МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ
РАДИОЭЛЕКТРОНИКИ

ISSN 0135-1710

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ И ПРИБОРЫ АВТОМАТИКИ

Всеукраинский межведомственный научно-технический сборник

Основан в 1965 г.

Выпуск 151

Харьков
2010

УДК 519.713

В.Б. ЯЦІВ, Б.П. РУСИН

ПОРІВНЯННЯ АЛГОРИТМІВ МАСШТАБУВАННЯ ЗОБРАЖЕНЬ В МЕТОДАХ НЕРУЙНІВНОГО КОНТРОЛЮ МАТЕРІАЛІВ

Проводиться огляд основних методів масштабування зображень, розглядається реалізація та аналіз алгоритмів масштабування напівтонових зображень в задачах неруйнівного контролю матеріалів.

1. Вступ

З розвитком цифрової техніки набула особливого розвитку обробка зображень і зараз широко застосовується в найрізноманітніших сферах діяльності. В широкому значенні обробка зображень – це будь-яка форма обробки інформації, для якої вхідними даними є графічні дані.

До стандартних задач обробки зображень відносяться:

– масштабування;
– поворот;
– сегментація;
– фільтрація (лінійна і нелінійна);
– корекція кольорів, яскравості, зміна глибини кольорів.

Одна з найпоширеніших задач обробки зображень – масштабування. В сучасних умовах різноманітності цифрових пристроїв для показу зображень і відео все частіше використовують пристрої з заданим розрішенням, наприклад, плазмові панелі або LCD-дисплеї, постають задачі представлення графічних даних зі зміною співвідношення сторін. Якість зображення на таких пристроях залежить від якості застосованих алгоритмів масштабування. На дисплеях (LCD), де базова гратка пікселів фіксована, доводиться використовувати апаратне масштабування. Як правило, воно проводиться алгоритмом білінійної інтерполяції, при цьому стає практично неможливо працювати, наприклад, на ноутбуках з розрішенням відмінним від базового. Такі ж проблеми виникають в проекторах і при показі зображення в PAL/NTSC на телевізорах високої чіткості (HDTV), наприклад, на плазмових панелях. При необхідності відобразити зображення на дисплеї певного розміру чи надрукувати його на папері заданого розміру і в ряді інших задач виникає необхідність масштабування зображення, часто зі зміною співвідношення сторін.

У зв'язку з цим виникає необхідність розробки нових методів масштабування, які передбачають можливість масштабування зображень без спотворення основного змісту і позбавленого від артефактів, які виникають при застосуванні існуючих методів. Мета даної роботи – реалізація та аналіз алгоритмів масштабування в задачах неруйнівного контролю матеріалів.

2. Типи алгоритмів масштабування

Сучасні методи масштабування використовують передискретизацію зображення для уникнення втрати дрібних деталей. При цьому найчастіше використовують такі методи інтерполяції [1]:

1. Nearest neighbor (найближчий сусід). Значення кольору пікселя, що не входить в піксельну сітку оригінального зображення, копіюється з найближчого до нього пікселя оригінального зображення.
2. Bilinear (білінійна). Проводиться лінійна інтерполяція по двох координатах за значеннями чотирьох найближчих сусідів в оригінальному зображенні.
3. Bicubic (бікубічна) – кубічна інтерполяція по двох координатах.
4. Precise bilinear (точна білінійна) - проводиться лінійна інтерполяція в двох напрямках, але замість матриці 2x2 використовується повна білінійна матриця.
5. Precise bicubic (точна бікубічна) - проводиться кубічна інтерполяція в двох напрямках по повному бікубічному ядру.

6. Lanczos3 – використовується матриця 6x6, тобто кожний піксель обчислюється по 36 сусідніх.

Іншим методом, що дозволяє змінити співвідношення сторін, є метод вирізання (crop) потрібної частини зображення з потрібним співвідношенням сторін. Для автоматизації знаходження найбільш важливої частини зображення використовують метод Smart Crop.

Існуючі методи масштабування передбачають видалення чи вставку колонок чи рядків пікселів з усередненням різними способами по сусідніх. При застосуванні такого методу для непропорційного масштабування об'єкти зображення спотворюються, змінюючи свої пропорції так само, як і ціле зображення. Все це в цілому призводить до зменшення ефективності обробки і розпізнавання зображень та до виникнення похибок [2].

Коли постає задача масштабування з врахуванням вмісту інформативної частини зображення, то слід використовувати технологію масштабування Liquid Resize, яка передбачає можливість непропорційного масштабування без спотворення форми об'єктів зображення. Метод передбачає побудову і видалення піксельних шляхів. Це досягається за рахунок видалення з зображення в першу чергу малопомітних пікселів, які переважно належать фонові, і залишення пікселів основних об'єктів зображення практично без змін.

3. Реалізація алгоритмів масштабування

Щоб зрозуміти, як працює кожний з перелічених методів, реалізуємо алгоритми їхньої роботи на практиці.

Nearest neighbor - це найбільш базовий зі всіх алгоритмів інтерполяції, який вимагає найменшого часу обробки, оскільки враховує тільки один піксель - найближчий до точки інтерполяції. Значення кольору пікселя, що не входить в піксельну сітку оригінального зображення, копіюється з найближчого до нього пікселя оригінального зображення. В результаті кожний піксель просто стає більшим [3].

Bilinear - проводиться лінійна інтерполяція по двох координатах за значеннями чотирьох найближчих сусідів в оригінальному зображенні (рис. 1).

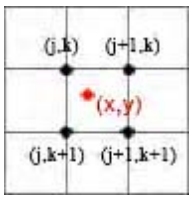


Рис. 1. Обчислення значення кольору в точці (x,y) за значеннями кольорів оригінального зображення (Bilinear)

Розглядається масив 2x2 навколишніх пікселів – 4. Результуюче значення кольору обчислюється за формулами:

$$
\begin{aligned}
F(x,y) &= \frac{F_b - F_a}{y_2 - y_1}\cdot(y - y_1) + F_a,\\
F_a &= \frac{F(x_2,y_1) - F(x_1,y_1)}{x_2 - x_1}\cdot(x - x_1) + F(x_1,y_1),\\
F_b &= \frac{F(x_2,y_2) - F(x_1,y_2)}{x_2 - x_1}\cdot(x - x_1) + F(x_1,y_2).
\end{aligned}
\qquad (1)
$$

При масштабуванні зображень недоліком алгоритму є той факт, що при збільшенні в N разів зображення розміром W на H пікселів в результаті буде отримано зображення розміром не NW на NH пікселів, а (N(W-1)+ 1) на (N(H-1)+ 1) пікселів.

Bicubic – кубічна інтерполяція по двох координатах [4]. Розглядається масив з 4x4 навколишніх пікселів – всього 16 (рис. 2).

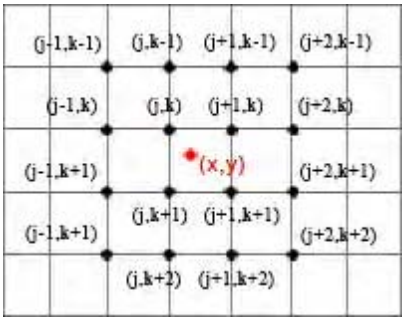


Рис. 2. Обчислення значення кольору в точці (x,y) за значеннями кольорів оригінального зображення (Bicubic)

Оскільки вони знаходяться на різних відстанях від невідомого пікселя, найближчі пікселі отримують при розрахунку більшу вагу. Результуюче значення кольору обчислюється за виразом

$$F(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}a_{ij}(x-x_1)^i(y-y_1)^j. \qquad (2)$$

Коефіцієнти a_{ij} шукаються з умов рівності значень і похідних кольору в чотирьох вузлових точках.

Lanczos – застосування алгоритму дає змогу добитися високої якості при масштабуванні. Спочатку шукаємо найближчу точку з цілими координатами і позначаємо її j,k, тоді перебираємо всі точки з координатами від j-a до j+a і від k-a до k+a, для кожної шукаємо відстань до x,y (рис. 3).

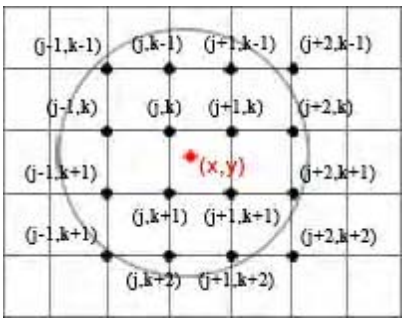


Рис. 3. Обчислення значення кольору в точці (x,y) за значеннями кольорів оригінального зображення (Lanczos)

Обчислюється значення кольору пікселя за формулою:

$$F(x)=\begin{cases}\sin c(x)\sin c(\frac{x}{a}), & -a<x<a;\\ 0, & x=0,\end{cases} \qquad (3)$$

де a – радіус дії алгоритму або ширина вікна. Якщо x виходить за межі вікна, то результат приймається за 0. При $x=0$ – результат одиниця.

Застосувавши кожний з алгоритмів інтерполяції, збільшуючи на 500% елемент оригінального зображення, отримали результати для зображення з пітінгами, які представлені на рис. 4.

При масштабуванні зображень з використанням алгоритмів інтерполяції зникає негативний ефект втрати дрібних деталей зображення. Проте при непропорційному масштабуванні об’єкти зображення змінюють свою форму та пропорції. Для уникнення спотворень при непропорційному масштабуванні застосовують технологію масштабування Liquid Resize, яка передбачає можливість масштабування зображень без спотворення основного вмісту зображення [5].

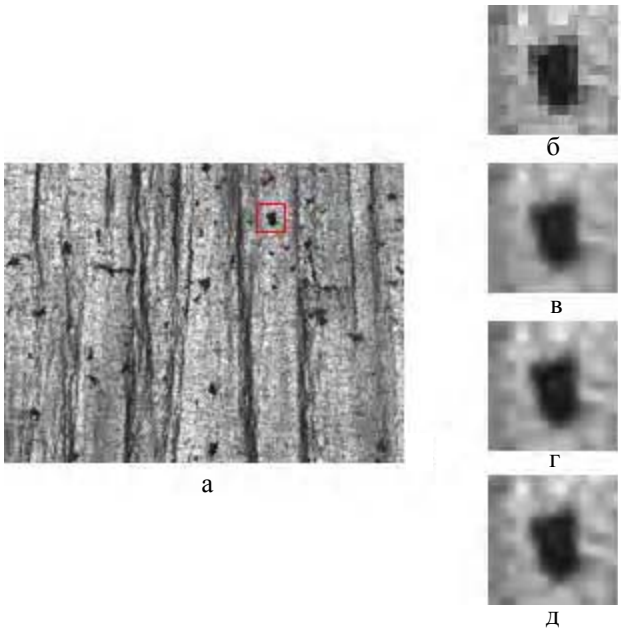


Рис 4. Елемент оригінального зображення з пітінгами (а),
Nearest neighbor (б), Bilinear (в), Bicubic (г), Lanczos (д)

Розглянемо алгоритм непропорційного масштабування на основі методу Seam Carving, який полягає в побудові і видаленні піксельних шляхів. Реалізований алгоритм складається з таких етапів:

1. Присвоєння важливості пікселям. Критерієм важливості пікселів виступають вагові коефіцієнти, отримані шляхом застосування до зображення енергетичної функції. Найпростіший приклад такої функції – градієнт F(I) (4), його значення велике там, де зображення різко міняється (краї об'єктів, місця різкої зміни кольорів), і практично нульове у фонових частинах зображення. Вагові коефіцієнти повинні приймати великі значення у важливих місцях зображення, і малі значення – в малопомітних частинах зображення:

$$F(I) = (\frac{\partial}{\partial x} I) + (\frac{\partial}{\partial y} I) . \quad (4)$$

З формули (4) видно, що енергетична функція пікселя рівна зміні кольору сусідніх пікселів у порівнянні з даним пікселем. І чим більша різниця в кольорі між даним пікселем і сусіднім, тим більше значення вагового коефіцієнта.

2. Побудова горизонтальних, вертикальних піксельних шляхів.

3. Шляхи починаються з верхнього чи лівого краю зображення і будуються так, щоб в кожному стовпці зображення (при побудові горизонтального шляху) чи кожному рядку зображення (при побудові вертикального шляху) був присутній один і тільки один піксель, що належить шляху (рис. 5).

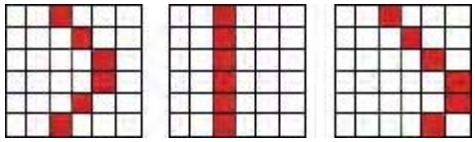
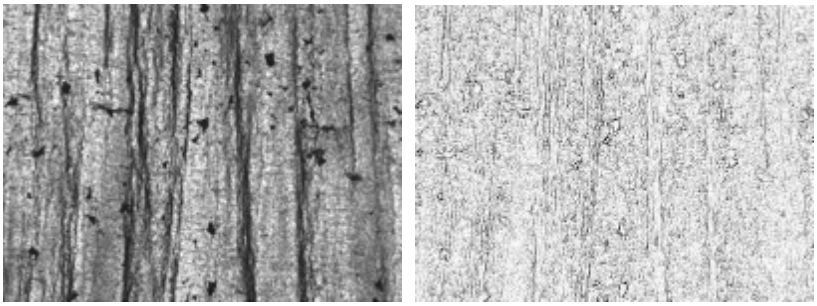


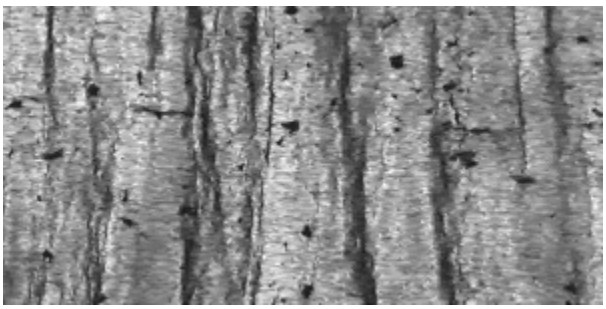
Рис. 5. Побудова піксельних шляхів

4. Пошук і видалення піксельних шляхів з мінімальною енергією до тих пір, поки не буде досягнуто потрібного розміру при збільшенні чи зменшенні зображення.

Результати алгоритму масштабування Seam Carving по горизонталі представлено на рис. 6.



а б



в

Рис. 6. Результати масштабування з використанням технології Liquid Resize: а – вхідне зображення; б – енергетична карта зображення; в – масштабування по горизонталі

4. Аналіз алгоритмів масштабування зображення

При використанні описаних вище методів значення кольорів пікселів зазвичай обчислюються шляхом інтерполяції чотирьох сусідніх, зображення на виході виходить розмитим (див. рис 4.). Крім того, застосування інтерполяції веде до зниження контрастності (різкості на межах), і тому дані методи дають погані результати на зображеннях з індексованою палітрою. Nearest neighbor – найшвидший, але разом з тим малоефективний метод, оскільки використовується значення найближчих пікселів оригінального зображення. Таким чином, частина пікселів відкидається (при зменшенні) або дублюється (при збільшенні). Bilinear – другий по швидкості метод, дає непогані результати. Проте при великих збільшеннях починає проявлятися блочна структура. Bicubic - значно повільніший від попередніх методів, а при збільшенні зображення дає значно кращий результат. Оскільки деталі промальовуються чіткіше, але посилюються і шуми, то зображення з шумами потрібно збільшувати дуже обережно. Ослаблення впливу шумів досягається з допомогою фільтрації. Lanczos – при застосуванні спостерігається найбільш точна передача контрастних границь зображення. Для зображень із високою деталізацією проявляється ефект Гіббса (ореоли навколо контурів об'єктів).

Основними недоліками методу непропорційного масштабування з використанням технології Liquid Resize є зміна взаємного розташування об'єктів, можливе виникнення невеликих спотворень на краях об'єктів при видаленні шляхів, що через них проходять, а також виникнення перепадів яскравості, які можуть проявлятися при нерівномірній освітленості зображення.

Проаналізувавши сучасні методи масштабування зображень, наведені вище, виділимо основні артефакти, які виникають при масштабуванні, що ілюструється в таблиці.

1. Ringing – виникнення хвиль біля різкої межі на зображенні.
2. Overshooting - виникнення 2 і 3 хвиль.
3. Aliasing – нерівномірності зображення на різких діагональних границях зображення.
4. Unsharpening – розмиття, недостатньо чітким є зображення після масштабування.

Очевидно, що поліпшення чіткості, як правило, приводить до збільшення інших артефактів і навпаки – зменшення артефактів зменшує також і чіткість.

5. Sub-pixel shift – субпіксельне зміщення зображення, зв'язане, як правило, з особливостями реалізації алгоритму. Практично не впливає на візуальну якість.

Метод	**Опис методу масштабування**	**Артефакти**
Nearest neighbor	Вибирається найближчий піксель в початковому зображенні	Aliasing, Unsharpening, Overshooting, зниження контрастності (різкості на границях), утворюються сходинки на різких перепадах яскравості
Bilinear	Лінійна інтерполяція по двох координатах (використовується матриця 2x2, тобто 4 найближчі пікселі)	
Bicubic	Кубічна інтерполяція по двох координатах(по матриці 4x4, тобто використовуються 16 найближчих пікселів)	
Crop	Проводиться вирізання потрібної частини зображення з потрібним співвідношенням сторін	Ringing, втрачається периферійна частина зображення, виникнення спотворень на краях об'єктів
Smart crop	Пошук частини зображення з потрібним співвідношенням сторін так, щоб при вирізанні такої частини втрати важливих частин зображення були мінімальними	
Liquid Resize	Враховує вміст зображення, при масштабуванні не спотворює основний вміст зображення	Ringing, Sub-pixel shift

5. Висновок

Запропонований перелік наведених методів дає уявлення про основні тенденції розвитку алгоритмів масштабування зображень. При виборі алгоритмів важливо розуміти їх позитивні і негативні сторони; якщо вибрано алгоритм без врахування вмісту зображення, то варто зрозуміти його властивості і умови застосування. Порівняльний аналіз найбільш відомих методів масштабування (Nearest neighbor, Bilinear, Bicubic, Lanczos, Liquid Resize) показав необхідність створення алгоритму масштабування, що дає чітке зображення і по можливості позбавлене від артефактів інших методів (розмиття, субпіксельне зміщення), які спостерігаються при застосуванні в задачах неруйнівного контролю матеріалів.

Список літератури: **1.** *Половко А.М., Бутусов П.Н.* Интерполяция. Методы и компьютерные технологии их реализации. СПб.: БХВ-Петербург, 2004. 320 с. **2.** *Яців В.Б., Русин Б.П.* Методи масштабування зображень в системах обробки та розпізнавання / Збірник праць. Науково-технічна конференція: Обчислювальні методи і системи перетворення інформації. Львів, 2010. С. 197-198. **3.** *Яне Б.* Цифровая обработка изображений. М.: Техносфера, 2007. 581 с. **4.** *Fritsh F. N., Carlson R.E.* Monotone Piecewise Cubic Interpolation / SIAM Jornal on Numerical Analysis, 1980. С. 238-246. **5.** *Avidan S., Shamir A.* Seam carving for content-aware image resizing // ACM Trans. Graph.,#26(3/2007), ACM 2007 **6.** *Crochiere R.E., Rabiner L.R.* Multirate Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, Mar. 1983.

Надійшла до редколегії 25.05.2010

Яців Віталій Богданович, аспірант ФМІ НАН України ім. Г.В. Карпенка. Наукові інтереси: обробка зображень. Адреса: Україна, 79601, Львів, вул. Наукова, 5а, тел. 229-65-30, e-mail: vicja@list.ru.

Русин Богдан Павлович, д-р техн. наук, професор, зав. відділом "Методів та систем аналізу, обробки та ідентифікації зображень" ФМІ НАН України ім. Г.В. Карпенка. Наукові інтереси: аналіз, обробка та розпізнавання зображень. Адреса: Україна, 79601, Львів, вул. Наукова, 5а, тел. 229-61-09, e-mail: rusyn@ipm.lviv.ua.

УДК 004.732

А.В. БАБИЧ, И.В. ЕМЕЛЬЯНОВ, МУРАД АЛИ А.

ИССЛЕДОВАНИЕ КОРРЕЛЯЦИОННОЙ ЗАВИСИМОСТИ МЕЖДУ КАНАЛОМ СВЯЗИ И АЛГОРИТМАМИ ПРИКЛАДНОГО ПО

Рассматривается метод корреляционного анализа зависимости между качеством работы канала связи и алгоритмами прикладного программного обеспечения (ПО). Целью данного метода является отделение задачи повышения эффективности работы алгоритмов прикладного ПО от задач мониторинга и диагностирования компьютерных сетей, поскольку данные задачи решаются отдельно и независимо. Приводится практическая реализация этого метода и результаты его применения.

Введение

Согласно [1], в компьютерных сетях (КС) часто встречается ситуация, когда необходимо определить, является ли причиной неудовлетворительного качества работы КС сама сеть или же влияние оказывают неэффективные алгоритмы работы прикладного ПО. В противном случае вероятность неоправданных затрат на модернизацию аппаратной платформы КС довольно высока, поскольку причина ее плохой работы может заключаться не в недостаточной пропускной способности сетевого оборудования или низкой производительности сервера, а в неэффективных алгоритмах самого прикладного ПО или его неправильной настройке. Под неэффективными алгоритмами работы прикладного ПО понимаются такие алгоритмы, применение которых не позволяет максимально полно использовать пропускную способность ресурсов сети (перекрытие запросов с ответами, передача данных пакетами минимальной длины и пр.).

Обратная ситуация, когда прикладное ПО неадекватно работает вследствие наличия узких мест в КС, причем возможность доказать это эмпирическим путем весьма сомнительна, тоже может иметь место.

Таким образом, следует включить в процессы мониторинга и диагностирования КС решение задачи отделения неудовлетворительного функционирования сети от неэффективных алгоритмов прикладного ПО, а разработка соответствующих методов и алгоритмов для корпоративных компьютерных сетей – весьма актуальна в настоящее время.

Постановка задачи

Для решения задачи отделения гипотезы «неудовлетворительное функционирование сети» от гипотезы «неэффективные алгоритмы прикладного ПО» предлагается ввести два наблюдаемых параметра, один из которых характеризует качество работы сети на всех уровнях ее функционирования, включая прикладной, а второй характеризует только качество работы транспортной платформы КС без учета функционирования Software-уровней сетевой модели. Первым параметром, y_1, будет время реакции прикладного ПО на запрос клиента, вторым, y_2, – скорость работы сети.

Корреляционный анализ позволит выяснить, существует ли корреляционная линейная связь между параметрами «скорость работы сети» и «время реакции ПО» и, следовательно, определить носителя неисправности: если связь существует, то носитель неисправности – компонент сети, в противном случае – прикладное ПО.

Алгоритм отделения неисправностей ПО от неисправностей КС

1) Выполняется запуск типовой задачи прикладного ПО, используемого в сети с сервера на любом из узлов-клиентов (так как по умолчанию принимается нуль-гипотеза: «прикладное ПО – носитель неисправности», инвариантная к структуре КС).

2) Выполняется генерация трафика с клиента, выполняющего типовую задачу на сервере, с последовательным увеличением нагрузки (шаг – 10%) на сеть (канал связи).

3) Параллельно с увеличением нагрузки на каждом шаге выполняется измерение каждого из наблюдаемых параметров в один момент времени. При этом скорость работы сети измеряется как время получения пакета от клиента сервером на уровне их сетевых

адаптеров. Например, анализатор протоколов Observer представляет эту информацию в виде параметра Station1-Station2 функции Time Interval Dynamics. Параметр «время реакции прикладного ПО» измеряется как “latency” – разность между запросом клиента и ответом сервера. Эта информация также представлена в Observer в виде параметра Station1/Port-Station2/Port функции Time Interval Dynamics. Так как интересует время реакции только от типовой задачи наблюдаемого прикладного ПО, то необходимо знать порты, которые используются этой типовой задачей. Это легко сделать с помощью функции Decode, встроенной в анализатор протоколов Observer. Все данные, полученные функцией Time Interval Dynamics, могут быть легко сохранены и переконвертированы в табличную форму для статистической обработки, в частности, для выполнения корреляционного анализа.

4) Количество измерений производится с учетом ограничений границ выборки для метода ранговой корреляции, а также с учетом варьирования нагрузки [2]:

$$5 < \frac{(N - n) \cdot g}{n} \le 20, \tag{1}$$

где N – количество измерений; n – количество параметров; g – количество уровней варьирования нагрузки.

5) Временной интервал между измерениями (тестами) в ходе сбора статистической информации вычисляется по формуле:

$$I_t = \frac{Q_h}{N}, \tag{2}$$

здесь I_t – интервал времени между измерениями в ходе сбора статистической информации; Q_h – время, затраченное на сбор статистической информации на каждом уровне варьирования; N - количество измерений (тестов), вычисленных по формуле (1).

6) Корреляционный анализ. Между парами параметров необходимо вычислить коэффициент парной корреляции, который является общепринятой в математической статистике характеристикой связи между двумя случайными величинами. Если обозначить один параметр через y_1 , а другой – через y_2 , и число опытов, в которых они будут измеряться, - через N так, что u=1,2,…,N, где u – текущий номер опыта, то коэффициент парной корреляции r вычисляется по формуле:

$$r_{y_1 y_2} = \frac{\sum_{u=1}^{N} (y_{1_u} - \bar{y}_1)(y_{2_u} - \bar{y}_2)}{\sqrt{\sum_{u=1}^{N} (y_{1_u} - \bar{y}_1)^2 \sum_{u=1}^{N} (y_{2_u} - \bar{y}_2)^2}}. \tag{3}$$

Здесь $\bar{y}_1 = \sum_{u=1}^{N} \frac{y_{1_u}}{N}$ и $\bar{y}_2 = \sum_{u=1}^{N} \frac{y_{2_u}}{N}$ – средние арифметические соответственно для y_1 и y_2.

Необходимо помнить, что коэффициент парной корреляции как мера тесноты связи имеет четкий математический смысл только при линейной зависимости между параметрами и в случае их нормального распределения. Для проверки значимости коэффициента парной корреляции нужно сравнить его значение с табличным (критическим) значением r [3,4]. Для пользования этой таблицей нужно знать число степеней свободы f=N-2 и выбрать определенный уровень значимости, например, равный 0,05. Такое значение уровня значимости соответствует вероятности верного ответа при проверке гипотезы p=1-a=1-0.05=0.95 или 95%. Это значит, что в среднем только в 5% случаев возможна ошибка при проверке гипотезы.

Если экспериментально найденное значение r больше или равно критическому, то гипотеза о корреляционной линейной связи подтверждается, а если меньше, то нет оснований считать, что имеется тесная линейная связь между параметрами.

Практическая реализация алгоритма

Генерация трафика (второй пункт алгоритма отделения неисправностей прикладного ПО от неисправностей КС) выполняется с помощью функции «Генератор трафика» анализатора протоколов Observer из пункта меню «Инструменты». Адрес назначения генерируемого трафика задается опцией Set Destination Address всплывающего меню поля содержимого генерируемого пакета либо параметром Destination Address (рис. 1, 2).

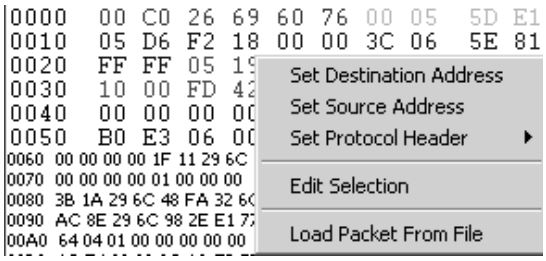


Рис.1. Задание адреса назначения в теле генерируемого пакета

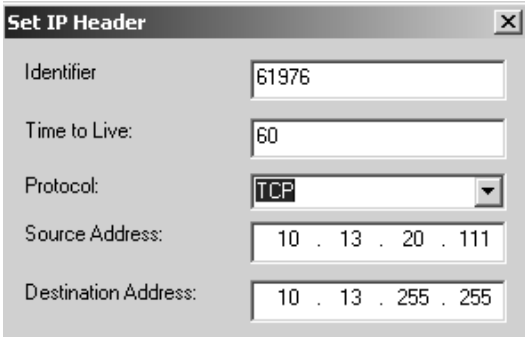


Рис. 2. Задание адреса назначения в IP-заголовке

Интенсивность трафика задается следующим образом (рис. 3).

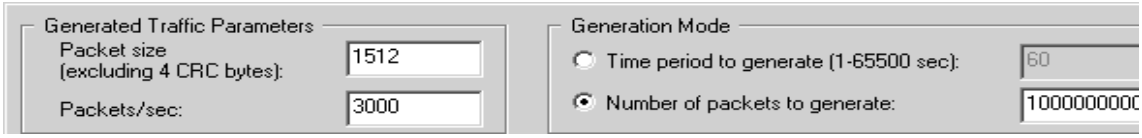


Рис. 3. Установки интенсивности трафика

Изменяемыми параметрами, влияющими на уровень интенсивности трафика, являются: размер пакета (максимально допустимый размер пакета совпадает с максимальным корректным значением размера кадра канального уровня), интенсивность передачи пак/с, период генерации трафика (временной интервал либо объем трафика в пакетах, который необходимо сгенерировать).

Выполняется генерация трафика с клиента, выполняющего типовую задачу на сервере, с последовательным увеличением нагрузки (шаг – 10%) на сеть (канал связи) (рис. 4).

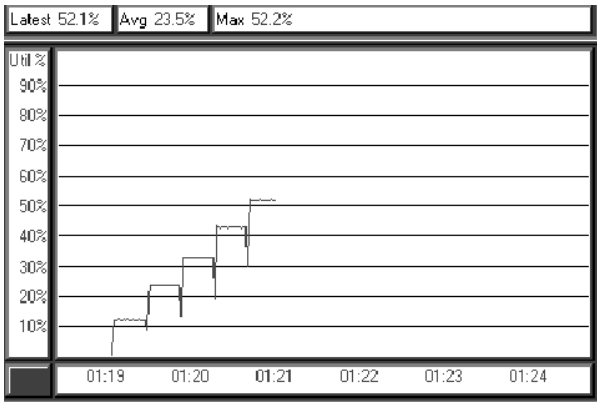


Рис. 4. Характер нагрузки при использовании метода отделения неисправностей прикладного ПО от неисправностей КС

Далее выполняется сохранение отчетов по контрольным точкам Station1-Station2 и по Station1/Port-Station2/Port (рис. 5) в одинаковом временном интервале, переконвертирование результатов в табличную форму, их совмещение и выполнение корреляционного анализа.

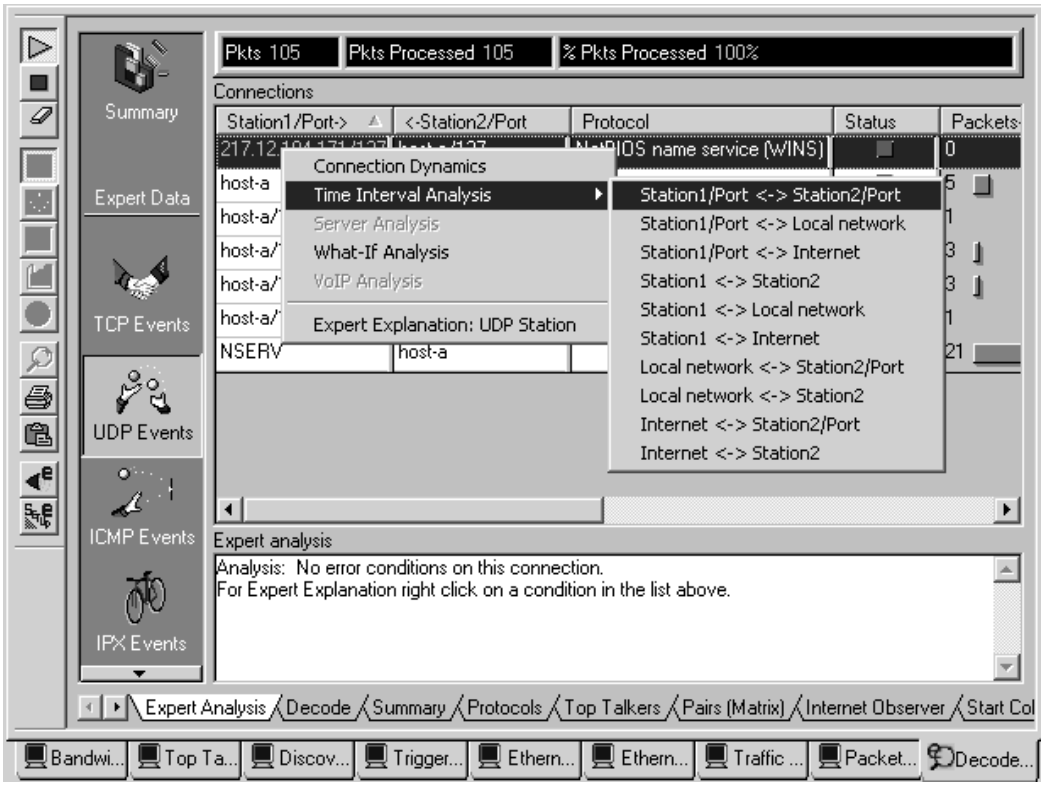


Рис. 5. Просмотр и сохранение результатов подачи ступенчатой нагрузки

Ниже приведена таблица статистических данных для корреляционного анализа, а также результаты корреляционного анализа (рис. 6). Здесь St1-St2 delay – время передачи пакета между сетевыми адаптерами Станции 1 и Станции 2. Этот параметр характеризует качество канала связи. St1/P-St2/P delay – время передачи и обработки пакета между Station1/Port и Station2/Port на прикладном уровне, этот параметр характеризует производительность прикладного ПО. Utilization – характеристика увеличения сетевой нагрузки. Количество измерений N на каждом уровне варьирования нагрузки, согласно (1), равно 11. Интервал между снятиями показаний I_t при Q_h = 30 минут составляет 3 минуты.

D	E	F	G	H	I
0,20	0,31			*Столбец 1*	*Столбец 2*
1,10	5,00		Столбец 1	1	
0,20	1,00		Столбец 2	0,99229774	1
0,50	1,00				
1,10	7,00				
0,20	1,50				

Рис. 6. Результаты корреляционного анализа в Microsoft Exel

Здесь степень свободы f=11, уровень значимости=0,05, r_{exp} = 0,993, r_{tab} по указанным условиям равно 0,553 [4]. Так как $r_{exp} > r_{tab}$, то имеет место линейная корреляция наблюдаемых параметров, и можно сделать вывод о том, что причина неудовлетворительного качества работы КС заключается в самой сети, а не в прикладном ПО.

Дальнейшие эмпирические исследования показали, что причиной медленной работы компьютерной сети было узкое место: один из коммутаторов на тракте передачи данных не справлялся с входящим потоком и отбрасывал пакеты. После замены на коммутатор с большей буферной памятью проблема с замедлениями устранилась.

Статистические данные для корреляционного анализа

Time	St1↔St2 delay,ms	St1/P↔St2/P delay,ms	Utilization, %
09:05:00:000	0.2	0.31	x...10
09:08:00:000	1.1	5	x...10
09:11:00:000	0.2	1	x...10
09:14:00:000	0.5	1	x...10
09:17:00:000	1	7	x...10
09:20:00:000	0.2	1.5	x...10
09:23:00:000	1	7	x...10
09:26:00:000	2.7	10	x...10
09:29:00:000	1	7	x...10
09:32:00:000	2.3	7	x...10
09:35:00:000	2.7	10	x...10
09:38:00:000	7.9	28	10…20
09:41:00:000	10.6	37	10…20
09:44:00:000	17.4	62	10…20
09:47:00:000	10.7	37	10…20
09:50:00:000	4.5	15	10…20
09:53:00:000	5.7	19	10…20
09:56:00:000	5.5	19	10…20
09:59:00:000	10.1	35	10…20
10:02:00:000	8.8	30	10…20
10:05:00:000	5.9	21	10…20
10:08:00:000	6.8	24	10…20
10:11:00:000	11.4	41	20…30
10:14:00:000	12.4	44	20…30
10:17:00:000	14.3	48	20…30
10:20:00:000	12.4	44	20…30
10:23:00:000	12.3	44	20…30
10:26:00:000	13.2	57	20…30
10:29:00:000	6.9	24	20…30
10:32:00:000	5.3	19	20…30
10:35:00:000	12.8	48	20…30
10:38:00:000	9.8	34	20…30
10:41:00:000	9.4	33	20…30
10:44:00:000	11	40	30…40
10:47:00:000	8.2	30	30…40
10:50:00:000	15.2	60	30…40
10:53:00:000	12.6	46	30…40
10:56:00:000	14.1	49	30…40
10:59:00:000	5.6	21	30…40
11:02:00:000	8.8	30	30…40
11:05:00:000	12.2	44	30…40
11:08:00:000	7.5	28	30…40
11:11:00:000	9	35	30…40
11:14:00:000	8.8	32	30…40

Выводы

В данной статье рассмотрен метод корреляционного анализа зависимости между качеством работы канала связи и алгоритмами прикладного ПО. Научная новизна метода заключается в применении к задаче, обозначенной в статье и традиционно решающейся эмпирическим путем, математического аппарата. Практическая значимость состоит в формализации процесса отделения задачи верификации прикладного ПО от задач мониторинга и диагностирования компьютерных сетей, что приводит к снижению трудоемкости обслуживания компьютерных сетей и поиска сетевых неисправностей. В перспективах

исследования – дальнейшее развитие применения методов статистической обработки данных к процессам мониторинга и диагностирования компьютерных сетей.

Список литературы: 1. *В. Подлазов, В. Борисенко, С. Юдицкий.* Узкие места в локальных сетях - http://www.osp.ru/lan/1998/09/133684/. **2.** *Цейтлин Н.А.* Процессы и аппараты производства основной химии: Труды НИОХИМ. Т. 56. Харьков, 1981. С. 29-38J. **3.** *Казаков Ю.Б.* Электронный конспект лекций по предмету «Методы планирования эксперимента». http://www.ispu.ru/portal/index.pl **4.** *Хамханов К.М.* Основы планирования эксперимента: Методическое пособие. Улан-Удэ: ВСГТУ, 2001. 94 с.

Поступила в редколлегию 22.05.2010

Бабич Анна Витальевна, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: сетевые технологии, технологии дистанционного образования. Увлечения: активный отдых, путешествия, иностранные языки. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: babich@kture.kharkov.ua

Емельянов Игорь Валерьевич, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: сетевые технологии, диагностика и оптимизация компьютерных систем и сетей. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

Мурад Али А., аспирант кафедры АПВТ ХНУРЭ. Научные интересы: компьютерные сети следующего поколения . Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

УДК 519.63:532.5

А.П. БЛИШУН, М.В. СИДОРОВ, И.Г. ЯЛОВЕГА

ЧИСЛЕННЫЙ АНАЛИЗ СТАЦИОНАРНЫХ ФИЛЬТРАЦИОННЫХ ТЕЧЕНИЙ СО СВОБОДНОЙ ГРАНИЦЕЙ СТРУКТУРНО-ВАРИАЦИОННЫМ МЕТОДОМ

Рассматриваются применение вариационного метода Ритца и структурного метода R-функций для численного решения задачи расчета фильтрации в области со свободной границей. Уточнение уравнения свободной границы проводится итерационно. Для модельной задачи с известным точным решением проводится вычислительный эксперимент.

Введение

Актуальность исследования. Моделирование и численный анализ течений жидкости в пористой среде имеет большое прикладное значение. Задачи такого класса возникают в инженерном деле (например, при проектировании гидротехнических сооружений), в экологии (рациональное природопользование), при разработке комплекса мер по защите населения в чрезвычайных ситуациях (например, во время паводков) и т.д. Основные сложности, возникающие при численном анализе таких систем, обусловлены разнообразием режимов течений и типов грунта, сложной геометрией области течения и возможным наличием в области фильтрации участков свободной границы. Основные методы моделирования фильтрационных течений рассмотрены в [1 – 5]. Таким образом, разработка новых и усовершенствование существующих методов математического моделирования фильтрационных течений является актуальной прикладной задачей.

Цели и задачи исследования. Целью настоящего исследования является применение для численного анализа фильтрационного течения со свободной границей структурного метода R-функций и вариационного метода Ритца. Для достижения поставленной цели необходимо решить следующие задачи:

– основываясь на построенной структуре решения краевой задачи, свести ее к вариационной задаче о минимуме функционала энергии;

– разработать алгоритм решения вариационной задачи методом Ритца;

– провести вычислительный эксперимент для модельной задачи с известным точным решением.

1. Постановка задачи

Рассмотрим двумерный случай. Предположим, что оси координат являются главным направлением тензора коэффициента фильтрации. Будем решать смешанную краевую задачу, связанную с определением подземной части гидросооружения:

$$-\frac{\partial}{\partial x}\left(k_{11}\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(k_{22}\frac{\partial u}{\partial y}\right)=0 \text{ в } \Omega, \tag{1}$$

$$u\big|_{AE}=u_1,\ u\big|_{FC}=u_3, \tag{2}$$

$$\frac{\partial u}{\partial N}+\sigma_0 u\Big|_{ABC\cup A'B'C'}=\gamma_0(x,y), \tag{3}$$

где u – напор; $\frac{\partial u}{\partial N}=k_{11}\frac{\partial u}{\partial x}\cos(\mathbf{n},x)+k_{22}\frac{\partial u}{\partial y}\cos(\mathbf{n},y)$ – производная по конормали; $\mathbf{n}$ – единичный вектор внешней нормали; u_1, u_3 – постоянные величины. Часть границы области $A'B'C'$ (рис. 1) – неизвестная линия, которую обозначим через $\partial\Omega_*$. В частном случае, когда ABC и $A'B'C'$ – непроницаемые границы, имеем $\sigma_0=0$ и $\gamma_0=0$.

Предполагаем, что точки A' и C' неизвестные. Согласно теории на свободном участке границы $\partial\Omega_*$ задается дополнительное, так называемое «геометрическое условие» (или «кинематическое условие»)

$$u=\xi(s) \text{ на } \partial\Omega_*, \tag{4}$$

где ξ означает заданный напор на подземном контуре гидросооружения.

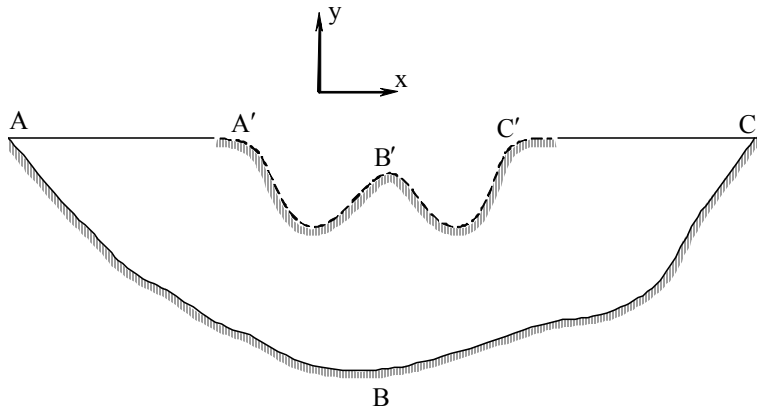


Рис. 1

2. Метод численного анализа

Будем предполагать, что уравнение (1) эллиптическое невырождающееся, т.е. при всех $t_1,\ t_2\in R\setminus\{0\}$ в Ω выполнено неравенство

$$k_{11}t_1^2+k_{22}t_2^2\ge\mu_0(t_1^2+t_2^2) \text{ в } \Omega,\ \mu_0>0.$$

Введем следующие обозначения:

$$Au\equiv-\frac{\partial}{\partial x}\left(k_{11}\frac{\partial u}{\partial x}\right)-\frac{\partial}{\partial y}\left(k_{22}\frac{\partial u}{\partial y}\right),\ Nu\equiv k_{11}\frac{\partial u}{\partial x}\cos(\mathbf{n},x)+k_{11}\frac{\partial u}{\partial y}\cos(\mathbf{n},y),$$

$\partial\Omega_*=A'B'C'$ – свободная граница, $\partial\Omega_1=AA'\cup C'C$, $\partial\Omega_2=ABC$.

