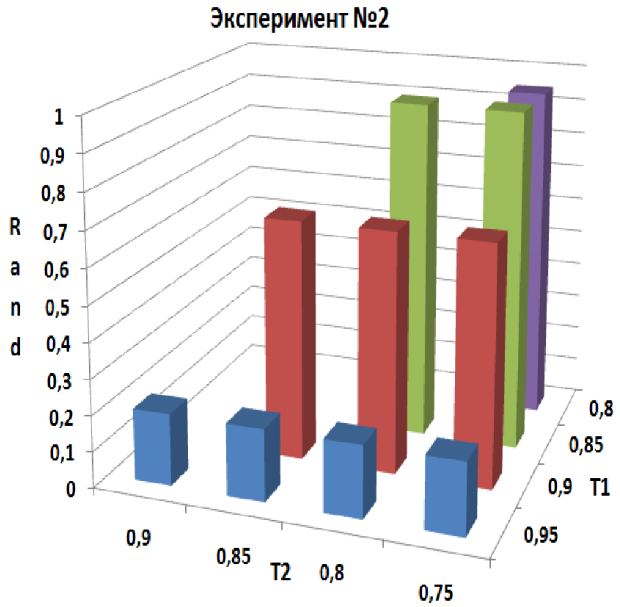


Рисунок 4. Результаты эксперимента про разбиению документов на 5 кластеров

Таблица. Значения индексов для всех экспериментов

	Индекс Ранда	Индекс Жаккарда	Индекс Фолька и Маллоу
Эксперимент 1 (1:1)	0,952196	0,908811	0,952228
Эксперимент 1 (2:1)	0,961564	0,940907	0,969602
Эксперимент 1 (1:2)	0,619687	0,35985	0,572497
Эксперимент 2	0,9503424	0,779689	0,876213

В таблице приведены значения всех трёх индексов при параметрах T1=0.85 и T2=0.8 для всех поставленных экспериментов. Кроме модели TF проводились эксперименты на модели TF*IDF, однако полученные результаты в целом оказались хуже, чем в модели TF.

Литература:

1. Интернет-ресурс: http://ru.wikipedia.org/wiki/Кластеризация

2. *А. Федоровский, М. Костин, А. Проскурин.* Mail.Ru на РОМИП-2005. Труды третьего российского семинара по оценке методов информационного поиска. Под ред. И.С. Некрестьянова - Санкт-Петербург: НИИ Химии СПбГУ, 2005, 226 с.

3. Интернет ресурс: http://ru.wikipedia.org/wiki/Okapi_BM25

4. *Зеленков Ю.Г, Сегалович И.В.* Сравнительный анализ методов определения нечетких дубликатов для Web-документов. Труды 9ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» - RCDL'2007, Переславль-Залесский, Россия, 2007.

5. *D. Fetterly, M. Manasse, M. Najork*. A Large-Scale Study of the Evolution of Web Pages, WWW2003, May 20-24, 2003, Budapest, Hungary.

6. Интернет-ресурс: http://lucene.apache.org/

7. Интернет-ресурс: http://uima.apache.org/

8. Интернет-ресурс: http://company.yandex.ru/technology/mystem

9. *Segalovich I.* "A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine.", MLMTA-2003

10. *S. Owen, R. Anil и др*. Mahout in Action. - Shelter Island: Manning Publications Co., 2012, 387 с.

11. Интернет-ресурс: http://mahout.apache.org//

12. Интернет-ресурс: https://cwiki.apache.org/MAHOUT/canopy-clustering.html

13. *Сивоголовко Е.* Оценка качества четкой кластеризации. Семинар Московской Секции ACM SIGMOD, 24 ноября 2011 г. – http://synthesis.ipi.ac.ru/sigmod/seminar/s20111124

УДК 004.8

ЭКСПЕРТНАЯ СИСТЕМА ЦЕНТРОВ ТЕЛЕФОННОЙ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЕЙ ПОРТАЛА ГОСУДАРСТВЕННЫХ УСЛУГ

Л.С. Тятюшкина, Г.С. Иванова

В информационном обществе современные компьютерные технологии все активнее вторгаются в работу властных структур, взаимоотношения граждан и государства. Повышение уровня компьютерной грамотности и широкое распространение информационно-коммуникационных технологий в социально-экономической сфере и органах государственной власти создало предпосылки для формирования электронного правительства.

Под электронным правительством понимается новая форма организации деятельности органов государственной власти, обеспечивающая за счет широкого применения информационно-коммуникационных технологий качественно новый уровень оперативности и удобства получения гражданами и организациями государственных услуг и информации о результатах деятельности государственных органов.