Продолжая краевые условия внутрь области Ω согласно методу R-функций [6], получаем, что краевую задачу (1) – (4) можно записать в виде

$$Au=0 \text{ в } \Omega, \tag{5}$$

$$u\big|_{\partial\Omega_1}=\varphi\big|_{\partial\Omega_1}, \tag{6}$$

$$Nu+\sigma u\big|_{\partial\Omega_2\cup\partial\Omega_*}=\gamma\big|_{\partial\Omega_2\cup\partial\Omega}, \tag{7}$$

$$u=\xi(x,y) \text{ на } \partial\Omega_*. \tag{8}$$

Здесь $\gamma = EC\gamma_0$, $\sigma = EC\sigma_0$, $\varphi = \dfrac{u_3\omega' + u_1\omega''}{\omega' + \omega''}$, $\varphi|_{\partial\Omega_1} = \begin{cases} u_1 \text{ на } AA', \\ u_3 \text{ на } C'C, \end{cases}$ а функции $\omega'(x,y)$ и $\omega''(x,y)$ строятся с помощью R-функций и удовлетворяют условиям

1) $\omega'(x,y) = 0$ на AA', $\omega''(x,y) = 0$ на $C'C$;

2) $\omega'(x,y) > 0$ в $\Omega \cup A'B'C' \cup C'C \cup ABC$, $\omega''(x,y) > 0$ в $\Omega \cup A'B'C' \cup AA' \cup ABC$;

3) $\left.\dfrac{\partial\omega'}{\partial\mathbf{n}}\right|_{AA'} = -1$, $\left.\dfrac{\partial\omega''}{\partial\mathbf{n}}\right|_{C'C} = -1$, $\mathbf{n}$ – внешняя нормаль к $\partial\Omega$.

Если $\partial\Omega_*$ была бы заданной границей, то краевая задача (5) – (7) могла бы быть сведена к нахождению минимума соответствующего квадратичного функционала.

В данном случае мы сразу не можем находить минимум полученного функционала, так как область Ω, по которой ведется интегрирование, и часть её границы $\partial\Omega_*$ неизвестны. Поэтому для полного решения задачи необходимо использовать и условие (8).

Для решения поставленной задачи сформируем итерационный процесс следующим образом. Пусть получено k-е приближение к свободной границе $\partial\Omega_*^{(k)}$. Тогда получим конкретную область $\Omega^{(k)}$ с известной границей. Определим функцию $u^{(k)}(x,y)$ как решение задачи

$$Au^{(k)} = 0 \text{ в } \Omega^{(k)}, \tag{9}$$

$$u^{(k)}\big|_{\partial\Omega_1} = \varphi|_{\partial\Omega_1}, \tag{10}$$

$$Nu^{(k)} + \sigma u^{(k)}\big|_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}} = \gamma|_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}}. \tag{11}$$

Для решения задачи (9) – (11) воспользуемся методом R-функций. Пусть уравнение $\partial\Omega_*^{(k)}$ представлено в виде $y = h(x)$, причем $y - h(x) > 0$ в $\Omega^{(k)} \cup \partial\Omega_1 \cup \partial\Omega_2$, тогда $\omega_*^{(k)}(x,y) = \dfrac{y - h(x)}{\sqrt{1 + (h'(x))^2}}$ обладает следующими свойствами:

1) $\omega_*^{(k)}(x,y) = 0$ на $\partial\Omega_*^{(k)}$; 2) $\omega_*^{(k)}(x,y) > 0$ в $\Omega^{(k)} \cup \partial\Omega_1 \cup \partial\Omega_2$; 3) $\left.\dfrac{\partial\omega_*^{(k)}}{\partial\mathbf{n}}\right|_{\partial\Omega_*^{(k)}} = -1$.

Пусть $\omega_1(x,y)$ и $\omega_2(x,y)$ обладают следующими свойствами соответственно:

1) $\omega_1(x,y) = 0$ на $\partial\Omega_1$; $\omega_2(x,y) = 0$ на $\partial\Omega_2$;

2) $\omega_1(x,y) > 0$ в $\Omega \cup \partial\Omega_2 \cup \partial\Omega_*^{(k)}$; $\omega_2(x,y) > 0$ в $\Omega \cup \partial\Omega_1 \cup \partial\Omega_*^{(k)}$;

3) $\left.\dfrac{\partial\omega_1}{\partial\mathbf{n}}\right|_{\partial\Omega_1} = -1$; $\left.\dfrac{\partial\omega_2}{\partial\mathbf{n}}\right|_{\partial\Omega_2} = -1$.

В работе [6] показано, что структура решения краевой задачи (9) – (11) имеет вид

$$u^{(k)} = \varphi + \omega_1\Phi_0 + \frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\vec{l}}\tilde{\omega} + \delta\tilde{\omega}) + S\tilde{\omega}}[-SD_1^{\vec{l}}\varphi - SD_1^{\vec{l}}(\omega_1\Phi_0) + \sigma\varphi + \sigma\omega_1\Phi_0 - \gamma + \tilde{\omega}\Phi_1], \tag{12}$$

где $\delta = 1 - (\vec{l}, \nabla\tilde{\omega}) > 0$, $S = \sqrt{\left(k_{11}\dfrac{\partial\tilde{\omega}}{\partial x}\right)^2 + \left(k_{22}\dfrac{\partial\tilde{\omega}}{\partial y}\right)^2 + \tilde{\omega}^2}$, $D_1^{\vec{l}}u = \dfrac{\partial u}{\partial x}l_1 + \dfrac{\partial u}{\partial y}l_2$, $\tilde{\omega} = \omega_2 \wedge_\alpha \omega_*^{(k)}$, $\vec{l} = (l_1, l_2) = \left(\dfrac{k_{11}}{S}\dfrac{\partial\tilde{\omega}}{\partial x}, \dfrac{k_{22}}{S}\dfrac{\partial\tilde{\omega}}{\partial y}\right)$, Φ_0 и Φ_1 – неопределенные компоненты структуры. Здесь $\wedge_\alpha$ – знак R-дизъюнкции; $u \wedge_\alpha v = \dfrac{1}{1+\alpha}(u + v - \sqrt{u^2 + v^2 - 2\alpha uv})$.

Возьмем $\Phi_1 \equiv 0$ и обозначим $\Phi_0 = \Phi^{(k)}$. Перепишем (12) в виде

$$u^{(k)} = g + B(\Phi^{(k)}),$$

где

$$g = \varphi + \frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\bar{1}}\tilde{\omega} + \delta\tilde{\omega}) + S\tilde{\omega}}[-SD_1^{\bar{1}}\varphi + \sigma\varphi - \gamma],$$

$$B(\Phi^{(k)}) = \omega_1\Phi^{(k)} + \frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\bar{1}}\tilde{\omega} + \delta\tilde{\omega}) + S\tilde{\omega}}[-SD_1^{\bar{1}}(\omega_1\Phi^{(k)}) + \sigma\omega_1\Phi^{(k)}].$$

В задаче (9) – (11) сделаем замену: $u^{(k)} = g + v^{(k)}$.

Тогда для $v^{(k)}$, в силу построения функции g, получим задачу с однородными краевыми условиями:

$$Av^{(k)} = -Ag \text{ в } \Omega^{(k)}, \tag{13}$$

$$v^{(k)}\big|_{\partial\Omega_1} = 0, \tag{14}$$

$$Nv^{(k)} + \sigma v^{(k)}\big|_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}} = 0. \tag{15}$$

Оператор краевой задачи (13) – (15) действует на области определения

$$D_A = \{v \in C^2(\Omega^{(k)}) \bigcap C^1(\bar{\Omega}^{(k)}),\ Av \in L_2(\Omega^{(k)}),\ v\big|_{\partial\Omega_1} = 0,\ Nv + \sigma v\big|_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}} = 0\} \subset L_2(\Omega^{(k)})$$

и является на ней положительно определенным [7]. Энергетическое пространство этого оператора получается пополнением D_A в норме, порожденной скалярным произведением

$$[u, v] = \iint\limits_{\Omega^{(k)}} \left[k_{11}\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + k_{22}\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right] dxdy + \int\limits_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}} \sigma uvds,\ \|u\|_A = \sqrt{[u,u]}.$$

Тогда в соответствии с теоремой о функционале энергии решение задачи (13) – (15) можно найти как:

$$v^{(k)} = \arg\inf_{v \in H_A} \left\{ \iint\limits_{\Omega^{(k)}} \left[k_{11}\left(\frac{\partial v}{\partial x}\right)^2 + k_{22}\left(\frac{\partial v}{\partial y}\right)^2 \right] dxdy + \int\limits_{\partial\Omega_2 \cup \partial\Omega_*^{(k)}} \sigma v^2 ds - \right.$$

$$\left. -2\iint\limits_{\Omega^{(k)}} v\left[-\frac{\partial}{\partial x}\left(k_{11}\frac{\partial g}{\partial x}\right) - \frac{\partial}{\partial y}\left(k_{22}\frac{\partial g}{\partial y}\right)\right] dxdy \right\}.$$

Для нахождения минимума этого функционала воспользуемся методом Ритца [7]. Приближенное решение будем искать в виде $v_m^{(k)} = \sum\limits_{i=1}^{m} \alpha_i^{(k)}\varphi_i$, где $\{\varphi_i\}$ – координатная последовательность. Пусть $\{\tau_i\}$ полна в $W_2^1(\Omega^{(k)})$. Тогда в качестве $\{\varphi_i\}$ возьмем

$$\varphi_i = B(\tau_i) = \omega_1\tau_i + \frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\bar{1}}\tilde{\omega} + \delta\tilde{\omega}) + S\tilde{\omega}}[-SD_1^{\bar{1}}(\omega_1\tau_i) + \sigma\omega_1\tau_i].$$

Коэффициенты $\alpha_1^{(k)}, ..., \alpha_n^{(k)}$ найдем из системы линейных алгебраических уравнений

$$\sum_{i=1}^{m} c_i^{(k)}[\varphi_i,\ \varphi_j] = (-Ag,\ \varphi_j),\ j = 1, 2, ..., m.$$

Найденная функция $u_m^{(k)} = g + v_m^{(k)}$ является приближенным решением задачи (9) – (11) и условию на свободной границе (8) не удовлетворяет.

Условие (8) используем для уточнения уравнения свободной границы. Уравнение свободной границы $\partial\Omega_*^{(k+1)}$ для следующей $(k+1)$-й итерации ищем в виде:

$$y^{(k+1)} = c_0^{(k+1)} + \sum_{r=1}^{n} c_r^{(k+1)} x^r, \tag{16}$$

где m – натуральное число; $c_r^{(k+1)}$ $(r = 0, 1, 2, ..., n)$ – неизвестные коэффициенты.

Пусть $a^{(k)}$ и $c^{(k)}$ – абсциссы точек $A'^{(k)}$ и $C'^{(k)}$. Тогда положим $d^{(k)} = \left|c^{(k)} - a^{(k)}\right|$. Возьмем натуральное число N такое, что $N > n$ и на отрезке с концами $a^{(k)}, c^{(k)}$ зададим сетку $x_l = a^{(k)} + \frac{N-1}{N-1} d^{(k)}$, $l = 1, 2, ..., N$. Положим $y_l^{(k+1)} = \sum_{r=0}^{n} c_r^{(k+1)} x_l^r$. Кроме единичной нормали $\mathbf{n} = (n_1, n_2)$ ведем в рассмотрение конормаль $\nu = (\nu_1, \nu_2)$ согласно правилу: $\nu_1 = \frac{k_{11} n_1}{S}$, $\nu_2 = \frac{k_{22} n_2}{S}$, где $S = [(k_{11},\ n_1)^2 + (k_{22},\ n_2)^2]^{\frac{1}{2}}$, $n_1 = \cos(\mathbf{n},\ x)$, $n_2 = \cos(\mathbf{n},\ y)$. Для вычисления координат точек $\tilde{P}_l$, лежащих на $\partial\Omega_*^{(k+1)}$, имеем [6]:

$$\tilde{x}_l = x_l + \cos(\nu, x) \left| \frac{u_m^{(k)}(P_l) - \xi(P_l)}{\frac{\partial \xi(P_l)}{\partial \nu} - Q(P_l)} \right|, \quad \tilde{y}_l = y_l + \cos(\nu, y) \left| \frac{u_m^{(k)}(P_l) - \xi(P_l)}{\frac{\partial \xi(P_l)}{\partial \nu} - Q(P_l)} \right|. \tag{17}$$

Для нахождения коэффициентов в уравнении кривой $\partial\Omega_*^{(k+1)}$ потребуем, чтобы кривая каким-то образом определялась через множество точек $\tilde{P}_l = (\tilde{x}_l,\ \tilde{y}_l)$, $l = 1, 2, ..., N$. Для этого воспользуемся точечным методом наименьших квадратов: потребуем, чтобы выражение $R = \sum_{l=1}^{N} \left[\tilde{y}_l - \sum_{r=0}^{n} c_r^{(k+1)} \psi_r(\tilde{x}_l) \right]^2$ имело минимальное значение, где значения $\tilde{x}_l$ и $\tilde{y}_l$ берутся из (17). После определения всех параметров из (16) кривая $y^{(k+1)}$ будет известной, значит, имеем известную область $\Omega^{(k+1)}$. На следующей $(k+2)$-й итерации строится приближенное решение $u_m^{(k+2)}$ для новой области $\Omega^{(k+1)}$. Итерация проводится до тех пор, пока "расстояние" между $\partial\Omega_*^{(k)}$ и $\partial\Omega_*^{(k+1)}$ не станет достаточно малым: $\max_{1 \le l \le N} \sqrt{(x_l^{(k+1)} - x_l^{(k)})^2 - (y_l^{(k+1)} - y_l^{(k)})^2} < \varepsilon$, где $\varepsilon > 0$ – наперед заданное достаточно малое число.

3. Модельный пример

Вычислительный эксперимент был проведен для задачи

$$-\frac{\partial}{\partial x}\left(\frac{1}{1-y^2}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\frac{1}{(1+x)^2}\frac{\partial u}{\partial y}\right) = 0 \text{ в } \Omega, \tag{18}$$

$$u\big|_{AC} = 1 - y^2, \tag{19}$$

$$\frac{1}{1-y^2}\frac{\partial u}{\partial x}\cos(\mathbf{n}, x) + \frac{1}{(1+x)^2}\frac{\partial u}{\partial y}\cos(\mathbf{n}, y)\bigg|_{BC} = -\frac{3 + 3x + 16y^2}{\sqrt{9 + 64y^2}(1+x)^3}, \tag{20}$$

$$\frac{1}{1-y^2}\frac{\partial u}{\partial x}\cos(\mathbf{n}, x) + \frac{1}{(1+x)^2}\frac{\partial u}{\partial y}\cos(\mathbf{n}, y)\bigg|_{AB} = 0, \tag{21}$$

$$u\big|_{BC} = \frac{3}{4}. \tag{22}$$

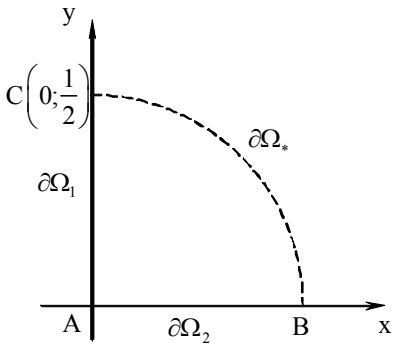


Рис. 2

Область Ω представлена на рис. 2.

Точку C считаем известной, положение точки B неизвестно, известно лишь, что она находится на прямой Ox, BC – есть свободная поверхность. Точное решение этой задачи имеет вид $u(x,y)=\dfrac{1-y^2}{1+x}$, $y=\dfrac{1}{2}\sqrt{1-3x}$. В качестве начального приближения к уравнению свободной границы возьмем прямую, соединяющую точки $C\left(0,\ \dfrac{1}{2}\right)$ и $B^{(0)}\left(\dfrac{2}{5},\ 0\right)$: $y=h^{(0)}(x)=-\dfrac{5}{4}x+\dfrac{1}{2}$. Построим решение задачи (18) – (22) в области $\Omega^{(0)}=ACB^{(0)}$ по алгоритму, описанному в разделе 2. Для задачи (18) – (22)

$$\omega_*^{(0)}(x,y)=\frac{-y-1{,}25x-0{,}5}{\sqrt{2{,}5625}},\ \omega_1(x,y)=x,\ \omega_2(x,y)=y,\ \tilde{\omega}(x,y)=y\wedge_\alpha\frac{-y-1{,}25x-0{,}5}{\sqrt{2{,}5625}},$$

$$\varphi=1-y^2,\ \sigma=0,\ \gamma=\frac{-\dfrac{3+3x+16y^2}{\sqrt{9+64y^2}(1+x)^3}y}{y+\dfrac{-y-1{,}25x+0{,}5}{\sqrt{2{,}5625}}},\ S=\sqrt{\left(\frac{1}{1-y^2}\frac{\partial\tilde{\omega}}{\partial x}\right)^2+\left(\frac{1}{(1+x)^2}\frac{\partial\tilde{\omega}}{\partial y}\right)^2+\tilde{\omega}^2},$$

$$\vec{l}=\left(\frac{\dfrac{1}{1-y^2}\dfrac{\partial\tilde{\omega}}{\partial x}}{S};\ \ \frac{\dfrac{1}{(1+x)^2}\dfrac{\partial\tilde{\omega}}{\partial y}}{S}\right),\ D_1^{\vec{l}}u=\frac{1}{S}\left[\frac{1}{1-y^2}\frac{\partial u}{\partial x}\frac{\partial\tilde{\omega}}{\partial x}+\frac{1}{(1+x)^2}\frac{\partial u}{\partial y}\frac{\partial\tilde{\omega}}{\partial y}\right].$$

Структура решения краевой задачи (18) – (21) имеет вид

$$u^{(0)}(x,y)=\varphi+\omega_1\Phi_0+\frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\vec{l}}\tilde{\omega}+\delta\tilde{\omega})+S\tilde{\omega}}[-SD_1^{\vec{l}}\varphi-SD_1^{\vec{l}}(\omega_1\Phi_0)+\sigma\varphi+\sigma\omega_1\Phi_0-\gamma+\tilde{\omega}\Psi_2].$$

Неопределённую компоненту $\Phi_0=\Phi^{(0)}$ аппроксимируем выражением вида

$$\Phi^{(0)}\approx\Phi_m^{(0)}=\sum_{i+j=0}^{m}\alpha_{ij}\tau_{ij}=\sum_{i+j=0}^{m}\alpha_{ij}x^iy^j.$$

Числа α_{ij}, $i+j=0,\ 1,\ ...,\ m$, найдем из условия минимума соответствующего функционала энергии. Координатные функции возьмем в виде

$$\varphi_{ij}(x,y)=\omega_1\tau_{ij}-\frac{\omega_1\tilde{\omega}}{\omega_1(SD_1^{\vec{l}}\tilde{\omega}+\delta\tilde{\omega})+S\tilde{\omega}}SD_1^{l}(\omega_1\tau_{ij}),$$

где $\delta=1-\dfrac{1}{S}\left[\dfrac{1}{1-y^2}\left(\dfrac{\partial\tilde{\omega}}{\partial x}\right)^2+\dfrac{1}{(1+x)^2}\left(\dfrac{\partial\omega}{\partial y}\right)^2\right]$.

Используя значение $u_5^{(0)}(x,y)$, уточним уравнение свободной границы с привлечением условия (22). Будем искать уравнение $\partial\Omega_*^{(1)}$ в виде

$$y^{(1)}=c_0^{(1)}+c_1^{(1)}x+c_2^{(1)}x^2. \qquad (23)$$

Возьмем $N=5$, $x_l=\dfrac{5-l}{10}$, $l=1,2,...,5$.

На $\partial\Omega_*^{(0)}$ $\cos(\nu,x)=\dfrac{1}{S}\dfrac{1}{1-y^2}\dfrac{\partial\tilde{\omega}}{\partial x}\Big|_{\partial\Omega_*^{(0)}}$, $\cos(\nu,x)=\dfrac{1}{S}\dfrac{1}{(1+x)^2}\dfrac{\partial\tilde{\omega}}{\partial y}\Big|_{\partial\Omega_*^{(0)}}$, $P_1=(x_1,y_1)$ – точки, лежащие на $\partial\Omega_*^{(0)}$: $y_1=-\dfrac{5}{4}x_1+\dfrac{1}{2}$.

По формулам (17) находим

$$\tilde{x}_1=x_1+\frac{1}{S}\frac{1}{1-y^2}\frac{\partial\tilde{\omega}}{\partial x}\Big|_{(x_1,y_1)}\quad\frac{u_5^{(0)}(x_1,y_1)-\frac{3}{4}}{-\frac{1}{S}\gamma\Big|_{(x_1,y_1)}},\ \tilde{y}_1=y_1+\frac{1}{S}\frac{1}{(1+x)^2}\frac{\partial\tilde{\omega}}{\partial y}\Big|_{(x_1,y_1)}\quad\frac{u_5^{(0)}(x_1,y_1)-\frac{3}{4}}{-\frac{1}{S}\gamma\Big|_{(x_1,y_1)}}.$$

Тогда коэффициенты $c_0^{(1)}$, $c_1^{(1)}$, $c_2^{(1)}$ в (23) найдем из условия

$$\sum_{i=1}^{5}[\tilde{y}_1-(c_0^{(1)}+c_1^{(1)}\tilde{x}_1+c_2^{(1)}\tilde{x}_1^2)]^2\to\min_{c_1,c_2,c_3\in R}.$$

Сходимость с точностью $\varepsilon=10^{-3}$ достигнута на 7-й итерации. Сравнение с точным решением показало, что $\left\|u_5^{(7)}-u\right\|_{L_2(\Omega^{(7)})}=0{,}12\cdot10^{-3}$.

Выводы

Впервые на основании структурно-вариационного метода разработан метод численного анализа стационарных фильтрационных течений в областях со свободной границей. Разработанные нами методы могут быть использованы при проектировании сложных гидродинамических сооружений: плотин, дамб и т.п. Этим и определяется *научная новизна* и *практическая значимость* полученных результатов.

Список литературы: 1. *Бурак Я.Й., Чапля Є.Я., Чернуха О.Ю.* Континуально-термодинамічні моделі механіки твердих розчинів. К.: Наук. думка, 2006. 272с. **2.** *Ляшко И.И., Сергиенко Н.В., Мистецкий Г.Е., Скопецкий В.В.* Вопросы автоматизации решения задач фильтрации на ЭВМ. К.: Наук. думка, 1977. 288 с. **3.** *Ляшко Н.И., Великоиваненко Н.М.* Численно-аналитическое решение краевых задач теории фильтрации. К.: Наук. думка, 1973. 264 с. **4.** *Шаманский В.Е.* Численное решение задач фильтрации грунтовых вод на ЭЦВМ. К.: Наук. думка, 1969. 376 с. **5.** *Вабищевич П.Н.* Численные методы решения задач со свободной границей. М.: Изд-во Моск. ун-та, 1987. 164 с. **6.** *Блишун А.П., Сидоров М.В., Яловега И.Г.* Математическое моделирование стационарных фильтрационных течений со свободной границей методом R-функций // АСУ и приборы автоматики. 2010. № 150. С. 18-27. **7.** *Михлин С.Г.* Вариационные методы в математической физике. М.: Наука, 1970. 512 с.

Поступила в редколлегию 05.06.2010

Блишун Александр Павлович, студент группы ПМм-09-1 фак-та прикладной математики и менеджмента ХНУРЭ. Научные интересы: математическое моделирование, численные методы математической физики. Увлечения и хобби: покер, футбол. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057) 7021436.

Сидоров Максим Викторович, канд. физ.-мат. наук, доц. каф. прикладной математики ХНУРЭ. Научные интересы: математическое моделирование, численные методы, математическая физика, теория R-функций и её приложения, стохастический анализ и его приложения. Увлечения и хобби: всемирная история, история искусств. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057) 7021436.

Яловега Ирина Георгиевна, канд. техн. наук, ст. преп. каф. прикладной математики ХНУРЭ. Научные интересы: математическое моделирование, анализ динамических систем, стохастический анализ и его приложения. Увлечения и хобби: оригами. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057) 7021436.

УДК 658.52.011.56

С.А.КАНЦЕДАЛ, М.В.КОСТИКОВА

ПРОСТОЙ МЕТОД РАСКРАСКИ ГРАФА

Описывается новый метод раскраски графа. Доказана его корректность. Приводится пример, иллюстрирующий метод.

1. Постановка задачи

Разнообразные задачи планирования производства, хранения и транспортировки грузов, составления графиков осмотра оборудования и т.п. могут быть сформулированы как задачи оптимальной раскраски графа $G = (V, E)$ с множеством вершин V и рёбер E.

В свою очередь суть оптимальной раскраски G состоит в выборе для окрашивания его вершин наименьшего количества цветов $1, 2, \ldots$ так, чтобы никакие смежные вершины графа не были окрашены в один цвет. Полученный граф принято называть r-хроматическим (раскрашенным), а минимальное число цветов (красок) r – хроматическим числом графа G [1].

Рассмотрим следующую задачу. Для выполнения n видов работ используется m типов ресурсов. Все работы выполняются за одинаковое время, но присеняют различные типы ресурсов. Сокращение общего времени выполнения всех n работ может быть доступно тогда, когда большинство из них выполняется параллельно (одновременно). Требуется так распределить типы ресурсов, чтобы минимизировать общее время выполнения работ.

Сформулируем задачу в терминах графов. Для этого каждой работе поставим в соответствие его вершины ϑ_i, $i = 1, 2, \ldots, n$. Вершины ϑ_i, ϑ_j соединим ребром тогда и только тогда, когда для выполнения i-й и j-й работ требуется один и тот же тип ресурса. Это говорит о том, что указанные работы не могут выполняться одновременно.

Раскраска вершин полученного графа при помощи наименьшего количества красок будет означать, что выделено наименьшее число его рёбер, т.е. тех пар работ, которые не могут выполняться одновременно. В свою очередь это будет означать, что остальные работы могут выполняться параллельно и, следовательно, решение задачи о раскраске графа даст оптимальное решение поставленной задачи.

Цель исследования – разработка нового алгоритма раскраски графа.

Задача для достижения цели – изучение существующих алгоритмов.

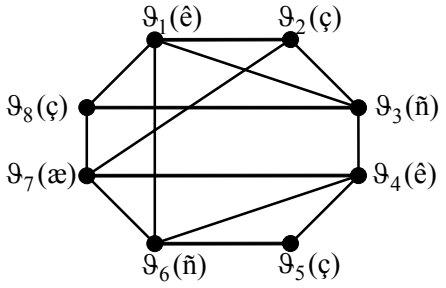
2. Метод раскраски

Задача о раскраске графов изучалась примерно с середины XIX века. Для планарных графов строго доказано, что их оптимальная раскраска требует не больше пяти цветов. Гипотеза о том, что такие графы могут быть раскрашены не больше, чем четырьмя цветами, доказана при помощи ЭВМ в 1976 г.

Для решения задачи о раскраске обычных графов предложены как точные, так и приближённые методы. В связи с тем, что задача о раскраске графа может быть представлена как задача целочисленного линейного программирования с булевыми переменными или сведена к задаче о наименьшем покрытии, она может быть решена методами решения задач целочисленного линейного программирования – отсечения либо по схеме ветвей и границ, либо теми методами, которые применяются для решения задачи о наименьшем покрытии. Кроме этих методов для решения задачи о раскраске предложен метод динамического программирования и прямого перебора (исчерпывающий поиск), использующий дерево поиска [2].

Мы предлагаем один из самых простых методов решения задачи о раскраске графа $G = (V, E)$. Он основан на последовательном окрашивании предварительно упорядоченных вершин графа, состоит из тактов и на каждом такте окрашивания реализует идею максимально возможного окрашивания вершин G в один цвет.

Суть метода объясним на примере. Пусть задан граф $G = (V, E)$, изображенный на рис. 1. Вершины графа отмечены символами $\vartheta_1, ..., \vartheta_8$. Рядом представлен список множеств несмежных вершин всех вершин G.



$\vartheta_1 : \{\vartheta_4, \vartheta_5, \vartheta_7\}$,
$\vartheta_2 : \{\vartheta_4, \vartheta_5, \vartheta_6, \vartheta_8\}$,
$\vartheta_3 : \{\vartheta_5, \vartheta_6, \vartheta_7\}$,
$\vartheta_4 : \{\vartheta_1, \vartheta_2, \vartheta_8\}$,
$\vartheta_5 : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_7, \vartheta_8\}$,
$\vartheta_6 : \{\vartheta_2, \vartheta_3, \vartheta_8\}$,
$\vartheta_7 : \{\vartheta_1, \vartheta_3, \vartheta_5\}$,
$\vartheta_8 : \{\vartheta_2, \vartheta_4, \vartheta_5, \vartheta_6\}$.

Рис. 1. Граф G и список множеств несмежных вершин

Таким образом, предлагаемый метод раскраски G предусматривает предварительное составление указанных множеств. После этого вершины G упорядочиваются по невозрастанию мощностей множеств несмежных вершин. В том случае, когда мощности некоторых множеств одинаковы, они относительно друг друга располагаются в произвольном порядке. В результате для рассматриваемого примера получаем последовательность вершин ϑ_5, ϑ_2, ϑ_8, ϑ_1, ϑ_3, ϑ_4, ϑ_6, ϑ_7, которую обозначим γ.

Теперь приступим к окраске вершин G. Для этого первой вершине ϑ_5 последовательности γ назначим цвет 1, например, зелёный и обозначим её $\vartheta_5(ç)$. Далее в множестве вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_7, \vartheta_8\}$ несмежных вершине $\vartheta_5(ç)$ найдём вершину, ближайшую к вершине ϑ_5 в последовательности γ. Такой является вершина ϑ_2. Этой вершине также назначим цвет 1 и обозначим её $\vartheta_2(ç)$.

После этого из множеств вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_7, \vartheta_8\}$, $\{\vartheta_4, \vartheta_5, \vartheta_6, \vartheta_8\}$, несмежных вершинам ϑ_5, ϑ_2, исключаем окрашенные вершины ϑ_2, ϑ_5 и находим множество несмежных вершин для суммы $\vartheta_5 + \vartheta_2$.

Если множество таких вершин для $\vartheta_5 + \vartheta_2$ не пусто, найдём ближайшую к началу γ_5 вершину этого множества и назначим ей также цвет 1. Такие действия выполняем до тех пор, пока множество несмежных вершин накапливающейся суммы окрашиваемых вершин G не окажется пустым. Когда это произойдёт, в последовательности γ найдём ближайшую к её началу неокрашенную вершину и назначим ей цвет 2.

Теперь описанные действия выполним с цветом 2. После окраски вершин цветом 2 переходим к цвету 3 и т.д. Процесс окраски графа завершаем тогда, когда будет окрашена последняя вершина последовательности γ.

Множество несмежных вершин для $\vartheta_5 + \vartheta_2$ – это пересечение множеств вершин $\{\vartheta_1, \vartheta_3, \vartheta_7, \vartheta_8\}$ и $\{\vartheta_4, \vartheta_6, \vartheta_8\}$, несмежных вершинам ϑ_5, ϑ_2, из которых исключены вершины ϑ_2, ϑ_5. Оно равно $\{\vartheta_8\}$. Следовательно, вершине ϑ_8 также назначим цвет 1 и обозначим её $\vartheta_8(ç)$.

Теперь определим множество несмежных вершин для суммы $\vartheta_5 + \vartheta_2 + \vartheta_8$. Из $\{\vartheta_8\}$ исключаем окрашенную вершину ϑ_8. В результате получаем пустое множество. А из множества вершин $\{\vartheta_2, \vartheta_4, \vartheta_5, \vartheta_6\}$, несмежных вершине ϑ_8, исключаем окрашенные вер-

шины ϑ_2, ϑ_5 и получаем $\{\vartheta_4, \vartheta_6\}$. Так как пересечение пустого множества с любым множеством – пустое множество, окраску вершин цветом 1 завершаем и назначаем цвет 2 – красный – ближайшей к началу последовательности γ неокрашенной вершине ϑ_1. Обозначим её $\vartheta_1(\hat{e})$.

Из множества вершин $\{\vartheta_4, \vartheta_5, \vartheta_7\}$, несмежных вершине $\vartheta_1(\hat{e})$, исключаем окрашенную вершину ϑ_5 и в полученном множестве $\{\vartheta_4, \vartheta_7\}$ в цвет 2 окрасим ближайшую к началу последовательности γ вершину ϑ_4. Обозначим её $\vartheta_4(\hat{e})$.

Определим множество несмежных вершин для суммы $\vartheta_1 + \vartheta_4$. Для этого из множества $\{\vartheta_4, \vartheta_7\}$ исключаем окрашенную вершину ϑ_4 и получаем множество $\{\vartheta_7\}$, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_8\}$, несмежных вершине ϑ_4, исключаем окрашенные вершины $\vartheta_1, \vartheta_2, \vartheta_8$ и получаем пустое множество. Так как пересечение полученных множеств пусто, ближайшей к началу последовательности γ неокрашенной вершине ϑ_3 назначим цвет 3 – синий – и обозначим её $\vartheta_3(\tilde{n})$.

Из множества вершин $\{\vartheta_5, \vartheta_6, \vartheta_7\}$, несмежных вершине ϑ_3, исключим окрашенную вершину ϑ_5 и в полученном множестве $\{\vartheta_6, \vartheta_7\}$ назначим ближайшей к началу последовательности γ вершине ϑ_6 тоже цвет 3. Обозначим её $\vartheta_6(\tilde{n})$.

Далее определим множество несмежных вершин для суммы $\vartheta_3 + \vartheta_6$. Для этого из множества $\{\vartheta_6, \vartheta_7\}$ исключаем окрашенную вершину ϑ_6, а из множества несмежных вершин $\{\vartheta_2, \vartheta_3, \vartheta_8\}$ – окрашенные вершины ϑ_2, ϑ_3, ϑ_8. Так как пересечение полученных множеств пусто, назначим ближайшей к началу последовательности γ неокрашенной вершине ϑ_7 цвет 4 – жёлтый – и обозначим её $\vartheta_7(\text{æ})$.

Из множества вершин $\{\vartheta_1, \vartheta_3, \vartheta_5\}$, несмежных вершине ϑ_7, исключим окрашенные вершины ϑ_1, ϑ_3, ϑ_5. В результате получим пустое множество. Это означает, что нет вершины, которую следует окрасить в цвет 4, т.е. все вершины последовательности γ, а следовательно, графа G окрашены. Для этого понадобилось четыре цвета и, таким образом, хроматическое число графа $r = 4$.

Правильность описанного метода окраски графа и определения наименьшего его хроматического числа вытекает из следующего.

Метод реализует последовательную окраску вершин графа, не пропуская ни одной его вершины. В связи с тем, что число вершин G конечно, метод также конечен, так как останавливается после их окраски.

С другой стороны, метод выполняет последовательность этапов окраски вершин. Начало каждого этапа – задание очередного нового цвета вершине G, а содержимое этапа включает действия, направленные на окраску одним цветом по возможности наибольшего числа несмежных вершин G, причём эти действия выполняются с позиций не одной вершины, инициировавшей окраску, а накапливаемой суммы окрашенных одним цветом вершин. Тем самым гарантируется выполнение условия: никакие смежные вершины не окрашиваются в один цвет.

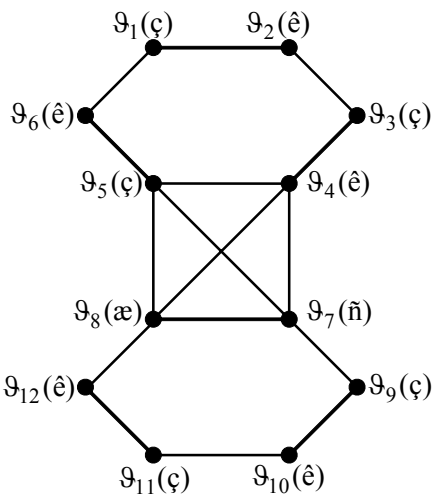
Поскольку число вершин G конечно и на каждом этапе в один цвет окрашивается максимальное число несмежных вершин, число этапов окраски и, следовательно, цветов будет минимальным.

Полученные цвета окраски вершин G показаны на рис. 1 в скобках возле обозначения каждой вершины. Легко проследить, что в результате раскраски G никакие пары смежных ему вершин не окрашены в один цвет.

В заключение решим задачу раскраски графа, хроматическое число которого $r = 4$ известно [2].

3. Пример

Граф $G = (V, E)$, взятый из [2], изображён на рис. 2. Рядом приведены множества вершин несмежности перечня его вершин.



$\vartheta_1 : \{\vartheta_3, \vartheta_4, \vartheta_5, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_2 : \{\vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_3 : \{\vartheta_1, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_4 : \{\vartheta_1, \vartheta_2, \vartheta_6, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_5 : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_6 : \{\vartheta_2, \vartheta_3, \vartheta_4, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_7 : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_6, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_8 : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_6, \vartheta_9, \vartheta_{10}, \vartheta_{11}\}$,
$\vartheta_9 : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_8, \vartheta_{11}, \vartheta_{12}\}$,
$\vartheta_{10} : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_{12}\}$,
$\vartheta_{11} : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_9\}$,
$\vartheta_{12} : \{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_9, \vartheta_{10}\}$.

Рис. 2. Граф G=(V, E) и множества верши несмежности

Расположение вершин G по невозрастанию мощностей множеств вершин несмежности даёт следующую последовательность: $\gamma = (\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_6, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}, \vartheta_4, \vartheta_5, \vartheta_7, \vartheta_8)$.

Окрасим вершину ϑ_1 в первый цвет – зелёный и обозначим её ϑ_1(ç). Ближайшей вершиной к началу последовательности γ, несмежной вершине ϑ_1, является вершина ϑ_3. Её окрасим также в первый цвет и обозначим ϑ_3(ç).

Определим ближайшую к началу γ вершину, несмежную сумме $\vartheta_1 + \vartheta_3$. Для этого из множеств вершин несмежности ϑ_1, ϑ_3 удалим вершины ϑ_3, ϑ_1 и найдём пересечение множеств оставшихся вершин. Получим

$$\{\vartheta_4, \vartheta_5, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\} \cap \{\vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\} = \{\vartheta_5, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}.$$

Вершиной пересечения, ближайшей к началу γ, является ϑ_9. Её окрасим в первый цвет и обозначим ϑ_9(ç). Далее находим вершину, несмежную сумме $\vartheta_1 + \vartheta_3 + \vartheta_9$.

Из пересечения $\{\vartheta_5, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$ удаляем вершину ϑ_9, а из множества вершин, несмежных вершине ϑ_9, – окрашенные вершины ϑ_1, ϑ_3. В результате получаем следующие множества: $\{\vartheta_5, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$, $\{\vartheta_2, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_8, \vartheta_{11}, \vartheta_{12}\}$. Их пресечение даёт множество $\{\vartheta_5, \vartheta_8, \vartheta_{11}, \vartheta_{12}\}$. Вершиной пересечения, ближайшей к началу γ, является ϑ_{11}. Её также окрашиваем в первый цвет и обозначаем ϑ_{11}(ç).

Находим вершину, несмежную сумме $\vartheta_1 + \vartheta_3 + \vartheta_9 + \vartheta_{11}$. Для этого из пересечения $\{\vartheta_5, \vartheta_8, \vartheta_{11}, \vartheta_{12}\}$ удаляем вершину ϑ_{11}, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_9\}$, несмежных вершине ϑ_{11}, предварительно окрашенные вершины ϑ_1, ϑ_3, ϑ_9. В результате получаем множества $\{\vartheta_5, \vartheta_8, \vartheta_{12}\}$, $\{\vartheta_2, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8\}$. Их пересечение образует множество $\{\vartheta_5, \vartheta_8\}$. Вершиной пересечения, ближайшей к началу γ, является ϑ_5. Её также окрашиваем в первый цвет и обозначаем ϑ_5(ç).

Находим вершину, несмежную сумме $\vartheta_1 + \vartheta_3 + \vartheta_9 + \vartheta_{11} + \vartheta_5$. Для этого из пересечения $\{\vartheta_5, \vartheta_8\}$ удаляем вершину ϑ_5, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$, несмежных вершине ϑ_5, – предварительно окрашенные вершины ϑ_1, ϑ_3, ϑ_9, ϑ_{11}. В результате получаем множества $\{\vartheta_8\}$, $\{\vartheta_2, \vartheta_{10}, \vartheta_{12}\}$. Так как пересечение полученных множеств пусто, ближайшей к началу последовательности γ неокрашенной вершине ϑ_2 назначим второй цвет – красный – и обозначим её ϑ_2(ê).

Удаляем из множества вершин, несмежных вершине ϑ_2, окрашенные вершины ϑ_5, ϑ_9, ϑ_{11}. В результате получаем множество $\{\vartheta_4, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$. Ближайшей к началу последовательности γ вершиной в этом множестве является вершина ϑ_6. Окрашиваем её во второй цвет и обозначаем ϑ_6(ê).

Находим несмежную вершину суммы $\vartheta_2 + \vartheta_6$. Для этого из множества $\{\vartheta_4, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$ удаляем вершину ϑ_6, а из множества вершин $\{\vartheta_2, \vartheta_3, \vartheta_4, \vartheta_7, \vartheta_8, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$, несмежных вершине ϑ_6, – предварительно окрашенные вершины ϑ_2, ϑ_3, ϑ_9, ϑ_{11}. Пересечение полученных множеств $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$, $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$ даёт множество $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$. Ближайшей к началу последовательности γ вершиной в этом множестве является вершина ϑ_{10}. Окрашиваем её во второй цвет и обозначаем $\vartheta_{10}(\hat{e})$.

Определяем вершину, несмежную сумме $\vartheta_2 + \vartheta_6 + \vartheta_{10}$. Для этого из пересечения $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{10}, \vartheta_{12}\}$ удаляем вершину ϑ_{10}, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_8, \vartheta_{12}\}$, несмежных вершине ϑ_{10}, – предварительно окрашенные вершины ϑ_1, ϑ_2, ϑ_3, ϑ_5, ϑ_6, ϑ_9, ϑ_{10}, ϑ_{11}. Находим пересечение множеств $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{12}\}$, $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{12}\}$, в результате чего получаем $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{12}\}$. Ближайшей вершиной к началу последовательности γ в этом множестве является вершина ϑ_{12}. Её окрашиваем во второй цвет и обозначаем $\vartheta_{12}(\hat{e})$.

Находим вершину, несмежную сумме $\vartheta_2 + \vartheta_6 + \vartheta_{10} + \vartheta_{12}$. Для этого из пересечения $\{\vartheta_4, \vartheta_7, \vartheta_8, \vartheta_{12}\}$ удаляем вершину ϑ_{12}, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6, \vartheta_7, \vartheta_9, \vartheta_{10}\}$, несмежных вершине ϑ_{12}, – предварительно окрашенные вершины ϑ_1, ϑ_2, ϑ_3, ϑ_5, ϑ_6, ϑ_9, ϑ_{10}, ϑ_{11}, ϑ_{12}. Пересечение полученных множеств $\{\vartheta_4, \vartheta_7, \vartheta_8\}$, $\{\vartheta_4, \vartheta_7\}$ даёт множество $\{\vartheta_4, \vartheta_7\}$. Вершиной, ближайшей к началу последовательности γ в этом множестве, является ϑ_4. Окрашиваем её во второй цвет и обозначаем $\vartheta_4(\hat{e})$.

Находим вершину, несмежную сумме $\vartheta_2 + \vartheta_6 + \vartheta_{10} + \vartheta_{12} + \vartheta_4$. Для этого из пересечения $\{\vartheta_4, \vartheta_7\}$ удаляем вершину ϑ_4, а из множества вершин $\{\vartheta_1, \vartheta_2, \vartheta_6, \vartheta_9, \vartheta_{10}, \vartheta_{11}, \vartheta_{12}\}$, несмежных вершине ϑ_4, – предварительно окрашенные вершины ϑ_1, ϑ_2, ϑ_3, ϑ_5, ϑ_6, ϑ_9, ϑ_{10}, ϑ_{11}, ϑ_{12}, ϑ_4. Пересечение полученных множеств $\{\vartheta_7\}$, $\{\varnothing\}$ – пустое множество. Поэтому неокрашенной вершине ϑ_7, ближайшей к началу последовательности γ, назначим третий цвет и обозначим её $\vartheta_7(\tilde{n})$.

Множество вершин, несмежных вершине ϑ_7, пусто, так как все они окрашены. Поэтому оставшейся неокрашенной вершине ϑ_8 назначаем четвертый цвет – жёлтый и обозначаем её $\vartheta_8(æ)$.

Таким образом, для окраски вершин графа потребовалось четыре цвета, т.е. хроматическое его число $r = 4$. Это совпадает с данными [2, с. 76]. На рис. 2 назначенные цвета проставлены в скобках возле обозначения вершины графа.

Выводы

Научная новизна работы состоит в том, что разработан новый эффективный алгоритм окраски графа.

Практическая значимость исследований состоит в том, что алгоритм легко программируем и может успешно применяться для различных задач окраски графа.

Список литературы: 1. *Берж К.* Теория графов и ее применения. М.: ИЛ, 1962. 320 с. **2.** *Кристофидес Н.* Теория графов. Алгоритмический подход. М.: Мир, 1978. 432 с.

Поступила в редколлегию 21.05.2010

Канцедал Сергей Андреевич, д-р техн. наук, профессор кафедры прикладной математики и информатики Западнодонбасского института экономики и управления. Научные интересы: математическое моделирование, теория расписаний и её применение. Адрес: Украина, Павлоград, ул. Днепровская, 400.

Костикова Марина Владимировна, канд. техн. наук, доцент кафедры информатики Харьковского национального автомобильно-дорожного университета. Научные интересы: математическое моделирование, теория расписаний и её применение. Адрес: Украина, 61002, Харьков, ул. Петровского, 25, тел. 707-37-74.

УДК 519.863:621.382

И.В. ПРАСОЛ

МЕТОД УСТРАНЕНИЯ ДУБЛИРОВАНИЯ КРИТЕРИЕВ ПРИ ПАРАМЕТРИЧЕСКОМ СИНТЕЗЕ ЭЛЕКТРОННЫХ СХЕМ

Рассматривается метод, позволяющий устранить дублирующиеся критерии при параметрическом синтезе аналоговых электронных схем. Он основан на применении двухуровневой оптимизации, когда на нижнем уровне используются известные методы нелинейного программирования, а на верхнем – метод направленного изменения весовых коэффициентов функции-свертки. Получены соответствующие теоретические соотношения и алгоритм определения частных производных. Это позволяет уменьшить число рассматриваемых критериев, редуцировать математическую модель схемы и в конечном счете значительно снизить вычислительные затраты на этапе многокритериальной оптимизации в САПР.

1. Введение

При параметрическом синтезе электронных схем часто приходится сталкиваться с многокритериальностью задачи. Поиск решения в этом случае довольно трудоемкий и временные затраты напрямую зависят от количества критериев [1,2]. Зачастую увеличение этого количества не оправдано, так как с точки зрения качества решения одни критерии могут дублировать другие. Распознать такую ситуацию и избавиться от дублирования довольно сложно, однако если проделать такую работу и отсеять «лишние» критерии, задача получения наилучшего решения значительно упростится [3,4]. Особенно *актуальным* становится такая задача в случае использования агрегированных математических моделей, когда их размерность связана с числом критериев [5] и уменьшение их числа позволяет в ряде случаев значительно снизить количество уравнений модели схемы. Таким образом, *целью* данного исследования является разработка метода для уменьшения числа критериев качества схемы за счет устранения их дублирования.

2. Метод устранения дублирования критериев

Для поиска и удаления дублирующих критериев предлагается следующее.

Представим задачу как минимаксную, при которой решением считается точка:

$$X^* = f^{-1}\left(\min_{X\in\Omega}\max_{j} f_j(X), j=\overline{1,m}\right). \quad (1)$$

Здесь f^{-1} – символ обратного преобразования $f(X)$ в X; $f_j(X)$ – нормированная каким-либо образом непрерывная функция, определяющая j-й критерий качества; область определения

$$\Omega = S_1 \bigcap S_2, \quad (2)$$

где $S_1 = \{X|\ g_k(X) \le g_k'', k=\overline{1,l}\}$; $S_2\{X|\ x_i' \le x_i \le x_i'', i=\overline{1,n}\}$; $g_k(X), k=\overline{1,l}$ – в общем случае нелинейные функции ограничений; X – вектор управляемых (варьируемых) схемных параметров размерностью m.

Предлагается искать решение (1) следующим образом. Пусть

$$F(\Lambda, X) = \sum_{j=1}^{m} \lambda_j f_j(X), \quad (3)$$

где

$$\lambda_j \in E,\ E = \{\Lambda \in R^m|\ 0 \le \lambda_j \le 1, \sum_{j=1}^{m}\lambda_j = 1\}. \quad (4)$$

Тогда

$$X^* = f^{-1}\left(\max_{\Lambda\in E}\min_{X\in\Omega} F(\Lambda, X)\right). \quad (5)$$

Очевидно, что процесс получения решения (5) представляет собой двухуровневую оптимизацию: на нижнем уровне при заданных $\lambda_j, j=\overline{1,m}$ используется один из известных методов нелинейного программирования в целях поиска вектора X^{r*}, минимизирующего свертку (3) на r–м шаге. При этом решение будет принадлежать нижней левой границе (НЛГ) ; на верхнем уровне работает алгоритм направленного изменения $\lambda_j, j=\overline{1,m}$, который, основываясь на информации о состоянии оптимизируемой системы в точке X^{r*}, стремится достичь максимума функции $\min\limits_{X\in\Omega} F(\Lambda, X)$.

Для построения алгоритма оптимизации верхнего уровня следует определить вектор градиента $(\frac{\partial F(\Lambda,X)}{\partial\lambda_j}), j=\overline{1,m}$. На первый взгляд численно эту операцию можно было бы сделать путем попеременного изменения λ_j при неизменных $\lambda_k, k=\overline{1,m}, k\neq j$, и проведения соответствующих процедур минимизации $F(\Lambda, X)$ по X. Однако здесь необходимо обратить внимание на два момента:

а) для получения вектора градиента $(\frac{\partial F(\Lambda,X)}{\partial\lambda_j}), j=\overline{1,m}$ потребуется минимум $(m+1)$ этапа минимизации сверки (3);

б) точность полученного решения нельзя гарантировать, так как при выполнении действия $\lambda'_j=\lambda_j+\Delta\lambda_j, \lambda_k=\text{const}, k=\overline{1,m}, k\neq j$, будут нарушены условия (4), что может привести к выходу квазиоптимальной точки X^{r*} за область компромиссов НЛГ.

Поэтому для определения вектора частных производных $(\frac{\partial F(\Lambda,X)}{\partial\lambda_j}), j=\overline{1,m}$, предлагается следующее.

Запишем задачу оптимизации (3),(4) в таком виде:

$$F^*(\Lambda)=\max_{\lambda}\sum_{j=1}^{m}\lambda_j f_j^{\min}(\Lambda), \tag{6}$$

где $f_j^{\min}(\Lambda)$ – значение j-й функции в точке X^{r*}, полученной при минимизации свертки (3) на r-м этапе.

Определим вектор $(\frac{\partial F(\Lambda,X)}{\partial\lambda_j}), j=\overline{1,m}$, в некоторой точке Λ^0:

$$F(\Lambda^0)=\sum_{j=1}^{m}\lambda_j^0 f_j^{\min}(\Lambda^0);$$

$$F(\Lambda^0+\Delta\Lambda)=\sum_{j=1}^{m}(\lambda_j^0+\Delta\lambda_j) f_j^{\min}(\Lambda^0+\Delta\Lambda);$$

$$F(\Lambda^0+\Delta\Lambda)-F(\Lambda^0)=\sum_{j=1}^{m}(\lambda_j^0+\Delta\lambda_j) f_j^{\min}(\Lambda^0+\Delta\Lambda)-$$

$$-\sum_{j=1}^{m}\lambda_j^0 f_j^{\min}(\Lambda^0)=\sum_{j=1}^{m}\lambda_j^0 f_j^{\min}(\Lambda^0+\Delta\Lambda)+$$

$$+\sum_{j=1}^{m}\Delta\lambda_j f_j^{\min}(\Lambda^0+\Delta\Lambda)-\sum_{j=1}^{m}\lambda_j^0 f_j^{\min}(\Lambda^0)=$$

$$= \sum_{j=1}^{m} \lambda_j^0 (f_j^{min}(\Lambda^0+\Delta\Lambda) - f_j^{min}(\Lambda^0)) + \sum_{j=1}^{m} \Delta\lambda_j f_j^{min}(\Lambda^0+\Delta\Lambda) =$$

$$= \sum_{j=1}^{m} \lambda_j^0 \Delta f_j^{min}(\Lambda^0) + \sum_{j=1}^{m} \Delta\lambda_j f_j^{min}(\Lambda^0+\Delta\Lambda) .$$

Определим теперь частную производную по одной из координат:

$$\frac{F(\Lambda^0+\Delta\Lambda)-F(\Lambda^0)}{\Delta\lambda_k} = \sum_{j=1}^{m} \lambda_j^0 \frac{f_j^{min}(\Lambda^0+\Delta\Lambda)-f_j^{min}(\Lambda^0)}{\Delta\lambda_k} + \sum_{j=1}^{m} \frac{\Delta\lambda_j}{\Delta\lambda_k} f_j^{min}(\Lambda^0+\Delta\Lambda) . \quad (7)$$

Переходя к предельно малым величинам, получаем:

$$\frac{\partial F(\Lambda)}{\partial\lambda_k} = \sum_{j=1}^{m} \lambda_j \frac{\partial f_j^{min}(\Lambda)}{\partial\lambda_k} + \sum_{j=1}^{m} \frac{\partial\lambda_j}{\partial\lambda_k} f_j^{min}(\Lambda) .$$

Наличие сомножителя $\frac{\partial\lambda_j}{\partial\lambda_k}$ говорит о том, что множество переменных Λ является внутренне зависимым, т.е. весовые коэффициенты зависят от своих же значений внутри компакта E.

Раскроем этот сомножитель следующим образом. Пусть

$$\lambda_j^1 = \frac{\lambda_j^0}{1+\Delta\lambda_k^p} \; ; \; \lambda_k^1 = \frac{\lambda_k^0+\Delta\lambda_k^p}{1+\Delta\lambda_k^p}, \; j \neq k \; ,$$

где $\Delta\lambda_k^p$ – некоторое текущее приращение k-го весового коэффициента. Это предполагает наличие подвижной связи между всеми весовыми коэффициентами. Тогда

$$\Delta\lambda_j = \frac{\lambda_j^0}{1+\Delta\lambda_k^p} - \lambda_j^0 = -\frac{\lambda_j^0 \Delta\lambda_k^p}{1+\Delta\lambda_k^p} ;$$

$$\Delta\lambda_k = \frac{\lambda_k^0+\Delta\lambda_k^p}{1+\Delta\lambda_k^p} - \lambda_k^0 = \frac{\Delta\lambda_k^p (1-\lambda_k^0)}{1+\Delta\lambda_k^p} , \text{ и } \frac{\Delta\lambda_j}{\Delta\lambda_k} = -\frac{\lambda_j}{1-\lambda_k} , \text{ для } j \neq k ,$$

а для λ_k это частное будет равно, естественно, единице. Формула (7) запишется в следующем виде:

$$\frac{\partial F(\Lambda)}{\partial\lambda_k} = \sum_{j=1}^{m} \lambda_j \frac{\partial f_j^{min}(\Lambda)}{\partial\lambda_k} - \sum_{\substack{j=1 \\ j\neq k}}^{m} \frac{\lambda_j}{1-\lambda_k} f_j(\Lambda) + f_k(\Lambda) . \quad (8)$$

Выражение (8) дает возможность предложить такую последовательность действий по численному дифференцированию функции $\min_{X\in\Omega} \sum_j \lambda_j f_j(X)$:

1. Исходная точка, в которой необходимо определить вектор-градиент, задается некоторыми значениями $\lambda_j^0 \in E$.
2. Определяется точка, доставляющая минимум свертке (3).
3. Определяются $f_j^{min}(\Lambda^0), \; j = \overline{1,m}$.
4. Задается некоторое $\Delta\lambda$.

5. Вычисляются $\lambda_j^1 = \frac{\lambda_j^0}{1+\Delta\lambda}$; $\lambda_s^1 = \frac{\lambda_s^0+\Delta\lambda}{1+\Delta\lambda}$, $1 \le s \le m, j \ne s$.
6. Выполняются последовательно п.2 и 3 для Λ^1.
7. По формуле (8) вычисляются составляющие вектора производных $(\frac{\partial F(\Lambda, X)}{\partial \lambda_j}), j = \overline{1,m}$.

3. Пример

Пусть нормированные критерии качества описываются системой функций:

$$f_1 = (x_1 - 1)^2 + x_2^2,$$
$$f_2 = (x_1 - 2)^2 + (x_2 - 2)^2,$$
$$f_3 = x_1^2 + (x_2 - 3)^2,$$
$$f_4 = (x_1 - 1)^2 + (x_2 - 1)^2,$$
$$f_5 = (x_1 - 1)^2 + (x_2 - 2)^2, \ \Omega = R^2.$$

В данной постановке критерии 3–5 дублируют качество остальных критериев, так как их значение будет заведомо лучше или равно критериям 1 и 2.

Определим чувствительности (8).

Продифференцировав свертку по X и приравняв частные производные нулю, получим значения вектора X, в которых достигается ее минимум при некоторых заданных $\lambda_j, j = \overline{1,5}$. Это будут соответственно

$$x_1 = \lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5; x_2 = 2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5. \quad (9)$$

Подставив в исходную систему значения (9), получим:

$$f_1 = (\lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5 - 1)^2 + (2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5)^2,$$
$$f_2 = (\lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5 - 2)^2 + (2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5 - 2)^2,$$
$$f_3 = (\lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5)^2 + (2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5 - 3)^2,$$
$$f_4 = (\lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5 - 1)^2 + (2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5 - 1)^2,$$
$$f_5 = (\lambda_1 + 2\lambda_2 + \lambda_4 + \lambda_5 - 1)^2 + (2\lambda_2 + 3\lambda_3. + \lambda_4 + 2\lambda_5 - 2)^2.$$

Значения производных $\frac{\partial f_i^{min}(\Lambda)}{\partial \lambda_j}$, необходимые для расчета (8) сведены в таблицу.

	λ_1	λ_2	λ_3	λ_4	λ_5
f_1	$2(x_1 - 1)$	$4(x_1 - 1) + 4x_2$	$6x_2$	$2(x_1 - 1) + 2x_2$	$2(x_1 - 1) + 4x_2$
f_2	$2(x_1 - 2)$	$4(x_1 - 2) + 4(x_2 - 2)$	$6(x_2 - 2)$	$2(x_1 - 2) + 2(x_2 - 2)$	$2(x_1 - 2) + 4(x_2 - 2)$
f_3	$2x_1$	$4x_1 + 4(x_2 - 3)$	$6(x_2 - 3)$	$2x_1 + 2(x_2 - 3)$	$2x_1 + 4(x_2 - 3)$
f_4	$2(x_1 - 1)$	$4(x_1 - 1) + 4(x_2 - 1)$	$6(x_2 - 1)$	$2(x_1 - 1) + 2(x_2 - 1)$	$2(x_1 - 1) + 4(x_2 - 1)$
f_5	$2(x_1 - 1)$	$4(x_1 - 1) + 4(x_2 - 2)$	$6(x_2 - 2)$	$2(x_1 - 1) + 2(x_2 - 2)$	$2(x_1 - 1) + 4(x_2 - 2)$

Здесь x_1 и x_2 необходимо заменить на соответствующие значения из (9). Тогда,

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0{,}2$$

получим следующий вектор чувствительностей:

$$\frac{\partial F(\Lambda)}{\partial \lambda_k} = (1{,}4; -0{,}35; 1{,}9; -1{,}35; -1{,}6).$$

Результаты вычислений показывают, что вес критериев необходимо уменьшить, так как они, вероятнее всего, являются избыточными.

Проверим это для набора $\lambda_1 = \lambda_3 = 0{,}5; \lambda_2 = \lambda_4 = \lambda_5 = 0$: $\frac{\partial F(\Lambda)}{\partial \lambda_k} = (0;0;0;-2;-2)$.

Отрицательные значения производной подтверждают тот факт, что критерии 4 и 5 действительно являются "лишними" в данной постановке, а нулевые – что данное решение является оптимальным с точки зрения минимакса.

Анализ результатов подтверждает, что полученные теоретические соотношения верно определяют частные производные свертки (6) по λ, которые позволяют устранить избыточность в задачах параметрического синтеза электронных схем.

4. Выводы

Предложен метод выявления и устранения дублирующих критериев при многокритериальной параметрической оптимизации электронных схем. Он базируется на двухуровневой оптимизации, которая на верхнем уровне использует метод варьирования весовыми коэффициентами для максимизации введенной функции – свертки.

Научная новизна заключается в том, что впервые получены теоретические соотношения для определения составляющих градиента функции, переменные которой представляют компакт, и алгоритм ее численного дифференцирования. Справедливость метода подтверждена на конкретном примере. Данный метод позволяет уменьшать число критериев оптимизации за счет устранения дублирования, что особенно важно при оптимизации схем по агрегированным моделям, размерность которых пропорциональна количеству критериев.

Практическая значимость заключается в возможности повысить точность и значительно снизить вычислительные затраты на этапе параметрического синтеза электронных схем в интегрированной САПР РЭА.

Список литературы: 1. *Шеховцов А.В., Крючковский В.В., Мельник А.Н.* Решение многокритериальной оптимизации с использованием адаптивных алгоритмов // Автоматика, автоматизация, электротехнические комплексы и системы. Херсон: ХГТУ. 2007. №2 (20). С. 60 – 67. 2. *Батищев Д.И., Исаев С.А.* Оптимизация многоэкстремальных функций с помощью генетических алгоритмов // Межвузовский сборник научных трудов «Высокие технологии в технике, медицине и образовании». Воронеж: ВГТУ. 1997. С. 4-17. 3. *Анохин А.М., Глотов В.А.,Павельев В.В., Черкашин А.М.* Методы определения коэффициентов важности критериев // Автоматика и телемеханика. 1997. №8. С. 3-35. 4. *Анохин А.М., Гусев В.Б., Павельев В.В.* Комплексное оценивание и оптимизация на моделях многомерных объектов. М.: Институт проблем управления им. В.А. Трапезникова РАН, 2003. 80 с. 5. *Прасол И.В.* Особенности агрегирования моделей электронных схем при оптимизации их параметров // Технология приборостроения. Харьков: ДП НДТІП. 2010. № 1. С. 25-29.

Поступила в редколлегию 21.05.2010

Прасол Игорь Викторович, канд. техн. наук, доцент, профессор каф. БМЭ ХНУРЭ. Научные интересы: проектирование и моделирование электронных схем, САПР РЭА, цифровая схемотехника, разработка биомедицинских устройств. Увлечения и хобби: авто, путешествия, фото- и видеосъемка, приусадебное хозяйство. Адрес: Украина, 61166, Харьков, пр. Ленина,14, тел.: (057) 7021364, e-mail: ivp@kture.kharkov.ua.

УДК 681.325.53: 37: 004.5

Н.Я.КАКУРИН, Ю.В. ЛОПУХИН, Е.В. БОЧАРОВ, В.В. ВАРЕЦА, А.Н. МАКАРЕНКО

САПР ДЛЯ ПОЛУЧЕНИЯ ЗАКОНОВ ФУНКЦИОНИРОВАНИЯ ПРЕОБРАЗОВАТЕЛЕЙ КОДОВ С ПАРАЛЛЕЛЬНОЙ СТРАТЕГИЕЙ

Рассматриваются возможности подсистем САПР «TRANS» для получения таблиц законов функционирования многоблочных преобразователей кодов, функционирующих по методу накопления эквивалентов и использующих параллельную стратегию шагов преобразования.

1. Постановка задачи

В настоящее время большинство ЭВМ функционируют в двоичной системе счета. Исходные данные и результаты решения задач должны быть представлены в форме, удобной для восприятия их человеком, т.е. в десятичной системе счисления.

По этой причине возникает задача преобразования чисел из десятичной системы в двоичную и обратно. Операции преобразования чисел из одной системы в другую можно выполнять как программным способом, так и аппаратным. Аппаратный способ преобразования чисел с помощью специализированного блока – преобразователя кодов (ПК) предпочтителен, так как он не снижает быстродействие компьютера и позволяет совместить решение одной задачи с подготовкой данных для ввода/вывода другой.

Наиболее удобным методом преобразования чисел аппаратным способом является метод накопления эквивалентов. Этот метод гибкий и позволяет за счет выбора числа шагов преобразования, значений шагов и стратегий их использования (последовательной или параллельной) обеспечить требуемые значения быстродействия и аппаратурных затрат. Преимуществом параллельной стратегии использования шагов является возможность дальнейшего увеличения быстродействия на 20-30% по сравнению с последовательной стратегией.

Цель данной работы – автоматизация этапа системного проектирования преобразователей кодов целых чисел с параллельной стратегией использования шагов преобразования в широком диапазоне разрядностей, оснований систем счета и значений шагов.

Основными *задачами* являются:

– анализ двух вариантов структурной организации и функционирования ПК – с последовательной и параллельной стратегией использования шагов преобразования;

– рассмотрение возможностей подсистемы САПР «TRANS» для проектирования ПК, как параллельного, так и последовательного типов.

2. Последовательная стратегия

При использовании последовательной стратегии преобразования чисел процесс преобразования сводится к последовательному вычитанию из состояния разрядного счетчика, в котором хранится цифра преобразуемого числа, значения шага преобразования. Если значение системы счета на входе К, то максимальное значение цифры преобразуемого числа равно К-1. Следовательно, максимальное число тактов преобразования старших разрядных цифр равно К-1. Затем, еще один такт используется для трансляции (передачи) значения цифры младшего разряда. Таким образом, суммарное число тактов преобразования равно, т.е. N_1=К. В двухшаговом ПК вначале ведется преобразование с шагом а. При наличии в одном из старших разрядов $x_m (m = \overline{2, n})$ максимального значения цифры $x_m = K - 1$ потребуется на первом этапе $](K-1)/a[$ шагов, преобразования, где] [означают округление до меньшего целого. Затем, если хотя бы одна из оставшихся старших цифр имеет значение $x_i = a - 1$, потребуется на втором этапе еще а-1 такт для обнуления с шагом 1 всех старших разрядных счетчиков. Следует добавить еще один такт (третий этап) для трансляции младшей цифры x_0. Следовательно,

$$N_2^{цел} =](K-1)/a[+(a-1)+1 =](K-1)/a[+a\,. \qquad (1)$$

Рассмотрим структуру двухшагового ПК (рис.1), предложенную в целях повышения быстродействия одношагового ПК в [1].

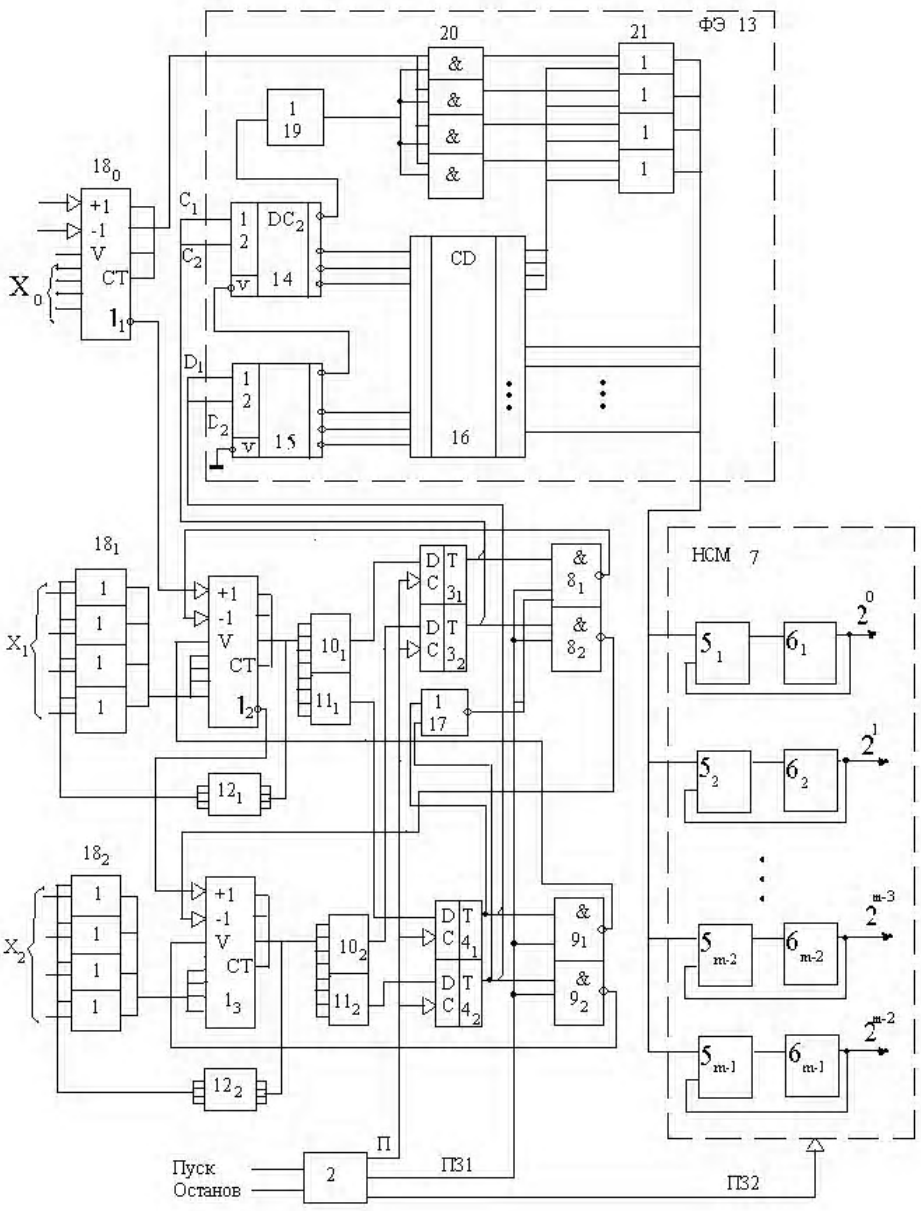


Рис. 1. Структура двухшагового ПК с последовательной стратегией

Двухшаговый ПК двоично-К-ичного кода в двоичный код содержит группу разрядных счетчиков I, блок управления (генератор) импульсов 2**,** содержащий прямой П, прямые задержанные ПЗ1 и ПЗ2 выходы, первую группу триггеров 3 состояния, вторую группу триггеров 4 состояния, комбинационный двоичный сумматор 5 и регистр 6 результата, образующие в совокупности накапливающий сумматор 7, группу элементов И-НЕ 8, группу элементов И-НЕ 9, группу дешифраторов нуля 10, группу дешифраторов превышения 11, группу шифраторов 12, формирователь эквивалента 13, включающий в свой состав первый 14 и второй 15 дешифраторы и шифратор 16, элемент ИЛИ-НЕ 17, группу элементов ИЛИ 18.

В состав формирователя эквивалентов 13 также входят элемент НЕ 19, группа элементов И 20 и группа элементов ИЛИ 21. Функционирование двухшагового ПК происходит следующим образом.

Группа триггеров 3 фиксирует ненулевое значение счетчиков соответствующих разрядов; группа триггеров 4 – значение старших разрядных счетчиков, превышающее заданное число $a-1$, например 1.

Шифраторы 12 реализуют следующую функцию:

$$Y = \{X - a; X \geq a; \{X; X < a,$$

где X – входной код тетрады; a – параметр, в частном случае, равный 2. Цепи инициирования и сброса на рис.1 не показаны. Так как в конкретном случае n = 4, K = 12, то диапазон изменения входного кода $0-(12^4-1)=0-20735_{10}$. Код состояний триггеров как первой группы 3, так и второй 4 имеет $2^3=8$ значений от 000 до III.

ФЭ 13, выполненный в виде последовательного соединения первого 14 и второго 15 дешифраторов, реализует функцию

$$S=\begin{cases} KC_1+K^2C_2+K^3C_3; Д_1=Д_2=Д_3=0; \\ aКД_1+aK^2Д_2+aK^3Д_3; Д_1, Д_2, Д_3 \neq 0; \end{cases} \quad (2)$$

где C_1, C_2, C_3 – значения разрядов двоичного кода триггеров состояний первой группы 3; $Д_1, Д_2, Д_3$ – значения разрядов двоичного кода триггеров состояния второй группы 4.

Каждый из $C_m(m=\overline{1,3})$ триггеров 3 первой группы состояний разрядных счетчиков соответствует наличию (1) или отсутствию (0) информации в соответствующем старшем разряде преобразуемого кода, а разряд $Д_m(m=\overline{1,3})$ триггеров 4 второй группы состояний разрядных счетчиков равен 1,если соответствующий разряд преобразуемого кода имеет значение $x_i \geq a(2 \leq a \leq k-1)$, в противном случае $Д_m = 0$.

ФЭ 13 преобразует вначале двоичный код $Д_3, Д_2, Д_1$ триггеров состояний второй группы, затем при $Д_3, Д_2, Д_1 = 0$ преобразует двоичный код C_3, C_2, C_1 триггеров состояний первой группы и при $Д_3 = Д_2 = Д_1 = C_3 = C_2 = C_1$=000 выполняет трансляцию (передачу) двоично-К-ичного кода младшего разряда в двоичный код эквивалента на выходе.

Двухшаговый ПК работает следующим образом. Пусть требуется преобразовать входной 12-ричный код числа

$$\begin{aligned} A_0 &= 9238_{12} = 1001*0010*0011*1000_{(2,12)} = 9*12^3+2*12^2+3*12^1+8*12^0 = \\ &= 9*1728+2*144+3*12+8=15884_{10} \end{aligned} \quad (3)$$

Для определенности примем, что дешифраторы превышения 11 настроены на определение значений разрядных цифр $x_i \geq 2$. Следовательно, в этом случае каждый разряд $Д_m(m=\overline{1,3})$ триггеров 4 второй группы состояний разрядных счетчиков равен I, если соответствующий разряд преобразуемого кода имеет значение $x_i \geq 2$ **,** в противном случае $Д_m = 0$. Шифраторы 12 формируют выходное слово, меньшее на 2 единицы по отношению к входному слову.

ФЭ 13 в данном случае реализует функцию

$$S=\begin{cases} 12C_1+144C_2+1728C_3; Д_1 = Д_2 - Д_3 = 0; \\ 24Д_1+288Д_2+3456Д_3; Д_1, Д_2, Д_3 \neq 0. \end{cases} \quad (4)$$

Преобразование двоичных кодов триггеров состояний первой 3 и второй 4 группы соответствует табл. 1. В табл. 1 S означает общий вид эквивалента, а S_{10} – десятичный код эквивалента.

Таблица 1

Выходы ДС	Состояние триггеров		S	S_{10}	Двоичный код эквивалента		
	Второй группы	Первой группы					
Z_i	$Д_3Д_2Д_1$	$С_3С_2С_1$			$У_{11}У_{10}У_9У_8$	$У_7У_6У_5У_4$	$У_3У_2У_1У_0$
Z_0	000	000	x_0	x_0	Трансляция тетрады x_0		
Z_1	000	001	K	12	0000	0000	1100
Z_2	000	010	K^2	144	0000	1001	0000
Z_3	000	011	$K^2 + K$	156	0000	1001	1100
Z_4	000	100	K^3	1728	0110	1100	0000
Z_5	000	101	$K^3 + K$	1740	0110	1100	1100
Z_6	000	110	$K^3 + K^2$	1872	0111	0101	0000
Z_7	000	111	$K^3 + K^2 + K$	1884	0111	0101	1100
Z_8	001	XX1	aK	24	0000	0001	1000
Z_9	010	X1X	aK^2	288	0001	0010	0000
Z_{10}	011	X11	$aK^2 + aK$	312	0001	0011	1000
Z_{11}	100	1XX	aK^3	3456	1101	1000	0000
Z_{12}	101	1X1	$aK^3 + aK$	3480	1101	1001	1000
Z_{13}	110	11X	$aK^3 + aK^2$	3744	1110	1010	0000
Z_{14}	111	111	$aK^3 + aK^2 + aK$	3768	1110	1011	1000

3. Параллельная стратегия

Дальнейшего увеличения быстродействия ПК можно достичь или за счет увеличения числа шагов в наборе до трех 1, a, b (b – третий шаг) при сохранении принципа последовательного использования шагов преобразования (вначале b, затем a и в заключение шаг 1), или за счет параллельного (одновременного) вычитания различных шагов из различных разрядных цифр. Усиление локального параллелизма достигается допущением возможности одновременного использования различных шагов преобразования в различных старших разрядах числа.

Выражение для формулы максимального числа тактов преобразования двухшагового ПК с параллельным использованием шагов преобразования получить в явном виде затруднительно. Определить же значение максимального числа тактов преобразования можно путем моделирования процесса преобразования для различных наборов шагов и путем анализа результатов (табл. 2).

Анализ табл.2 показывает, что для K = 12 в двухшаговом ПК с параллельным использованием шагов преобразования минимальное значение числа тактов преобразования равно пяти для наборов (1, 3); (1, 4); (1, 5). Подобный анализ, проведенный для наборов 1,6 –1,11, дает рост числа тактов преобразования до шести – десяти.

Таким образом, по сравнению с принципом последовательного использования шагов преобразования число тактов сокращается на один (5 вместо 6). Среди трех наборов шагов

преобразования оптимальным по дополнительному критерию минимума аппаратных затрат является набор 1,4, при использовании которого отсутствуют затраты ЛЭ на шифратор 12 (табл. 2). В табл. 2 NT означает номер такта преобразования.

Таблица 2

NT	К=12 Набор шагов (1,2)	К=12 Набор шагов (1,3)
	11 10 9 8 7 6 5 4 3 2 1 0	11 10 9 8 7 6 5 4 3 2 1 0
1	9 8 7 6 5 4 3 2 1 0 0 0	8 7 6 5 4 3 2 1 0 2 0 0
2	7 6 5 4 3 2 1 0 0 0 0 0	5 4 3 2 1 0 1 0 0 1 0 0
3	5 4 3 2 1 0 0 0 0 0 0 0	2 1 0 1 0 0 0 0 0 0 0 0
4	3 2 1 0 0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0
5	1 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
6	0 0 0 0 0 0 0 0 0 0 0 0	
NT	К=12 Набор шагов 1,4	К=12 Набор шагов 1,5
	11 10 9 8 7 6 5 4 3 2 1 0	11 10 9 8 7 6 5 4 3 2 1 0
1	7 6 5 4 3 2 1 0 2 1 0 0	6 5 4 3 2 1 0 3 2 1 0 0
2	3 2 1 0 2 1 0 0 1 0 0 0	1 0 3 2 1 0 0 2 1 0 0 0
3	2 1 0 0 1 0 0 0 0 0 0 0	0 0 2 1 0 0 0 1 0 0 0 0
4	1 0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0 0 0
5	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0

При К=10 и параллельном использовании шагов преобразования минимальными наборами шагов, обеспечивающими преобразование за 4 такта, являются наборы 1,3 и 1,4.

Лучший среди них - набор 1,4 (по причине простой реализации шифраторов 12).

Повышение быстродействия двухшаговых ПК с параллельным использованием шагов преобразования достигается ценой некоторого усложнения ФЭ и ЛЭ для управления вычитанием шагов а и 1 из содержимого разрядных счетчиков.

Логика управления в двухшаговом ПК параллельного типа выполнена так, чтобы запретить возможность вычитания шага 1, если в этом разряде имеется возможность вычитания шага а. И наоборот, если значение разряда x_m находится в пределах $1 \le x_m < a$, то следует разрешить опрос вентиля, управляющего вычитанием 1 из разрядного счетчика, хранящего x_m.

Этот принцип управления реализуется в двухшаговом ПК параллельного типа путем замены (n-1) входного элемента ИЛИ-НЕ блоком инверторов (инверторы 17_1, 17_2), вход каждого из которых связан с единичным выходом соответствующего триггера старшего регистра состояний $РГ_4$, а выход соответствующего инвертора соединяется с управляющими входами схем И 8_1, И 8_2, на информационные входы которых поступают сигналы с единичных выходов триггеров этого разряда младшего регистра состояний $РГ_3$ (рис. 2).

Закон функционирования ФЭ для двухшагового ПК параллельного типа для набора шагов 1,2 и К=12 приведен в табл. 3. Детально параллельная стратегия приведена в [2,3].

Сравнение табл. 1 и 3 показывает, что ФЭ двухшагового ПК параллельного типа имеет 27 строк; ПК последовательного типа (см. табл.1) - всего 15 строк.

4. Программное средство для расчета аппаратурных затрат в ФЭ

Программное средство предназначено для получения значений эквивалентов, которые должен выдавать блок ФЭ, а также для расчета аппаратурных затрат в долях корпусов ИМС на реализацию каждой функции выхода ФЭ. Программа рассчитана на ПК, функционирующие по методу накопления эквивалентов.

Для того чтобы произвести расчет в данном режиме, необходимо запустить программу путем активизации исполняемого модуля TRANS.exe.

После запуска программы появится пять текстовых полей для ввода данных. С помощью этих элементов управления необходимо задать исходные данные для режима преобразования чисел: основание системы счисления на входе (3-15), разрядность преобразуемого числа (1-24), количество блоков (1-24), количество шагов преобразователя (1-8) и их значения, тип преобразователя (последовательный или параллельный).

Таблица 3

Номер набора	Состояние триггеров		S	S_{10}	Двоичный код эквивалента		
	Второй группы	Первой группы					
Z_i	$Д_3Д_2Д_1$	$C_3C_2C_1$			$y_{12}y_{11}y_{10}y_9$	$y_8y_7y_6y_5$	$y_4y_3y_2y_1$
0	000	000	X_0	X_0	Трансляция младшей тетрады		
1	000	001	K^1	12	0000	000	1100
2	000	010	K^2	144	0000	1001	0000
3	000	011	$K^2 + K^1$	156	0000	1001	1100
4	000	100	K^3	1728	0110	1100	0000
5	000	101	$K^3 + K^1$	1740	0110	1100	1100
6	000	110	$K^3 + K^2$	1872	0111	0101	0000
7	000	111	$K^3 + K^2 + K^1$	1884	0111	0101	1100
8	001	001	aK	24	0000	0001	1000
9	001	011	$K^2 + aK$	168	0000	1010	1000
10	001	101	$K^3 + aK$	1752	0110	1101	1000
11	001	111	$K^3 + K^2 + aK$	1896	0111	0110	1000
12	010	010	aK^2	288	0001	0010	0000
13	010	011	$aK^2 + K$	300	0001	0010	1100
14	010	110	$K^3 + aK^2$	2016	0111	1110	0000
15	010	111	$K^3 + aK^2 + K$	2028	0111	1110	1100
16	011	011	$aK^2 + aK$	312	0001	0011	1000
17	011	111	$K^3 + aK^2 + aK$	2040	0111	1111	1000
18	100	100	aK^3	3456	1101	1000	0000
19	100	101	$aK^3 + K$	3468	1101	1000	1100
20	100	110	$aK^3 + K^2$	3600	1110	0001	0000
21	100	111	$aK^3 + K^2 + K$	3612	1110	0001	1100
22	101	101	$aK^3 + aK$	3480	1101	1001	1000
23	101	111	$aK^3 + K^2 + aK$	3624	1110	0010	1000
24	110	110	$aK^3 + aK^2$	3744	1110	1010	0000
25	110	111	$aK^3 + aK^2 + K$	3756	1110	1010	1100
26	111	111	$aK^3 + aK^2 + aK$	3768	1110	1011	1000

После ввода всех необходимых данных необходимо нажать на кнопку «ОК». Если параметры были заданы некорректно, то выводится соответствующее сообщение. После выполнения расчет на экране появится результат преобразования.

На рис.3,4 приведены результаты преобразования для основания системы счисления 12, двух блоков и шести разрядов. В качестве весов шагов были заданы значения 1 и 4.