В качестве элемента инфраструктуры электронного правительства, был создан Единый портал государственных услуг (далее — ЕПГУ). Единый портал государственных услуг обеспечивает доступ граждан и организаций к федеральным, муниципальным и государственным услугам, а в случае возникновения вопросов и трудностей при заказе услуг — возможность обратиться на «горячую линию».

Для обеспечения эффективной информационной поддержки граждан и организаций, обратившихся на «горячую линию» ЕПГУ, разработана

Экспертная Система Центров Телефонного Обслуживания (далее — ЭС ЦТО) [1].

ЭС ЦТО представляет собой федеральную государственную информационную систему, обеспечивающую информационную поддержку специалистов ЦТО в ходе консультирования граждан и организаций по вопросам предоставления государственных и муниципальных услуг федеральных органов исполнительной власти (далее — ФОИВ) и органами местного самоуправления, а также по вопросам функционирования информационных систем, входящих в инфраструктуру электронного правительства.

Поиск информации

Условия функционирования и организационные структуры исполнительной власти отличаются специфическими особенностями, не позволяющими без изменений переносить понимание рационализации, типичное для технической сферы, на административное управление. Органы исполнительной власти значительно различаются между собой, что обусловливает большие различия в спектре оказываемых электронных услуг, а так же и в терминологии.

Эта же проблема находит отражение в базе данных ЭС ЦТО, которая наполняется инструкциями и нормативно-правовыми актами, быстрый поиск которых затрудняется терминологией и стилем написания официальных документов.

Таким образом, актуальной становится задача выбора эффективного поискового средства. Для выбора оптимального поискового средства, проведем сравнительный анализ типов информационного поиска. В частности, рассмотрим: булевый поиск, поиск по релевантности, поиск по сходству.

При использовании булевого поиска запросы строятся на основе элементарных слов и словоформ, находящихся между собой в отношениях, определенных предикатами (отрицание, дизъюнкция или конъюнкция). Данный тип запросов обычно используется в случаях, когда необходимо найти точное соответствие, например поиск в документах того или иного слова.

Поиск по релевантности осуществляется в некотором множестве электронных текстовых документов, по поисковому запросу, заданному

пользователем. Результатом поиска по релевантности является как можно более полная и точная выборка подмножества документов наиболее близких по смыслу поисковому запросу.

Поиск по сходству представляет собой «усовершенствованный» булевый поиск, где поисковая система учитывает возможные неточности формулирования поискового запроса или представления документа в электронном виде. При использовании данного типа поиска увеличивается вероятность выборки документов, степень релевантности которых мала, хотя не стоит забывать, что так же увеличивается объем выборки.

Исходя из сравнительного анализа типов информационного поиска и учитывая специфику системы, можно сделать вывод, что необходимо использовать как поиск по релевантности, так и поиск по сходству.

Для реализации поиска по сходству необходимо использовать словари русского языка, в которых слово «разбивается» на части: основу и окончание, а поиск осуществляется по основе слова. Словарь состоит из слов, терминов и указаний, в каких местах они встречаются. При индексировании [2] в словарь могут попасть не только ключевые слова, но и короткие слова (что, как, или), чтобы избежать наполнения словаря избыточными данными нужно установить значение минимальной длины индекса.

В следствии применения выбранных типов информационного поиска, полученные результаты представляются оператору в соответствии с учетом:

- весов полей документа (заголовок, таблица, сноска), в которых найдено соответствие;
- полным или частичным соответствием тексту поискового запроса;
- количеством найденных соответствий.

Были выявлены недостатки, которые можно исправить введением следующих средств:

- таблицы мапинга [3]: при вводе неформализованного поискового запроса «получить загранпаспорт» или «оформить загранник» должна предоставляться информация о правилах оформления заграничного паспорта. Таблица мапинга хранит соответствия нестандартных словоформ ключевым словам для того, чтобы они не обрабатывались по стандартной схеме индекса.

- листов исключений. Несмотря на установленные ограничения индекса, результаты поиска включали в себя документы, по смыслу не относящиеся к запросу, но содержащие предлоги или частицы, входящие в текст поискового запроса. Добавление листов исключений позволит отсечь результаты, содержащие общие с запросом: предлоги, союзы и частицы; слова, не несущие смысловой нагрузки (когда, здесь).

Типовые сценарии

Для обеспечения консультации пользователя в режиме он-лайн необходимо реализовать типовые сценарии, позволяющие полностью избавить оператора от формализации запроса и сократить время на его обработку.

Наиболее популярными запросами, требующими от оператора работы с персональными данными пользователя, являются запросы, связанные со статусом регистрации на ЕПГУ или статусом выполнения электронной услуги, заказанной на портале, а именно:

- проверка статуса почтового отправления;
- проверка статуса регистрации на портале;
- проверка состояния заказанной услуги.