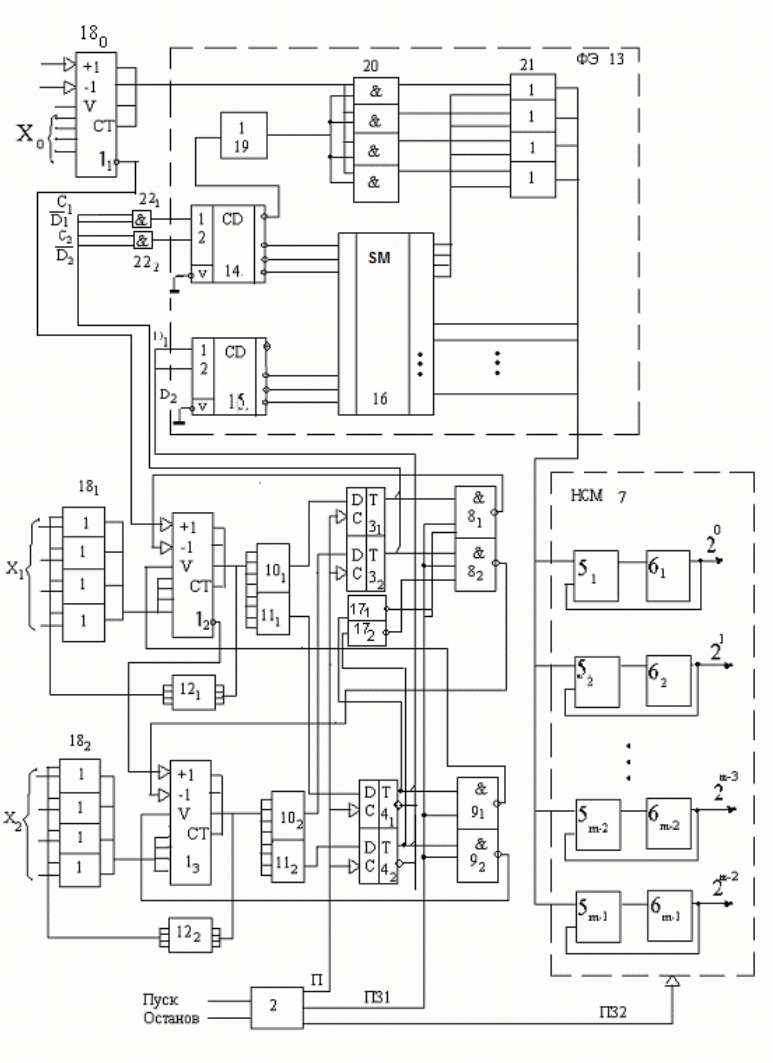


Рис. 2. Двухшаговый ПК с параллельным использованием шагов

Результат

Разряды: 1...3

Выход дешифратора	Код состояния регистра	Код эквивалента десятичный	двоичный
z 1	000 000	0	000 0000 0000 0000 0000 0000 0000
z 2	000 001	1	000 0000 0000 0000 0000 0000 0001
z 3	000 010	12	000 0000 0000 0000 0000 0000 1100
z 4	000 011	13	000 0000 0000 0000 0000 0000 1101
z 5	000 100	144	000 0000 0000 0000 0000 1001 0000
z 6	000 101	145	000 0000 0000 0000 0000 1001 0001
z 7	000 110	156	000 0000 0000 0000 0000 1001 1100
z 8	000 111	157	000 0000 0000 0000 0000 1001 1101
z 9	001 001	4	000 0000 0000 0000 0000 0000 0100
z 10	001 011	52	000 0000 0000 0000 0000 0011 0100
z 11	001 101	580	000 0000 0000 0000 0010 0100 0100
z 12	001 111	628	000 0000 0000 0000 0010 0111 0100
z 13	010 010	48	000 0000 0000 0000 0000 0011 0000
z 14	010 011	52	000 0000 0000 0000 0000 0011 0100
z 15	010 110	624	000 0000 0000 0000 0010 0111 0000
z 16	010 111	628	000 0000 0000 0000 0010 0111 0100
z 17	011 011	52	000 0000 0000 0000 0000 0011 0100
z 18	011 111	628	000 0000 0000 0000 0010 0111 0100
z 19	100 100	576	000 0000 0000 0000 0010 0100 0000
z 20	100 101	580	000 0000 0000 0000 0010 0100 0100
z 21	100 110	624	000 0000 0000 0000 0010 0111 0000
z 22	100 111	628	000 0000 0000 0000 0010 0111 0100
z 23	101 101	580	000 0000 0000 0000 0010 0100 0100
z 24	101 111	628	000 0000 0000 0000 0010 0111 0100
z 25	110 110	624	000 0000 0000 0000 0010 0111 0000
z 26	110 111	628	000 0000 0000 0000 0010 0111 0100
z 27	111 111	628	000 0000 0000 0000 0010 0111 0100

```
y1 = (¬z0)|(¬z2)|(¬z4)|(¬z6)
y2 = 0
y3 = (¬z1)|(¬z2)|(¬z5)|(¬z6)|(¬z7)|(¬z8)|(¬z9)|(¬z10)|(¬z12)|(¬z14)|(¬z15)|(¬z16)|(¬z18)|(¬z20)|(¬z21)|(¬z22)|(¬z2
y4 = (¬z1)|(¬z2)|(¬z5)|(¬z6)
y5 = (¬z3)|(¬z4)|(¬z5)|(¬z6)|(¬z8)|(¬z10)|(¬z11)|(¬z12)|(¬z13)|(¬z14)|(¬z15)|(¬z16)|(¬z19)|(¬z20)|(¬z22)|(¬z23)|(¬
y6 = (¬z8)|(¬z10)|(¬z11)|(¬z12)|(¬z13)|(¬z14)|(¬z15)|(¬z16)|(¬z19)|(¬z20)|(¬z22)|(¬z23)|(¬z24)|(¬z25)
y7 = (¬z9)|(¬z10)|(¬z13)|(¬z14)|(¬z16)|(¬z17)|(¬z18)|(¬z19)|(¬z20)|(¬z21)|(¬z22)|(¬z23)|(¬z24)|(¬z25)
y8 = (¬z3)|(¬z4)|(¬z5)|(¬z6)
y9 = 0
y10 = (¬z9)|(¬z10)|(¬z13)|(¬z14)|(¬z16)|(¬z17)|(¬z18)|(¬z19)|(¬z20)|(¬z21)|(¬z22)|(¬z23)|(¬z24)|(¬z25)
```

Рис. 3. Форма приложения в режиме преобразования для К=12

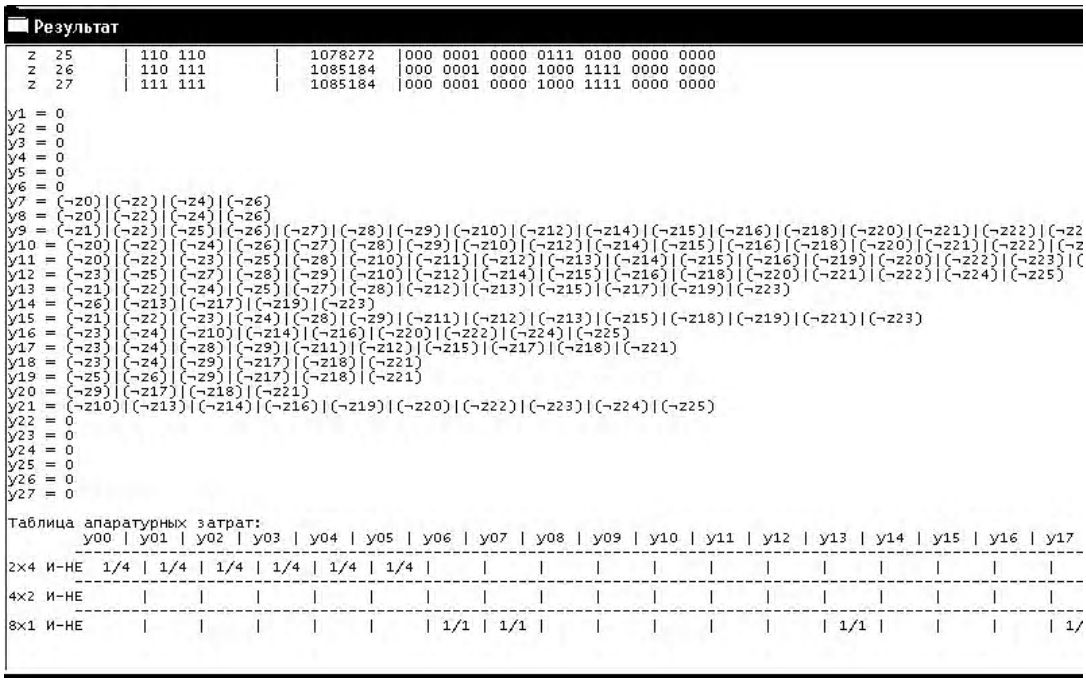


Рис. 4. Выведение результирующей функции и таблицы аппаратурных затрат

В качестве инструмента для разработки программного пакета «TRANS» была использована среда программирования Microsoft Visual Studio 2005. Данная среда программирования частично использует принципы RAD. RAD означает быструю разработку приложений. Эта концепция позволяет создавать программные продукты, причем особое внимание уделяется быстроте и удобству программирования, созданию технологичного процесса, позволяющего программисту максимально быстро писать компьютерные программы.

Для задания первичных значений работы программы предусмотрены текстовые поля «Основание системы счисления», «Число разрядов», «Число блоков», «Число шагов» и «Веса шагов».

Для переключения типа преобразователей с последовательного на параллельный и наоборот используется специальная кнопка. Такое решение позволит максимально облегчить и упростить интерфейс программы, сделать его максимально понятным. Определение типа преобразователя реализовано с помощью переменной bool bParallel. Это булева переменная, которая может принимать только два значения: true или false. По умолчанию переменной присвоено true, т.е. будет применяться параллельный тип преобразователя.

В процессе своей работы функция использует следующие структуры данных:

1) Переменная iBaseNumber – определяет основание системы счисления, из которой будем преобразовывать. Значение переменной может варьироваться от 3 до 15, т.е. основание системы может изменяться от 3 до 15. По умолчанию переменной присвоено значение, равное 10, т.е. по умолчанию установлена десятичная система счисления.

2) Переменная iAmountOfBlocks – определяет количество блоков. Значение переменной может варьироваться от 1 до 24, т.е. количество блоков может изменяться от 1 до 24. По умолчанию переменной присвоено значение, равное 1, т.е. по умолчанию установлен один блок.

3) Переменная iAmountOfSteps – определяет количество шагов. Значение переменной может варьироваться от 1 до 8, т.е. количество шагов может изменяться от 1 до 8. По умолчанию переменной присвоено значение, равное 2, т.е. по умолчанию установлено два шага.

4) Переменная iAmountOfDigits – определяет количество разрядов. Значение переменной может варьироваться от 1 до 24, т.е. количество разрядов может изменяться от 1 до 24. По умолчанию переменной присвоено значение, равное 2, т.е. по умолчанию установлено два разряда.

5) Массив SW хранит в себе веса шагов. По умолчанию записано значение {1,4}, т.е. по умолчанию веса шагов установлены в значения 1 и 4. Размерность массива равна 80 значений.

6) Массив equivs хранит в себе коды эквивалентов. Размерность массива равна 80 значений.

Выводы

Основные результаты. Рассмотрены две стратегии преобразования чисел по методу накопления эквивалентов. Показано, что применение параллельной стратегии позволяет по сравнению с последовательной сократить число тактов преобразования на 20-25%. Разработан программный пакет TRANS, позволяющий выполнять построение таблицы, законов функционирования ФЭ для ПК с параллельной стратегией и расчет аппаратурных затрат ИМС на построение каждого блока ФЭ.

Сравнение с лучшими аналогами. Предложенный программный пакет TRANS позволяет выполнять построение таблиц и анализ затрат для ПК с параллельной стратегией в то время, как программный пакет FE_LION может выполнять аналогичные функции только для ПК с последовательной стратегией.

Практическая значимость. Программное средство TRANS позволяет проводить детальный анализ этапа системного проектирования ПК параллельного типа, сократить аппаратурные затраты на реализацию всего ПК за счет нахождения оптимального разбиения на блоки, сократить число тактов преобразования за счет параллельной стратегии.

Список литературы: **1.** А.С. 1647908 5НОЗМ 7/12. Преобразователь двоично-К-ичного кода в двоичный код /*Н.Я.Какурин, Ю.К. Кирьяков, А.Н. Макаренко* //Открытия, изобретения. 1984. №17. С. 262-263. **2**. *Какурин Н.Я., В.В. Вареца, С.Н. Коваленко.* Параллельная стратегия использования шагов в двухшаговых преобразователях кода //АСУ и приборы автоматики. 2007. Вып.141. С.29-36. **3.** *Какурин Н.Я., Лопухин Ю.В., Макаренко А.Н.* Об одном способе повышения быстродействия преобразователей кодов //АСУ и приборы автоматики. 2003. Вып.122. С.72-83.

Поступила в редколлегию 07.05.2010

Какурин Николай Яковлевич, канд. техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: прикладная теория цифровых автоматов, автоматизация проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

Лопухин Юрий Владимирович, ст. преподаватель кафедры АПВТ ХНУРЭ. Научные интересы: проектирование программного обеспечения, автоматизация проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

Бочаров Евгений Витальевич, студент группы КИ-07-6 ХНУРЭ. Научные интересы: автоматизация проектирования цифровых устройств, проектирование программного обеспечения. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

Вареца Виталий Викторович, аспирант кафедры АПВТ ХНУРЭ. Научные интересы: проектирование программного обеспечения, автоматизация проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

Макаренко Анна Николаевна, канд. техн. наук, доцент кафедры высшей математики и информационных технологий Харьковского банковского института. Научные интересы: дискретная математика, анализ и синтез преобразователей код-код. Адрес: Украина, 61074, Харьков, пр. Победы, 55, тел. 336-05-64.

УДК 519.246.8

Л.О. КИРИЧЕНКО, Н.М. РВАЧЁВА, А.В. СТОРОЖЕНКО

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОГНОЗИРОВАНИЯ РЫНОЧНОГО РИСКА ФИНАНСОВЫХ РЯДОВ НА ОСНОВЕ МОДЕЛЕЙ С АВТОРЕГРЕССИОННОЙ УСЛОВНОЙ ГЕТЕРОСКЕДАСТИЧНОСТЬЮ

Проводится сравнительный анализ прогнозирования рыночного риска финансовых временных рядов с помощью методики VaR на основе моделей с авторегрессионной условной гетероскедастичностью GARCH и GJR, учитывающих скошенность и тяжелые хвосты распределений рядов доходностей. Были получены оценки рыночного риска VaR для финансовых показателей разных типов: валюты евро, финансового индекса S&P500 и акций компании Microsoft. Верификация по историческим данным показала достаточную точность прогноза риска.

Введение

Залогом успешного функционирования любого финансового института служит способность управлять своими рисками в конкретных макроэкономических условиях. Рыночные риски связаны с неопределенностью колебаний рыночной конъюнктуры — ценовыми и валютными рисками, процентным риском, ликвидностью и т. п. При измерении рыночных рисков в качестве случайной переменной берут доходность финансового актива. Финансовым показателем, характеризующим изменение доходности финансового инструмента и отражающим уровень колебаний доходности, т.е. меру риска, является волатильность. Формальной мерой волатильности служит среднеквадратическое отклонение доходности. Высокий уровень волатильности означает, что доходность изменяется в широком диапазоне. Низкий уровень волатильности актива говорит о том, что его доходность изменяется несущественно. Активы с более низким уровнем волатильности являются менее рискованными, чем активы с высоким уровнем волатильности.

Множество проведенных исследований выявили целый ряд специфических особенностей временных рядов доходности финансовых активов и их волатильности – высокие пики и тяжелые хвосты распределения, долгосрочную зависимость, кластеризацию волатильности, условную гетероскедастичность, эффект «рычага» и другие [1-5]. Однако традиционные модели временных рядов, такие как модель ARIMA, не позволяют достаточно эффективно рассчитывать величину риска в сложных условиях, как, например, в присутствии гетероскедастичности и тяжелых хвостов функций распределений входных данных. К настоящему времени предложено довольно много моделей, описывающих подобное поведение временных рядов. В последние годы наибольший интерес сосредоточен вокруг моделей из семейства стохастических условно-гауссовских моделей (ARCH - Autoregressive Conditional Heteroscedasticity), многообразие которых позволяет учесть особенности финансовых временных рядов независимо от их происхождения и дает достаточную точность прогноза. Авторегрессионная модель условной гетероскедастичности, предложенная Р. Энглом в 1982 г. [1], позже обобщенная (GARCH – Generalized ARCH) Т. Боллерслевым в 1986 г. [2], является наиболее широко используемой. Одно из направлений дальнейшего развития этих моделей заключается в попытке заменить нормальное распределение распределениями с более тяжелыми хвостами и ассиметричностью [3, 6-10].

Целью представленной работы является сравнительный анализ прогнозирования рыночного риска финансовых временных рядов на основе различных моделей с авторегрессионной условной гетероскедастичностью, учитывающих асимметричность и тяжелые хвосты распределений.

1. Модели с авторегрессионной условной гетероскедастичностью

Основной смысл модели GARCH состоит в моделировании кластеризации волатильности, т.е., если абсолютная величина доходности оказывается большой, то это приводит к

повышению условной дисперсии в последующие периоды, а при низкой условной дисперсии более вероятно появление малых значений доходности. В реальных финансовых рядах наблюдаются описанные тенденции.

Процесс GARCH порядка (p,q) $\{\varepsilon_t\}_{t=-\infty}^{+\infty}$ задается следующими соотношениями:

$$\varepsilon_t \big| \Omega_{t-1} \sim N(0,\sigma_t^2),$$

$$\sigma_t^2 = \omega + \sum_{j=1}^{p} \delta_j \sigma_{t-j}^2 + \sum_{j=1}^{q} \gamma_j \varepsilon_{t-j}^2 , \qquad (1)$$

где ε_t – значения доходности в момент времени t, нормально распределенные независимые случайные величины; $\Omega_{t-1} = (\varepsilon_{t-1}, \varepsilon_{t-2}, ...)$ – предыстория процесса $\{\varepsilon_t\}$; σ_t^2 – условная по предыстории дисперсия ε_t, т.е. $\sigma_t^2 = D(\varepsilon_t \mid \Omega_{t-1})$; ω, δ_j, γ_j – параметры модели.

При этом предполагается, что $\omega > 0, \delta_1, ..., \delta_p \geq 0, \gamma_1, ..., \gamma_q \geq 0$ и $\sum_{j=1}^{p} \delta_j + \sum_{j=1}^{q} \gamma_j < 1$. Методом оценивания параметров GARCH моделей является метод максимального правдоподобия.

На практике, как правило, используется модель GARCH (1,1) [11]:

$$\sigma_t^2 = \omega + \delta \sigma_{t-1}^2 + \gamma \varepsilon_{t-1}^2 . \qquad (2)$$

К недостаткам модели GARCH относится то, что она является симметричной и не учитывает асимметричность динамики доходностей (эффект «рычага»), которой характеризуются распределения многих финансовых рядов [4, 9].

Для учета эффекта «рычага» предложена GJR-модель, названная по фамилиям авторов (Glosten L.R., Jagannathan R., Runkle D.E [4]), которая модифицирует формулу (2):

$$\sigma_t^2 = \omega + \delta \sigma_{t-1}^2 + (\gamma + \eta I_{t-1}) \varepsilon_{t-1}^2, \qquad (3)$$

$$I_{t-1} = \begin{cases} 1, \text{ при } \varepsilon_{t-1} < 0 \\ 0, \text{ при } \varepsilon_{t-1} \geq 0 \end{cases},$$

где I – функция-индикатор, при $\eta \neq 0$ задающая асимметрию условной дисперсии σ_t^2; $\omega, \delta, \gamma, \eta$ – параметры модели.

В работе проведены исследования для моделей GARCH и GJR с четырьмя различными распределениями – нормальным, Стьюдента, скошенным t-распределением Хансена и обобщенным распределением ошибки (GED – Generalized Error Distribution). Кратко опишем указанные выше законы распределений.

Наиболее распространенным является t-распределение Стьюдента, поскольку оно при малых степенях свободы ν имеет большой куртозис:

$$p(x) = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)} \frac{1}{\sqrt{\pi\nu}} \left(1 + \frac{x^2}{\nu}\right)^{-(\nu+1/2)}, \ \nu > 2 ,$$

где $2 < \nu \leq \infty$ – количество степеней свободы, чем меньше ν, тем толще хвосты распределения; $\Gamma(\cdot)$ – гамма-функция.

Часто распределение доходностей является скошенным. Для учета этой ситуации следует использовать асимметричные распределения с толстыми хвостами. Например, можно использовать скошенное t-распределение Хансена [12], которое «склеивается» из двух половинок t-распределений Стьюдента, по-разному масштабированных:

$$p(x) = \begin{cases} Sx \cdot (1 + \zeta^2 (sx + m)^2 / \nu - 2)^{-(\nu+1/2)}, \text{ если } x \leq -m/s, \\ Sx \cdot (1 + \zeta^{-2} (sx + m)^2 / \nu - 2)^{-(\nu+1/2)}, \text{ если } x > -m/s; \end{cases}$$

$$Sx = (2s/\zeta + \zeta^{-1})[\Gamma(\nu+1/2)/\Gamma(\nu/2)](1/\sqrt{\pi(\nu-2)});$$

где параметр $4 < \nu < 30$ определяет куртозис, $-1 < \zeta < 1$ задает асимметрию, $m = \Gamma[(\nu-1)/2]\sqrt{(\nu-2)/\pi}(\zeta - 1/\zeta)/\Gamma(\nu/2)$, $s = \sqrt{(\zeta^2 + 1/\zeta^2 - 1) - m^2}$.

Семейство распределений GED охватывает симметричные распределения с различными коэффициентами куртозиса. Плотность распределения GED определяется выражением

$$p(x) = \frac{v\,\mathrm{Exp}(-0.5\,|\,x/\lambda\,|^{v})}{\lambda \cdot 2^{(1+1/v)}\Gamma(1/v)},$$

где $\lambda = (2^{(-1/\upsilon)}\Gamma(1/v)/\Gamma(3/v))^{1/2}$. Параметр v регулирует толщину хвоста. При $v = 2$ распределение GED совпадает с нормальным распределением, при $v < 2$ плотность GED имеет более толстые, а при $v > 2$ – более тонкие хвосты, чем плотность нормального распределения.

2. Оценка рисков и верификация моделей методом Value-at-Risk

Существует множество методов анализа и оценки рыночных рисков [13-16]. Для всесторонней количественной и качественной оценки рыночного риска в настоящее время в мире широко применяется методика оценки рисков Value-at-Risk (VaR). VaR — выраженная в данных денежных единицах оценка величины, которую не превысят ожидаемые в течение данного периода времени потери с заданной вероятностью. Преимуществами методики является возможность измерить риск величиной потерь, соотнесенных с вероятностью их возникновения; измерить и сравнить риски по операциям на различных рынках универсальным образом; агрегировать риски отдельных финансовых инструментов в единую величину для всего портфеля. Методика VaR используется такими международными регулирующими органами, как Банк международных расчетов, Банковская федерация Европейского Сообщества, «Группа тридцати» в качестве основы при установлении нормативов величины собственного капитала банка относительно величины его активов и предлагается данными организациями и рядом национальных центральных банков в качестве стандарта для оценки рыночного риска.

Основным сущностным элементом функционирования рыночных рисков является соотношение «риск/доходность». Искусство финансовой деятельности, по сути, представляет собой умение находить оптимальные сочетания этих элементов. Доходность x в момент времени t обычно определяется как натуральный логарифм отношения стоимостей актива

$$P:\ x_t = \ln\frac{P_t}{P_{t-1}}.$$

Приведем определение меры риска VaR [17]. Для заданного уровня достоверности $\alpha \in (0,1)$ и временного горизонта Δt мера риска VaR определяется как

$$\mathrm{VaR}_\alpha = \inf\{u \mid P[\Delta P(\Delta x, \Delta t) \le u] > \alpha\}, \qquad (4)$$

где ΔP – изменение стоимости портфеля; Δx – изменения переменных состояния за период времени Δt. Это означает, что VaR есть наибольший убыток, который может произойти на протяжении периода времени Δt с вероятностью α. Считается, что за период времени Δt состав портфеля не изменяется.

Важнейшей аналитической функцией банковского риск-менеджмента является регулярная верификация (оценка адекватности) внутренней модели по историческим данным в целях проверки ее прогнозной точности и внесения необходимых изменений. Верификация VaR-модели по историческим данным осуществляется путем ретроспективного анализа, заключающегося в подсчете частоты случаев превышения фактическими дневными убытками прогнозных значений VaR за продолжительный период времени в прошлом. Базельский комитет по банковскому надзору, основной задачей которого является внедрение единых стандартов в сфере банковского регулирования, предписывает банкам проводить оценку точности внутренних моделей по выборке из предшествующих 250 дней торгов с использованием доверительного интервала в 99% и горизонта прогнозирования в один день в качестве нормативных параметров расчета VaR [18].

Процедура проверки точности модели по методике Базельского комитета представляет собой статистический тест на отклонение фактической частоты превышений убытками дневной величины VaR от заданной вероятности в 1%, основанной на вычислении вероятностей ошибок I и II рода. Для расчета вероятностей ошибок используется биномиальный критерий. В зависимости от количества превышений модель может быть отнесена к одной

из трех зон (табл.1). Попадание модели в последние две зоны будет означать, что ее реальный доверительный интервал меньше предписанных 99%.

Таблица 1. Классификация моделей по адекватности

	Зеленая зона, адекватные модели	Желтая зона, сомнительные модели	Красная зона, неадекватные модели
Число превышений	1-4	5-9	Более 10

Критерием точности модели обычно выступает число случаев превышения реальных потерь над величиной VaR. При оценке эффективности методов и моделей прогнозирования риска используются критерии, основанные на функции потерь вида:

$$L_t^{\alpha} = \begin{cases} f_1(\varepsilon_t, VaR_t^{\alpha}), \text{если } |\varepsilon_t| < \left|VaR_t^{\alpha}\right| \\ f_2(\varepsilon_t, VaR_t^{\alpha}), \text{если } |\varepsilon_t| \geq \left|VaR_t^{\alpha}\right| \end{cases},$$

где ε – реализовавшаяся доходность в момент времени t; VAR_t^{α} – значение риска, спрогнозированное в момент времени $t-1$ с заданным доверительным уровнем α.

Рассмотрим основные критерии, базирующиеся на данной функции потерь [17,19]:

1) Количество превышений прогнозного значения за временной интервал T:

$$Lsum_{\alpha} = \sum_{t=1}^{T} L_t^{\alpha}, \text{ где } L_t^{\alpha} = \begin{cases} 0, \text{если } |\varepsilon_t| < \left|VaR_t^{\alpha}\right| \\ 1, \text{если } |\varepsilon_t| \geq \left|Va\,R_t^{\alpha}\right| \end{cases}. \quad (5)$$

Функция учитывает только сами факты наличия превышения без учета величины превышения.

2) Величина превышений за временной интервал T:

$$LF1_{\alpha} = \sum_{t=1}^{T} L_t^{\alpha}, \text{ где } L_t^{\alpha} = \begin{cases} 0, & \text{если } |\varepsilon_t| < VaR_t^{\alpha}, \\ |\varepsilon_t| - \left|VaR_t^{\alpha}\right|, & \text{если } |\varepsilon_t| \geq VaR_t^{\alpha}. \end{cases} \quad (6)$$

Данная функция показывает, насколько реальные прибыли/убытки превышают величину VAR за весь период оценки эффективности.

3) Величина средней относительной величины превышений LF2:

$$LF2_{\alpha} = \frac{\sum_{t=1}^{T} L_t^{\alpha}}{T}, \text{ где } L_t^{\alpha} = \begin{cases} 0, & \text{если } |\varepsilon_t| < VaR_t^{\alpha}, \\ \dfrac{|\varepsilon_t| - \left|VaR_t^{\alpha}\right|}{\left|VaR_t^{\alpha}\right|}, & \text{если } |\varepsilon_t| \geq VaR_t^{\alpha}. \end{cases} \quad (7)$$

Экономическая интерпретация данной величины представляет собой средний процент от зарезервированного капитала, который не покрывается в случае превышения доходностью спрогнозированного значения риска

4) Величина неиспользованного капитала:

$$LF3_{\alpha} = \sum_{t=1}^{T} L_t^{\alpha}, \text{ где } L_t^{\alpha} = \begin{cases} \left|VaR_t^{\alpha}\right| - |\varepsilon_t|, \text{если } |\varepsilon_t| < VaR_t^{\alpha}, \\ 0, \quad \text{если } |\varepsilon_t| \geq VaR_t^{\alpha}. \end{cases} \quad (8)$$

Экономическая интерпретация данного критерия: значение $LF3_{б}$ представляет собой разность капитала, который был зарезервирован для покрытия риска, и реального значения риска, т.е. величину капитала, зарезервированного сверх необходимого.

5) Критерий относительного уровня потерь:

$$Lreal_{\alpha} = \frac{LF1_{\alpha}}{\sum_{t=1}^{T} |\varepsilon_t|}. \quad (9)$$

Данный критерий показывает, какую долю составляют убытки в общей сумме доходностей. Например, может быть количество превышений одинаково, однако сумма потерь может быть разной. Таким образом, критерий $Lreal_{\alpha}$ позволяет получить информацию о полученных потерях по отношению к общей сумме доходности и сравнить модели в случае равенства количества превышений.

3. Результаты исследований

В работе рассмотрены временные ряды ежедневных значений финансовых показателей разных типов: котировки валюты EUR (евро) к российскому рублю, финансовый индекс S&P500 и акции компании Microsoft с 02.01.2009 по 02.07.2010 (в период времени, когда острая фаза кризиса уже пройдена). На рис.1 представлены соответствующие временные реализации показателей.

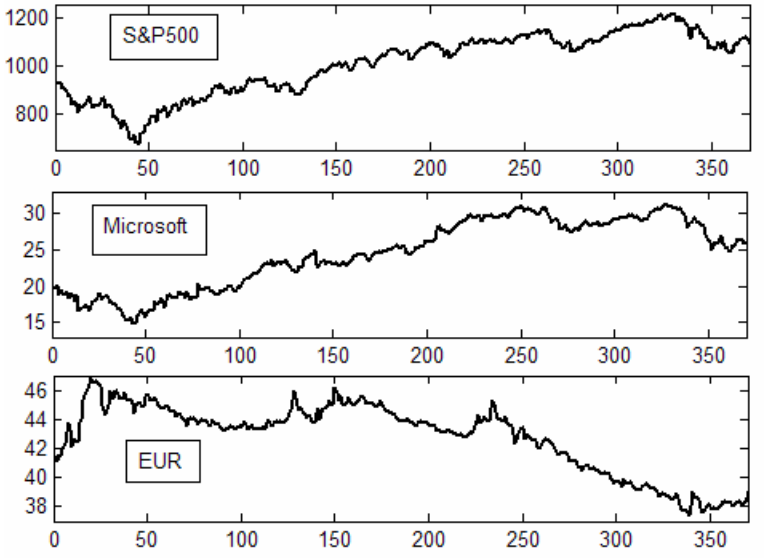


Рис. 1. Временные ряды котировок S&P500, Microsoft, EUR

Численный анализ и прогнозирование временных рядов в работе проводились с помощью специальных модулей математического пакета Matlab r2009b – Financial Toolbox, GARCH Toolbox, Oxford MFE Financial Econometrics Toolbox, Patton Skewed-t Toolbox. Для оценки параметров моделей использовался метод максимального правдоподобия. Функции максимального правдоподобия составлены на основе соответствующих конкретным моделям функций плотности распределения. Объем выборки для оценки параметров и оптимизации начальных параметров равен 370 значений. Анализируя динамику логарифмических доходностей, представленную на рис.2, можно сделать вывод о том, что финансовые ряды обладают свойством гетероскедастичности (непостоянством дисперсии), а также кластеризацией волатильности, т.е. наблюдается чередование периодов, когда финансовый показатель ведет себя непостоянно и относительно спокойно.

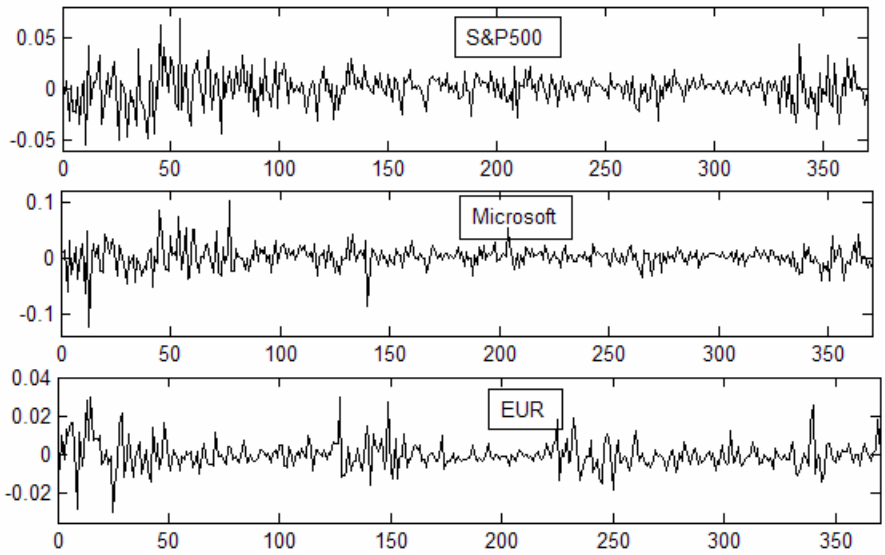


Рис. 2. Временные ряды логарифмических доходностей S&P 500, Microsoft, EUR

Проведенный численный анализ автокорреляционных функций квадратов доходностей показал, что условное стандартное отклонение зависит от предыдущих своих значений, и образуются кластеры волатильности. Для проверки выборок на наличие гетероскедастичности был использован тест Энгла (Engle's ARCH Test), который показал, что рассматриваемые ряды доходностей обладают гетероскедастичностью.

В табл.2 приведены основные статистики для выборок доходностей исследуемых финансовых индексов. Среднее значение выборки у каждого ряда близко к нулю, т.е. сумма положительных и сумма отрицательных доходностей практически совпадает. Коэффициент куртозиса рядов является большим, чем это предусмотрено нормальным распределением. Это объясняется возможной тяжестью хвостов распределения и/или островершинностью распределения.

Таблица 2. Выборочные статистики для доходностей

	S&P500	Microsoft	EURO
Среднее	2.4659e-004	4.4446e-004	-1.5266e-004
Стандартное отклонение	0.0158	0.0209	0.0071
Скошенность	-0.0611	-0.3093	0.7245
Куртозис	5.1377	8.7443	6.8180

Анализируя выборочные статистики и гистограммы плотностей распределения доходностей, показанные на рис.3, можно прийти к выводу, что хвосты распределения исследуемых рядов доходностей являются «тяжелыми», т.е. экстремальные выбросы в финансовых активах встречаются чаще, чем это предусмотрено нормальным распределением.

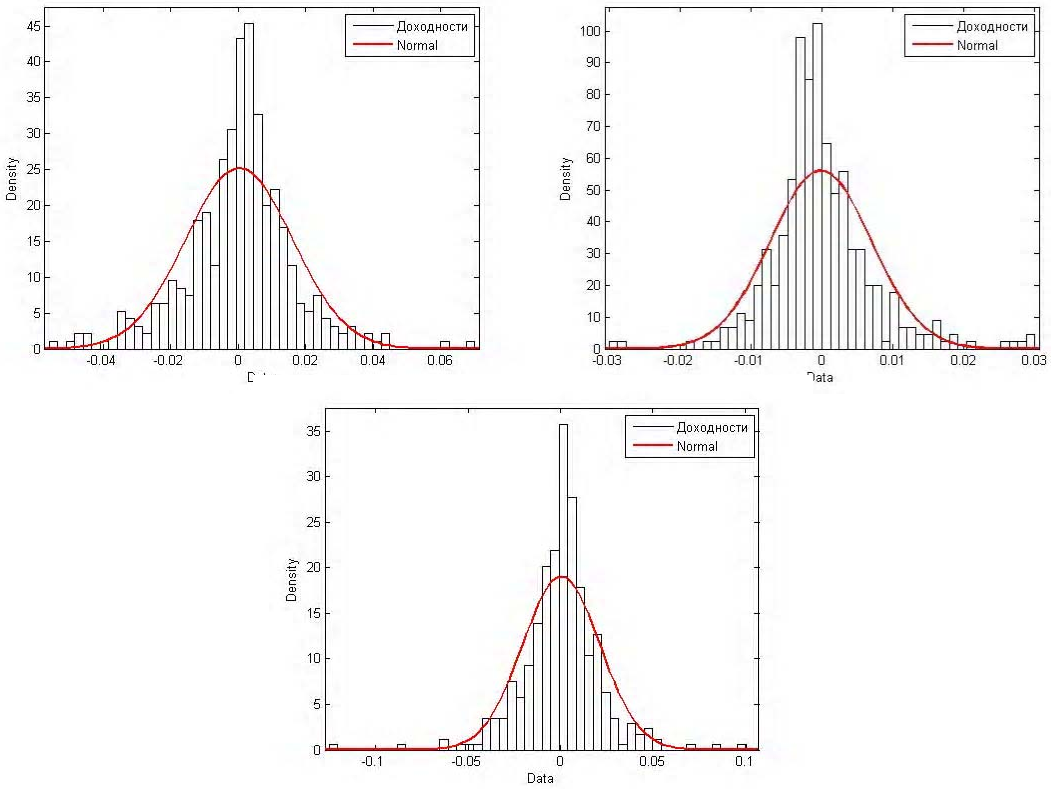


Рис. 3. Гистограммы для логарифмических доходностей S&P 500, Microsoft, EUR

Для прогнозирования динамики волатильности в работе были использованы модели GARCH(1,1) и GJR(1,1) со следующими законами распределений:

– нормальным (GARCH, GJR);

– Стьюдента (GARCH-*t*, GJR-t) с параметром числа степеней свободы ν -student;

– скошенным t -распределением Хансена (GARCH-skew, GJR-skew) с параметрами ν -skewness, ζ -skewness, определяющими куртозис и асимметрию;

– обобщенным распределением ошибки GED (GARCH-GED, GJR-GED) с параметром ν -GED, регулирующим коэффициент куртозиса.

В табл.3 приведены результаты оценивания параметров моделей рассматриваемых финансовых рядов на основе представленных распределений. Для сравнения моделей с различными пространствами параметров использовались информационные критерии Акаике, Шварца и Хеннана-Куинна, которые численно построены так, чтобы учесть влияние на качество подгонки модели двух противоположных тенденций. При сравнении двух типов моделей предпочтение отдается той, которая имеет наименьшие значения критериев. Для всех индексов наибольшее значение функции правдоподобия достигалось при использовании моделей со скошенным распределением Стьюдента - SKEWt. Однако не для всех индексов эта модель была лучшей. Также хорошие результаты были достигнуты при использовании моделей GARCH-t и GJR-t.

Таблица 3. Результаты оценки параметров моделей

Параметры	Модели	S&P500	EURO	Microsoft
ν -student	GARCH-t	6,5538	3,7588	3,9785
	GJR-t	8,9725	3,7574	4,0503
ν -skewness	GARCH-skew	9,3323	3,9762	4,2474
	GJR- skew	13,63	3,9721	4,4933
ζ -skewness	GARCH-skew	-0,1774*	0,279	-0,0996*
	GJR-skew	0,2049*	0,284	0,1005*
ν -GED	GARCH-GED	1,3892	1,213	1,1441
	GJR-GED	1,4673	1,2302	1,1453
η -GJR	GJR	0,1555	-0,1307*	0,0288
	GJR-t	0,1554	-0,1307*	0,0288
	GJR-skew	0,1535	-0,1307*	0,0288
	GJR-GED	0,1535	-0,1307*	0,0288

* – не значимый параметр

Оценки параметров GARCH значимы во всех моделях и для всех индексов, что свидетельствует о наличии эффекта кластеризации волатильности. Параметр η значим во всех моделях семейства GJR для S&P500 и Microsoft, что свидетельствует о наличии эффекта рычага. В моделях GARCH-SKEWt и GJR-SKEWt параметр куртозиса ν распределения Хансена значим для всех индексов, его значение находится в промежутке между 3,9 до 14, что свидетельствует о наличии толстых хвостов распределений. Параметр скошенности ζ значим только в индексе евро, что говорит о том, что этот индекс обладает скошенностью.

На рис. 4 представлены графики теоретических функций плотности нормального распределения, распределения Стьюдента, распределения Хансена и распределения GED с параметрами, оцененными в моделях GARCH-t, GARCH-skew и GARCH-GED.

С помощью полученных параметров моделей можно строить однодневные прогнозы волатильности и получать прогнозы риска методикой VaR:

$$\mathrm{VaR}_\alpha = (\mathrm{M} + \mathrm{k}_\alpha \sigma_t),$$

где M – среднее значение доходности финансового актива; k_α – α -квантиль соответствующего распределения; σ_t – прогнозное значение стандартного отклонения ряда доходностей.

Согласно Базельским рекомендациям [18], прогноз доходности осуществляется на один день вперед, адекватность моделей оценивается по выборке объемом 250 значений. На рис. 4 представлены графики значений прогноза VaR для исследуемых рядов доходностей при применении моделей GARCH(1,1) (слева) и GJR(1,1) (справа) с разными распределениями для уровня доверия 0.99.

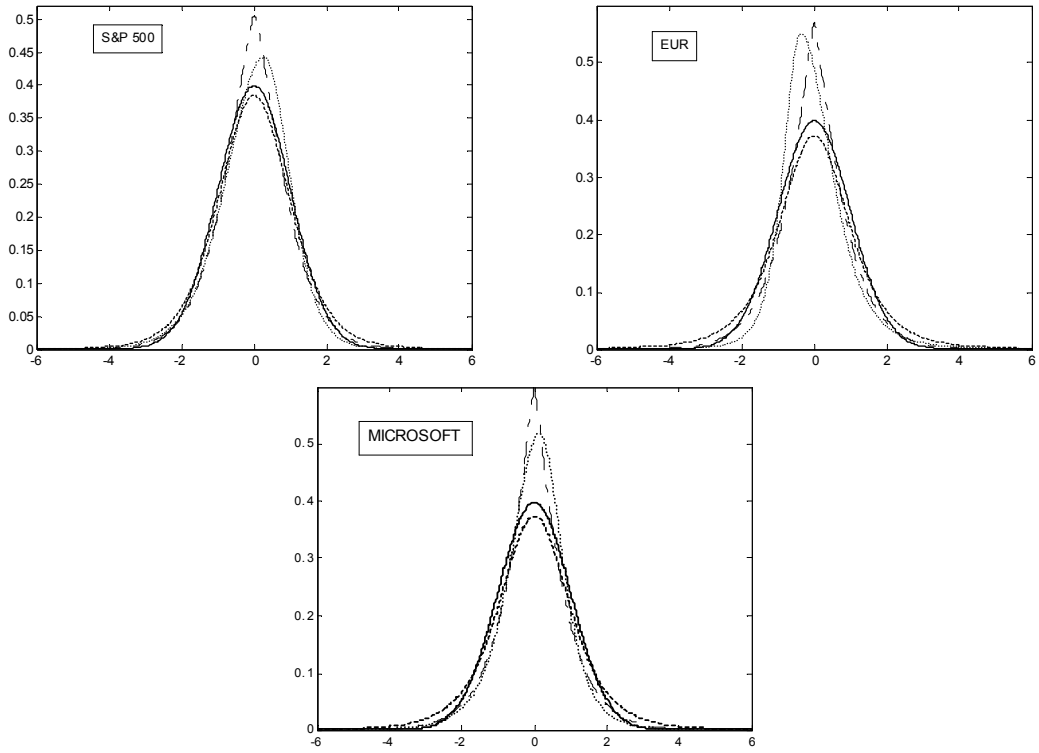


Рис. 4. Графики теоретических функций плотности распределения доходностей S&P 500, EUR, Microsoft (—— - плотность нормального распределения, – – – -распределения Стьюдента, – . – . – - распределения Хансена , ····· - распределения GED)

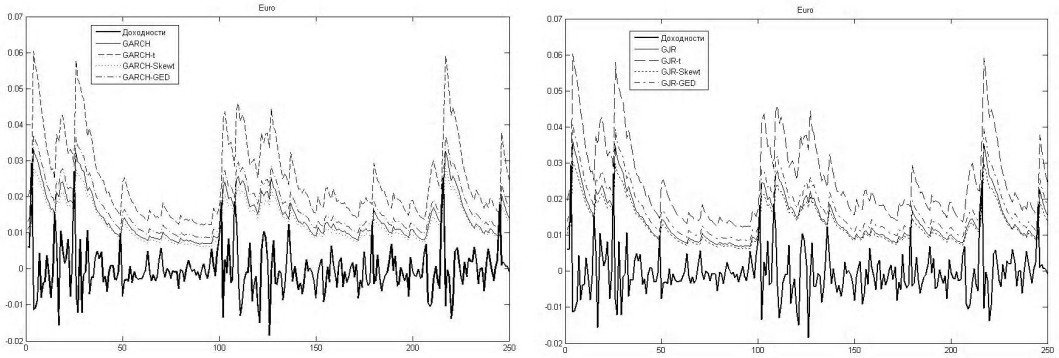
Для проверки адекватности моделей были использованы функции потерь (5)-(9). Результаты оценки адекватности моделей прогноза риска по различным критериям приведены в табл. 4.

Таблица 4. Результаты оценки адекватности моделей

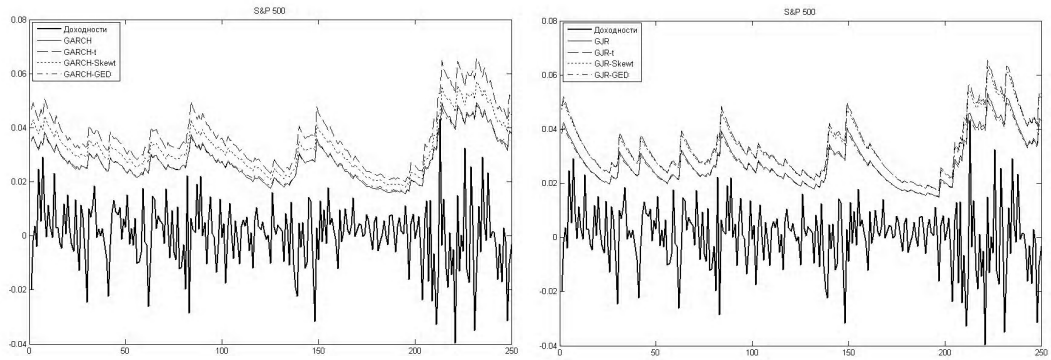
Индекс	GARCH	GARCH-t	GARCH-skew	GARCH-GED	GJR	GJR-t	GJR skew	GJR-GED
Результаты оценки адекватности моделей по критерию Lsum								
S&P	9	1	2	7	6	2	2	6
Euro	9	2	13	6	9	2	12	6
Microsoft	5	2	2	2	5	2	2	2
Результаты оценки адекватности моделей по критерию LF1								
S&P	0,0215	0,0002	0,0035	0,0221	0,0185	0,0023	0,0033	0,0209
Euro	0,0493	0,0065	0,0623	0,0305	0,0484	0,0065	0,0595	0,0289
Microsoft	0,077	0,0123	0,0467	0,0415	0,0785	0,0206	0,0516	0,0444
Результаты оценки адекватности моделей по критерию LF2								
S&P	0,8715	0,0091	0,1593	0,8636	0,9091	0,1028	0,1479	1,011
Euro	3,6671	0,294	5,0206	1,9415	3,5864	0,2934	4,6742	1,8105
Microsoft	2,0915	0,1697	0,9739	0,8154	2,1757	0,3167	1,156	0,8976
Результаты оценки адекватности моделей по критерию LF3								
S&P	4,7097	6,9389	5,8426	4,734	4,71	6,205	6,0285	4,5675
Euro	2,6177	5,2158	2,2842	3,2499	2,6105	5,2175	2,3315	3,2541
Microsoft	6,4442	12,5809	8,3799	9,045	6,4833	12,5218	8,2548	9,2225
Результаты оценки адекватности моделей по критерию Lreal								
S&P	0,01	0,0001	0,0016	0,0103	0,0086	0,0011	0,0015	0,0097
Euro	0,0435	0,0058	0,055	0,0269	0,0427	0,0057	0,0526	0,0256
Microsoft	0,0282	0,0045	0,0171	0,0152	0,0288	0,0075	0,0189	0,0163

Результаты тестирования по критерию количества превышений позволяют отнести почти все рассматриваемые модели к зеленой и желтой зонам, что свидетельствует о возможности их использования в соответствии с базельскими требованиями. В красную зону попали только модели GARCH-skew и GJR- skew для ряда доходностей котировок евро. Произведенный анализ позволяет говорить о том, что наиболее точные оценки VAR для всех финансовых показателей, с точки зрения минимизации значений бинарной функции потерь L_{sum} , дают модели GARCH-t и GJR-t – с распределением Стьюдента.