В качестве решения была разработана группа типовых сценариев, при запуске которых, оператор должен ввести учетные данные пользователя портала (фамилия, имя, отчество и страховой номер индивидуального лицевого счета (далее — СНИЛС)) и, при необходимости, дополнительную информацию, что в свою очередь потребовало создания и настройки конфигурации, позволяющей хранить в системе персональные данные. Для организации контура, полностью удовлетворяющего требованиям информационной безопасности, требуются существенные трудозатраты и финансовые вложения.

Возможность работы с персональными данными пользователей была реализована менее затратным способом с помощью следующего механизма: данные пользователя в зашифрованном виде пересылаются на авторизованный сервер, где производится сравнение учетных данных, находящихся на сервере и данных, поступивших от оператора. Далее, сервер высылает результат операции: если ответ отрицательный — оператор

получает уведомление о некорректных данных, если положительный — авторизация прошла успешно.

При отработке сценария, с учетом успешной авторизации, сервер предоставляет результаты запроса ЭС ЦТО. Результаты запроса доступны оператору на странице ЭС ЦТО, а так же он получить их в виде отдельного html-файла.

Введение ряда типовых сценариев позволило оператору осуществлять консультацию пользователя исходя из конкретного обращения, а не руководствоваться общими рекомендациями и инструкциями. Отсутствие необходимости формализации ряда запросов [4] оператором, привело к существенному сокращению времени на их обработку.

Интеллектуализация системы ЭС ЦТО путем введения средств морфологического поиска, таблиц мапинга и типовых сценариев позволяет существенно сократить временные затраты и максимально упростить процесс получения оператором информации с портала.

Литература:

1. О внесении изменений в федеральную целевую программу "Электронная Россия (2002 - 2010 годы)": постановление Правительства РФ от 10 сентября 2009 г. N 721 // Собрание законодательства Российской Федерации, 2002, N 5, ст. 531; 2006, N 37, ст. 3875.

2. *Моисеенко Е.В., Лаврушина Е.Г.* Интеллектуальные технологии и системы// Информационные технологии в экономике. 2010. URL: http://abc.vvsu.ru/Books/up_inform_tehnol_v_ekon/page0017.asp (дата обращения 15.02.2012).

3. Лекции 7-8: Экспертные системы // Курс лекций по дисциплине "Системы искусственного интеллекта". 2001. URL: http://www.mari-el.ru/mmlab/home/AI/7_8/index.html (дата обращения 18.02.2012).

4. *Л.А. Растригин.* Искусственный интеллект // Радио. 1988. №4. С. 22-23.

УДК 621.396.93 (075.8)

ЗАЩИТА ТРАФИКА IP-ТЕЛЕФОНИИ С ПОМОЩЬЮ МЕЖСЕТЕВОГО ЭКРАНА DFL

А.М. Суровов, И.В. Баскаков

Передача голосовых данных по IP-сетям занимает в настоящее время важное место в сетевом функционале. За несколько последних лет технологии IP-телефонии значительно эволюционировали: появились мощные магистральные и транзитные маршрутизаторы и высокоскоростные телекоммуникационные каналы, появились качественно новые технологии, такие, как динамическая маршрутизация с учетом качества обслуживания в мультисервисных IP-сетях и ряд других. Актуальность развития решений IP-телефонии обусловлена не только возможностью снижения затрат на телефонные переговоры и техническое обслуживание инфраструктуры. В стратегическом плане IP-телефония может стать единой технической платформой, которая позволит объединить решения для передачи данных и голоса, а также для обработки и последующего использования этой информации во всех бизнес-процессах. В определенном смысле IP-телефония является средством повышения производительности труда и развития бизнеса.

IP-телефония могла бы получить значительно большее распространение, однако многих останавливает существующая опасность несанкционированного доступа к служебной информации. В общем случае, проблема решается с помощью комплекса аппаратно-программных средств, в который, несомненно, должно войти одно из наиболее эффективных средств защиты информации – межсетевой фильтр.

Воспользуемся появившимся недавно на рынке межсетевым экраном D-Link серии DFL.

Ресурсы устройства управляются сетевой операционной системой безопасности (network security operating system), называемой NetDefendOS. С точки зрения системного администратора NetDefendOS визуализирована с использованием логических объектов (objects). Таким образом, возможно настроить различные варианты конфигурации устройства.

Межсетевой экран поддерживает различные виды преобразования адресов на основе политик. Поддерживается NAT, PAT, SAT с различными видами преобразований.

Application-level gateway, или ALG (англ. «шлюз прикладного уровня») — компонент NAT-маршрутизатора, который понимает какой-либо прикладной протокол, и при прохождении через него пакетов этого протокола модифицирует их таким образом, что находящиеся за NAT'ом пользователи могут пользоваться протоколом.