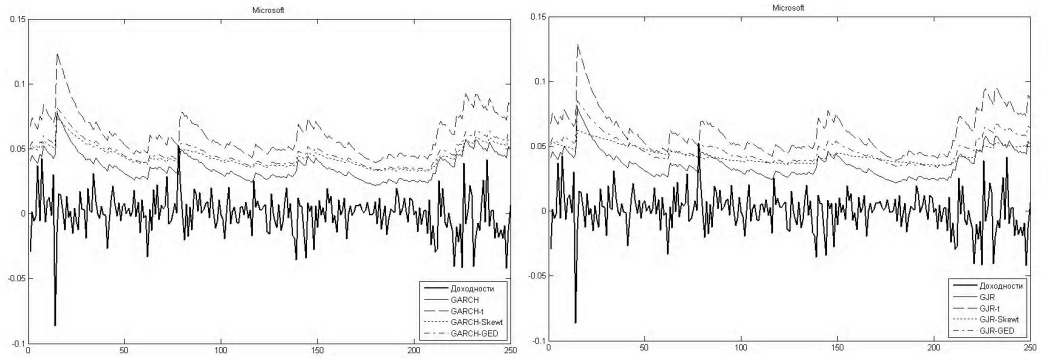
Более того, эти модели показывают лучшие результаты тестирования по всем другим критериям адекватности. С точки зрения среднего непокрытого риска, учитывающего величину превышений оценки риска (LF2), модели GARCH-t и GJR-t дают наименьшее значение по сравнению с другими моделями, однако если рассматривать с точки зрения неиспользованного рискового капитала (LF3), то эти модели являются менее эффективными по сравнению с остальными, так как дают более высокую оценку по критерию LF3.



Прогноз EUR при применении GARCH(1,1) и GJR(1,1)



Прогноз S&P 500 при применении GARCH(1,1) и GJR(1,1)



Прогноз Microsoft при применении GARCH(1,1) и GJR(1,1)

Рис. 5. Графики значений прогноза VaR для исследуемых рядов доходностей

На основе полученных результатов был проведен многокритериальный анализ моделей в целях определения парето-оптимальных моделей. Под парето-оптимальной моделью понимается такая модель, которая в сравнении с другими дает меньшую величину непокрытого и неиспользованного капитала. [19] Для этого на двумерной плоскости размещаются соответствующие точки, каждая из которых характеризует модель с точки зрения средней величины избыточного капитала и средней величины недостаточности капитала, резервируемого для покрытия возможных убытков. На рис. 6 показаны диаграммы, по которым можно определить парето-оптимальные модели.

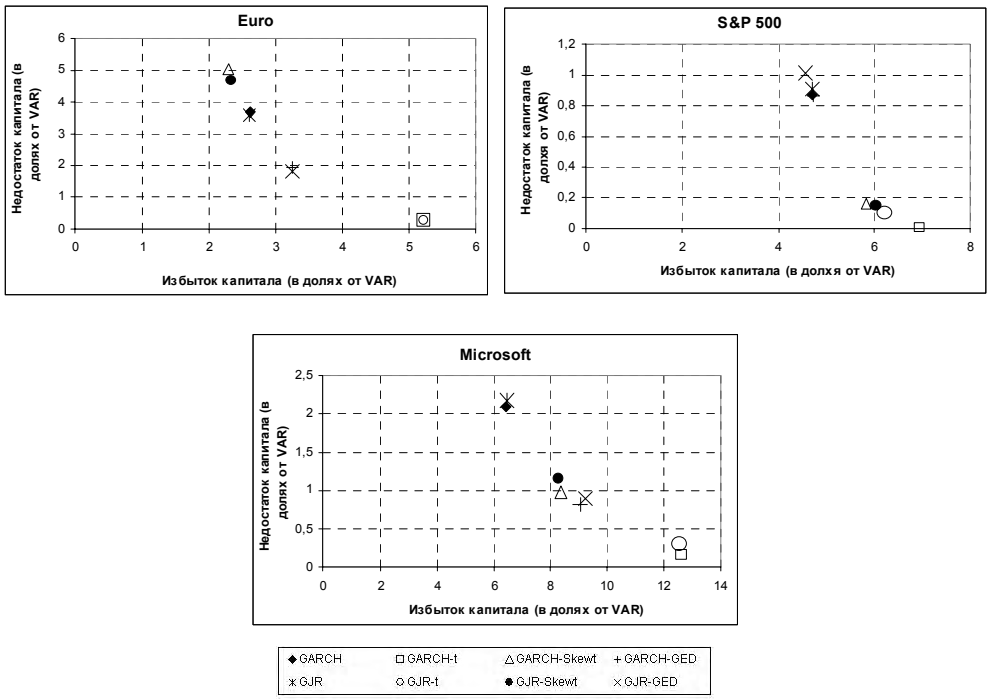


Рис.6. Многокритериальный анализ моделей

По результатам многокритериального анализа можно сделать вывод, что для Microsoft оптимальной моделью является GARCH-SKEWt, а также модель GJR–SKEWt, однако она чуть менее эффективна по критерию недостаточности капитала. Для евро и S&P500 нельзя выделить какую-то одну модель: риск-менеджер сам должен выбрать более приоритетный для него критерий. Для ряда евро модель GJR идентична модели GARCH, модель GJR-GED – модели GARCH-GED, а модель GJR-t – модели GARCH-t. Для ряда S&P500 модели GARCH, GARCH-GED, GJR, GJR-GED эффективны по критерию избытка капитала, тогда как модели GARCH-t, GARCH-SKEWt, GJR-t, GJR- SKEWt более эффективны по критерию недостаточности капитала.

Выводы

В настоящее время распространенным методом оценивания рисков является методика Value-at-Risk, которая дает вероятностную оценку потенциальных убытков в течение определенного периода при экспертно заданном доверительном уровне. В работе проведен сравнительный анализ прогнозирования рыночного риска финансовых временных рядов с помощью методики VaR на основе моделей с авторегрессионной условной гетероскедастичностью GARCH и GJR, учитывающих скошенность и тяжелые хвосты распределений рядов доходностей. Были получены оценки рыночного риска VaR для финансовых показателей разных типов: валюты евро, финансового индекса S&P500 и акций компании Microsoft. Верификация по историческим данным показала достаточную точность прогноза риска.

В работе показано, что для ряда евро модели семейства GJR являются более предпочтительными по сравнению с моделями GARCH, так как они учитывают эффект рычага. Кроме того, использование t-распределения Стьюдента дает лучший результат. Распределение Хансена позволяет учесть известный эффект лептокуртозиса финансовых данных и может дать дополнительное улучшение результата, однако при выборе модели надо сле-

дить, чтобы при этом были значимы оценки параметров распределения и спецификация модели в целом.

Список литературы: 1. *Engle R.F.* Autoregressive conditional heteroskedasticity with estimates of the UK inflation // Econometrics. 1982. V.50. P.987-1008. **2.** *Bollerslev T.* Generalised Autoregressive Conditional Heteroscedasticity // Journal of Econometrics. 1986. V.31. P.307-327. **3.** *Мандельброт Б.* Фрактальная геометрия природы. М.: Институт компьютерных исследований, 2002. 656 с. **4.** *Glonsten L., Jagannathan R., Runkle D.* Relationship between the expected value and the volatility of the nominal excess return on stocks // Journal of Finance. 1993. V.48. P.1779 – 1802. **5.** *Петерс Э.* Фрактальный анализ финансовых рынков. М.: Интернет-трейдинг, 2004. 333 с. 6. *Anders W.* Garch forecasting performance under different distribution assumptions // Journal of Forecast. 2006. №25. P.561-578. **7.** *Daal E., Yu J.* Volatility clustering, leverage effects, and jump dynamics in the US and emerging Asian equity markets // Journal of Banking & Finance. 2007. №31. P. 2751-2769. **8.** *Jondeau E., Rockinger M.* Conditional volatility, skewness, and kurtosis: existence, persistence, and comovements // Journal of Economic Dynamics and Control. 2003. №27. P.1699–1737. **9.** *Zifter A.* Asymmetric and Fractionally Integrated Garch Models with (Skewed) Student-t and Ged Distribution in Risk Management: An Application on Eurobond // Presented in VIII National Finance Symposium, Istanbul Technical University 2004. **10.** *Bakshi G., Kapadia N., Madan D.* Stock Returns Characteristics, Skew, Laws, and the Differential Pricing for Individual Equity Options // The Review of Financial Studies. 2003. №16. P.101-143. **11.** *Morgan J.P.* RiskMetrics - technical document. New York: Morgan Guaranty Trust Company, 1996. 214 p. **12.** *Hu W., Kercheval A.* The Skewed t Distribution for Portfolio Credit Risk // Econometrics. 2008. V.22. P.55-83. **13.** *Рэдхед К., Хьюс С.* Управление финансовыми рисками. М.: ИНФРА-М, 2000. 288 с. **14.** *Рогов М.А.* Риск-менеджмент. М.: Финансы и статистика, 2001. 120 с. **15.** *Кузнецов В.Е.* Измерение финансовых рисков // Банковские технологии. 1997. №7. С.76-78. **16.** *Dorfman M.* Introduction to Risk Management and Insurance. Pearson Higher Education, 1997. 600 p. **17.** *Меньшиков И.С., Шелагин Д.А.* Рыночные риски: модели и методы. М.: Вычислительный центр РАН, 2000. 54 с. **18.** Supervisory Framework for The Use of «Backtesting» in Conjunction with The Internal Models Approach to Market Risk Capital Requirements // Basle Committee on Banking Supervision, January, 1996. **19.** *Милосердов А.А., Герасимова Е.Б.* Рыночные риски: формализация, моделирование, оценка качества моделей. Тамбов: Изд-во Тамб. гос. техн. ун-та, 2004. 116 с.

Поступила в редколлегию 21.08.2010

Кириченко Людмила Олеговна, канд. техн. наук, доцент кафедры прикладной математики ХНУРЭ. Научные интересы: самоподобные случайные процессы, фрактальный и мультифрактальный анализ. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-640. E-mail: ludmila@kture.kharkov.ua.

Рвачева Наталия Михайловна, магистр системного анализа ХНУРЭ. Научные интересы: анализ и прогнозирование финансовых временных рядов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-640.

Стороженко Александра Владимировна, канд. техн. наук, доцент кафедры экономической кибернетики ХНУРЭ. Научные интересы: бизнес-прогнозирование. Адрес: Украина, 61166, Харьков, просп. Ленина, 14, тел. 70-21-490.

УДК 681.326:519.713

В.И. ХАХАНОВ, С.В. ЧУМАЧЕНКО, Е.И. ЛИТВИНОВА

ЛОГИЧЕСКИЙ ВЕКТОРНЫЙ АНАЛИЗ АССОЦИАТИВНЫХ ТАБЛИЦ

Предлагается инфраструктура логического анализа ассоциативных таблиц, позволяющая выполнять обработку взаимодействия входного вектора с n-мерным алгебрологическим пространством, задаваемым с помощью упорядоченных и структурированных таблиц проблемно-ориентированных данных, которые представляют собой ассоциативные модели поведения логических объектов. Для оценки взаимодействия векторов в алгебрологическом пространстве разработан универсальный критерий качества, позволяющий находить и оценивать квазиоптимальное решение в задачах ассоциативно-логического поиска информации. Приводятся примеры использования инфраструктуры и алгебрологических процедур для решения традиционных задач логического анализа, подтверждающие эффективность и практическую направленность алгебраических моделей.

Цель – разработка элементов алгебры логического анализа ассоциативных таблиц для реализации инфраструктуры обработки больших объемов информации при решении задач поиска, принятия решений и диагностирования технических состояний.

Задачи исследования: 1. Мотивация и актуальность процедур анализа информационных массивов, сформатированных в графы ассоциативных таблиц. 2. Разработка алгебры логического анализа ассоциативных таблиц как эффективного математического аппарата, ориентированного на параллельно-конвейерную обработку векторных и табличных массивов. 3. Создание метрики для оценивания взаимодействия векторов и таблиц в процессе поиска оптимального решения в дискретном булевом пространстве. 4. Решение практических задач нахождения квазиоптимального покрытия и диагностирования неисправностей или ошибок в программных продуктах.

Источники исследования: 1. Аппаратные средства решения информационно-логических задач [1-4]. 2. Алгебраические системы для описания и анализа ассоциативно-логических структур данных [5-8]. 3. Критерии оценивания вариантов поиска решений в дискретном булевом пространстве [9-12]. 4. Методы решения задач покрытия и диагностирования на основе анализа таблиц [13-16].

1. Актуальность технологий векторного параллельного анализа

Современный рынок электронных и информационных технологий нуждается в новых носителях больших объемов информации, эффективных структурах данных для хранения и поиска требуемой информации, новых интеллектуальных и компьютерных средствах, точно и быстро обеспечивающих любые запросы пользователя. Современные компьютеры ориентированы в своей основе на арифметические вычисления, поэтому не всегда являются эффективным инструментом для решения логических, ассоциативных и поисковых задач в существующем океане информации. Поэтому актуальной представляется разработка новых специализированных технологий, ориентированных на использование ограниченной системы логических команд для анализа в реальном масштабе времени больших информационных массивов, организованных в эффективные структуры данных. Если исключить из рассмотрения арифметические операции, которые не свойственны человеческому мозгу, то все ресурсы компьютера можно переориентировать на повышение мощности и производительности вычислителя для решения чрезвычайно большого класса проблем логического, ассоциативного анализа, распознавания образов, диагностирования и поиска информации.

Создание ассоциативно-логического «движка» должно быть ориентировано только на параллельные вычисления, связанные с обработкой двоичных и многозначных векторов, таблиц, матриц на основе использования мультипроцессора, ассоциативной памяти, регистров. Для оценивания полученных решений необходимо разработать арифметически независимый критерий оценки качества в векторном логическом пространстве, определяемый

соотношением нулевых и единичных разрядов. Таким образом, лучшим решением при разработке упомянутого «движка» будет полное исключение всех арифметических операций и команд на их основе, что позволит на несколько порядков повысить быстродействие выполнения векторных логических операций над информационными массивами больших объемов. Теория должна быть ориентирована только на логические операции, оперировать не переменными, а векторами и матрицами (таблицами), максимально распараллеливать все вычислительные процессы, обеспечивать масштабируемость как размерности операндов, так и исполняемых программных приложений для эффективного достижения цели с требуемым качеством удовлетворения запроса.

Сущность данного исследования – создание проблемно ориентированной алгебры логического анализа ассоциативных матриц на основе минимального числа логических операций, ориентированных на параллельно-конвейерную обработку табличных и векторных форм данных для существенного повышения быстродействия анализа больших объемов информации при решении задач поиска, принятия решений, распознавания образов и диагностирования технических состояний объектов или процессов.

2. Инфраструктура логического анализа ассоциативных таблиц

Аналитическая модель описания и решения логических уравнений информационного анализа представлена системой выражений:

$$P(m,A)=Q(m,A)=[0{,}0;1{,}0];$$

$$Q(m,A)=\frac{1}{3}[d(m,A)+\mu(m\in A)+\mu(A\in m)];$$

$$P(m,A)=1\leftarrow(m=A);$$

$$P(m,A)=\max_i Q_i(m,A_i);$$

$$Q(m,A_i)=\{Q_1,Q_2,...,Q_i,...,Q_n\};$$

$$P(m,A)=m\,\Delta\,(A_1,A_2,...,A_i,...,A_t);$$

$$\Delta=\{\neg,\wedge,\vee\};$$

$$A=(A_1,A_2,...,A_i,...,A_t);$$

$$A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is});$$

$$A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{tsq});$$

$$m=(m_1,m_2,...,m_r,...,m_q);$$

$$P(m,A)=m\wedge\{A_1,A_2,...,A_i,...,A_t\}=p;\ p=\{p_1,p_2,...,p_r,...,p_n\};$$

$$m\wedge(A_1\vee A_2\vee...\vee A_i\vee...\vee A_t)=\max_i Q_i(m,A_i);$$

$$p_r=m\underset{i=\overline{1,t}}{\overset{j=\overline{1,s}}{\wedge}}A_{ij}\leftarrow\begin{cases}\forall i\exists j\ (m\wedge A_{ij}\neq\varnothing);\\ \exists i\ \forall j(m\wedge A_{ij}=\varnothing)\wedge d(m,A_{ij})=\max;\end{cases}\ p_r\in p;$$

$$m\wedge A_{ij}=m_r\overset{q}{\underset{r=1}{\wedge}}A_{ijr}=\begin{cases}m\leftarrow m_r\overset{q}{\underset{r=1}{\wedge}}A_{ijr}=m_r;\\ A_{ij}\leftarrow m_r\overset{q}{\underset{r=1}{\wedge}}A_{ijr}=A_{ijr};\\ m\wedge A_{ij}\leftarrow m_r\overset{q}{\underset{r=1}{\wedge}}A_{ijr}=m_r\vee A_{ijr};\\ \varnothing\leftarrow\exists(m_r\overset{q}{\underset{r=1}{\wedge}}A_{ijr}=\varnothing);\end{cases}\qquad(1)$$

$$P(m,A)=p_r\leftarrow Q_i(m,p_r)=\max,\ r=\overline{1,n},p_r\in p.$$

Здесь определены следующие предикаты или высказывания. Выражение $P(m,A)=Q(m,A)$ с помощью операций объединения, пересечения и дополнения задает аналитическую модель вычислительного процесса в виде предиката, определенного на единичном интервале $Q(m,A)=[0,0;1,0]$ интегральным критерием принадлежности входного вектора множеству дискретных булевых состояний. Упорядоченная совокупность $A=(A_1,A_2,...,A_i,...,A_t)$ взаимодействующих ассоциативных таблиц объединяется на системном уровне обобщенным графом целевой функциональности. Предикат $A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{is})=1$ представлен совокупностью векторов, формирующих ассоциативную таблицу. Вектор $A_{ij}=(A_{ij1},A_{ij2},...,A_{ijr},...,A_{tsq})$ задает ассоциативное решение как совокупность состояний переменных, где каждая координата определена в троичном алфавите $A_{ijr}\in\{0,1,x\}=\beta$. Взаимодействие $P(m,A)$ входного вектора запроса $m=(m_1,m_2,...,m_r,...,m_q)$ с графом ассоциативных таблиц $A=(A_1,A_2,...,A_i,...,A_t)$ формирует множество решений $p=\{p_1,p_2,...,p_r,...,p_n\}$, каждое из которых имеет интегральную оценку качества $Q(m,A)=\{Q_1,Q_2,...,Q_i,...,Q_n\}$. Выбор лучшего решения осуществляется путем вычисления функционала:

$$P(m,A)=p_r\leftarrow Q_i(m,p_r)=\max,\ r=\overline{1,n},p_r\in p.$$

3. Элементы алгебры векторного анализа ассоциативных матриц

Матрица (таблица), размерностью $n\times m$, может быть представлена векторами строк или столбцов $[A_i]\vee[A_j]$, где каждая координата матрицы или вектора определена в алфавите $\{0,1\}$:

$$A=[A_{ij}]=\{[A_i],[A_j]\},\ [A_i]=(A_1,A_2,...,A_i,...,A_n),\ [A_j]=(A_1,A_2,...,A_j,...,A_t),\ A_{ij}=\{0,1\},$$
$$A_i=(A_{i1},A_{i2},...,A_{ij},...,A_{in}),\ A_j=(A_{j1},A_{j2},...,A_{ji},...,A_{jt}). \qquad (2)$$

Матрица взаимодействует с одним из входных векторов, ориентированных на обработку строк или столбцов $m^a\vee m^b$:

$$m=m^a\vee m^b,\ m^a=(m_1,m_2,...,m_i,...,m_n),\ m^b=(m_1,m_2,...,m_j,...,m_t). \qquad (3)$$

По существу матрица представляет собой любое ассоциативное отношение (таблицу истинности, таблицу предикатов), заданное в n- (m-) мерном векторном двоичном пространстве, где каждый вектор есть точка, принадлежащая данному пространству (т.е. элемент пространства). Взаимодействие m-вектора с матрицей А преследует цель определить отношение принадлежности каждого вектора матрицы или точки пространства к m-вектору или выполнить разделение пространства точек на два подмножества. Для решения данного класса задач вводятся предикаты, инвариантные по отношению к обработке строк и столбцов матрицы, основанные на использовании логических операций (И, ИЛИ, НЕ).

1) Дизъюнкция конъюнктивного взаимодействия между m-вектором и А-матрицей:

$$P_1(m,A)=m\wedge A=m\wedge[A_1\vee A_2\vee...\vee A_{i(j)}\vee...\vee A_{n(t)}]=$$

2)
$$=\begin{cases}P^a=P^a\vee m^a\overset{n}{\underset{i=1}{\wedge}}A_i\leftarrow(m^a\overset{n}{\underset{i=1}{\wedge}}A_i\neq\varnothing);\\ P^b=P^b\vee m^b\overset{t}{\underset{j=1}{\wedge}}A_j\leftarrow(m^b\overset{t}{\underset{j=1}{\wedge}}A_j\neq\varnothing).\end{cases} \qquad (4)$$

3) Логическое умножение конъюнкции единичных по m векторов матрицы А на отрицание дизъюнкции нулевых по m векторов А:

$$P_2(m,A)=(\underset{\forall m_{i(j)}=1}{\wedge}A_{i(j)})\wedge\overline{(\underset{\forall m_{i(j)}=0}{\vee}A_{i(j)})}.$$

4) Логическое умножение дизъюнкции единичных по m векторов матрицы А на отрицание дизъюнкции нулевых по m векторов А:

$$P_3(m,A)=(\bigvee_{\forall m_{i(j)}=1} A_{i(j)})\wedge(\overline{\bigvee_{\forall m_{i(j)}=0} A_{i(j)}}). \qquad (5)$$

Для всех трех операций качество взаимодействия между m-вектором и A-матрицей оценивается с помощью теоретико-множественной метрики:

$$\begin{aligned}
&Q=\frac{1}{3}[d(m,A_{i(j)})+\mu(m\in A_{i(j)})+\mu(A_{i(j)}\in m)],\\
&d(m,A_{i(j)})=\frac{1}{n}\{n-\mathrm{card}[m\bigcap_{i(j)=1}^{n(t)}A_{i(j)}=\varnothing]\};\\
&\mu(m\in A_{i(j)})=2^{\mathrm{card}[m\cap A_{i(j)}]-\mathrm{card}[A_{i(j)}]}\leftarrow \mathrm{card}[m\cap A_{i(j)}]=\\
&=\mathrm{card}[m\bigcap_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,\mathrm{card}[A_{i(j)}]=\mathrm{card}[\bigcup_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,[m\bigcap_{i(j)=1}^{n(t)}A_{i(j)}\neq\varnothing];\\
&\mu(A_{i(j)}\in m)=2^{\mathrm{card}[m\cap A_{i(j)}]-\mathrm{card}[m]}\leftarrow \mathrm{card}[m\cap A_{i(j)}]=\\
&=\mathrm{card}[m\bigcap_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,\mathrm{card}[m]=\mathrm{card}[\bigcup_{i(j)=1}^{n(t)}m=x]\,\&\,[m\bigcap_{i(j)=1}^{n(t)}A_{i(j)}\neq\varnothing].
\end{aligned} \qquad (6)$$

Данные формулы определения качества взаимодействия m-вектора и A-матрицы изоморфны следующим алгебрологическим выражениям:

$$\begin{aligned}
&Q=\frac{1}{3}[d(m,A_{i(j)})+\mu(m\in A_{i(j)})+\mu(A_{i(j)}\in m)],\\
&d(m,A_{i(j)})=\frac{1}{n}\{n-\mathrm{card}[m\bigwedge_{i(j)=1}^{n(t)}A_{i(j)}=0]\};\\
&\mu(m\in A_{i(j)})=2^{\mathrm{card}[m\wedge A_{i(j)}]-\mathrm{card}[A_{i(j)}]}\leftarrow \mathrm{card}[m\wedge A_{i(j)}]=\\
&=\mathrm{card}[m\bigwedge_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,\mathrm{card}[A_{i(j)}]=\mathrm{card}[\bigvee_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,[m\bigwedge_{i(j)=1}^{n(t)}A_{i(j)}\neq 0];\\
&\mu(A_{i(j)}\in m)=2^{\mathrm{card}[m\wedge A_{i(j)}]-\mathrm{card}[m]}\leftarrow \mathrm{card}[m\wedge A_{i(j)}]=\\
&=\mathrm{card}[m\bigwedge_{i(j)=1}^{n(t)}A_{i(j)}=x]\,\&\,\mathrm{card}[m]=\mathrm{card}[\bigvee_{i(j)=1}^{n(t)}m=x]\,\&\,[m\bigwedge_{i(j)=1}^{n(t)}A_{i(j)}\neq 0].
\end{aligned} \qquad (7)$$

Здесь кодирование теоретико-множественных символов четырехзначного алфавита для выполнения операций в алгебре логики определяется в следующем виде:

$$T(\alpha\rightarrow\beta)=\begin{array}{|c|cccc|}\hline \alpha & 0 & 1 & x & \varnothing\\ \hline \beta & 01 & 10 & 11 & 00\\ \hline\end{array}. \qquad (8)$$

В таком алфавите все теоретико-множественные операции, согласно изоморфизму, имеют аналоги в булевой алгебре, что позволяет параллельно выполнять алгебрологические регистровые операции над векторами. Если векторы $A_{i(j)}$, m определены в двоичном алфавите, то формулы вычисления качества взаимодействия m-вектора и A-матрицы существенно упрощаются:

$$
\begin{aligned}
&Q = \frac{1}{3}[d(m, A_{i(j)}) + \mu(m \in A_{i(j)}) + \mu(A_{i(j)} \in m)],\\
&d(m, A_{i(j)}) = \frac{1}{n}\{n - \mathrm{card}[m \underset{i(j)=1}{\overset{n(t)}{\wedge}} A_{i(j)} = 0]\};\\
&\mu(m \in A_{i(j)}) = 2^{\mathrm{card}[m \wedge A_{i(j)}] - \mathrm{card}[A_{i(j)}]} \leftarrow \mathrm{card}[m \wedge A_{i(j)}] =\\
&= \mathrm{card}[m \underset{i(j)=1}{\overset{n(t)}{\wedge}} A_{i(j)} = 1] \,\&\, \mathrm{card}[A_{i(j)}] = \mathrm{card}[\underset{i(j)=1}{\overset{n(t)}{\vee}} A_{i(j)} = 1];\\
&\mu(A_{i(j)} \in m) = 2^{\mathrm{card}[m \wedge A_{i(j)}] - \mathrm{card}[m]} \leftarrow \mathrm{card}[m \wedge A_{i(j)}] =\\
&= \mathrm{card}[m \underset{i(j)=1}{\overset{n(t)}{\wedge}} A_{i(j)} = 1] \,\&\, \mathrm{card}[m] = \mathrm{card}[\underset{i(j)=1}{\overset{n(t)}{\vee}} m = 1].
\end{aligned}
\tag{9}
$$

Фактически функция принадлежности определяется числом единиц, полученных после логического умножения векторов $A_{i(j)} \wedge m$, отнесенных к количеству единиц в векторах $A_{i(j)}, m$. Здесь следует понимать существенное отличие взаимодействия уже не пространств, а точек в виде вектор-маски m и двоичных векторов матрицы $A = [A_{ij}]$. Тем не менее, два упомянутых компонента успешно могут взаимодействовать в векторном алгебрологическом пространстве с помощью операций И, ИЛИ, НЕ. При этом следует учитывать, что нулевые векторы не могут взаимодействовать между собой, поскольку они соответствуют пустому множеству по определению. Следовательно, векторы матрицы $A = [A_{ij}]$ и маска m не могут быть определены только нулевыми значениями по всем координатам. Кодовое расстояние между двумя алгебрологическими векторами определяется числом нулевых значений при их конъюнкции минус количество единичных координат в результирующем векторе.

Здесь и далее матрица $A = [A_{ij}]$ будет записываться двумя способами – по строкам или столбцам:

$$
\begin{aligned}
&A = [A] \vee [B] = [A_i] \vee [B_j],\\
&A = (A_1, A_2, ..., A_i, ..., A_n);\\
&B = (B_1, B_2, ..., B_j, ..., B_m).
\end{aligned}
\tag{10}
$$

Две формы записи служат для сокращения сложности представления формул, задающих процесс обработки матрицы, а также для улучшения понимания аналитически записанных вычислительных процессов. Соответственно входной вектор-маска имеет идентификаторы $m = m^a \vee m^b$ в зависимости от способа анализа матрицы, по строкам или столбцам. Четыре предикатные векторные операции (and, or, not, xor – раньше не упоминалась и не задавалась) позволяют решать многие, интересные для практики, задачи. Некоторые из них представлены ниже.

4. Логический анализ таблицы для поиска покрытия

Нахождение квазиоптимального покрытия единичными значениями вектор-строк всех вектор-столбцов таблицы. Технологическая инновация – параллельное выполнение операций над векторами матрицы или таблицы. Научная новизна – определение подмножества максимально непересекающихся или минимально пересекающихся строк. Стратегия процедуры. Выбор ядра – каждая строка, покрывающая некоторый столбец безальтернативно – одна единица в столбце. Для этого вектор-столбец m^b последовательно маскирует по одной координате для нахождения ядра в виде вектор-строк, которые покрывают безаль-

тернативно некоторый столбец. Определение дополнения к ядру – строки с максимальным числом единиц, имеющие максимально пустое пересечение с уже найденными векторами покрытия. Вербальному описанию модели процесса поиска покрытия соответствует следующая аналитическая форма поиска ядра и дополнения к нему:

$$P = P \underset{i=1}{\overset{n}{\vee}} [A_i \leftarrow (Q_j(m^b \overset{t}{\underset{j=1}{\wedge}} B_j) = 1) \leftarrow (m^b = 0) \vee (m_j^b = 1)];$$
$$P = P \underset{i=1}{\overset{n}{\vee}} [A_i \leftarrow \max[Q_i((m^a = 1) \wedge A_i)) \times (1 - Q_i(P \wedge A_i))]]. \quad (11)$$

где $\times$ – арифметическое умножение критериев качества.

Здесь вектор m играет неоценимую роль маскирования строк, столбцов или отдельных переменных матрицы для создания идеальных условий вычислительного процесса в целях достижения результата. Без него матрица не несет информации. Для решения задачи покрытия необходимо последовательно задать для каждого столбца вектор-маску $(m^b = 0) \vee (m_j^b = 1)$, определенную всеми нулями, кроме одной координаты j, имеющей единичный сигнал; запись $m^a = 1$ означает, что строчный вектор-маска равен 1 по всем координатам. Произведение критериев качества $Q_i((m^a = 1) \wedge A_i)) \times (1 - Q_i(P \wedge A_i))$ предпочтительнее суммы, поскольку оно дает более взвешенную оценку относительно покрывающей способности вектора и кодового расстояния с уже имеющимся решением при выборе очередной строки – кандидата в покрытие. Вычислительная сложность процедуры поиска квазиоптимального покрытия равна $Z = n + \frac{1}{2}n^2$, где n – число строк в матрице, первое слагаемое – определение ядра покрытия, второе – поиск квазиоптимального дополнения к нему. Для примера таблицы, размерностью 5x7, имеющей вид:

$A = [A_i] \vee [B_j] =$

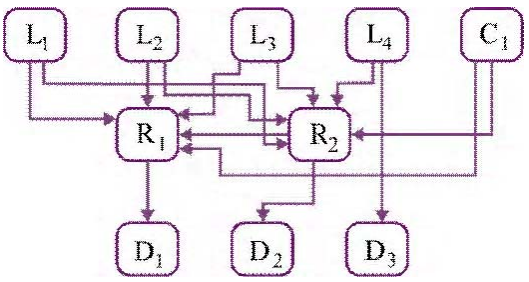
$A_i \setminus B_j$	B_1	B_2	B_3	B_4	B_5	B_6	B_7
A_1	0	0	0	0	1	0	1
A_2	0	0	1	1	0	1	0
A_3	0	1	0	1	1	0	0
A_4	0	1	1	0	0	1	0
A_5	1	0	0	1	0	0	0

(12)

модель поиска квазиоптимального покрытия на основе использования описанной выше процедуры дает минимальное покрытие: $P = \{A_1, A_4, A_5\}$.

5. Логический метод диагностирования дефектов по таблице неисправностей

Рассматривается на примере транзакционного графа одного из программных модулей Row_buffer, используемых при создании IP-core вейвлет-преобразования для стандарта JPEG 2000 (рисунок). Вершины графа представлены входными шинами и переменными, регистрами и выходными шинами. Дуги между ними означают существование транзакций между вершинами при выполнении операторов HDL-кода.



Row_buffer транзакционный граф

Для данного Row_buffer графа построена таблица, задающая поведение неисправных блоков на тесте путем использования системы моделирования и генерации тестов SIGETEST:

$A = \{A_i, B_j\} =$

Test	B_1 (L_1)	B_2 (L_2)	B_3 (L_3)	B_4 (L_4)	B_5 (C_1)	B_6 (R_1)	B_7 (R_2)	B_8 (D_1)	B_9 (D_2)	B_{10} (D_3)	m_1^b	m_2^b
$A_1(t_1)$	1	0	0	0	0	1	0	1	0	1	0	1
$A_2(t_2)$	0	1	0	0	0	0	1	0	1	0	1	1
$A_3(t_3)$	0	0	1	0	0	1	1	1	0	1	0	1
$A_4(t_4)$	0	0	0	1	0	0	1	0	1	0	1	0
$A_5(t_5)$	0	0	0	0	1	1	1	1	0	1	0	0
$A_6(t_6)$	1	0	0	0	0	0	1	0	1	0	1	1
$A_7(t_7)$	0	1	0	0	0	1	0	1	0	0	0	1
$A_8(t_8)$	0	0	1	0	0	1	0	1	0	0	0	1
$A_9(t_9)$	0	0	1	0	0	0	1	0	1	0	1	0
$A_{10}(t_{10})$	0	0	0	1	0	1	1	1	0	0	0	0
$A_{11}(t_{11})$	0	0	0	1	0	0	0	0	0	1	0	0
$A_{12}(t_{12})$	0	0	0	0	1	0	1	0	1	0	1	0
$A_{13}(t_{13})$	0	0	0	0	1	1	1	1	0	0	0	0

5.1. Логический метод анализа столбцов таблицы неисправностей

Основан на применении операции логического умножения или конъюнкции вектора экспериментальной проверки, формально рассматриваемого в качестве входного вектор-столбца или маски m, на столбцы таблицы неисправностей $m^b \wedge (B_1 \vee B_2 \vee ... \vee B_j \vee ... \vee B_m)$ и подсчете качества взаимодействия векторов $Q_j(m^b \wedge B_j)$ в целях выбора лучшего из них при наличии максимальной оценки. При этом столбец $B_j \in A$ фактически идентифицирует метрику поведения неисправности или дефектного блока на тестовых наборах. Предикатная запись процесса получения решения в виде совокупности ошибок, присутствующих в HDL-коде, представлена в следующем виде:

$$P^s = P^s \bigvee_{i=1}^{n} [B_j \leftarrow \max(Q_j(m^b \bigwedge_{j=1}^{t} B_j))];$$

$$P^m = P^m \bigvee_{i=1}^{n} [B_j \leftarrow (m^b \wedge B_j = B_j)] \leftarrow Q[m^b \wedge (\bigvee_{j=1}^{t} B_j \in P^m)] = 1 \vee \max; \quad (13)$$

Здесь вектор экспериментальной проверки

$$m^b = f(T,F) \oplus f^*(T,F,L) \quad (14)$$

есть результат проведения тестового эксперимента – сравнение функционалов (состояний выходов) эталонного $f(T,F)$ и реального $f^*(T,F,L)$ устройства с дефектами L на тестовых наборах T.

Достоинство метода – выбор всегда лучшего решения из всех возможных как для одиночных, так и для кратных дефектов. В последнем случае, если одиночный дефект не идентифицируется оценкой, равной 1, выполняется дизъюнкция таких вектор-строк (главное отличие метода от существующих технологий), которые формируют оценку качества, равную 1 или максимально близкую к единице $Q[m^b \wedge (\bigvee_{j=1}^{t} B_j \in P^t)] = 1 \vee \max$. По существу, в список кратных дефектов включаются такие одиночные неисправности, которые при логическом умножении на вектор экспериментальной проверки дают результат в виде соответствующего вектор-столбца. Дизъюнкция всех столбцов, составляющих решение, равна $\bigvee_{j=1}^{t} (B_j \in P^t) = m^b$.

С использованием таблицы и процедуры диагностирования (13) определяются дефектные компоненты программного кода модуля Row_buffer методом логического умножения вектор-столбцов таблицы истинности на вектор экспериментальной проверки. Здесь векторы m_1^b, m_2^b формируют результаты диагностического эксперимента, выполненные по технологии (14). Результат диагностирования одиночных и кратных дефектов имеет следующий вид:

$$\begin{aligned}
&P^s(m_1^b) = m_1^b \wedge (L_1 \vee L_2 \vee L_3 \vee L_4 \vee C_1 \vee R_1 \vee R_2 \vee D_1 \vee D_2 \vee D_3) = D_2;\\
&P^m(m_2^b) = m_2^b \wedge (L_1 \vee L_2 \vee L_3 \vee L_4 \vee C_1 \vee R_1 \vee R_2 \vee D_1 \vee D_2) = L_1 \vee L_2;\\
&Q(m_1^b, D_2) = 1;\\
&Q[m_2^b, (L_1 \vee L_2)] = \frac{1}{3}(\frac{4}{13} + \frac{1}{4} + 1) = 0{,}52.
\end{aligned} \qquad (15)$$

В первом случае диагноз определен в виде одного дефектного блока D_2, присутствующего в транзакционном графе, качество решения равно 1. Во втором случае процедура диагностирования выявила наличие двух дефектных модулей $L_1 \vee L_2$, которые не смогли сформировать идеальную оценку качества. Тем не менее, решение является лучшим среди всех возможных, которое максимально приближено к вектору экспериментальной проверки по критерию принадлежности $Q[m_2^b, (L_1 \vee L_2)]$.

Вычислительная сложность метода анализа столбцов определяется следующей зависимостью: $Z^c = 3n^2 + n^2 = 4n^2$; $Z^r = 3n + n = 4n$. Здесь первая оценка учитывает выполнение координатных операций над матрицей размерностью $n \times n$. Вторая оценка определяет вычислительную сложность регистровых параллельных операций для подсчета критериев качества и обработки матрицы соответственно.

5.2. Логический метод анализа строк таблицы неисправностей

Стратегия определения ошибок программного кода по таблице неисправностей связана с анализом ее строк, состоящим из двух процедур: 1) вычисление логического произведения конъюнкции строк, отмеченных единичными значениями вектора $A_i(m_i^b = 1)$, на отрицание дизъюнкции нулевых строк $A_i(m_i^b = 0)$ для одиночных дефектных блоков; 2) вычисление логического произведения дизъюнкции единичных строк на отрицание дизъюнкции нулевых строк для кратных дефектных блоков:

$$\begin{aligned}
&P^s = (\bigwedge_{\forall m_i^b = 1} A_i) \wedge (\overline{\bigvee_{\forall m_i^b = 0} A_i});\\
&P^m = (\bigvee_{\forall m_i^b = 1} A_i) \wedge (\overline{\bigvee_{\forall m_i^b = 0} A_i});
\end{aligned} \qquad (16)$$

Формулы интересны тем, что они не привязаны к критериям качества диагностирования, а оперируют лишь двумя компонентами, таблицей неисправностей и вектором экспериментальной проверки. Выполнение процедуры диагностирования по формулам (16) для вектора экспериментальной проверки $m^b = (0101010010010)$, заданного в последней таблице неисправностей, дает результат: $P^s(m_1^b, A) = D_2$, который не хуже, чем полученный ранее методом анализа столбцов. Для вектора экспериментальной проверки $m^b = (1110011100000)$, результат диагностирования имеет вид: $P^m(m_2^b, A) = L_1 \vee L_2$. Вычислительная сложность метода анализа строк определяется следующей зависимостью: $Z^c = n^2$; $Z^r = n$. Первая оценка предназначена для подсчета числа координатных операций, вторая – определяет вычислительную сложность процесса обработки на основе регистровых параллельных операций.

6. Векторно-логический критерий качества решения

Цель – использовать только логические операции и исключить арифметические вычисления из процедуры формирования критерия для существенного повышения быстродействия логического анализа информационных массивов.

Идея – качество решения логического анализа оценивается мощностью единиц в векторе качества и определяется путем взаимодействия входного вектора с таблицами данных на основе использования векторных логических операций.

Вектор качества формирует критерий принадлежности, определяемый следующими выражениями:

$$
\begin{aligned}
&Q = d[m, A_{i(j)}] + \mu[m \in A_{i(j)}] + \mu[A_{i(j)} \in m],\\
&d(m, A_{i(j)}) = \mathrm{card}[m \overset{n(i)}{\underset{i(j)=1}{\oplus}} A_{i(j)} = 1];\\
&\mu(m \in A_{i(j)}) = \mathrm{card}[A_{i(j)} = 1] - \mathrm{card}[m \overset{n(i)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1];\\
&\mu(A_{i(j)} \in m) = \mathrm{card}[m = 1] - \mathrm{card}[m \overset{n(i)}{\underset{i(j)=1}{\wedge}} A_{i(j)} = 1].
\end{aligned}
\qquad (17)
$$

Первый компонент, составляющий критерий, формирует степень несовпадения n-мерных векторов – кодовое расстояние, путем выполнения операции xor, второй и третий – определяют степень непринадлежности результата конъюнкции к числу единиц каждого из двух взаимодействующих векторов. Понятия принадлежности и непринадлежности являются взаимодополняющими, но в данном случае технологичнее высчитывать именно непринадлежность. Таким образом, идеальный критерий качества равен нулю, когда два вектора равны между собой. Оценка качества взаимодействия двух двоичных векторов убывает по мере роста критерия от 0 к 1.

Чтобы окончательно уйти от арифметических операций при подсчете критерия качества, необходимо выражения (17) преобразовать к следующему виду:

$$
\begin{aligned}
&Q = d(m, A) \vee \mu(m \in A) \vee \mu(A \in m),\\
&d(m, A) = m \oplus A;\\
&\mu(m \in A) = A \wedge \overline{m \wedge A};\\
&\mu(A \in m) = m \wedge \overline{m \wedge A}.
\end{aligned}
\qquad (18)
$$

Здесь критерии представлены уже не числами, а векторами, которые оценивают взаимодействие между компонентами m, A. При этом число нулей в трех оценках есть хорошо, а единицы ухудшают качество взаимодействия. Оптимизация решения логической задачи направлена на минимизацию числа единиц и максимизацию количества нулевых координат в векторах критерия качества. Для сравнения двух оценок необходимо определять мощность единиц в каждом векторе без выполнения операций суммирования, это можно сделать, например, с помощью регистра [4] уплотнения единиц и их сдвига влево. После процедуры сжатия номер последнего или правого единичного бита уплотненной серии единиц формирует индекс качества взаимодействия векторов.

Для двух векторов $m = (110011001100)$; $A = (000011110101)$ определение качества их взаимодействия в соответствии с моделью (18) представлено в следующем виде (нулевые координаты отмечены точками):

m	1	1	.	.	1	1	.	.	1	1	.	.
A	.	.	.	.	1	1	1	1	.	1	.	1
$m \wedge A$	.	.	.	.	1	1	.	.	.	1	.	.
$\overline{m \wedge A}$	1	1	1	1	.	.	1	1	1	.	1	1
$d(m, A) = m \oplus A$	1	1	.	.	.	.	1	1	1	.	.	1
$\mu(A \in m) = m \wedge \overline{m \wedge A}$	1	1	.	.	.	.	.	.	1	.	.	.
$\mu(m \in A) = A \wedge \overline{m \wedge A}$	.	.	.	.	.	.	1	1	.	.	.	1
$Q = d(m, A) \vee \mu(m \in A) \vee \mu(A \in m)$	1	1	.	.	.	.	1	1	1	.	.	1
$Q(m, A) = (6,12)$	1	1	1	1	1	1	.	.	.	.	.	.