NAT-маршрутизатор ретранслирует пакеты, поступающие изнутри сети, с внешнего IP-адреса. Также может подменяться порт. Но некоторые сетевые протоколы передают IP-адрес или порт отправителя. Конечно же, после прохождения NAT эти параметры становятся неверными — а значит, удалённая сторона не может наладить соединение.

ALG, идентифицировав пакет как относящийся к данному протоколу, подставляет в качестве IP-адреса и порта свои адрес и порт. ALG подобен прокси-серверу, который выполняет дополнительные операции наподобие кэширования, задача ALG — обеспечить, чтобы клиенты могли пользоваться протоколом SIP.

Механизм SIP ALG позволяют защитить VoIP-телефонию организации.

Предположим, что имеются три рабочие станции, которые защищены межсетевым протоколом DFL-210/800/1600/2500.

Обеспечим защиту трафика VoIP-телефонии между VoIP-терминалами и VoIP-прокси-серверами.

Настройка DFL-210/800/1600/2500

Настройка ALG

Зайдем в *Objects->ALG*. Добавим новый SIP-ALG.

Name	SIP-ALG
Maximum Sessions per ID	2 (только 2 одновременные сессии разрешено, по умолчанию – 5)
Maximum Registration Time	1200 (по умолчанию 3600 секунд)
SIP Signal Timeout	60 (по умолчанию 43200 секунд)

Data Channel Timeout 100 (по умолчанию 120 секунд)

Allow Media Bypass Disable

Примечание: Allow Media Bypass разрешает трафик RTP/RTCP данных напрямую между клиентами без межсетевого экрана, клиенты должны быть на одном интерфейсе и в одной подсети. Разрешение этой опции позволяет сократить время на обработку данных и улучшить качество голосового сигнала, но означает отсутствие защиты трафика со стороны межсетевого экрана.

Настройка TCP/UDP service

Зайдем в *Objects->Services*. Создадим новую службу SIP-Service, для этого введем следующие параметры:

Name SIP-service

Type TCP/UDP (выберите из списка)

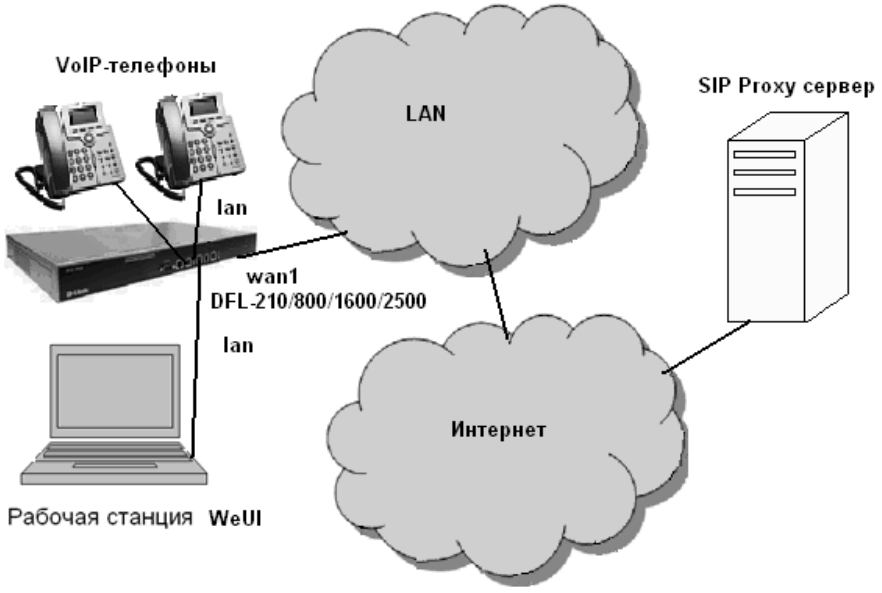
Destination 5060 (порт сигнализации SIP по умолчанию)

ALG SIP-ALG (выберите из списка ранее созданную)

Защита локальных клиентов – прокси в Интернете

Описание сценария Обеспечим защиту трафика VoIP-телефонии, размещенной за межсетевым экраном с NAT. SIP-Proxy server размещается в Интернете.

Рисунок 1



Создадим объект IP-адрес SIP proxy server. Для этого зайдем в *Objects->Address book->InterfaceAddresses->Add->IP4 Address*. Введем следующие параметры:

Name ip_proxy

IP Address 81.100.55.2

Настройка IP Rule

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-outbound
Action NAT
Service SIP-service
Source Interface lan
Source Network lannet
Destination Interface wan1
Destination Network ip_proxy

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-inbound
Action Allow
Service SIP-service
Source Interface wan1
Source Network ip_proxy
Destination Interface core
Destination Network wan1_ip (настраиваем для внешнего IP-адреса межсетевого экрана)

Зайдем в *Configuration* и выберем *Save and Activate*.

Примечание: 1. NAT Traversal не настраивается на строне VoIP-терминалов и SIP Proxy, т.к. SIP ALG берет функцию преобразования адресов на себя.

2. SAT-правило для трафика от прокси не требуется, т.к. межсетевой экран будет автоматически перенаправлять SIP-запросы конкретному внешнему пользователю.

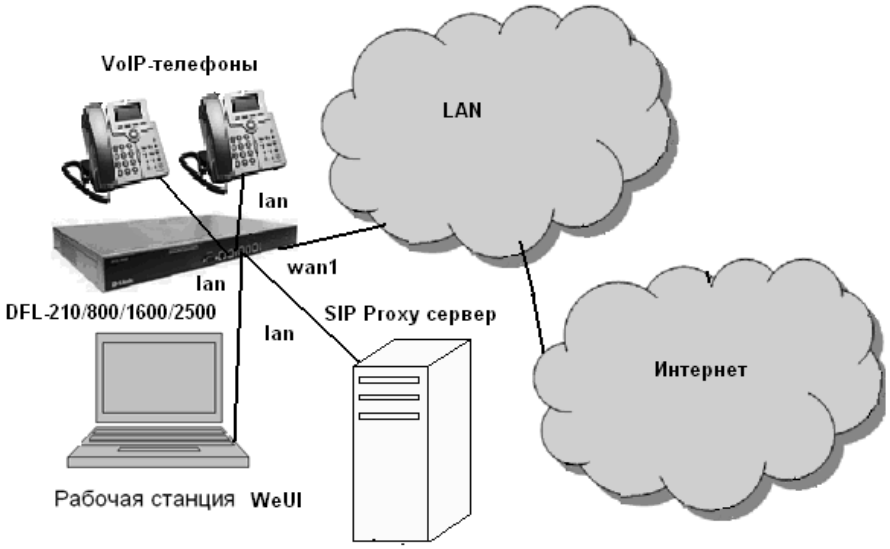
Упражнение Проверим работосопособность VoIP-телефонии, сделаем пробные звонки на внутренние телефоны и на внешние телефоны.

Защита прокси и локальных клиентов – полагаем, что прокси и

клиенты находятся в одной подсети.

Описание сценария Обеспечим защиту трафика VoIP-телефонии, размещенной за межсетевым экраном с NAT. SIP Proxy server размещается также в приватной сети **lan**.

Рисунок 2



Настройка DFL-210/800/1600/2500

Создадим объект IP-адрес SIP-proxy server. Для этого зайдем в *Objects->Address book->InterfaceAddresses->Add->IP4 Address*. Введем следующие параметры:

Name ip_proxy

IP Address 192.168.1.2

Настройка IP Rule

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-outbound

Action NAT

Service SIP-service

Source Interface lan

Source Network ip_proxy

Destination Interface wan1

Destination Network all-nets

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-inbound-SAT

Action	SAT
Service	SIP-service
Source Interface	wan1
Source Network	all-nets
Destination Interface	core
Destination Network	wan1_ip

Введем параметры во вкладке *SAT*:

Translate the	Destination IP Address
To New IP Address	ip_proxy

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name	SIP-inbound-allow
Action	Allow
Service	SIP-service
Source Interface	wan1
Source Network	all-nets
Destination Interface	core
Destination Network	wan1_ip

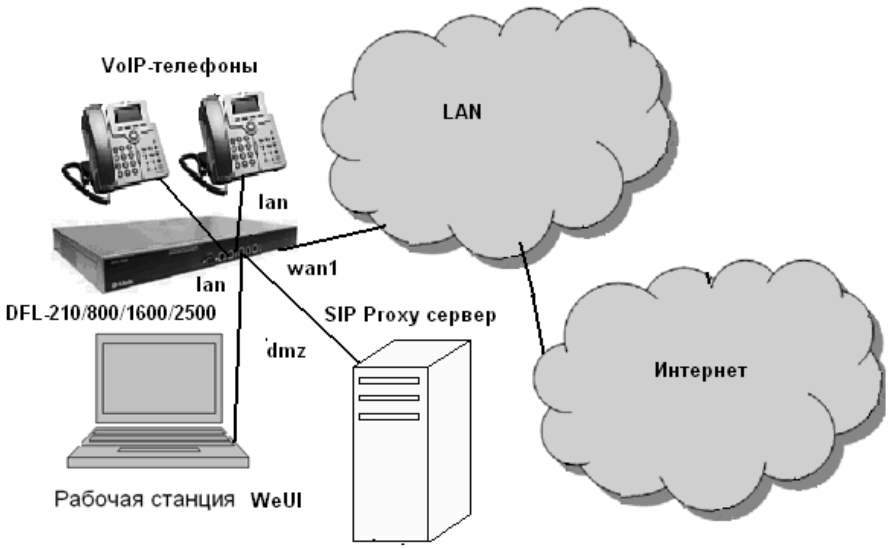
Зайдем в *Configuration* и выберите *Save and Activate*.