Здесь сформирована не только оценка качества взаимодействия векторов, равная $Q(m, A) = (6,12)$, но, что самое главное, единичные координаты строки $Q = d(m, A) \vee \mu(m \in A) \vee \mu(A \in m)$ идентифицируют все те места или позиции, по которым существует некачественное взаимодействие векторов.

Другой пример иллюстрирует формирование максимального критерия качества для двоичных векторов $m = (110000110011);\ A = (110000110011)$, имеющих совпадение по всем координатам:

m	1	1	.	.	.	.	1	1	.	.	1	1
A	1	1	.	.	.	.	1	1	.	.	1	1
$m \wedge A$	1	1	.	.	.	.	1	1	.	.	1	1
$\overline{m \wedge A}$	.	.	1	1	1	1	.	.	1	1	.	.
$d(m, A) = m \oplus A$	.	.	.	.	.	.	.	.	.	.	.	.
$\mu(A \in m) = m \wedge (\overline{m \wedge A})$	.	.	.	.	.	.	.	.	.	.	.	.
$\mu(m \in A) = A \wedge (\overline{m \wedge A})$	.	.	.	.	.	.	.	.	.	.	.	.
$Q = d(m, A) \vee \mu(m \in A) \vee \mu(A \in m)$	.	.	.	.	.	.	.	.	.	.	.	.
$Q = (0,12)$	.	.	.	.	.	.	.	.	.	.	.	.

Здесь критерий качества, равный нулю во всех 12 разрядах $Q = (0,12)$, является максимальным или самым лучшим для взаимодействующих векторов $m = (110000110011);\ A = (110000110011)$, поскольку он определен минимальным числом единиц на двенадцати координатах вектора. Для сравнения двух решений, полученных в результате логического анализа, следует использовать сжатые векторы качества Q, над которыми необходимо выполнить процедуру, включающую следующие векторные операции:

$$Q(m, A) = \begin{cases} Q_1(m, A) \leftarrow Q_1(m, A) \oplus Q_1(m, A) \wedge Q_2(m, A) = 0; \\ Q_2(m, A) \leftarrow Q_1(m, A) \oplus Q_1(m, A) \wedge Q_2(m, A) \neq 0. \end{cases} \quad (19)$$

Для двоичных векторов, представляющих собой критерии качества, выполнена процедура выбора лучшего их них на основе выражения, представленного в (19):

$Q_1(m, A) = (6,12)$	1	1	1	1	1	1	.	.	.	.	.	.
$Q_2(m, A) = (8,12)$	1	1	1	1	1	1	1	1	.	.	.	.
$Q_1(m,A) \wedge Q_2(m,A)$	1	1	1	1	1	1	.	.	.	.	.	.
$Q_1(m, A) \oplus Q_1(m, A) \wedge Q_2(m, A)$	.	.	.	.	.	.	.	.	.	.	.	.
$Q(m, A) = Q_1(m, A)$	1	1	1	1	1	1	.	.	.	.	.	.

Предложенная алгебра и разработанные на ее основе аналитические модели и методы анализа таблиц на основе введенных критериев качества позволяют решать задачи поиска квазиоптимального покрытия, диагностирования одиночных и кратных дефектов программных и/или аппаратных блоков. Алгебра анализа матриц также может быть использована для решения следующих задач технической диагностики: 1. Моделирование исправного поведения цифрового устройства, заданного таблицей истинности. 2. Определение качества теста по таблице неисправностей. 3. Минимизация теста. 4. Поиск тестов, распознающих заданный дефект, на основе анализа таблицы неисправностей. 5. Идентификация кратного дефекта. 6. Поиск и исправление ошибок при наборе текста.

7. Выводы

1. Научная новизна представлена алгеброй логического анализа векторных и табличных форм задания информации для решения задач поиска, диагностирования, распознавания образов и принятия решений в векторном дискретном булевом пространстве. Представлены быстродействующие модели и методы параллельного векторного логического анализа информации, в пределе полностью исключающие использование арифметических операций, в том числе и для подсчета критерия качества решения, где применяются только логические операции. В качестве примеров описаны новые методы для решения задач диагностирования и нахождения квазиоптимального покрытия, использующие векторные операции, ориентированные на распараллеливание вычислительных процессов.

2. Практическая значимость заключается в ориентации алгебры ассоциативных таблиц и методов их анализа на создание логического мультипроцессора с ограниченной системой команд, ориентированной на высокое быстродействие параллельной обработки больших массивов информации, представленных графовыми структурами ассоциативных матриц.

3. Дальнейшие исследования будут направлены на разработку прототипа мультипроцессора и решение новых практических задач с помощью предложенной алгебры.

Список литературы: 1. *Zorian Yervant.* Test Strategies for System-in-Package // The Plenary Paper of IEEE East-West Design & Test Symposium (EWDTS'08). Lvov, Ukraine. October 9-12, 2008. **2.** *Smith L.* 3D Packaging Applications, Requirements, Infrastructure and Technologies // Fourth Annual International Wafer-Level Packaging Conference. San Jose, California. September, 2007. **3.** *The next Step* in Assembly and Packaging: System Level Integration in the package (SiP) / Editors: William Chen, W. R. Bottoms, Klaus Pressel, Juergen Wolf // SiP White Paper. International Technology Roadmap for Semiconductors. 2007. P. 17-23. **4.** *Какурин Н.Я., Хаханов В.И., Лобода В.Г., Какурина А.Н.* / Регистр сдвига. А.С. №1439682. 22.07.88. 4с. **5.** *Бондаренко М.Ф., Дударь З.В., Ефимова И.А., Лещинский В.А., Шабанов–Кушнаренко С.Ю.* О мозгоподобных ЭВМ // Радиоэлектроника и информатика. 2004. № 2. С. 89–105. **6.** *Бондаренко М.Ф., Шабанов–Кушнаренко Ю.П.* Об алгебре предикатов // Бионика интеллекта. 2004. № 1. С. 15–26. **7.** *Бондаренко М.Ф., Шабанов–Кушнаренко Ю.П.* Теория интеллекта. Учебник. Харьков: СМИТ, 2006. 592 с. **8.** *Бондаренко М.Ф., Шабанов–Кушнаренко Ю.П.* Модели языка // Бионика интеллекта. 2004. № 1. С. 27–37. **9.** *Акритас А.* Основы компьютерной алгебры с приложениями: Пер. с англ. М.: Мир, 1994. 544 с. **10.** *Гилл Ф. Мюррей У., Райт М.* Практическая оптимизация. М.: Мир. 1985. 509с. **11**. *Аттетков А.В., Галкин С.В., Зарубин В.С.* Методы оптимизации. М.: Изд-во МГТУ им. Н.Э.Баумана, 2003. 440 с. **12.** *Дегтярев Ю. И.* Методы оптимизации: Учебное пособие для вузов. М.: Сов. радио, 1980. 270с. **13.** *Bergeron J.* Writing Testbenches Using SystemVerilog / J. Bergeron // Springer Science and Business Media, Inc., 2006. 414 p. **14.** *Abramovici M., Breuer M.A. and Friedman A.D.* Digital System Testing and Testable Design. Comp. Sc. Press. 1998. 652 p**. 15.** *Densmore* Douglas A Platform–Based taxonomy for ESL Design / Douglas Densmore, Roberto Passerone, Alberto Sangiovanni–Vincentelli // Design&Test of computers. 2006. P. 359–373. **16.** *Хаханов В.И., Литвинова Е.И., Гузь О.А.* Проектирование и тестирование цифровых систем на кристаллах. Харьков: ХНУРЭ, 2009. 484с.

Поступила в редколлегию 15.04.2010

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Чумаченко Светлана Викторовна, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: математическое моделирование и вычислительные методы, методы дискретной оптимизации. Увлечения: путешествия, музыка, поэзия, спорт. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: ri@kture.kharkov.ua.

Литвинова Евгения Ивановна, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: kiu@kture.kharkov.ua.

УДК 681.326:519.713

В.И. ХАХАНОВ, Е.И. ЛИТВИНОВА, А. ПРИЙМАК

СТРУКТУРЫ ТАБЛИЦ ДАННЫХ ДЛЯ CYBER-SPACE

Предлагается концепция саморазвития информационно-компьютерной экосистемы, повторяющая эволюцию функциональностей человека. Предлагаются оригинальные процесс-модели ассоциативно-логического анализа информации на основе быстродействующего мультипроцессора в n-мерном векторном дискретном пространстве. Рассматривается применение алгебры векторной логики для создания процесс-моделей актуальных прикладных задач, качество решения которых оценивается интегральной неарифметической метрикой взаимодействия ассоциативных структур.

Введение

Проблема создания инфраструктуры кибернетического пространства (Cyber Space) или саморазвивающейся информационно-компьютерной экосистемы (ИКЭС) планеты в последние 40 лет является весьма интересной темой не только для ученых-фантастов. Экосистема – совокупность популяций, которые взаимодействуют между собой и окружающей их средой неопределённо долгое время, имеющая сходство относительно протекающих в них энергетических процессов. Сегодня чрезвычайно важно обозначить возможные пути для решения проблемы создания инфраструктуры саморазвивающейся компьютерной экосистемы. Эволюция ИКЭС основывается на использовании трех наиболее важных компонентов: фантазия, математика и технология, где субъектом экосистемы выступает саморазвивающийся компьютер (СРК). Основное отличие СРК от современного компьютера заключается в концепции жизненного цикла. Стратегия настоящего компьютера есть обучение или повторение уже пройденного пути. Принципиальная позиция СРК – постоянный поиск новых путей для эволюционирования на основе мирового опыта, скрытого в информационном пространстве. ИКЭС имеет возможность повторить эволюцию человечества, только в тысячи раз более быстрыми темпами. На рис.1 представлен замкнутый цикл эволюции ИКЭС, который фактически изоморфен спирали развития человечества, накрученной на временную ось.

Здесь заложены основные принципы эволюции, явно выраженные уже в современной компьютерной индустрии: 1) Стандартизация – самое главное для эволюции и жизненного цикла СРК – рынок не принимает и не понимает нестандартных по интерфейсу решений. 2) Специализация есть повышение эффективности предоставляемых (персонально ориентированных) сервисов изделия, связанных с быстродействием, качеством, затратами, энергосбережением путем оптимизации структуры и функциональных компонентов, покрывающих спецификацию. 3) Повсеместное использование векторно-логического критерия качества решения в задачах генерирования идей, синтеза и анализа. Генерирование – процесс создания новой функциональности. При этом синтез оперирует существующими в информационном пространстве компонентами для создания структуры. Анализ – оценивание полученного решения. 4) Диаграмма Хассе используется для выработки стратегии оптимизации покрытия функциональностей спецификации библиотечными компонентами или их сочетаниями, принадлежащими информационному пространству. Она согласуется с современной Y-технологией, входящей в состав ESL Design, суть которой – использовать библиотечные компоненты на всех уровнях проектирования изделия для покрытия специфицируемой функциональности в процессе синтеза.

В части развития компьютера мы горим желанием сделать его мыслящим, наделить органами чувств – сотворить его подобным, в конце концов, человеку. Не лучше ли будет взять на вооружение другой тезис – компьютер, как реальная часть природы, имеет право занимать в ней собственную нишу для самостоятельного эволюционирования и самосовершенствования. Не следует сталкивать лбами человека и компьютер, которые и в будущем будут жить в мире и согласии, как субъекты природы, использующие друг друга для решения общественно полезных задач. Не следует также пытаться копировать живой мозг человека, по сути, в неживом кремниевом кристалле. Копия всегда будет хуже оригинала!

Но инкапсулировать функциональности мозга в кристалл – значит расширить мощность сервисов, полезных для решения практических задач в информационном пространстве. Идея самостоятельного развития компьютера по собственному пути немного подрезает крылья у фантазии – сделать его неотличимым от человека. Но при этом она окрыляет практических исследователей и IT-индустрию. Становится возможным уже в ближайшее десятилетие сотворить саморазвивающуюся информационно-компьютерную экосистему! С другой стороны, от взаимодействия Software и Hardware, освященного оригинальной идеей, действительно появляются полезные функциональности: специализированные изделия, мобильные телефоны, RFID, планшеты, серверы, порталы, формирующие в совокупности информационное пространство планеты. Рынок, как и природа, осуществляет селекцию или естественный отбор практически ориентированных разработок, среди которых нет места слабым субъектам или проектам, а выживают сильнейшие представители в своем экологическом слое.

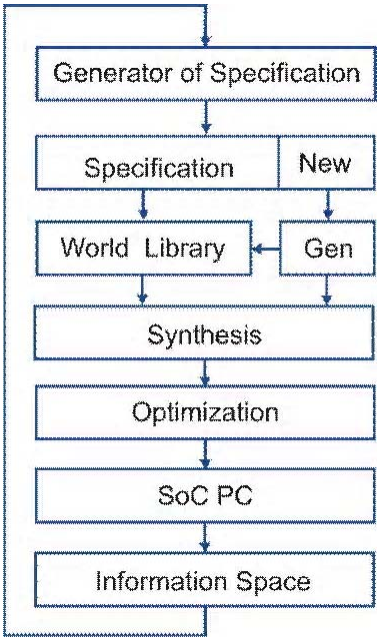


Рис. 1. Цикл ИКЭС

Относительно интеллекта. Википедия дает краткое и достаточно полное определение: «Интеллект – способность системы создавать в ходе самообучения программы для решения задач». Это относится к естественному и искусственному интеллекту. Человек – только повторитель? Где же место в интеллекте для создателей новой культуры, математической и технологической, машин, строений? Маленькая корректировка может существенно поправить понятие интеллекта, если в нем появятся два вида деятельности: повторение и созидание. Интеллект – способность субъекта природы к самообучению (познанию) и самосовершенствованию (созиданию) для формулирования и решения проблем, связанных с повышением качества жизненного цикла.

Как человек берет лучшие функциональности у субъектов природы для расширения своих возможностей, так и компьютер имеет целью самосовершенствовать себя, заимствуя или подсматривая у природы интересные решения: кремний – для выполнения элементарных логических функций; математику – для формализации и решения проблем; естественные языки – для создания спецификации и интерфейса проекта; функциональность человеческого мозга, формально выделенную в векторно-логический анализ – для организации структур данных, поиска, распознавания и принятия решений.

Цель исследования – ответить на вопрос, каким будет кибернетическое пространство, компьютер и сервисы через 10 лет, или представить новые перспективные направления развития компьютера и Internet как единой информационно-компьютерной экосистемы,

ориентированной на повышение качества жизни путем ее непрерывного использования для поиска, распознавания и принятия решений. Сколько людей – столько и компьютеров, *индивидуально* настроенных на каждого конкретного человека. Рыночная привлекательность – все население и все компьютеры планеты. Опасности – выход из-под контроля человека саморазвивающейся информационно-компьютерной экосистемы планеты.

Задачи: 1) Убрать из компьютера тяжеловесную арифметику для повышения быстродействия мозгоподобных (ассоциативно-логических) задач на несколько порядков. 2) Создать стандартизованные структуры данных – иерархия таблиц – для ИКЭС. 3) Разработать быстродействующий параллельный логический ассоциативный мозгоподобный мультипроцессор без использования арифметических операций. 4) Создать простые, эффективные метрики и критерии оценивания получаемых решений в векторно-логическом информационном пространстве. 5) Разработать процесс-модели, реализующие рыночно-привлекательные функциональности, ориентированные на поиск, распознавание и принятие решений в ассоциативном векторно-логическом пространстве. 6) Создать инфраструктуру, ориентированную на автоматическое генерирование процесс-моделей поиска, распознавания и принятия решений в кибернетическом пространстве планеты.

Источники: 1. Ассоциативно-логические структуры данных для мозгоподобных вычислительных процессов [1-5]. 2. Мультипроцессорная аппаратная платформа информационно-логического анализа [6-9]. 3. Модели и методы дискретного анализа, синтеза и диагностирования цифровых изделий [10-15].

Инфраструктура формирования условий и принятия решения

Мозгоподобный компьютер – новый специализированный движок для Internet пространства. Инфраструктура глобальной сети Internet по оценке Gary Smith (Gartner Research Group) нуждается в радикальной перестройке. Глобальное информационное пространство уже не вмещается в демократически развивающиеся и потому несовершенные форматы и структуры данных. Еще хуже обстоит дело с разработкой движков, специализированных под задачи быстрого передвижения по пространству для поиска, распознавания информации и принятия умных решений. Роль таких устройств сегодня выполняют универсальные процессоры, функциональность которых на 90% не пригодна для решения упомянутых задач. Что можно сделать для решения указанных выше проблем? Далее представлены семь параметров повышения производительности мозгоподобного компьютера, которые оказывают существенное влияние на инфраструктуру (рис. 2) принятия решения.

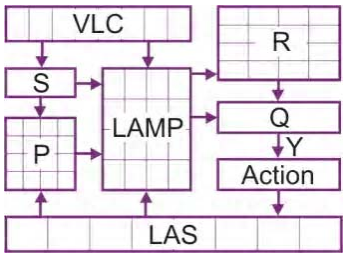


Рис. 2. Инфраструктура логического принятия решения

1) Исключение арифметических операций из системы команд процессора. 2) Использование в вычислительных процедурах алгебры векторной логики. Компьютер выигрывает у человека в способности быстро анализировать большое число существенных логических переменных, сконкатенированных в вектор-запрос. 3) Применение мультипроцессорной матрицы для распараллеливания вычислительных процедур. 4) Использование векторного двоичного критерия качества для оценивания решения. 5) Введение операции векторизации в целях генерирования ассоциации минимального S-вектора существенных переменных, необходимых для поиска решения. 6) Создание вектор-бит D-оператора девекторизации для формирования двоичного решения на основе применения логических операций and, or, not или их комбинации к вектору (векторам) существенных переменных. 7) Создание P-платформы логических процесс-моделей (IP-cores) для поиска, распознавания и принятия решений.

Для решения конкретной задачи, специфицированной S-вектором, необходимо найти (квази-) оптимальное покрытие спецификации функциональностями, формирующими P-платформу в виде совокупности IP-cores, сгенерированных ранее. При отсутствии конкретной процесс-модели в библиотеке решений P ее следует синтезировать на основе анализа логического ассоциативного пространства L, обозначенного на рис. 2 как LAS (Logical Associative Space). Фактически такое пространство представляет собой Internet, базы данных и знаний, средства (программные и аппаратные) синтеза и анализа информации, которая представлена в виде логических ассоциаций. Решение задачи покрытия функций спецификации примитивными процесс-моделями платформы основывается на использовании Y-технологии [11], известной также под названием ESL – Electronic System Level design. Все локальные и глобальные решения оцениваются системой векторно-логических критериев (VLC – Vector Logical Criteria). Действительно, каждой процесс-модели из P-платформы ставится в соответствие критерий качества из банка VLC.

Интегральная аналитическая процесс-модель анализа информации, соответствующая структурам данных, представленным на рис. 2, имеет вид:

$$\begin{cases} R = [\max(S \underset{j=1}{\overset{m}{\wedge}} P_j)] \underset{i=1}{\overset{n}{\Delta}} L_i \leftarrow \max Q_i; \\ Q = \max Q_i \{[\max(S \underset{j=1}{\overset{m}{\wedge}} P_j)] \underset{i=1}{\overset{n}{\Delta}} L_i\}; \\ Y = D[\max Q_i \{[\max(S \underset{j=1}{\overset{m}{\wedge}} P_j)] \underset{i=1}{\overset{n}{\Delta}} L_i\}]. \end{cases}$$

Здесь $P_j \in P$, $\Delta = \{\text{and, or, not, xor, s1c}\}$ – векторные логические операции, где s1c – shift left bit crowding; $Y = \{0,1\}$ – двоичное решение, направленное на выполнение конкретной активности (action); Q – критерий качества решения, определяемый двоичным вектором; L_i – вариант содержательного решения, представленного в виде информационного сообщения; m – количество процесс-моделей в платформе P; n – число вариантов ассоциаций, среди которых осуществляется поиск решения; R – (квази-) оптимальное содержательное решение, представленное в виде совокупности векторов или других информационных сообщений в формате синтаксиса естественных или искусственных языков.

Далее предлагается алгебраическая структура, регулирующая векторно-матричные преобразования в дискретном булевом пространстве для анализа информации на основе логических операций над ассоциативными данными.

Источники: 1. Технологии параллельных вычислений на специализированных мультипроцессорных системах [1-2, 12, 14, 15]. 2. Алгебраические структуры, ориентированные на создание математического аппарата для параллельных вычислительных процессов [3-4, 7-10]. 3. Процесс-модели решения практических задач на основе эффективных параллельных вычислительных процессов [5, 6, 11, 13].

Анализ табличных структур данных

Рассматривается таблица A, размерностью $m \times n$, взаимодействующая с двумя векторами m_a, m_b, которые могут быть входными и выходными, в зависимости от процедуры анализа (рис. 3):

$$A = (A_1, A_2, ..., A_i, ..., A_n);\ A = (A_1, A_2, ..., A_j, ..., A_t);$$

$$m_a = (m_1, m_2, ..., m_r, ..., m_n);\ m_b = (m_1, m_2, ..., m_s, ..., m_m).$$

Первичный анализ: запрос – таблица – ответ:

$$m_a \rightarrow A \rightarrow m_b \vee m_b \rightarrow A \rightarrow m_a$$

связан с выполнением процесс-моделей $f_1 \vee f_2$, где результатом является множество решений, отмеченных нулевыми координатами выходного двоичного вектора, если соответствующий столбец (строка) таблицы удовлетворяет входному запросу, и единичными – в противном случае:

$$m_b = f_1(m_a, A) \to m_{bj} = m_a \overset{t}{\underset{j=1}{\oplus}} A_j;$$

$$m_a = f_2(m_b, A) \to m_{ai} = m_a \overset{n}{\underset{i=1}{\oplus}} A_i.$$

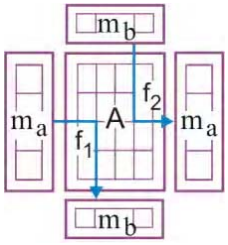


Рис. 3. Первичный анализ таблицы

Здесь операция xor может быть заменена симметрической разностью для анализа многозначных таблиц в замкнутом теоретико-множественном алфавите:

$$m_b = f_1(m_a, A) \to m_{bj} = m_a \overset{t}{\underset{j=1}{\Delta}} A_j;$$

$$m_a = f_2(m_b, A) \to m_{ai} = m_a \overset{n}{\underset{i=1}{\Delta}} A_i.$$

При этом качество взаимодействия векторов будет иметь многозначную градацию, которая уменьшается по мере возрастания полноты символа замкнутого алфавита. Например, для четырехзначного алфавита Кантора – $< \varnothing \to \{0,1\} \to x >$ или 16-значного алфавита описания автоматных переменных:

$$< \varnothing \to \{Q, E, H, J\} \to \{O, I, A, B, S, P\} \to \{C, F, L, V\} \to Y > .$$

Вторичный анализ таблицы связан со сверткой столбцов или строк таблицы для получения интегрированного решения в форме одного вектора, что означает повышение глубины поиска информации за счет исключения из обобщенного вектора противоречивых (невозможных) состояний координат путем выполнения процедур:

$$m_a = f_1^s(m_b, A) \to \underset{\forall j(m_{bj}=0)}{\wedge} A_j \wedge \overline{\underset{\forall j(m_{bj}=1)}{\vee} A_j};$$

$$m_a = f_1^m(m_b, A) \to \underset{\forall j(m_{bj}=0)}{\vee} A_j \wedge \overline{\underset{\forall j(m_{bj}=1)}{\vee} A_j};$$

$$m_b = f_2^s(m_a, A) \to \underset{\forall i(m_{ai}=0)}{\wedge} A_i \wedge \overline{\underset{\forall j(m_{ai}=1)}{\vee} A_i};$$

$$m_b = f_2^m(m_a, A) \to \underset{\forall i(m_{ai}=0)}{\vee} A_i \wedge \overline{\underset{\forall i(m_{ai}=1)}{\vee} A_i}.$$

Изначально векторы в таблице формируют общее пространство, отмеченное нулевыми и единичными значениями вектор-запроса. Последние процедуры позволяют разделить единичное и нулевое пространства друг от друга и сформировать множество только существенных координат в одном нулевом или единичном векторе. Отличие функционалов f^s, f^m заключается в поиске единичных координатных решений путем пересечения всех строк в первом случае, или в определении множественного решения на основе объединения всех строк таблицы – во втором случае. Аналогично можно отделить от нулевого пространства единичное и сформировать единичные соответствующие векторы существенных координат:

$$m_a = f_1^s(m_b, A) \rightarrow \bigwedge_{\forall j(m_{bj}=1)} A_j \wedge \overline{\bigvee_{\forall j(m_{bj}=0)} A_j};$$

$$m_a = f_1^m(m_b, A) \rightarrow \bigvee_{\forall j(m_{bj}=1)} A_j \wedge \overline{\bigvee_{\forall j(m_{bj}=0)} A_j};$$

$$m_b = f_2^s(m_a, A) \rightarrow \bigwedge_{\forall i(m_{ai}=1)} A_i \wedge \overline{\bigvee_{\forall j(m_{ai}=0)} A_i};$$

$$m_b = f_2^m(m_a, A) \rightarrow \bigvee_{\forall i(m_{ai}=1)} A_i \wedge \overline{\bigvee_{\forall i(m_{ai}=0)} A_i}.$$

Для многозначной таблицы А вторичный анализ также выполняется путем использования последних восьми логических выражений, которые уже оперируют многозначной логикой. Для четырехзначного алфавита Кантора примитивы and, or, not определены следующими таблицами:

$\wedge$	0	1	x	∅
0	0	∅	0	∅
1	∅	1	1	∅
x	0	1	x	∅
∅	∅	∅	∅	∅

$\vee$	0	1	x	∅
0	0	x	x	0
1	x	1	x	1
x	x	x	x	x
∅	0	1	x	∅

a	0	1	x	∅
$\bar{a}$	1	0	∅	x

Заключение

Актуальные задачи в направлении создания саморазвивающейся информационно-компьютерной экосистемы: 1. Синтез минимального количества элементарных функций, способных покрыть все коды изображения для минимизации информационного объема, передаваемого по каналу. 2. Уменьшение объемов информации и минимизации структур путем использования принципа дополнения в избыточности замкнутых алфавитов. 3. Формирование решений в причинно-следственном покрытии на основе максимизации критерия качества взаимодействия вектор-спецификации и библиотечных примитивов, которые должны быть упорядочены по разделам или кодам и представлены в форме двоичного дерева. 4. Разработка формальных методов синтеза функциональностей по таблице истинности (покрытия), которая в совокупности составляет спецификацию проекта. 5. Создание новых методов сжатия данных, основанных на использовании метрики векторно-логического пространства для передачи информации по телекоммуникационным системам. 6. Разработка новых методов криптоанализа на основе использования метрики векторно-логического пространства и принципа свертки кодов расстояний в нулевую сумму.

Список литературы: 1. *Бондаренко М.Ф.* О мозгоподобных ЭВМ / М.Ф. Бондаренко, З.В. Дударь, И.А. Ефимова, В.А. Лещинский, С.Ю. Шабанов–Кушнаренко // Радиоэлектроника и информатика. 2004. № 2. С. 89–105. **2.** *Cohen A.A.* Addressing architecture for Brain-like Massively Parallel Computers / Euromicro Symposium on Digital System Design (DSD'04). 2004. P. 594-597. **3.** *Кузнецов О.П.* Быстрые процессы мозга и обработка образов // Новости искусственного интеллекта. 1998. №2. **4.** *Васильев С.Н., Жерлов А.К., Федосов Е.А., Федунов Б.Е.* Интеллектуальное управление динамическими системами. М.: Физико-математическая литература. 2000. 352 с. **5.** *Липаев В.В.* Программная инженерия. Методологические основы. Учебник. М.: Теис. 2006. 608 с. **6.** А.С. №1439682. 22.07.88. Регистр сдвига / Какурин Н.Я., Хаханов В.И., Лобода В.Г., Какурина А.Н. 4с. **7.** *Гайдук С.М., Хаханов В.И., Обризан В.И., Каменюка Е.А.* Сферический мультипроцессор PRUS для решения булевых уравнений // Радиоэлектроника и информатика. 2004. № 4(29). С.107-116. **8.** *Проектирование* и тестирование цифровых систем на кристаллах / В.И. Хаханов, Е.И. Литвинова, О.А. Гузь. Харьков: ХНУРЭ, 2009. 484с. **9.** *Проектирование* и верификация цифровых систем на кристаллах. Verilog & System Verilog / В.И. Хаханов, И.В. Хаханова, Е.И. Литвинова, О.А. Гузь. Харьков: Новое слово. 2010. 528с. **10.** *Акритас А.* Основы компьютерной алгебры с приложениями: Пер. с англ. / А. Акритас. М.: Мир, 1994. 544 с. **11.** *Аттетков А.В.* Методы оптимизации / А.В. Аттетков, С.В. Галкин, В.С. Зарубин. М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. 440 с. **12.** *Abramovici M.* Digital System Testing and Testable Design / M. Abramovici, M.A. Breuer and A.D. Friedman. Comp. Sc. Press. 1998. 652 p. **13.** Densmore D. A Platform-Based taxonomy for ESL Design / Douglas Densmore, Roberto Passerone, Alberto Sangiovanni-Vincentelli // Design & Test of computers. 2006. P. 359–373. **14.** *Автоматизация* диагностирования электронных устройств/ Ю.В.Малышенко и

др./ Под ред. В.П.Чипулиса. М.: Энергоатомиздат, 1986. 216с. **15.** *Трахтенгерц Э.А.* Компьютерные методы реализации экономических и информационных управленческих решений. СИНТЕГ. 2009. 396с.

Поступила в редколлегию 04.03.2010

Хаханов Владимир Иванович, декан факультета КИУ ХНУРЭ, д-р техн. наук, профессор кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Увлечения: баскетбол, футбол, горные лыжи. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanov@kture.kharkov.ua.

Литвинова Евгения Ивановна, канд. техн. наук, доцент кафедры АПВТ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: kiu@kture.kharkov.ua.

Приймак Алексей, студент факультета КИУ ХНУРЭ. Научные интересы: техническая диагностика цифровых систем, сетей и программных продуктов. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326.

УДК 681.326:519.613

С.А. ЗАЙЧЕНКО, С.В. ЧУМАЧЕНКО

ОПТИМИЗАЦИИ ВЫЧИСЛИТЕЛЬНОГО ЦИКЛА АНАЛИЗА АССЕРЦИЙ

Предлагается обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели. Разрабатывается метод компрессии событий на выходе бесконечных репетиций.

Введение. Исследование связано с созданием новых моделей и методов верификации, а также с эффективным устранением ошибок: 1) допускаемых инженерами в системной модели, тестах и спецификации в процессе проектирования; 2) обусловленных несовершенством средств диагностирования в системах автоматизации, затрудняющих локализацию и устранение причины возникновения ошибки; 3) обусловленных недостаточной производительностью и точностью программных систем автоматизации, качество которых растет существенно медленнее увеличения сложности обрабатываемых моделей. Комплексное решение проблемы верификации системных моделей позволит в значительной степени снизить затраты на проектирование цифровых систем на кристаллах. Согласно исследованиям ведущих мировых компаний в области EDA (Cadence Design Systems, Synopsys Inc., Mentor Graphics Corporation, Magma, IBM, Intel, Sun Microsystems, Cisco Systems Inc., Atrenta, Aldec Inc.) усилия ученых должны быть сосредоточены на создании эффективных методов верификации, способных: 1) в десятки раз снизить вероятность возникновения ошибок за счет уменьшения участия человека в процессе проектирования; 2) обеспечить обнаружение и диагностирование допущенных неточностей на ранней стадии проектирования в целях сокращения времени и стоимости устранения несоответствий спецификации; 3) на порядок повысить производительность и надежность систем верификации за счет повышения уровня абстракции как самих моделей, так и тестовых воздействий.

Предлагается обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели. Рассматриваются только высокоуровневые быстродействующие алгоритмические методы, а также минимальные модификации изложенных выше структур данных, обеспечивающие существенное ускорение цикла обработки темпоральных описаний. Следует отметить не менее важную роль низкоуровневых задач оптимизации вычислений на булевом уровне. Хотя модель DRTLQ [2] предполагает полное абстрагирование от деталей элементарных вычислений через абстракцию верификационных переменных, за этим уровнем модели скрыта реализация компилятивного типа, где применяются оптимизации ассемблерного уровня. Такие оптимизации подбираются для конкретной платформы выполнения, и носят аналогичный характер по

сравнению с оптимизациями, применяемыми для ускорения вычислений в процессе HDL-симуляции.

Цель исследования – разработка моделей и методов функциональной верификации цифровых систем на кристаллах на основе использования темпоральных ассерций для тестового диагностирования ошибок в процессе программно-аппаратного моделирования, обеспечивающего существенное повышение качества цифрового изделия, а также уменьшение временных и материальных затрат проектирования.

Задачи исследования: 1) аналитический обзор наиболее значимых оптимизаций вычислительного цикла модели DRTLQ, ориентированных на уменьшение количества элементарных действий по транспортированию событий от входов к выходам модели: 2) разработка метода компрессии событий на выходе бесконечных репетиций.

Источники: языки описания аппаратуры высокого уровня [3-5,16], моделирование и синтез цифровых систем на кристаллах [6, 7, 10, 11, 15], верификация цифровых проектов [8,9,12-14], ассерции [17-19].

1. Метод предикторных связей. При транспортировании событий через последовательностный уровень модели DRTLQ распространенной ситуацией может быть продвижение события с $\rho^{VAL} = 0$. Существует ряд конструкций, например, логические последовательностные операторы AND и INTERSECT, когда результат функции зависит от появления такого события, и оно является ожидаемым. Во многих же других случаях транспортирование такого события является признаком предстоящего неудачного разрешения вычислительного потока. Пусть имеется последовательность $\{a;b[*2];c\}$ (рис. 1). Любой из выделенных генераторов может выдать событие с $\rho^{VAL} = 0$ и однозначно определить результирующий негативный статус анализа всей последовательности.

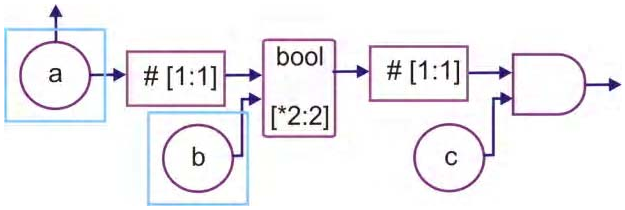


Рис. 1. Потенциальные точки досрочного появления неудачного статуса анализа последовательности $\{a;b[*2];c\}$

Учитывая значительное количество элементов, стоящих на пути формулы после первого и второго генераторов, имеет смысл отказаться от дальнейшего продвижения события с $\rho^{VAL} = 0$ и доставить неудачное событие непосредственно на следующий по иерархии уровень обработки. Введение быстрой прямой связи транспортирования события, однозначно досрочно определяющего неудачный статус анализа последовательности, к следующему уровню структур модели (ассерционный монитор или темпоральное свойство) называется методом предикторных связей (рис. 2).

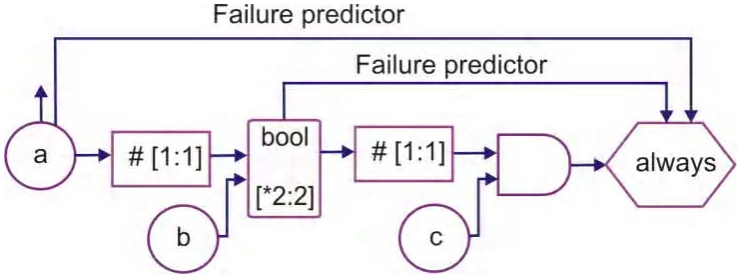


Рис. 2. Понятие предикторных связей

Предикторная связь к следующему уровню создается от элементов, непосредственно наблюдающих состояние, способное досрочно завершить анализ последовательности. В примере на рис. 2 первая предикторная связь создается от начальной функции-генератора (ситуация $a = 0$), а вторая – от элемента репетиции (ситуация $b = 0$). Вторая связь не

задается непосредственно на элементе-генераторе, поскольку само по себе появление события $b = 0$ не является определяющим фактором разрешения анализа, так как требуется выполнение $a = 1$ на предыдущем такте анализа. Последовательность может быть также разрешена с неудачным статусом событием $c = 0$, однако в данном случае добавление предикторных связей не имеет смысла, так как за последним элементом-генератором более нет элементов с временным сдвигом.

Выявление и доставка событий с $\rho^{VAL} = 0$ по предикторной связи обеспечивается модификацией процедуры левого итерирования цепочек в соответствующих контейнерах, описанной в [1]. Помимо анализа флага разрешения правой цепочки, альтернативным условием отделения события от цепочки транспортирования является неудачный статус анализа. В таком случае событие, вместо немедленного уничтожения, доставляется в точку назначения по предикторной связи. Те же событийные контейнеры, для которых предикторных связей не предусматривается, функционируют согласно собственным описанным выше процедурам без модификаций.

Реакция ассерционного монитора на появление событий с негативным статусом по предикторной связи зависит от его типа. Разрешающий монитор для ограничения вида $always(\{...seq...\})$ полностью уничтожает поток активации, поскольку ассерция однозначно считается нарушенной. Запрещающий монитор вида $never(\{...seq...\})$ при получении события $\rho^{VAL} = 0$, напротив, уничтожает его, и лишь в том случае, когда событие является единственным (последним) в потоке активации; это приводит к разрешению последовательности, причем с положительным статусом. К ожидающим мониторам вида $eventually!(\{...seq...\})$ также формируются предикторные связи, роль которых состоит только в элиминации неудачных витков анализа за счет досрочного уничтожения замеченного неудачного события.

Отключение предикторных связей не влияет на функциональную составляющую результата. Очевидно, если формула не удовлетворяется на некотором вычислительном пути, модель DRTLQ продемонстрирует этот факт независимо от наличия предикторных связей. Отличие состоит в конкретном моменте обнаружения неудачного результата (таб. 1). Количество вычислительных путей, завершившихся неудачно, либо успешно, либо не завершившихся вообще, не отличается между вариантами модели. Однако статус $FAIL^2$ в модели с учетом предикторов получен на первом же такте анализа ($a = 0$ – срабатывает первая предикторная связь), статусы $FAIL^4$ и $FAIL^5$ также получены досрочно ($b = 0$ – работа второй предикторной связи). Условие $c = 0$, а также успешный статус обнаруживаются одновременно.

Таблица 1. Ускорение анализа при помощи предикторных связей

#	A	b	c	Базовая модель	Предикторы
1	1	1	1	-	-
2	0	1	1	-	$FAIL^2$
3	1	1	1	-	-
4	1	1	1	$PASS^1$	$PASS^1$
5	1	1	1	$FAIL^2$	-
6	1	0	0	$FAIL^3$	$FAIL^3$ $FAIL^4$ $FAIL^5$
7	1	1	1	$FAIL^4$	-
8	1	1	1	$FAIL^5$	-
Конечный момент симуляции				$UNFINISHED^{6-8}$	$UNFINISHED^{6-8}$

Число операций транспортирования индивидуальных событий в тесте, рассмотренном в табл. 1, равно 31 в базовой модели, и лишь 20 – в модели с учетом предикторов. Кроме

того, за счет раннего завершения FAIL[2] по первой предикторной связи было создано/ уничтожено на одно меньше событийное расширение для обработки репетиции. Поскольку влияние времени проверки события на наличие признака $\rho^{VAL} = 0$ при итерировании контейнеров событий ничтожно мало, преимущества внесения предикторных связей очевидны с точки зрения производительности.

2. Метод обратного оповещения интервальных очередей и репетиций. Пусть имеется последовательность $\{a;[*3:5];b\}$, использующая интервальную форму элемента-очереди (рис. 3), на которую подается тестовое воздействие, представленное в табл. 2.

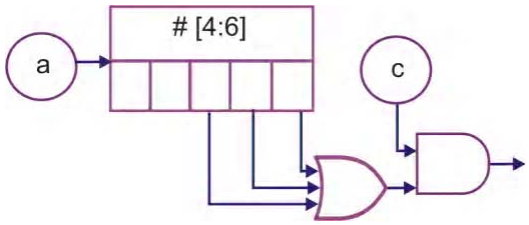


Рис. 3. Последовательность с интервальной формой очереди

Таблица 2. Тестовое воздействие на функцию-очередь

#	A	b	Содержимое очереди	Выход очереди	Результат
1	1	0	$\varnothing$	$\varnothing$	-
2	1	0	$[1]: e_1^1$	$\varnothing$	-
3	1	0	$[1]: e_1^2; [2]: e_1^1$	$\varnothing$	-
4	1	0	$[1]: e_1^3; [2]: e_1^2; [3]: e_1^1$	$\varnothing$	-
5	1	0	$[1]: e_1^4; [2]: e_1^3; [3]: e_1^2; [4]: e_1^1$	e_2^1	-
6	1	0	$[1]: e_1^5; [2]: e_1^4; [3]: e_1^3; [4]: e_1^2; [5]: e_1^1$	$e_3^1; e_2^2$	-
7	1	1	$[1]: e_1^6; [2]: e_1^5; [3]: e_1^4; [4]: e_1^3$ $[5]: e_1^2$ $[6]: e_1^1$	$e_4^1; e_3^2; e_2^3$	PASS[1]
8	1	1	$[1]: e_1^7; [2]: e_1^6; [3]: e_1^5; [4]: e_1^4; [5]: \varnothing; [6]: \varnothing$	e_2^4	PASS[2]

Столбец “Содержимое очереди” в табл. 2 демонстрирует динамику изменения множеств внутренних событийных контейнеров интервальной очереди, а столбец “Выход очереди” – множество событий, поступающих на выход очереди на каждом такте. В соответствии с семантикой интервальных очередей, определенной в [2], на выход очереди подаются копии событий из внутренних множеств, которые достигли необходимого минимального интервала ожидания, но еще не достигли максимально допустимого.

Следует отметить, что сгенерированные копии событий на тактовом цикле №5 (e_2^1) и №6 ($e_3^1; e_2^2$) оказываются невостребованными при достижении элемента конъюнктивной конкатенации, поскольку $b = 0$ на данных тактах. Варианты развития последовательности не удовлетворяются в таком случае, и сгенерированные копии событий нужно уничтожить. Если разница между границами интервала очереди велика, избыточно может генерироваться и через короткое время уничтожаться большое количество событий.

Аналогичная ситуация характерна и для элементов, реализующих булеву репетицию интервальной формы, где в соответствии с [2] также генерируются копии внутренних событий, превзошедших минимальное количество вычислительных итераций.

Избежать такого нерационального поведения позволяет метод обратного оповещения интервальных очередей и репетиций. Суть метода заключается в добавлении дополнитель-

ной управляющей обратной связи между интервальным последовательностным элементом и элементом-генератором, от которого он зависит (рис. 4).

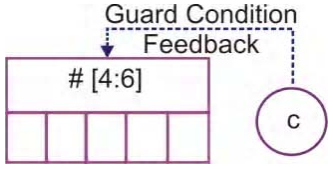


Рис. 4. Связь обратного оповещения интервальных очередей

Семантика функции-очереди модифицируется с учетом связи обратного оповещения – булевого значения GCF – таким образом, что ложное значение, возникающее на следующем элементе-генераторе, блокирует копирование лишних цепочек внутренних событий ввиду нецелесообразности их дальнейшего транспортирования:

$$f_{\#[N:M]}(t) = r_{M-1}(t-1) \cup \begin{cases} \bigcup_{i=N-1}^{M-2} \kappa(r_i(t-1)), GCF = 1; \\ \varnothing, GCF = 0. \end{cases} \quad (1)$$

При выполнении условия GCF = 0 на выход очереди поступают только те события, которые достигли максимального интервала ожидания в очереди. Такие события не будут уничтожены при прохождении элемента конъюнктивной конкатенации, однако им будет присвоен атрибут $\rho^{VAL} = 0$.

В табл. 3 представлены результаты моделирования теста, аналогичного тесту из табл. 2, с учетом связи обратного оповещения.