Упражнение Проверим работоспособность VoIP-телефонии, сделаем пробные звонки на внутренние телефоны и на внешние телефоны.

Защита прокси и локальных клиентов – прокси подключен к DMZ-интерфейсу.

Описание сценария Обеспечим защиту трафика VoIP-телефонии, размещенной за межсетевым экраном с NAT. SIP Proxy server размещается также в приватной сети **dmz**.

Рисунок 3



Настройка DFL-210/800/1600/2500

Создадим объект IP-адрес SIP-proxy server. Для этого зайдем в *Objects->Address book->InterfaceAddresses->Add->IP4 Address*. Введем следующие параметры:

Name	ip_proxy
IP Address	81.100.55.2

Настройка IP Rule

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введите параметры во вкладке *General*:

Name	SIP-outbound
Action	NAT
Service	SIP-service
Source Interface	lan
Source Network	lannet
Destination Interface	dmz
Destination Network	ip_proxy

Зайдем в *Rules->IP Rules*. Добавьте новое IP-Rule для SIP, введите параметры во вкладке *General*:

Name	SIP-outbound-allow
Action	Allow
Service	SIP-service
Source Interface	dmz
Source Network	ip_proxy
Destination Interface	wan1

Destination Network all-nets

Зайдем в *Rules->IP-Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-inbound-from-proxy
Action Allow
Service SIP-service
Source Interface dmz
Source Network ip_proxy
Destination Interface core
Destination Network dmz_ip

Зайдем в *Rules->IP Rules*. Добавим новое IP-Rule для SIP, введем параметры во вкладке *General*:

Name SIP-inbound-to-proxy
Action Allow
Service SIP-service
Source Interface wan1
Source Network all-nets
Destination Interface dmz
Destination Network ip_proxy

Зайдем в *Configuration* и выберем *Save and Activate*.

Примечание: При регистрации клиентов на прокси используется IP-адрес dmz интерфейса устройства.

Упражнение Проверим работоспособность VoIP-телефонии, сделайте пробные звонки на внутренние телефоны и на внешние телефоны.

Литература:

1. IP-телефония в компьютерных сетях: Учебное пособие/ *И.В. Баскаков и др*. – М: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2008. – 184 с.
2. *Гольдштейн Б.С., Пинчук А.В., Суховицкий А.Л.* IP-телефония. – М: Радио и связь, 2001 – 336 с.
3. http://www.mipk.ru
4. http://www.dlink.ru

5. http://www.taile.ru

УДК 519.216.1/2

МЕТОД СИНТЕЗА И СВОЙСТВА ПАРАМЕТРИЧЕСКОГО БАЗИСА НА ОСНОВЕ ВОЗВРАТНЫХ ЧИСЛОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Д.Ю. Карамнов, В.В. Сюзев

Использование спектральных методов цифровой обработки сигналов ставит задачу выбора рациональной дискретной системы базисных функций, адекватной требованиям, предъявляемым к процессу обработки. Ее решение существенно упрощается при наличии в структуре базисных функций изменяемых параметров, влияющих на их свойства. Например, в системах функций Виленкина-Крестенсона таким параметром является основание системы счисления, используемой для представления номеров и аргументов функций [1]. В данной работе предлагается метод построения параметрических дискретных базисов на основе возвратных числовых последовательностей с изменяемыми начальными значениями.

Воспользуемся числовой последовательностью, вычисляемой по правилу Фибоначчи [2]

$$u_k = u_{k-1} + u_{k-2}, \quad k > 2 \tag{1}$$

при общих начальных условиях

$$u_1 = u_2 = a, \tag{2}$$

где a – произвольное целое число. Из чисел этой последовательности сформируем конечную базисную систему, каждую функцию $\varphi(k,i)$ которой выразим следующим соотношением:

$$\varphi(k,i) = \begin{cases} u_{i+1}, & 0 \le i \le k, \\ -u_{i,} & i = k+1, \\ 0, & k+1 < i \le N-1. \end{cases} \tag{3}$$

В соответствии с ним при $0 \le k \le N-2$ все функции $\varphi(k,i) = \{u_1, u_2, ..., u_{k+1}, -u_{k+1}, 0, ..., 0\}$, а при $k = N-1$ функция $\varphi(N-1,i) = \{u_1, u_2, ..., u_N\}$.

Система функций (3) является ортогональной системой. Для доказательства этого найдем сумму произведений двух произвольных базисных функций $\varphi(k, i)$ и $\varphi(m, i)$, где $m \neq k$ и $m > k$. С учетом соотношения (3) эту сумму можно записать так:

$$\sum_{i=0}^{N-1} \varphi(k,i)\varphi(m,i) = \sum_{i=0}^{k} u_{i+1}^2 - u_{k+1}u_{k+2}.$$

Для определения суммы квадратов чисел u_i, найдем их связь с числами Фибоначчи f_i. Последние вычисляются по тому же правилу (1), но при начальных условиях $f_1 = f_2 = 1$. Поэтому несложно показать, что $u_3 = u_1 + u_2 = 2a = af_3$, $u_4 = u_2 + u_3 = 3a = af_4$ и в общем случае $u_k = u_{k-1} + u_{k-2} = af_k$. Тогда

$$\sum_{i=0}^{k} u_{i+1}^2 = a^2 \sum_{i=0}^{k} f_{i+1}^2 .$$

Но из теории чисел Фибоначчи [2] известно, что

$$\sum_{i=0}^{k} f_{i+1}^2 = f_{k+1} \cdot f_{k+2}$$

Следовательно,

$$\sum_{i=0}^{k} u_{i+1}^2 = u_{k+1} \cdot u_{k+2} \tag{4}$$

и

$$\sum_{i=0}^{N-1} \varphi(k,i)\varphi(m,i) = 0.$$

Все функции этого базиса являются ненормированными. Их мощность для $0 \le k \le N-2$ равна

$$P_k = \frac{1}{N}\sum_{i=0}^{N-1} \varphi^2(k,i) = \frac{1}{N}\left(\sum_{i=0}^{k} u_{i+1}^2 + u_{k+1}^2\right) =$$

$$= \frac{1}{N} u_{k+1}(u_{k+1} + u_{k+2}) = u_{k+1}u_{k+3} / N, \tag{5}$$

а для $k = N-1$

$$P_{N-1} = \frac{1}{N}\sum_{i=0}^{N-1} u_{i+1}^2 = u_N u_{N+1} / N. \tag{6}$$

При выводе этих формул использовано соотношение (4).

Числовой базис (3) является полным базисом и может быть использован для спектрального представления любых решетчатых сигналов

и функций ограниченной мощности. Дискретные преобразования Фурье в этом базисе имеют обычный вид записи

$$x(i) = \sum_{k=0}^{N-1} X(k)\varphi(k,i), \tag{7}$$

$$X(k) = \frac{1}{NP_k} \sum_{i=0}^{N-1} x(i)\varphi(k,i), \tag{8}$$

а равенство Парсеваля

$$\frac{1}{N} \sum_{i=0}^{N-1} x^2(i) = \sum_{k=0}^{N-1} X^2(k) P_k \tag{9}$$

отражает энергетическую равноправность временного $x(i)$ и спектрального $X(k)$ представления сигналов.

Особенности аналитической записи (3) функций $\varphi(k,i)$ позволяют прямое преобразование Фурье (8) представить в более удобной для вычислений форме:

$$X(k) = \begin{cases} [\sum_{i=0}^{k} x(i)u_{i+1} - x(k+1)u_{k+1}] / (u_{k+1}u_{k+3}), k \neq N-1, \\ [\sum_{i=0}^{N-1} x(i)u_{i+1}] / (u_N u_{N+1}), k = N-1. \end{cases} \tag{10}$$

Как следует из зависимости (10) спектр сигнала в базисе (3) также тесно связан с самими числами последовательности (1).

Его можно связать с числами Фибоначчи f_k, если снова учесть, что $u_k = af_k$. Тогда после преобразований получаем

$$X(k) = \begin{cases} [\sum_{i=0}^{k} x(i)f_{i+1} - x(k+1)f_{k+1}] / (f_{k+1} f_{k+3} a), k \neq N-1, \\ [\sum_{i=0}^{N-1} x(i)f_{i+1}] / (f_N f_{N+1} a), k = N-1. \end{cases} \tag{11}$$

Сравнение выражений (10) и (11) приводит к важному выводу, что параметр a играет роль масштабирующего множителя и с его помощью можно влиять на величину спектральных коэффициентов.

При этом, если спектр вычисляется с учетом мощности P_k базисных функций (т.е. по формулам (10) и (11)), то параметр a оказывается в знаменателе выражения для спектра $X(k)$ и его увеличение приводит к уменьшению модуля $X(k)$. Если же спектр вычислять без учета P_k (перенести все P_k в формулу (7) обратного преобразования Фурье), то параметр a будет уже в числителе выражения для спектра $X(k)$ и его

увеличение приведет к увеличению модуля $X(k)$. Это свойство спектра числового базиса может оказаться полезным для практики спектрального анализа, особенно при одновременной обработке сигналов различной интенсивности и мощности.