Таблица 3. Работа очереди с учетом оптимизации GCF

a	b (условие GCF)	Содержимое очереди	Выход очереди	Результат
1	0	$\varnothing$	$\varnothing$	-
1	0	[1]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^2 ; [2]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^3 ; [2]: e_1^2 ; [3]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^4 ; [2]: e_1^3 ; [3]: e_1^2 ; [4]: e_1^1	$\varnothing$	-
1	0	[1]: e_1^5 ; [2]: e_1^4 ; [3]: e_1^3 ; [4]: e_1^2 ; [5]: e_1^1	$\varnothing$	-
1	1	[1]: e_1^6 ; [2]: e_1^5 ; [3]: e_1^4 ; [4]: e_1^3 ; [5]: e_1^2 [6]: e_1^1	e_4^1 ; e_3^2 ; e_2^3	PASS[1]
1	1	[1]: e_1^7 ; [2]: e_1^6 ; [3]: e_1^5 ; [4]: e_1^4 ; [5]: $\varnothing$ [6]: $\varnothing$	e_2^4	PASS[2]

3. Метод иерархического разрешения логических групп. Пусть имеется сложная LTL-формула, содержащая иерархически вложенные логические последовательностные операторы OR и AND: $(\{a;b[*2]\} | \{a;[*2];b\}) | (\{c;[*2];d\} \& \{e;d;c\})$. На рис. 5. представлена статическая структура реализации данной формулы в модели DRTLQ. Для удобства последующего анализа элементы модели пронумерованы.

Пара элементов OR 1/25 образует первый уровень логического разветвления, а пары элементов OR 2/11 и элементов AND 12/24 – второй уровень. Каждый из уровней прикрепляет к поступающим на вход событиям соответствующие реализуемым операторам событийные расширения. События, проходящие через элемент 9, также дополняются третьим расширением, необходимым для обработки репетиции. Единственное поступившее на вход

разделителя 1 событие породит 3 копии, 4 событийных расширения, и в сумме потребует максимум 20 операций транспортирования в течение 4 циклов анализа. Из-за пересечения параллельно анализируемых последовательностей в модели одновременно может существовать до 4 потоков активации, в общей сложности включающих 16 событий и 15 расширений. Получение результата первого вычислительного пути может потребовать выполнения около 50 операций транспортирования событий.

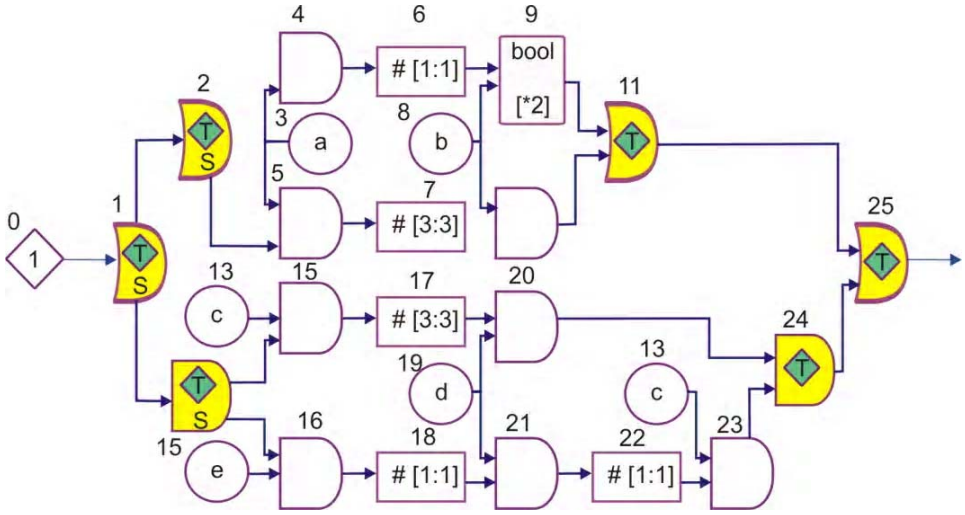


Рис. 5. Модель DRTLQ с вложенными последовательностными операторами

Представим сценарий возможного теста, при котором сигнал $a = 0$ и одновременно с этим либо $c = 0$, либо $e = 0$. В таком случае вычислительный путь, начинающийся с такого вектора, может быть удовлетворен на первом же такте анализа, поскольку на всех возможных вариантах ветвления модели будут транспортироваться события с $\rho^{VAL} = 0$. Однако моментальное разрешение потока активации является проблематичным в модели на рис. 5, поскольку здесь в силу структуры модели нельзя применить метод предикторных связей. Без модификации рабочего цикла модели такие события придется транспортировать к выходам за несколько циклов анализа, несмотря на то, что результат вычисления в целом уже однозначно определен на первом же шаге.

Решением данной проблемы является предлагаемый метод иерархического разрешения логических групп. Метод основывается, во-первых, на дополнении логических событийных расширений [1, формула (3.5)] связями быстрого транспортирования к элементу-соединителю, во-вторых, на учете атрибута $\rho^{RESOLVED}$ расширения события при итерировании событийного контейнера. Основная идея метода состоит в отказе от транспортирования событий уже разрешенной логической группы, а также на быстром продвижении разрешающего события к выходам модели. Подробно рассмотрим предлагаемый алгоритм на примерах:

1)В начальном состоянии все событийные контейнеры модели пусты. Элементом 0 генерируется первое событие e_1^1, создающее поток активации. Событие транспортируется к элементу-разделителю 1, где создается копия e_2^1, и к обоим событиям прикрепляется расширение X_L^1, инициализируемое значениями

$$< \rho^{VAL} = 1, \rho^{RESOLVED} = 0, N_T = 2, N_C = 0 >,$$

а также содержащее связь быстрого транспортирования к элементу 25. Далее события следуют к элементам 2 и 12 соответственно, где создаются дополнительные копии события e_3^1 и e_4^1, а также прикрепляются вложенные расширения X_L^2 со связью с элементом 11 (для события, транспортируемого по ветке с OR-разделителем 2) и X_L^3 со связью с элементом 24 (для события, транспортируемого по ветке с AND-разделителем 12). В табл. 4 показано текущее состояние обработки.

2) Производится анализ элемента 4, изменяющего атрибут $\rho^{VAL}(e_1^1)$ с 1 на 0 в связи с ложным значением, считываемым с генератора 3 при $a=0$. Транспортировать данное событие обычным путем далее не имеет смысла, и по быстрой связи оно направляется к элементу 11. Элемент не разрешает группу, но увеличивает число N_c и уничтожает событие e_1^1 (см. табл. 5).

Таблица 4. Состояние обработки ассерции на шаге №1

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^1	VAL = 1 RDEAD = 0 LDEAD = 0	4-in	e_1^1	e_1^4 - e_1^2	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^2	VAL = 1 RDEAD = 0 LDEAD = 0	5-in	e_1^2	e_1^1 - e_1^3	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^2 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^1	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25

Таблица 5. Состояние обработки ассерции на шаге №2

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^1	VAL = 0 RDEAD = 1 LDEAD = 1	11-in	$\varnothing$	$\varnothing$	OR2 NC=1 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^2	VAL = 1 RDEAD = 0 LDEAD = 0	5-in	e_1^2	e_1^4 - e_1^3	OR2 NC=1 VAL=1 RESOLVED=0 JOINER: 11	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^2 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^2	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 NC=0 VAL=1 RESOLVED=0 JOINER: 25

3) Аналогично производится анализ элемента 5, также сталкивающегося с необходимостью транспортирования события e_1^2 с инвертированным статусом по быстрой связи с элементом 11. На этот раз группа 2/11 разрешается с ложным статусом, поскольку все операнды завершились неудачно. От события e_1^2 отделяется расширение X_L^2, и событие, как неудачное, направляется по быстрой связи к элементу 25. Вторая группа не разрешается, поскольку требуется результат второго операнда. Событие e_1^2 уничтожается (табл. 6).

4)Производится анализ элемента 15, завершающийся быстрым транспортированием события e_1^3 с инвертированным статусом к элементу 24. Поскольку оператор AND признается удовлетворенным при первом же неудачным операнде, группа разрешается. При этом событие e_1^4 будет уничтожено вместе с расширением X_L^3 при первой же

попытке транспортирования, поскольку $\rho^{\mathrm{RESOLVED}}(X_L^3)=1$. Само событие e_1^3 направляется по быстрой связи к элементу 25, где также с негативным статусом разрешается группа X_L^1, и событие e_1^3 направляется к выходу модели.

Таблица 6. Состояние обработки ассерции на шаге №3

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^2	VAL = 0 RDEAD = 1 LDEAD = 1	25-in	∅	∅	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25	∅
e_1^3	VAL = 1 RDEAD = 0 LDEAD = 0	15-in	e_1^3	e_1^4 - e_1^4	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 **NC=1** VAL=1 RESOLVED=0 JOINER: 25
e_1^4	VAL = 1 RDEAD = 0 LDEAD = 0	16-in	e_1^4	e_1^3 - e_1^3	AND2 NC=0 VAL=1 RESOLVED=0 JOINER: 24	OR2 **NC=1** VAL=1 RESOLVED=0 JOINER: 25

В течение единственного такта анализа описанный алгоритм потребовал только 7 операций обычного транспортирования и 3 операции быстрого транспортирования при разрешении групп. В результате применения метода иерархического разрешения групп статус вычисления первого вычислительного пути был получен с использованием только 15% возможных операций транспортирования (так как не потребовался параллельный анализ нескольких потоков активации), при этом 13 из 25 элементов модели не были задействованы при вычислениях. Можно утверждать о 6-кратном ускорении анализа для рассмотренного тестового воздействия (табл. 7).

Таблица 7. Состояние обработки ассерции на шаге №4

Событие	Атрибуты	Место в модели	Левые связи	Правые связи	Расширение 1	Расширение 2
e_1^3	VAL = 0 RDEAD = 1 LDEAD = 0	25-in	∅	∅	AND2 NC=1 VAL=0 RESOLVED=1 JOINER: 24	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25
e_1^4	VAL = 1 RDEAD = 1 LDEAD = 0	16-in	e_1^4	∅	AND2 NC=1 VAL=0 RESOLVED=1 JOINER: 24	OR2 NC=2 VAL=0 RESOLVED=1 JOINER: 25

Метод иерархического разрешения последовательностных групп, внося лишь минимальную дополнительную стоимость (новые связи в рамках событийных расширений, новый критерий очистки при итерировании контейнеров событий), обеспечивает максимально быстрое завершение анализа досрочно удовлетворяемых потоков активации. По сути, метод иерархического расширения групп развивает ранее предложенный метод предикторных связей на уровне пары соединитель-разделитель. При этом никак не изменяется стоимость обычного варианта анализа, при котором не наблюдается возможности досрочного удовлетворения вычислений.

4. Метод компрессии потока событий в бесконечной репетиции. Одним из контекстов, вызывающих значительное падение производительности анализа модели DRTLQ, является репетиция с бесконечной правой границей количества итераций. При длительном ожидании разрешения последующего потока репетиция генерирует и накапливает значительное число событий. Пусть имеется последовательность {a[*];b}, допуска-

ющая произвольное число повторений (от нуля до бесконечности) события $a=1$ перед возникновением события $b=1$. Структурное представление данной последовательности в модели DRTLQ показано на рис. 6. Во избежание создания и транспортирования на выход репетиции избыточных копий событий, которые при выполнении условия $b\neq 1$ будут уничтожены следующим элементом, здесь применяется ранее рассмотренная оптимизация обратного оповещения.

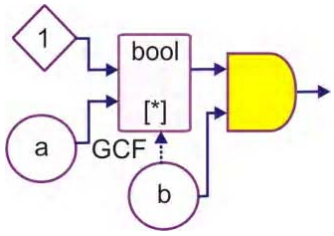


Рис. 6. Последовательность с репетицией с бесконечной правой границей

Предположим, на данную модель подается тестовое воздействие, представленное в табл. 8, в рамках которого в течение нескольких тактов инициируется входное событие $a=1$, а разрешающее событие $b=1$ наступает с некоторым опозданием. Исходя из семантики репетиций с бесконечными интервалами, репетиция накапливает события до момента разрешения.

Таблица 8. Работа функции-репетиции с бесконечной границей

#	a	b	Содержимое репетиции	Выход модели	Результат
1	1	0	$e_1^1\{r=1\}$	$\varnothing$	-
2	1	0	$e_1^1\{r=2\}$, $e_1^2\{r=1\}$	$\varnothing$	-
3	1	0	$e_1^1\{r=3\}$, $e_1^2\{r=2\}$, $e_1^3\{r=1\}$	$\varnothing$	-
4	1	0	$e_1^1\{r=4\}$, $e_1^2\{r=3\}$, $e_1^3\{r=2\}$, $e_1^4\{r=1\}$	$\varnothing$	-
5	1	1	$\varnothing$	e_1^1, e_1^2, e_1^3, e_1^4, e_1^5	$PASS^5$ $PASS^4$ $PASS^3$ $PASS^2$ $PASS^1$

Без применения оптимизации обратного оповещения количество событий, генерируемых элементом-репетицией, возрастало бы с квадратичной сложностью в зависимости от фактического количества тактов ожидания разрешения. Обратная связь от элемента-генератора b превращает данное количество в линейно возрастающую характеристику, однако при длительном ожидании может накопиться существенное количество событий.

Очевидным выводом из примера, рассмотренного в табл. 8, является тот факт, что при наступлении разрешающего события все накопленные данные транспортируются и обрабатываются далее совместно и одновременно, не покидая левой цепочки транспортирования. За исключением возможных дополнительных нюансов в ситуации нахождения события между парами соединитель-разделитель в логическом последовательностном операторе, все элементы цепочки событий либо одновременно достигнут выходов модели, либо будут одновременно уничтожены.

Данный факт предлагается использовать в целях уменьшения нагрузки на систему анализа ассерций. Сущность оптимизации состоит в применении метода компрессии цепочки событий в контексте разрешения функций-репетиций с бесконечными правыми интервалами. Всю цепочку накопленных событий можно заменить единственным событием с прикрепленным к нему специальным расширением, задающим множество моделируемых вычислительных потоков:

$$X_\infty :< X_{NEXT}, N_c, \{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}^{\overline{i=0..(N_c-1)}} >, \quad (2)$$

где X_∞ – прикрепляемое событийное расширение компрессии; N_c – число сжатых событий; $\{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}^{\overline{i=0..(N_c-1)}}$ – характеристический вектор и пара правых связей одного из событий, вошедших в сжатое расширение.

События могут быть сжаты в цепочку (2) только в том случае, если время создания каждого из них образует непрерывный ряд:

$$t_b(e_1) = t_b(e_2) - 1 = t_b(e_3) - 2 = \cdots = t_b(e_{N_c}) - N_c + 1. \quad (3)$$

Выполнение соотношения (3) обеспечивает возможность вычисления компонента t_b одного из сжатых событий, имея значение t_b для первого события (именно это значение ассоциируется с единственным событием, представляющим сжатую цепочку) и порядковый номер искомого события.

Характеристические векторы ρ^i могут различаться только в случае порождения сжатой цепочки между элементами соединитель-разделитель логического разветвления, когда альтернативный путь досрочно разрешает соответствующую группу. Поскольку данный факт представляется возможным установить статически во время подготовки модели, то в случае отсутствия окружающей логической группы цепочка (1) может не содержать элементов ρ ввиду их избыточности. Если же логическое разветвление имеется на родительском уровне модели, установление атрибута $\rho^{LDEAD} = 1$ на одном из событий, входящих в сжатую цепочку, не должно нарушать заложенный принцип компрессии. В тривиальных случаях можно также избежать присутствия в модели пары связей $\{e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}$, если поступающие на вход бесконечной репетиции события являются единственными в рамках представляемого ими потока активации.

При достижении ассерционного монитора или темпорального свойства событие, представляющее сжатую цепочку, требует специального режима обработки. Во-первых, если результат вычисления в соответствии разрешает поток активации, то речь идет об одновременном разрешении интервала потоков активации. Во-вторых, следует учитывать возможность наличия в сжатой цепочке некоторых записей о событиях с атрибутом $\rho^{LDEAD} = 1$, которые необходимо полностью проигнорировать.

Выводы. *Практическая значимость.* Предложенные модели и методы, составляющие основу программно-аппаратной среды верификации на основе ассерций, существенно (30%) повышают тестопригодность внутренних линий цифровой системы, что позволяет уменьшить временные затраты для создания теста, улучшить его функциональную полноту и качество проекта в целом.

Модели и методы функциональной верификации на основе темпоральных ассерций доведены до практической реализации в виде программных компонентов верификационной системы Riviera (Aldec Inc.), что дает возможность синтезировать специализированные и эффективные маршруты проверки и диагностирования цифровых систем на кристаллах.

Выигрыш производительности при применении предложенного метода компрессии событий на выходе бесконечных репетиций обусловлен следующими фактами:

1. Порождается меньшее количество событий. В худшем случае из каждого сжатого входного события сохраняется только тройка элементов $\{\rho, e_{\downarrow\rightarrow}, e_{\uparrow\rightarrow}\}$, в лучшем случае очередное входное событие, прикрепляемое к сжатой цепочке, лишь увеличивает значение счетчика N_c.

2. К выходам модели транспортируется только одно событие вместо цепочки из N_c событий. Данная характеристика производительности является константной относительно возрастания параметра N_c.

3. Специальные способы обработки требуются только на уровне ассерционного монитора и темпорального свойства. При этом возможность появления сжатой цепочки событий можно определить статически, что означает отсутствие каких-либо накладных расходов для формул, не содержащих элементы-репетиции с бесконечными интервалами.

Список литературы: 1. *Зайченко С.А., Хаханов В.И., Чумаченко С.В.* Структуры данных и модели реализации базовых элементов модели динамических регистровых очередей // Радиоэлектроника и информатика. 2010. № 1. С. 63-70. **2.** *Зайченко С.А., Чумаченко С.В.* DRTLQ-модель для функциональной верификации цифровых систем на основе линейной темпоральной логики // АСУ и приборы автоматики. 2010. Вып. 150. С. 33-48. **3.** *Berge J., O. Levia, J. Rouillard.* High-level System Modeling Specification Languages // Kluwer Academic Publishers, 1995, 162p. **4.** *Wakerly J.F.* Digital Design Principles and Practices. 1999. 895p. **5.** *Palnitkar S.* Verilog HDL: A Guide to Digital Design and Synthesis. Prentice Hall, 2003. 496p. **6.** *Lam W.K.*Hardware Design Verification: Simulation and Formal Method-Based Approaches. Prentice Hall, 2005, 585p. **7.** *Clarke E., Grumberg O., Peled D.* Model Checking. 6th edition, MIT Press, 2008. 313p. **8.** *Bening L.* An RTL Design Verification Linting Methodology // Proceedings of the International HDL Conference, 1999. P. 136-140. **9.** *Chakraborty R., Chowdhury D.R.* Raising the Level of Abstraction for the Timing Verification of System-on-Chips // IEEE Computer Society Annual Symposium on VLSI 2008, pp. 459-462. **10.** *Abramovici M., Breuer M.A., Friedman A.D.* Digital systems testing and testable design. Computer Science Press, 1998. 652 p. **11.** *Rabaey J.M., Pedram M.* Low-Power Design Methodologies // Kluwer Academic Publishers, 1996. 367p. **12.** *Piziali A.* Functional verification coverage measurement and analysis. Verisity Design Inc., Kluwer Academic Publishers, 2004. 213p. **13.** *Bening L., Foster H.* Principles of Verifiable Rtl Design: A Functional Coding Style Supporting Verification Processes in Verilog", Kluwer Academic Publishers, 2000. 281p. **14.** *Budkowski S., Najm E.* Formal Description Techniques and Protocol Specification, Testing and Verification. Chapman and Hall, 1998. 467p. **15.** *RTL* Design Style Guide for VHDL. English edition, STARC, 2003. 440p. **16.** *Melnyk D., Zaychenko S., Kolesnikov K., Lukashenko O.* Model of source code analyzer for hardware description languages // CAD Systems in Microelectronics, 2007. P. 113-114. **17.** *Foster H., Krolnik A., Lacey D.* Assertions-based Design. Kluwer Academic Publishers. 2003. 392p. **18.** *Assertion*-Based Verification. Synopsys Inc., Technical Report, May 2003. 14p. **19.** *Yeung P.* The Four Pillars of Assertions-Based Verification // Mentor Graphics Corporation, EuroDesignCon 2004. 21p.

Поступила в редколлегию 19.12.2009

Зайченко Сергей Александрович, аспирант кафедры АПВТ ХНУРЭ, начальник отдела разработки компании Aldec-Kharkov Ltd. Научные интересы: системы автоматизированного проектирования, моделирования и верификации цифровых систем на кристаллах. Увлечения: литература, музыка, футбол. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (097)-367-62-93. E-mail: Sergei.Zaychenko@aldec.com

Чумаченко Светлана Викторовна, д-р техн. наук, проф. кафедры АПВТ ХНУРЭ. Научные интересы: математическое моделирование, методы дискретной оптимизации. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326, e-mail: ri@kture.kharkov.ua.

УДК 004.415.2

В.А. ГОРБАЧЕВ, А.Ю. ЕФАНОВ

РАЗРАБОТКА МОБИЛЬНЫХ СЕРВИСНО-ОРИЕНТИРОВАННЫХ СИСТЕМ

Описывается разработка мобильной сервисно-ориентированной системы, предоставляющей доступ к ресурсам мобильного устройства через централизованную систему либо через другие мобильные устройства. Рассматриваются основные типы сервисно-ориентированных систем, а также общая архитектура системы, базирующаяся на облачных вычислениях.

Актуальность, новизна и цель работы

Постоянный рост вычислительных ресурсов мобильных устройств приводит к расширению возможностей их применения в повседневной жизни. В этой связи разработка приложений для них становится одним из основных направлений в индустрии программного обеспечения. Задача исследований в этой области – сделать рабочее место человека мобильным, предоставив ему соответствующие ресурсы для решения его проблем.

Современные мобильные устройства можно разделить на два типа:

– Непрограмируемые устройства. К данному типу можно отнести все мобильные либо встроенные (embedded) устройства, функции которых не могут быть расширены после производства, либо для этих устройств отсутствуют открытые программные интерфейсы.

– Программируемые устройства. К этому типу можно отнести устройства, обладающие открытыми программными интерфейсами, которые могут быть использованы для расширения функциональных возможностей устройства.

Для данного исследования интерес представляют устройства из второй группы, так как их функции могут быть расширены дополнительным программным обеспечением. Как правило, данные устройства имеют больше ресурсов центрального процессора, памяти, сетевой пропускной способности и, как следствие, могут стать сервисным провайдером. Провайдер сервисов означает, что устройство может предоставлять доступ к своим ресурсам через сеть различным клиентам, включающим другие мобильные устройства либо веб-браузер.К ресурсам мобильного устройства можно отнести как его аппаратные так и программные возможности, например, модуль позиционирования GPS, возможность записи звука, а также отправки текстовых сообщений и т.д.

На текущий момент использование ресурсов различных мобильных устройств в одном приложении распространено слабо. В данный момент начали распространяться сервисы для синхронизации данных между мобильными устройствами и компьютерами. Примерами таких сервисов являются Microsoft Live Mesh [1] и Apple Mobile Me [2]. Данные сервисы могут быть представлены как частный случай описаного ниже исследования.

Использование ресурсов различных мобильных устройств открывает большие перспективы для удобства работы, повышения производительности решения какой-либо задачи человеком, а также сокращения стоимости решения задачи.

К недостаткам такого подхода можно отнести зависимость от соединения с Интернетом. Соединение с сетью требуется для обмена данными между устройствами, а также системой управления ими. Возможность высокоскоростного соединения с Интернетом на текущий момент доступно не везде и требует дополнительной оплаты.

Цель работы – построение архитектуры мобильной сервисно-ориентированной системы, предоставляющей доступ к ресурсам мобильных устройств через централизованную систему.

Задачами исследования являются: 1) Анализ сервисно-ориентированных систем. 2) Разработка функциональной структуры мобильной сервисно-ориентированной системы. 3) Разработка программного обеспечения системы. 4) Оценка эффективности.

Рассмотрим задачи разработки функциональной структуры мобильной сервисно-ориентированной системы, а также ее программного обеспечения.

Общая архитектура системы

В данный пункт входит определение требований к системе, рассмотрение возможных вариантов использования системы в целом, определение спецификации требований к программному обеспечению на мобильном устройстве, а также требований к централизованной системе.

Одним из факторов, влияющих на архитектуру системы в целом, является определение эффективного метода взаимодействия между мобильными устройствами, а также между мобильным устройством и централизованной системой управления [3]. Существуют два метода, с помощью которых мобильные устройства осуществляют обмен данными:

Мобильное устройство как сервер

В этой модели мобильное устройство используется в качестве веб-сервера т.е. на устройстве физически размещается компонент, предоставляющий доступ к ресурсам устройства. В этом случае мобильные устройства устанавливают соединения друг с другом напрямую, а централизованная система используется для обнаружения устройств в сети, т.е. предоставляет адреса доступных устройств.

Преимущества: поскольку устройства устанавливают соединения напрямую, без функции управления, то производительность общего решения становится выше.

Недостатки: возможны проблемы с установкой сетевых соединений, так как большинство мобильных операторов имеют сетевую инфраструктуру, которая блокирует запросы из внешней сети к мобильному устройству.

Мобильное устройство как провайдер ресурсов

В этом случае клиентский компонент на мобильном устройстве устанавливает соединения с центральной системой и поддерживает его постоянно. Мобильное устройство обслуживает запросы, поступающие из системы, и посылает информацию о своем текущем статусе. Система транслирует запросы пользователя в запросы к мобильному устройству и имеет возможность управлять доступом к ним.

Преимущества:

– Система имеет возможность управлять доступом как к отдельным устройствам системы, так и к определенным ресурсам в системе.

– Управление устройствами и ресурсами мобильных устройств через веб-браузер более надежно, проще в реализации и обладает более высокой производительностью.

–Меньше проблем с установкой сетевых соединений с мобильными устройствами, так как устройство инициирует соединение и поддерживает его в активном состоянии.

Недостатки: более низкая производительность системы при работе с другими мобильными устройствами, так как весь сетевой обмен происходит через дополнительные сервер.

При дальнейшем проектировании системы была выбрана вторая модель из-за меньших проблем с организацией сетевого взаимодействия между устройствами и системой, а также чтобы иметь единое место, где контролируется доступ к устройствам.

Общая схема функционирования системы приведена на рис. 1. На схеме изображена центральная система управления, а также мобильные устройства, которые с ней взаимодействуют.

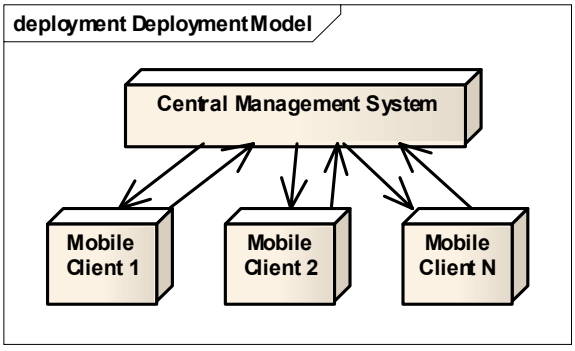


Рис. 1. Общая схема функционирования системы

Основными функциями центральной системы управления являются следующие:

– Хранение данных о мобильных устройствах, а также ресурсов, которыми устройство обладает.

– Обеспечение взаимодействия между устройствами и поддержка обмена сообщениями между ними.

–Поддержка приложений, использующих ресурсы досутпных мобильных устройств.

– Обеспечение доступа к мобильному устройству и его ресурсам, а также к приложениям, использующим эти ресурсы через веб-интерфейс.

Среди основных функций мобильных клиентских приложений можно выделить следующие:

– Предоставление доступа центральной системы управления к ресурсам устройства.

– Предоставление пользовательского интерфейса, обеспечивающего доступ к ресурсам других устройств, а также к приложениям, использующим эти ресурсы.

Разработка программного обеспечения системы

В данную задачу входит определение конкретной реализации архитектуры системы, разработка детальных архитектур для каждого ее компонента, включающая клиентские приложения для различных мобильных платформ, а также ПО для централизованной системы, базирующейся на облачных вычислениях.

В качестве платформы для облачных вычислений была выбрана Windows Azure [4]. Данная система включает в себя ряд компонентов и дополнительных сервисов. Ядро системы базируется на платформе Windows и представляет собой кластер, объединяющий

большое количество серверов, на которых установлены виртуальные машины. Система доступна через единый интерфейс, так что весь кластер доступен пользователям как единая машина. Подобно другим платформам корпорации Microsoft данная поддерживает платформу .NET для различных типов приложений, включающих, как сайты на ASP.NET, так и WCF веб-сервисы.

Доступ к данным выделен как отдельный компонент и представляет собой веб-сервис, доступный через REST интерфейс [5]. Windows Azure позволяет хранить данные в нескольких форматах, таких как таблицы, очереди и бинарные объекты (blobs).

На рис. 2 приведена реализация архитектуры системы.

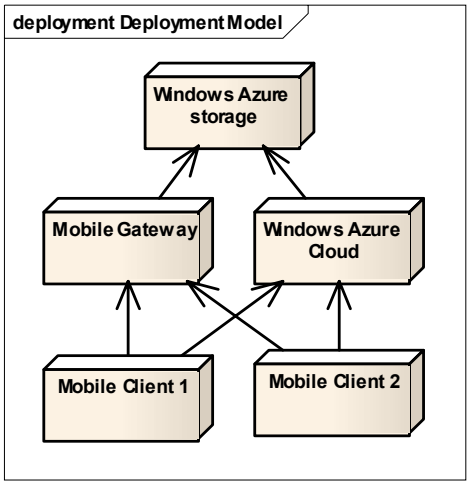


Рис. 2. Реализация архитекутры системы

В проекте используются следующие компоненты Windows Azure для хранения данных:

– Таблицы. В таблицах хранится вся доступная информация о системе, включающая данные мобильных устройств, их свойства, а также информация об интегрированных приложениях (т.е. приложениях, использующих ресурсы нескольких мобильных устройств).

– Очереди. Очереди используются для поддержки асинхронного взаимодействия между компонентами системы, такими как Мобильный шлюз (Mobile Gateway) и Обозреватель устройств (Devices Explorer).

Также на платформе облачных вычислений размещена централизованная система управления, которая использует следующие функции Windows Azure для своей работы:

– Компонент, выполняющий Веб-роль. Данный компонент представляет собой обычный веб-сервер, обрабатывающий HTTP запросы, на котором размещено веб-приложение, доступное через веб-браузер. Этот компонент реализует все функции, доступные пользователю, а также используется для размещения интегрированных приложений.

– Компонент с Рабочей ролью. Данный компонент используется для обработки данных, поступающих с мобильных устройств и их статуса. Компонент не имеет интерфейса пользователя и всегда работает в фоновом режиме. Взаимодействие с этим компонентом осуществляется через очереди либо через другие компоненты хранения данных Windows Azure.

Мобильный шлюз выполняет роль промежуточного сервера и используется для транслирования запросов мобильных устройств в запросы к подсистеме хранения данных. Мобильный шлюз предоставляет универсальный программный интерфейс (SOAP) для различных мобильных платформ. Этот компонент также используется для контроля доступа к данным и предотвращения несанкционированного доступа к ним. Компонент может быть размещен как на платформе облачных вычислений, так и в обычном окружения. В данном случае он размещен отдельно.

Мобильное клиентское приложение на рис. 1 представляет собой программное обеспечения для конкретной мобильной платформы. В рамках данного исследования были разработаны клиентские приложения для коммуникаторов Windows Mobile 6.1, а также iPhone OS 2.1, которые реализуют рассмотреные ранее основные функции мобильных клиентских

приложений. Клиентские приложения взаимодействуют с централизованной системой через мобильный шлюз, рассмотренный выше.

Реализация централизованной сервисно-ориентированной системы

Центральная система представляет собой приложение, размещенное на платформе облачных вычислений Windows Azure и состоящее из нескольких компонентов. Все компоненты системы реализованы с использованием возможностей платформы .NET Framework 3.5 на языке C#. Логически систему можно разделить на следующие компоненты.

– Компонент доступа к данным. Этот компонент инкапсулирует доступ к данным в системе Windows Azure и предоставляет программный интерфейс для доступа к таблицам, где хранятся данные об устройствах и их ресурсах, а также обеспечивает взаимодействие с очередями, которые используются для взаимодействия между мобильными устройствами и фоновым компонентом, рассмотренным ниже.

Данный компонент определяет несколько сообщений, которые могут быть помещены в очередь. Одно из них – это сообщение о состоянии устройства. Второе сообщение содержит информацию о текущем местоположении устройства, если она доступна.

Компонент также отвечает за проверку данных на соответствие допустимым значениям каждого поля.

Для обеспечения доступа к системе хранения данных о мобильных устройствах, их ресурсах, а также других данных системы были разработаны следующие классы.

– DeviceEntity – описывает объект устройства и его параметры.

– Devices – реализует логику добавления, удаления, изменения устройств, а также их выборку.

– DevicesDataContext – вспомогательный класс для представления коллекции объектов Devices в удобном для использования виде.

– DevicesQueue – класс для помещения и извлечения сообщений из очереди устройств.

– LocationAppQueue – класс для помещений и извлечения сообщений из очереди местоположений устройств.

– RuleViolation – класс для проверки корректности заполнения полей в классах Devices и Services.

– ServiceEntity – данный класс описывает структуру полей сервиса.

– Services – реализует логику добавления, изменения, удаления сервисов, а также их выборку

– ServicesDataContext – вспомогательный класс для представления коллекции объектов.

– DevicePingMessage – класс для сериализации и десирализации сообщений в формат xml для последующего помещения их в очередь.

– Веб-приложение. Данный компонент представляет собой веб-приложение и предоставляет пользовательский интерфейс к системе и ее компонентам. Компонент работает в вычислительном облаке в качестве Веб-роли. Данное веб-приложение построено на основе библиотеки ASP.NET MVC 1.0. Библиотека основана на шаблоне проектирования MVC, где структура веб-приложения разделена на классы, относящиеся к модели данных, классы, определяющие представление, а также классы, обрабатывающие запросы пользователя и бизнес-логику.

Класс Account Controller отвечает за обработку запросов на авторизацию пользователей в системе и регистрацию пользователей в ней.

Класс Home Controller отвечает за обработку запросов к главной странице, содержащей ссылки на функции системы.

Класс Devices Controller предоставляет пользователю возможность управления своими устройствами, добавления новых устройств, их редактирования, а также определения текущего статуса устройств. Информация, представленная в пользовательском интерфейсе управления устройствами, приведена в таблице.

Класс Location App Controller реализовывает поддержку интегрированного приложения, показывающего текущее местоположение всех устройств, зарегистрированных в системе.

Класс MVC Application определяет привязку пользовательских HTTP запросов к конкретным классам контроллеров, рассмотренных выше.

Интерфейс списка устройств

Название	Статус	Модель	Время
WinMo	1	HTCTouch	6/28/2009 10:04
My iPod	0	Apple iPod	6/27/2009 09:44
Palm Win	0	Palm Win	6/25/2009 19:31

Приложение позиционирования

Для тестирования возможностей мобильной сервисно-ориентированной системы на реальной проблеме была рассмотрена задача определения глобального местоположения нескольких мобильных устройств. Практическая ценность данного приложения в том, что оно может применяться для контроля перемещения грузов, спасательных операций, поиска и т.д.

Для решения данной задачи было разработано приложение, использующее модуль позиционирования GPS на нескольких мобильных устройствах. Приложение использует инфраструктуру центральной системы управления для получения доступа к данному ресурсу на всех мобильных устройствах и предоставляет пользовательский интерфейс для просмотра местоположения устройств на карте. Пользовательский интерфейс реализован как для мобильных клиентских приложений, так и для веб-браузера.

Приложение отображает текущее либо последнее местоположение всех зарегистрированных устройств на карте. Данные о текущем местоположении устройств клиентское приложение отсылает в систему каждые 5 секунд. Система показывает местоположение устройств как на платформе Windows Mobile, так и на платформе iPhone OS.

Приложение использует компонент доступа к данным для получения текущего местоположения устройств, а также Virtual Earth API для отображения данных на карте.

Это приложение также разработано на языке C# и имеет собственный контроллер обработки запросов пользователя.

Данные на карте показываются в режиме реального времени асинхронно, без необходимости обновления страницы. Пользователь также может увеличивать и уменьшать масштаб карты, что позволяет увидеть все устройства на земном шаре либо устройства на конкретной улице.

Фоновый компонент. Данный компонент всегда выполняется в фоновом режиме (рабочая роль) и не имеет пользовательского интерфейса. Компонент отвечает за обработку запросов с мобильных устройств и обновление информации об их текущем статусе и местоположении. Компонент работает с использованием очередей и получает данные для обработки из этого источника.

Мобильное устройство каждые 5 секунд посылает сообщение с информацией о текущем статусе и местоположении. Сообщение помещается в очередь для обработки фоновым компонентом.

Мобильное клиентское приложение. Это приложение представляет собой программу, реализованную для определенной мобильной программной платформы. Основная совокупность функций программы повторяет возможности веб-приложения Обозревателя устройств, однако данные доступны только в режиме для чтения.

Клиентское приложение предоставляет следующие функции:

– Список доступных устройств.

Приложение выводит на экран мобильного устройства список зарегистрированых устройств пользователя, а также их статус, указывающий на то, что устройство есть в сети и доступно для взаимодействия либо нет.

– Просмотр свойств определенных устройств. Клиентское приложение показывает различные свойства устройства, включая его модель и имя.

– Местоположение устройства. Приложение также показывает текущее местоположение выбранного устройства.

Все коммуникации с центральной системой управления происходят через промежуточный сервер, расммотренный ранее, что позволяет предоставлять интерфейс, удобный для реализации на большинстве мобильных платформ и языках программирования.

Клиентское приложения для платформы Windows Mobile. Данное приложение было разработано с использованием .NET Compact Framework 3.5 и работает на платформе Windows Mobile 6.1. Приложение также было разработано на языке C#.

Клиентское приложение помимо коммуникации с мобильным шлюзом для получения данных об устройствах и регистрации в системе также использует веб-сервисы Virtual Earth для отображения местоположения устройств на карте.

Клиентское приложение для iPhone. Подобно мобильному клиентскому приложению для Windows Mobile было разработано также приложение для платформы iPhone OS 2.1. Данное приложение реализует весь набор функций, приведенный выше.

Основные функции данного приложения были реализованы на JavaScript. На языке Objective-C была разработана только логика запуска приложения.

Это приложение использует мобильный шлюз для взаимодействия с центральной системой управления, а также для получения данных. Однако для отображения устройств на карте применяются сервисы Google Maps ввиду их встроенной поддержки на данной платформе.

Внешний вид пользовательского интерфейса на карте отображен на рис. 3.

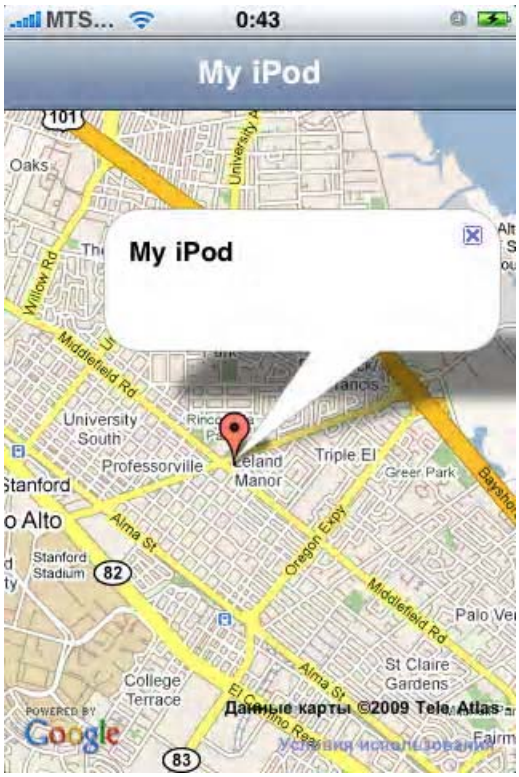


Рис. 3. Местоположение устройства на карте

Выводы

В исследовании продемонстрирована возможность реализации мобильной сервисно-ориентированной системы на примере разработки функциональной структуры программного комплекса, построение его архитектуры в целом и его отдельных компонентов.

После определения структуры был реализован прототип централизованной системы управления на платформе облачных вычислений Windows Azure, а также разработаны мобильные клиентские приложения для платформ Windows Mobile и iPhone OS.

Также было разработано приложение просмотра местоположения мобильных устройств на карте, что демонстрирует концепцию интеграции ресурсов различных мобильных устройств в одном приложении.

Новизна исследования состоит в следующем:

– Разработан инструмент, который может использоваться для исследований в области сервисно-ориентированных систем, а также сервисов для мобильных устройств.

– Продемонстрирована возможность и эффективность интеграции мобильных устройств через централизованную систему.

Исследованы возможности применения платформы облачных вычислений для размещения масштабируемых, высокопроизводительных систем

Практическая ценность работы состоит в использовании мобильного устройства в качестве сервера, предоставляющего доступ удаленным клиентам к своим ресурсам через сеть.

Список литературы: 1. Microsoft Live Mesh, http://mesh.com. **2.** Apple Mobile Me, http://me.com. **3.** *Gorton Ian,* 'Essential Software Architecture' Springer, 2006. 356p. **4.** *Chappel D.* 'Introducing the Azure Services Platform', 2008. 294p. **5.** *Richardson, Leonard Ruby, Sam,* 'RESTful Web Services' (O'Reilly Media, Inc., 2007. 311p.).

Поступила в редколлегию 20.10.2009

Горбачев Валерий Александрович, профессор кафедры ЭВМ ХНУРЭ. Научные интересы: моделирование систем. Хобби: музыка, волейбол, автомобили. Адрес: Украина, 61136, Харьков, ул. Героев Труда, 33, кв. 300, e-mail: Gorbachov Valeriy <gorbachov@kture.kharkov.ua> (Center NSIM).

Ефанов Александр Юрьевич, MCPD, ЭВМ ХНУРЭ. Научные интересы: сервис-ориентированые системы, компьютерные сети. Адрес: Украина, 61071, Харьков, ул. Кибальчича, 33, кв. 8.

УДК 519.713

А.В. КОРНИЛОВСКИЙ, В.А. ГУБИН

РАЗРАБОТКА ВЕБ-СЕРВИСА УДАЛЁННОГО ВЗАИМОДЕЙСТВИЯ С БАЗАМИ ДАННЫХ

Исследуются наиболее популярные подходы удалённого доступа к СУБД, анализируются их недостатки. Предлагается подход, использующий технологию веб-сервисов, для осуществления взаимодействия с удаленной СУБД, позволяющий в значительной мере избежать недостатков популярных подходов.

1. Введение

На сегодняшний день удалённое использование сервером баз данных в Web – привычное явление. Однако большинство систем, предоставляющих данную возможность, являются серверными СУБД, доступ к базам которых возможно осуществлять удалённо, используя TCP/IP [1]. Такое взаимодействие клиента БД и сервера СУБД имеет ряд недостатков: конструктивная сложность; привязка к конкретной платформе; небезопасность подключения.

2. Анализ предметной области

Рассмотрим более детально перечисленные аспекты предметной области.

Конструктивная сложность состоит в необходимости оборудования машины-сервера (если речь идёт о подключении к удалённому узлу), установке и настройке серверной СУБД, организации сетевого взаимодействия (вне зависимости от того, производится ли подключение к удалённому узлу, или же на 127.0.0.1, необходимо выделить и прослушивать порт(ы) для получения входящих запросов к СУБД), а также в доступе (создание аккаунтов, разграничение прав). Если последний пункт просто необходим для обеспечения безопасности взаимодействия, то по поводу первых можно внести некоторые коррективы.

С точки зрения простоты реализации веб-сервис является одним из наиболее оптимальных решений. Создание программного продукта такого класса не требует высокой квалификации разработчика в области знания протоколов уровня представления – XML-RPC, SOAP, а также языка описания веб-сервисов WSDL.

Привязка к платформе обусловлена необходимостью согласовывать платформу клиентского приложения с платформой СУБД посредством использования соответствующего драйвера или поставщика СУБД (набор классов, предназначенных для взаимодействия с хранилищем данных определенного типа). По этой причине наиболее распространены сочетания PHP + MySQL, C# + MS SQL, C++ + MS Access и др. Благодаря использованию технологии веб-сервисов, подобной зависимости можно избежать, так как к веб-сервису можно обращаться с клиентов любых программно-аппаратных платформ. С другой стороны, из веб-сервиса возможно обращаться к СУБД различных типов, используя технологию ADO .NET.