Приведем два примера спектров в числовом базисе для постоянного и линейного сигналов. Для постоянного сигнала $x(i) = b$ в соответствии с формулами (10) и (11) получим

$$X(k \neq N-1) = [b\sum_{i=0}^{k-1} u_{i+1}] / (u_{k+1}, u_{k+3}) = [\frac{b}{a}\sum_{i=0}^{k-1} f_{i+1}] / (f_{k+1} f_{k+3}),$$

$$X(N-1) = [b\sum_{i=0}^{N-1} u_{i+1}] / (u_N u_{N+1}) = [\frac{b}{a}\sum_{i=0}^{N-1} f_{i+1}] / (f_{N+1} f_{N+1}).$$

Но [2]

$$\sum_{i=0}^{n-1} f_{i+1} = \sum_{j=1}^{n} f_j = f_{n+2} - 1, \qquad (12)$$

поэтому

$$X(k) = \begin{cases} \dfrac{b(f_{k+2}-1)}{a f_{k+1} f_{k+3}}, & 0 \le k \le N-2, \\ \dfrac{b(f_{N+2}-1)}{a f_N f_{N+1}}, & k = N-1. \end{cases}$$

Для постоянного сигнала только одна нулевая спектральная составляющая равна нулю (в этом случае $f_{k+2} = f_2 = 1$). Значения всех остальных $X(k)$ являются ненулевыми и связаны с параметром *a* и числами Фибоначчи.

Для линейного сигнала $x(i) = bi$ из формул (10) и (11) получаем

$$X(k) = \frac{b}{NP_k}\sum_{i=0}^{N-1} i\varphi(k,i) = \frac{b}{u_{k+1}u_{k+3}}[\sum_{i=0}^{k} iu_{i+1} - (k+1)u_{k+1}] =$$

$$= \frac{b}{u_{k+1}u_{k+3}}[\sum_{i=0}^{k-1} iu_{i+1} - u_{k+1}] = \frac{b}{af_{k+1}f_{k+3}}[\sum_{i=0}^{k-1} if_{i+1} - f_{k+1}].$$

Если сделать подстановку $i = j-1,\ j = 1,2,...,k,$ то из последнего выражения можно получить:

$$X(k) = \frac{b}{u_{k+1}u_{k+3}}[\sum_{j=1}^{k} ju_j - \sum_{j=1}^{k+1} u_j] = \frac{b}{af_{k+1}\cdot f_{k+3}}[\sum_{j=1}^{k} jf_j - \sum_{j=1}^{k+1} f_j].$$

Но известно [2], что

$$\sum_{j=1}^{k} jf_j = kf_{k+2} - f_{k-3} + 2,$$

поэтому с учетом формулы (12) окончательно имеем:

$$X(k) = \frac{b}{af_{k+1} \cdot f_{k+3}}(kf_{k+2} - 2f_{k+3} + 3), \quad k \neq N-1.$$

При $k = N-1$

$$X(N-1) = \frac{b}{af_N \cdot f_{N+1}}[(N-1)f_{N+2} - f_{N+3} + 3].$$

В спектре линейного сигнала все составляющие *X(k)* не равны нулю и зависят от значений параметра *a* и соответствующих чисел Фибоначчи.

Числовые базисные функции $\varphi(k,i)$ и их спектры дискретных сигналов можно записать в виде функций параметра *a* и индексов *i* и *k*. Для этого нужно воспользоваться формулой Бине для чисел Фибоначчи [2]

$$f_i = (\alpha^i - \beta^i)/\sqrt{5}, \tag{13}$$

где

$$\alpha = \frac{1+\sqrt{5}}{2} \approx 1,618, \quad \beta = \frac{1-\sqrt{5}}{2} \approx -0,618,$$

и переписать её для последовательности u_i:

$$u_i = a(\alpha^i - \beta^i)/\sqrt{5}. \tag{14}$$

Из формул (13) и (14) следует, что числа в числовых последовательностях (1) имеют экспоненциальный характер изменения. Поэтому такой же характер изменения будут иметь и сами функции числового базиса и их спектры различных сигналов. Для этих базисных функций адекватными в смысле обеспечения концентрации энергии в минимальном числе спектральных коэффициентов будут экспоненциальные сигналы вида α^i, поскольку для них числовой спектр близок к дельта-функции. Поэтому числовые базисные функции (3) могут оказаться полезными при обработке сигналов с экспоненциальным законом изменения во времени.

Литература:

1. *Трахтман А.М., Трахтман В.А.* Основы теории дискретных сигналов на конечных интервалах. – М.: Сов. радио, 1975. - 208 с.

2. *Воробьев Н.Н.* Числа Фибоначчи. -М.: Наука, 1984. - 144с.

Содержание

Подписано в печать 21.06.2012. Формат 60x84/16.
Усл. печ. л. 12.45. Тираж 100 экз. Заказ

105005, Москва, 2-я Бауманская ул., д. 5
НИИ Радиоэлектроники и лазерной техники