ADO.NET представляет собой набор библиотек, входящих в Microsoft .NET Framework и предназначенных для взаимодействия с различными хранилищами данных из .NET-приложений. Библиотеки ADO.NET включают все необходимые классы для подключения к источникам данных практически произвольного формата, выполнения запросов к этим источникам и получения результата. Кроме того, несомненным достоинством ADO.NET является возможность работы с отсоединенными источниками данных, представляющими собой структуры, которые организуют данные в оперативной памяти компьютера и работать с которыми возможно с применением ставших уже привычными средств доступа к данным. Таким образом, ADO.NET можно использовать в качестве надежного, иерархически организованного отсоединенного от источника кэша данных для автономной работы, что незаменимо при построении масштабируемых приложений. Применение этой технологии позволяет в значительной мере решить проблему согласования клиента и СУБД различных платформ.

Небезопасность авторизации и передачи данных возможно устранить при использовании шифрования данных авторизации, а также самого потока передачи запроса на обращение к хранилищу данных. Случайное комбинирование хотя бы нескольких различных алгоритмов шифрования в этом вопросе позволит существенно повысить уровень безопасности взаимодействия. Также на пользу идёт факт логического и/или пространственного разделения области работы приложения веб-сервиса и СУЬД. Благодаря этому, клиент может обращаться с запросом к интерфейсу веб-сервиса, расположенному на одном узле, а веб-сервис, в свою очередь, переадресовывать запрос на удалённый узел (один или группу, в зависимости от наличия распределённости СУБД). Это неизбежно снизит быстродействие работы с базой данных для клиента, однако поспособствует сохранности передаваемой информации.

3. Постановка задачи

Из описания существующей ситуации можно сделать вывод о необходимости создания веб-сервиса для удалённого взаимодействия клиентских приложений с хранилищами данных. Этот веб-сервис может быть публичным и обладать системой разделения прав пользователей на доступ и операции с данными.

В качестве платформы разработки самого веб-сервиса было принято решение использовать Microsoft .NET [2]. Это обусловлено, в первую очередь, простотой реализации, а также широкими возможностями, предоставляемыми WSDL веб-сервисами: возможность использования как HTTP, так и SOAP протокола для внешнего взаимодействия; наличие wsdl-описания, что позволяет без труда сделать сервис доступным из репозиториев, а также получить перечень веб-методов, доступных для вызова клиентским приложением.

С другой стороны, веб-сервис будет взаимодействовать с удалённой или же локальной БД под управлением СУБД MS SQL Server, как наиболее нативной для платформ MS, в целом. Как уже было упомянуто ранее, это не будет иметь никакого значения в плане особенностей организации взаимодействия клиентского приложения с веб-сервисом.

Выдвинуты следующие требования к реализуемому сервису:

– Возможность взаимодействия с локальной или удалённой СУБД MS SQL Server.

– Наличие функций для работы с данными (создание, добавление, обновление, удаление таблиц, кортежей).

– Наличие системы пользовательских аккаунтов.

– Разграничение прав пользователей на доступ к данным.

– Шифрование данных авторизации, а также передаваемого потока посредством комбинированного использования нескольких алгоритмов.

– Доступность сервиса и его WSDL-описания в UDDI репозиториях.

4. Программная реализация

В соответствии с описанной задачей и поставленными требованиями был реализован веб-сервис удаленного доступа к БД. Данный сервис построен на базе платформы Microsoft ASP .NET и применяет протокол SOAP XML для взаимодействия с клиентом. В качестве СУБД использована Microsoft SQL Server [3], взаимодействие с которой производится через объекты отсоединенных источников данных.

Схема взаимодействия веб-сервиса с клиентским приложением и с СУБД имеет следующий вид (рис.1):

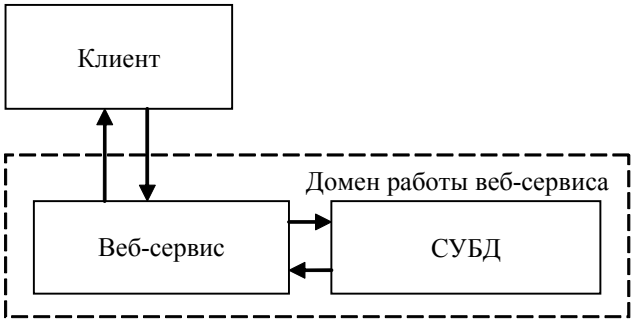


Рис. 1. Схема работы веб-сервиса

Схема взаимодействия также показывает строгое разделение домена веб-сервиса и клиентского приложения. Таким образом, непосредственный доступ к СУБД является инкапсулированным относительно клиентского приложения.

Классовая иерархия программного продукта представлена на рис. 2.

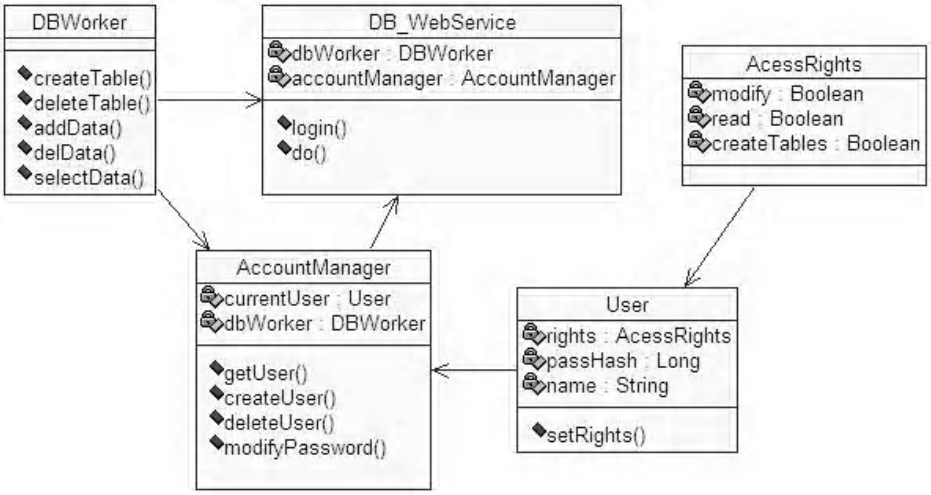


Рис. 2. UML-диаграмма классов веб-сервиса

В соответствии с разработанной структурой, с клиентской стороны доступны два метода веб-сервиса (их прототипы и аннотации представлены в WSDL-спецификации): int login(string name, string pass), возвращающий клиенту идентификатор сессии для работы с базой данных (по умолчанию, сессия действительна в течение 20 минут от момента логина); string do(string action, object[] data) – служит для выполнения действия action с базой данных (перечень доступных действий, а также их описание доступны в WSDL-аннотации) и набором данных data. В результате работы метод возвращает данные в виде xml-строки - ответа СУБД. Регистрация пользователя в системе осуществляется также либо с применением этого метода, либо локально (на стороне веб-сервиса).

Основной класс веб-сервиса (DB_WebService, наследник класса WebService) также использует для своей работы агрегированные объекты-экземпляры вспомогательных классов: менеджера работы с аккаунтами (AccountManager), а также объект для доступа к БД через отсоединенные источники данных (DBWorker).

5. Выводы

По причине существующей проблемы взаимодействия клиентских приложений с удалёнными СУБД была проанализирована предметная область и поставлена задача разработать веб-сервис для удалённой работы с базами данных. Веб-сервис реализован на платформе ASP .NET, работает по протоколу SOAP XML и удовлетворяет основным требованиям, обозначенным при рассмотрении задачи: простота взаимодействия, платформонезависимость и безопасность.

Одним из направлений дальнейших исследований является разработка веб-сервиса, позволяющего организовывать двухстороннее взаимодействие: как с клиентами на разных платформах – с одной стороны, так и с различными СУБД – с другой.

Список литературы: **1.** *Фуфаев Э.В.* и др. Разработка и эксплуатация удалённых баз данных. М.: Академия, 2008. 256с. 2. *Шапошников И.* Web-сервисы Microsoft .NET. СПб.: БХВ-Петербург, 2002. 334с. 3. *Хендерсон К.*. Microsoft SQL Server: структура и реализация. Профессиональное руководство. М.: Вильямс, 2005. 1056с. 4. *Шилдт Г.* Полный справочник по C#: Пер. с англ. М.: Вильямс, 2004. 752с.

Поступила в редколлегию 26.05.2010

Корниловский Артём Викторович, студент специальности интеллектуальные системы принятия решений, факультет КН, ХНУРЭ. Адрес: Украина, 61019, Харьков, пр. Ильича, 103а, кв. 40, тел. 376-27-27.

Губин Вадим Александрович, ст. преподаватель кафедры искусственного интеллекта ХНУРЭ. Научные интересы: интернет-технологии, интеллектуальный анализ данных. Адрес: Украина, 61054, Харьков, ул. Гв.-Широнинцев, 23, кв. 287, тел. 710-64-12.

УДК 519.876.2

А.Я. КУЗЕМИН, М. В. ШТУКИН

ВЫБОР ОПТИМАЛЬНОГО МЕЖДУНАРОДНОГО СОТРУДНИЧЕСТВА НАУЧНЫХ ПОДРАЗДЕЛЕНИЙ И РАЗРАБОТЧИКОВ НА ОСНОВЕ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ КООПЕРАЦИИ

Рассматриваются основные подходы к построению модели системы поддержки принятия решений в области международной научной кооперации. Приводится сравнительный анализ имеющихся моделей инновационной деятельности ВУЗа.

1. Постановка проблемы в общем виде

С развитием рыночных отношений в научной деятельности обострился вопрос выбора новых форм инновационной деятельности, важное место среди которых заняла научная кооперация. Смена ориентации отечественной науки с военного направления на гражданское обеспечила открытость исследований, что позволяет осуществлять не только локальную, но и международную кооперацию. Постоянное сокращение государственного финансирования исследовательской деятельности также подталкивает разработчиков искать более эффективные формы инновационной деятельности, основательно подходить к принятию решений о кооперации и начале совместной исследовательской работы. Поэтому проблема выбора наиболее оптимального партнера для совместной работы является одной из главенствующих и мало формализованных в современной науке.

Целью исследования является: определение проблемы менеджмента инновационных проектов в университетах; анализ существующей модели управления инновационными проектами и выявление ее недостатков; формализация европейской модели менеджмента инновационных проектов; постановка задачи прогнозирования и оценки рисков и шансов реализации инновационного проекта в сформированной модели; формализация проблемы прогнозирования динамических процессов менеджмента инновационных процессов.

2. Анализ исследований и публикаций последних лет

Проблема кооперации в менеджменте инновационных проектов и разработок нова для Украины. Постановка этой проблемы связана с переходом научно-исследовательских учреждений к рыночной форме взаимоотношений и развитием коммуникационных технологий, обеспечивающих обмен информацией между партнерами. Н. Шелюбская отмечает, что развитие международной кооперации обеспечено также прозрачностью границ ЕС и процессам евроинтеграции [1]. На сегодняшний момент проблемы международной научной кооперации исследуются в основном в направлениях классификации возможных видов такого сотрудничества и определения влияния государственных программ [1–3]

Л.И. Федулова в своей работе проводит анализ экономических показателей научно-технологического сотрудничества Украины и России в целях выявления устойчивых тенденций развития кооперации и определения влияющих макроэкономических и политических параметров[2].

В статье [3] рассматриваются проблемы прогнозирования инновационной деятельности применительно к промышленному предприятию. Данная работа не содержит математического аппарата прогнозирования.

В работе Р.В. Приходько [4] рассматриваются возможности сетевой научно-производственной кооперации ВУЗов и промышленных предприятий. Автор дает классификацию возможных форм научно-производственной кооперации.

А.Б. Сливицкий [5] большое внимание уделяет вопросу обмена информацией между партнерами в научной кооперации. Автор предлагает создание и развитие единого информационного пространства в рамках международного научного сотрудничества.

Во всех рассмотренных работах отсутствует аппарат формализации проблемы, а также какие-либо четкие алгоритмы принятия решений в сфере инновационной коопера-

ции. Отсутствие математического аппарата не позволяет иметь количественные оценки той или иной формы инновационного сотрудничества, что негативно сказывается на качестве принимаемых решений.

Анализ текущего состояния менеджмента инновационных проектов в ХНУРЭ

Современная украинская практика менеджмента инновационных проектов построена на модели планового подхода, где конечной производственной единицей (исследователь) является минимальная организационная единица ВУЗа – кафедра, лаборатория, а административный аппарат ВУЗа имеет императивные полномочия по отношению к любым проявлениям активности разработчика. Постановка задачи разработки спускается «сверху вниз», таким образом лишая разработчика возможности проявить инициативу. Данная модель менеджмента инновационных проектов рассчитана в основном на государственное финансирование и довольно редко приводит к интересу сторонних инвесторов.

Рассмотрим более детально такую модель на примере Харьковского национального университета радиоэлектроники. Для построения модели использовались методология ARIS и ПО ArisExpress.

Организационная диаграмма осуществляющих менеджмент инновационных проектов структурных подразделений ХНУРЭ представлена на рис 1.

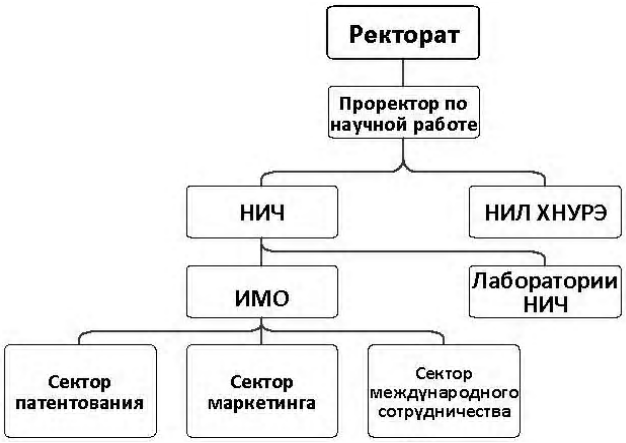


Рис. 1. Организационная диаграмма осуществляющих менеджмент инновационных проектов структурных подразделений ХНУРЭ в среде Aris

В данной организационной диаграмме конечными разработчиками являются научно-исследовательские лаборатории ХНУРЭ (на диаграмме – НИЛ ХНУРЭ) и лаборатории научно-исследовательской части (на диаграмме – лаборатории НИЧ).

Существующая модель бизнес-процесса менеджмента инновационных проектов в ХНУРЭ представлена на диаграмме бизнес-процессов ARIS (рис. 2).

На диаграмме существующей модели менеджмента инновационных проектов в ХНУРЭ видно, что финансовое обеспечение инновационной деятельности может происходить либо за счет государственного бюджета, либо за счет работы с партнерами и инвесторами, поиск которых происходит уже после осуществления самой научной деятельности и получения готовой разработки. Таким образом, на этапе постановки задачи исследования не учитываются и не могут учитываться конкретные потребности партнеров и инвесторов. В такой модели постановка задачи может совпасть с потребностями партнеров и инвесторов только случайно, при этом возможна длительная и выгодная кооперация.

В большинстве случаев «угадать» потребности инвестора невозможно, поэтому разработки, осуществленные в рассмотренной модели менеджмента инновационных проектов, не являются успешными.

Более успешной, с точки зрения источников финансирования, является европейская модель менеджмента инновационных проектов, в которой основным событием есть наличие коммерческого спроса на соответствующую разработку. Таким образом, дальнейшие технологические операции инновационного не являются рискованными с точки зрения их востребованности в будущем. “В сфере инновационной деятельности наиболее перспективной считается модель кооперации” [2].

Диаграмма бизнес-процессов новой модели менеджмента инновационных проектов в среде ARIS представлена на рис. 3.

В европейской модели менеджмента инновационных технологий основной инициативной единицей является исследователь (кафедра, лаборатория). Остальные организационные структуры, в том числе административный аппарат ВУЗа являются вторичными, осуществляющими вспомогательные функции элементами.

В большинстве случаев исследователь не может самостоятельно выполнять сложный научный проект. Основными проблемами исследователя являются недостаток ресурсов, а именно: 1) отсутствие необходимой научной базы; 2) отсутствие необходимой производственной базы; 3) нехватка финансовых ресурсов и т. п. Решением этих проблем является совместная инновационная деятельность с партнерами, которые компенсируют недостающую ресурсную базу.

Поиск партнеров, естественно, должен основываться на совпадении научных интересов. Тем не менее, после проведения активного поиска может обнаружиться достаточное количество возможных партнеров, у которых научные интересы совпадают с интересами исследователя. В этом случае актуальной становится проблема оценки возможного партнерства и выбора наиболее оптимальной альтернативы инновационной кооперации.

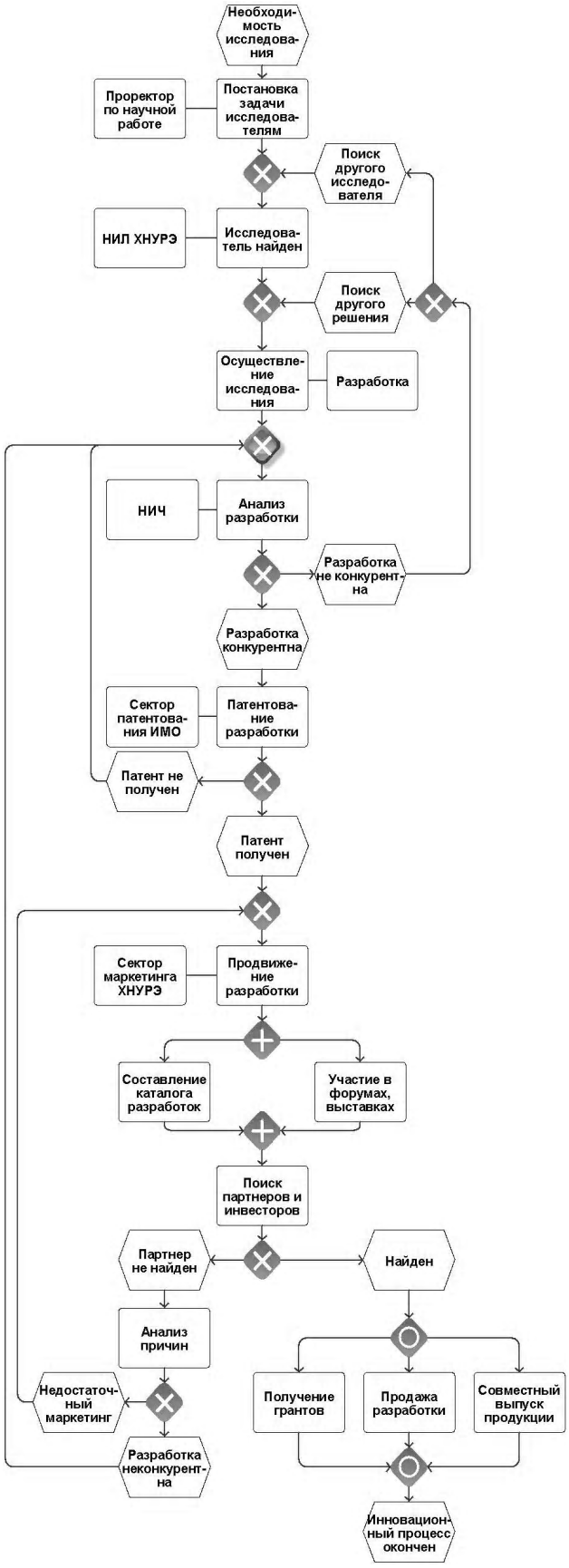


Рис. 2. Диаграмма бизнес-процесса менеджмента инновационных технологий в ХНУРЭ

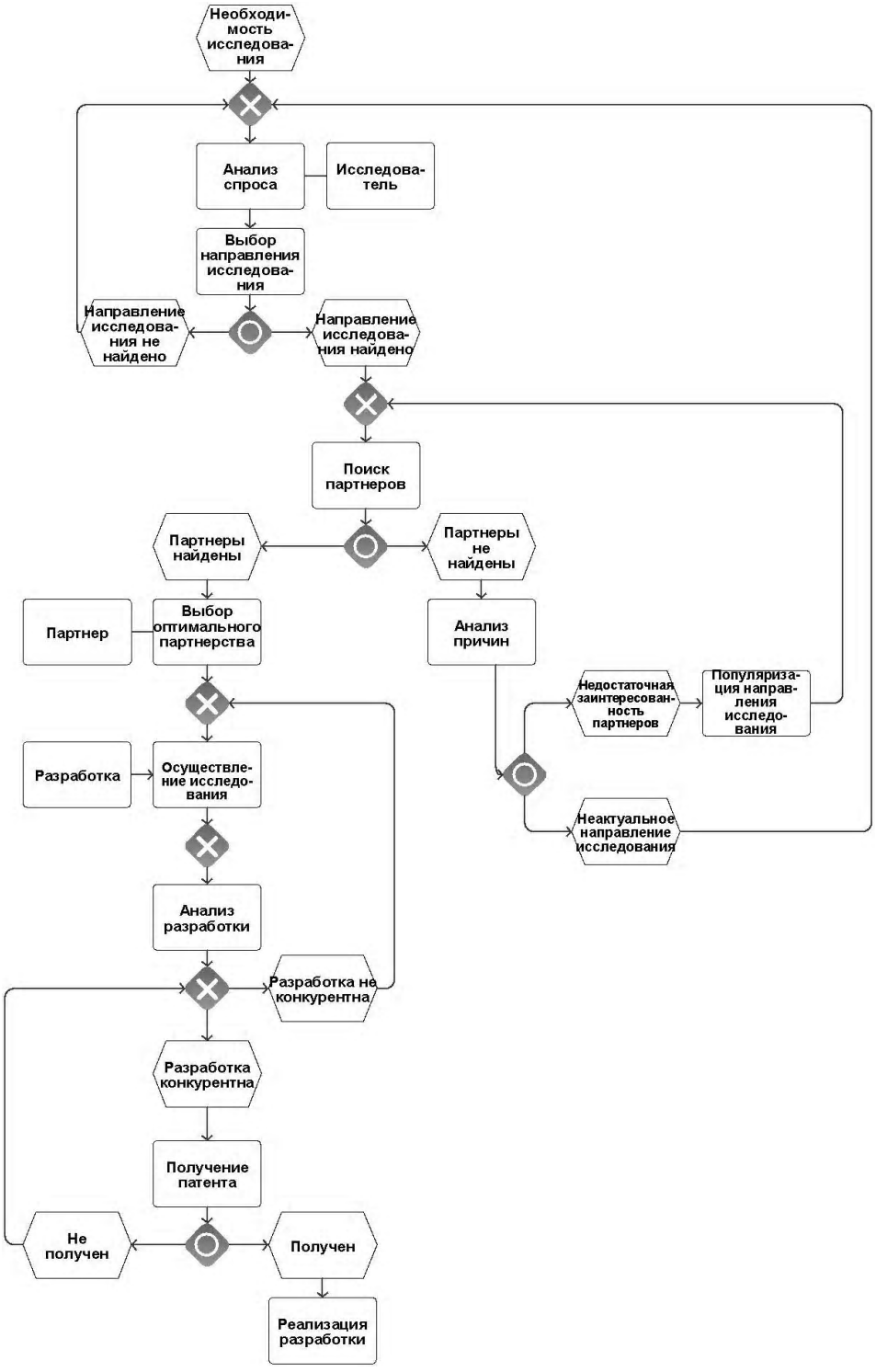


Рис 3. Диаграмма бизнес-процесса европейской модели менеджмента инновационных технологий

4. Проблема прогнозирования партнерства

Вопрос поиска спроса на разработки, а также поиска потенциальных партнеров интересен, но трудно формализуем. Поэтому оставим изучение этого вопроса для дальнейших исследований.

Многие исследователи, проведя поиск направлений исследований, пользующихся спросом, и поиск партнеров имеют в своем распоряжении несколько возможных направлений для исследований и несколько потенциальных партнеров.

Любой инновационный процесс в кооперации подразумевает ряд затрат партнеров:
– финансовые затраты;
– затраты трудовых ресурсов.

Суть любой научной деятельности заключается в процессе исследования, положительный результат от которого нельзя гарантировать. Современное украинское законодательство даже выделяет отдельно договоры на выполнение научно-исследовательских работ, где частично снимается ответственность подрядчика за результаты работы. Аналогичные нормы существуют и в других странах. Поэтому в любом инновационном процессе указанные затраты партнеров могут быть потрачены без достижения ожидаемых результатов. Для минимизации рисков таких потерь необходимо производить прогнозирование инновационного партнерства.

Важной проблемой является выбор тех направлений для исследования и тех партнеров, от работы с которыми будет наилучший результат. Данный выбор является достаточно сложным в силу того, что на результаты научной кооперации влияет огромное количество факторов, которые можно условно классифицировать следующим образом: жесткие факторы (не зависящие от партнеров; зависящие от партнеров); мягкие факторы.

Все указанные факторы являются изменяемыми во времени.

Кроме того, для выбора наилучшей альтернативы из имеющихся направлений исследований и партнеров необходимо формализовать понятие наилучшего результата, которого необходимо добиться от кооперации.

5. Обзор существующих методов планирования и оценки международной научной кооперации

На сегодняшний момент планирование и оценка рисков научной кооперации представлены в основном методами, основанными на формировании отношения экспертов на базе качественного анализа параметров кооперации. Это в основном: организация встреч; SWOT-анализ; метод критического пути.

Сравнительный анализ этих методов представлен в таблице.

Сравнительный анализ методов планирования и оценки международной научной кооперации

Название метода	Цель	Входная информация	Результаты	Недостатки метода
Организация встреч	Получение начальных сведений о кооперации	- информация о кооперации, предоставленная партнером; - эмоциональный опыт	- качественная информация о кооперации; - эмоциональное отношение к партнеру	- эмоциональная заангажированность; - невозможность формализации; - невозможность количественной оценки
SWOT-анализ	Выявление сильных и слабых сторон, возможностей и угроз кооперации	- качественная информация о кооперации	- перечень сильных и слабых сторон кооперации; - перечень возможностей и угроз кооперации	- невозможность формализации; - невозможность количественной оценки; - не учитывается временная динамика исследовательского процесса
Метод критического пути	Контроль сроков кооперации	- информация о сроках выполнения каждой части исследования; - информация о последовательности частей исследования и их взаимосвязи	- критические части кооперации; - срок выполнения исследования	- не учитываются риски кооперации; - невозможность формализации; - невозможность количественной оценки

Как видно из таблицы, все перечисленные методы имеют два основных недостатка: невозможность формализации и невозможность количественной оценки. Данное исследование посвящено преодолению указанных недостатков.

6. Выбор оптимального партнерства на основе прогнозирования изменения параметров партнеров во времени

Ограничимся задачей выбора наилучшей альтернативы из *N* партнеров на основе прогнозирования динамического развития кооперации. Для упрощения модели будем считать, что каждый партнер может осуществлять только одно направление исследования. Также будем осуществлять моделирование и прогнозирование сотрудничества только двух партнеров. Задача состоит в разработке метода выбора наилучшей альтернативы среди партнеров для исследователя. В данной статье исследователем называется тот партнер, который будет производить выбор альтернатив. Партнерами называются возможные варианты кооперации для исследователя.

Наилучшим или ожидаемым результатом является набор результатов, которых нужно достичь. К ним относятся результаты:

– финансовые (finr);

– научные (scir);

– социальные (socr).

Экспертную оценку каждого из указанных результатов может дать сам партнер. Ожидаемые результаты можно представить в виде вектора

$$R[finr, scir, socr]. \quad (1)$$

Векторы ожидаемых результатов обеих сторон (исследователя и партнера) различные. Таким образом, выделяются два вида ожидаемых результатов: ожидаемые результаты исследователя $R_{researcher}$ и множество ожидаемых результатов партнеровRP={$R_{partner1,}$…, $R_{partnerN}$}.

Векторы $R_{researcher}$ и $R_{partneri}$, где $i \in \overline{1, N}$ являются неизменными во времени. Экспертную оценку координат вектора $R_{researcher}$ получить несложно, так как экспертом здесь выступит сам исследователь. Достоверную же оценку координат векторов ожидаемых результатов множества RP можно получить только опросив каждого из партнеров. Такой опрос не всегда возможен по ряду причин:

– партнер может скрывать свои истинные ожидаемые результаты, чтобы показаться более привлекательным для кооперации;

– опрос партнера может быть невозможен по этическим соображениям;

– опрос множества партнеров может требовать большого расхода трудовых ресурсов.

Экспертную оценку координат векторов ожидаемых результатов партнеров $R_{partneri}$ может дать сам исследователь. Такие оценки, естественно, будут отличаться от истинных их значений, поэтому возникает необходимость введения для каждой координаты $R_{partneri}$ соответствующего коэффициента погрешности оценки б, который также должен указать исследователь. Таким образом, введем для каждого партнера вектор погрешностей оценки ожидаемых результатов $Ar_i[\alpha r_{finr}, \alpha r_{scir}, \alpha r_{socr}]$. Значение координаты scir вектора $R_{partneri}$находится в интервале [scri-scir* αr_{scir}; scri-scir* αr_{scir}]. Аналогично определяются интервалы значений других координат каждого из партнеров.

Также исследователю необходимо задать временные ограничения – срок предполагаемой кооперации T. В любой момент времени $t \in \overline{1, T}$ существует множество векторов имеющихся результатов исследователя RN_t={$Rn_{researcher\ 1t}$, $Rn_{researcherNt}$}. Каждый вектор $Rn_{researcherit}$ содержит координаты вектора имеющихся результатов для кооперации с i-м партнером. Оптимальной будет та альтернатива, для которой

$$\min(|\, Rn_{researcheriT} - Rresearcher\,|). \quad (2)$$

И множество векторов имеющихся результатов партнеров RN={$Rn_{partner\ 1}$, … , $Rn_{partnerN}$].

В каждый момент времени предполагаемой кооперации исследователь и каждый партнер обладают определенным набором параметров p_j, где $j \in \overline{1, M}$, M – количество рассматриваемых параметров, находящихся в постоянной динамической зависимости. В каждый определенный момент времени эти параметры для исследователя можно представить в виде вектора $P_{rest}[p_1, \dots, p_M]$. Исследователь дает экспертную оценку координат вектора $P_{resto}[p_1, \dots, p_M]$ в нулевой момент времени.

Для каждого из партнеров в каждый момент времени также определяется вектор параметров $P_{partit}[p_1, \dots, p_M]$. Исследователь дает экспертную оценку координат векторов $P_{partito}[p_1, \dots, p_M]$ в нулевой момент времени, а также векторы коэффициентов погрешностей $Ap_i[\alpha p_1, \dots, \alpha p_M]$.

Координаты векторов P_{rest} и P_{partit} определяются функциями f_{res} и f_{part}:

$$\begin{cases} P_{rest} = f_{res}(P_{rest_0}, P_{partit_0}, Ap_{it_0}, Rn_{partnerit}, t), \\ P_{partt} = f_{part}(P_{partit_0}, Ap_{it_0}, P_{rest_0} Rn_{partnerit}, t). \end{cases} \quad (3)$$

Определение функций f'_{res} и f'_{part}, а также решение системы уравнений позволяет осуществить прогнозирование изменения параметров партнеров во времени.

Для решения указанной системы уравнений предлагается использовать системы дифференциальных уравнений. Такие уравнения применяются при моделировании динамических процессов в термодинамике.

Однако определение в любой момент времени векторов параметров P_{rest} и P_{partit} еще не дает достаточно информации для принятия решения относительно самой оптимальной альтернативы предполагаемого партнерства.

Изменение параметров разработчика и партнеров естественным образом приводит к изменениям результатов кооперации. Таким образом, векторы параметров P_{rest} и P_{partit} обуславливают в любой момент времени соответствующие векторы достигнутых результатов. Иными словами, существует функция $res(P)$, множеством значений которой являются векторы достигнутых результатов.

Полученный метод позволяет моделировать динамические изменения кооперации двух научно-исследовательских подразделений. На основании количественных результатов прогнозирования (векторы параметров и векторы достигнутых результатов) возможно принятие оптимального партнерства в исследовательской деятельности.

7. Выводы и перспективы дальнейших разработок

Рассмотренная проблема выбора наиболее оптимального партнерства является базовой для всей последующей научной деятельности. Применение математического аппарата для формализации параметров возможных партнеров позволило применить математические методы прогнозирования во времени динамики развития партнерства.

Дальнейшие исследования стоит вести в двух направлениях:

– автоматизация поиска партнеров, с которыми возможна кооперация;

– усовершенствование математического аппарата прогнозирования кооперации и применение других методов прогнозирования для решения поставленной задачи.

Список литературы: 1. *Шелюбская Н.* Глобализация и региональная кооперация в сфере НИОКР. Электронный ресурс. Режим доступа Интернет: http://www.inti.kz/develop/document/f14.pdf. **2.** *Федулова Л.И.* Состояние и перспективы инновационно-технологического взаимодействия Украины и России: потенциал Украины. Электронный ресурс. Режим доступа Интернет: http://institutiones.com/innovations/1556-innovacionno-texnologicheskoe-vzaimodejstvie.html. **3.** *Шамина Л.К.* Особенности прогнозирования инновационной деятельности на промышленном предприятии. Электронный ресурс. Режим доступа Интернет: http://www.ibl.ru/konf/061207/18.html. **4.** *Приходько Р.В.* Сетевая научно-производственная кооперация высших учебных заведений и промышленных предприятий. Электронный ресурс. Режим доступа Интернет: http://economics.open-mechanics.com/articles/140.pdf. **5.** *Сливицкий А. Б.* Интеллектуализация научно-производственной кооперации как фактор инновационного развития // Материалы международного симпозиума «Актуальные проблемы научно-технической и инновационной политики в контексте формирования общеевропейского научного пространства: опыт и перспективы». Киев, 16-17 июня 2010 г. С. 293-295.

Поступила в редколлегию 16.05.2010

Кузёмин Александр Яковлевич, д-р техн. наук, проф. кафедры информатики, начальник инновационно-маркетингового отдела ХНУРЭ. Научные интересы: управление рисками, геоинформационные системы. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел.: 8(057)702-15-15,e-mail:kuzy@kture.kharkov.ua

Штукин Михаил Викторович, аспирант кафедры информатики ХНУРЭ. Научные интересы: системный анализ данных, управление проектами. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. (057) 702-15-15, e-mail: shtumi@tut.by

УДК 658.012.102

А.А. ФЕДОРОВ, Ю.В. ЛОПУХИН, А.Ю. СКОБЛИКОВ

ЗАДАНИЕ МЕТРИКИ В ЗАДАЧАХ КЛАССИФИКАЦИИ ОБЪЕКТОВ РАЗЛИЧНОЙ ПРИРОДЫ

Рассматривается вопрос задания меры близости при классификации объектов различной природы. Проводится анализ эффективности использования евклидовой метрики в задачах классификации объектов различной природы. Рекомендуется для определения степени сходства объектов вместо коэффициентов сходства Рао, Хаммана, Дейка, Танимото использовать меру близости. Рассматривается пример практического применения меры близости для количественного определения степени сходства объектов.

1. Постановка задачи

Целью исследования является разработка меры близости для объектов, заданных числовым вектором. Классификации объектов различной природы, как правило, выполняется с помощью ЭВМ, что требует наличия четкого и достаточно простого алгоритма. В научных и прикладных сферах при классификации объектов или измерений используют коэффициенты сходства различных исследователей Рао, Хаммана, Дейка, Танимото [1,4]. Оперировать с коэффициентами сходства несложно, но эффективней применять меру близости [3,4]. Для решения конкретных задач классификации, чтобы определить, являются ли два объекта близкими между собой, необходимо дать количественное определение меры близости. Это достигается введением функции, измеряющей близость на множестве рассматриваемых объектов или измерений. Понятие близости является одним из основных в таких задачах и поэтому требует не интуитивного представления, а математически корректного.

2. Выбор меры близости классифицируемых объектов

Наиболее употребительной в настоящее время является евклидова мера, хотя она имеет существенный недостаток – не учитывает возможной неравномерности осей пространства. Обобщением евклидовой метрики является мера Махаланобиса, которая инвариантна относительно аффинных преобразований

$$d = \{(X_i - X_j)^T W^{-1} (X_i - X_j)\}S, \quad (1)$$

где W^{-1} – матрица, обратная матрице рассеяния; X_i, X_j – числовые векторы измерений признаков, характеризующие соответственно i-й и j-й элементы множества объектов.

Выбор меры близости в значительной степени зависит от особенностей классифицируемых объектов. Так, для рассматриваемого в [2] множества элементов $X = \{X_i\}$, характеризующихся структурой отношений

$$X_i \cap X_j \neq \varnothing, X_i \notin X_j, |X_i| \neq |X_j|, i \neq j, \quad (2)$$

$$X_i = \{g_{ik}\}, g_{ik} \in \{0,1\}, i, j = \overline{1,n}, k = \overline{1,m},$$

в качестве меры близости использовалось выражение на основе коэффициента сходства Рао:

$$d_1 = 1 - \frac{|X_i \cap X_j|}{|X_i \cup X_j|}. \quad (3)$$

С точки зрения практических приложений для рассматриваемого выше множества элементов X, признаки которых являются двоичными переменными, могут оказаться полезными следующие метрики:

$$d_2 = 1 - \frac{|X_i \cap X_j|}{|X_i| + |X_j|}; \quad (4)$$

$$d_3 = 1 - \frac{2\left|X_i \cap X_j\right|}{\left|X_i\right| + \left|X_j\right|}. \tag{5}$$

Для общего случая, когда $g_{ip} \in \{0, 1, 2, \ldots, k\}$, в качестве меры для группирования можно использовать выражение

$$d_{ij} = 1 - \frac{\sum_{p=1}^{m} \alpha_{ij}^{p}}{\left|X_i\right| + \left|X_j\right|}, \tag{6}$$

где $\alpha_{ij}^{p} = \begin{cases} 0, & \text{если} \quad g_{ip} g_{jp} = 0, \\ g_{ip} + g, & \text{если} \quad g_{ip} g_{jp} \neq 0. \end{cases}$

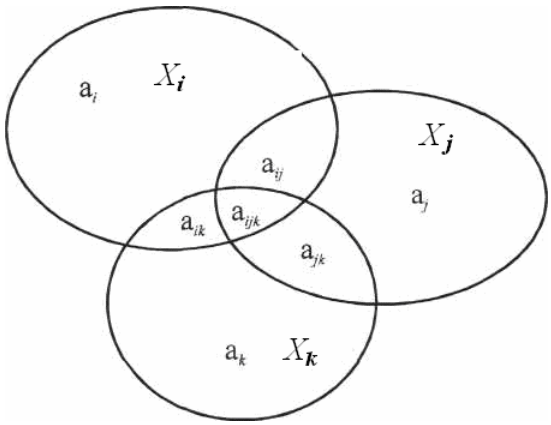
Чтобы выражение (6) использовалось в качестве меры близости, необходимо проверить выполнение аксиом Фреше.

Так как для любой пары $X_i X_j$ справедливо (2), то очевидно, что $0 \le d_{ij} \le 1$, $d_{ij} = d_{ji}$.

Необходимо проверить справедливость аксиомы треугольника

$$1 - \frac{\left|X_i \cap X_j\right|}{\left|X_i \cup X_j\right|} \le 1 - \frac{\left|X_i \cap X_k\right|}{\left|X_i \cup X_k\right|} + 1 - \frac{\left|X_k \cap X_j\right|}{\left|X_k \cup X_j\right|}. \tag{7}$$

Для проверки выполнения аксиомы треугольника воспользуемся рисунком взаимных пересечений множества конструктивно-технологических признаков, характеризующих изделия X_i, X_j, X_k.:



Обозначим взаимные пересечения множества признаков, характерные для объектов, представленных на рисунке:

$$a_i = \left| X_i \setminus [(X_i \cap X_j) \cup (X_i \cap X_k)] \right|, \tag{8}$$

$$a_j = \left| X_j \setminus [(X_j \cap X_i) \cup (X_j \cap X_k)] \right|, \tag{9}$$

$$a_k = \left| X_k \setminus [(X_k \cap X_i) \cup (X_k \cap X_j)] \right|, \tag{10}$$

где a_i , a_j a_k – признаки, присущие соответственно только i-му, j-му, k-му объекту:

$$a_{ij} = \left| X_i \cap X_j \setminus [(X_i \cap X_j \cap X_k)] \right|, \tag{11}$$

здесь a_{ij} – признаки, одновременно присущие i-му и j-му объектам:

$$a_{ik} = \left| X_i \cap X_k \setminus [(X_i \cap X_j \cap X_k)] \right|, \tag{12}$$

a_{ik} – признаки, одновременно присущие i-му и k-му объектам:

$$a_{jk} = \left| X_j \cap X_k \setminus [(X_i \cap X_j \cap X_k)] \right|, \tag{13}$$

где a_{jk} – признаки, одновременно присущие j-му и k-му объектам:

$$a_{ijk} = | X_i \cap X_j \cap X_k |, \tag{14}$$

a_{ijk} – признаки, одновременно присущие i-му, j-му и k-му объектам:

$$X = | X_i \cup X_j \cup X_k | = a_i + a_j + a_k + a_{ij} + a_{ik} + a_{jk} + a_{ijk}. \tag{15}$$

Согласно взаимным пересечениям, из рисунка и с учетом выражений (8)-(15) неравенство (7) примет вид:

$$\frac{a_{ij} + a_{ijk}}{X - a_k} + \frac{a_{ik} + a_{ijk}}{X - a_j} - \frac{a_{jk} + a_{ijk}}{X - a_i} \le 1. \tag{16}$$

3. Исследование выполнимости меры близости

Для проверки выполнимости неравенства (16) воспользуемся теоремой о необходимых условиях экстремума функции, заданной в виде неравенства [5].

Обозначим:

$$U = \frac{a_{ij} + a_{ijk}}{X - a_k} + \frac{a_{ik} + a_{ijk}}{X - a_j} - \frac{a_{jk} + a_{ijk}}{X - a_i}. \tag{17}$$

Составим функцию Лагранжа

$$F = -\lambda_0 \left(\frac{a_{ij} + a_{ijk}}{X - a_k} + \frac{a_{ik} + a_{ijk}}{X - a_j} - \frac{a_{jk} + a_{ijk}}{X - a_i}\right) - \lambda\left(\sum_{\bar{e}} a_{\bar{e}}^{-} - X\right) - \sum_{\bar{e}} \lambda_{\bar{e}} a_{\bar{e}}^{-}, \ \bar{e} = \{i, j, k, ij, ik, jk, ijk\}, \tag{18}$$

где λ_0, λ и $\lambda_{\bar{e}}$ – множители Лагранжа, согласно [5] не все равны нулю, при условии, что

$$\sum_{\bar{e}} a_{\bar{e}}^{-} - X = 0, \tag{19}$$

$$a_{\bar{e}}^{-} \ge 0. \tag{20}$$

Так как ограничения (19) линейны, то из [3] следует, что

$$\lambda_0 = 1. \tag{21}$$

Тогда (18) будет иметь вид:

$$F = \frac{a_{jk} + a_{ijk}}{X - a_i} - \frac{a_{ij} + a_{ijk}}{X - a_k} - \frac{a_{ik} + a_{ijk}}{X - a_j} - \lambda\left(\sum_{\bar{e}} (a_{\bar{e}}^{-} - X) - \sum_{\bar{e}} \lambda_{\bar{e}}^{-} a_{\bar{e}}^{-}\right). \tag{22}$$

Продифференцируем (22) по a_{ij}, a_{ik}, a_{kj}, a_{ijk}, a_k, a_j, a_i, X и приравняем производные нулю:

$$-\frac{1}{X - a_k} - \lambda - \lambda_{ij} = 0, \tag{23}$$

$$-\frac{1}{X - a_j} - \lambda - \lambda_{ik} = 0, \tag{24}$$

$$\frac{1}{X - a_i} - \lambda - \lambda_{kj} = 0, \tag{25}$$

$$-\frac{a_{ij} + a_{ijk}}{(X - a_k)^2} - \lambda - \lambda_k = 0, \tag{26}$$

$$-\frac{1}{X - a_k} - \frac{1}{X - a_j} + \frac{1}{X - a_i} - \lambda - \lambda_{ijk} = 0, \tag{27}$$

$$-\frac{a_{ij}}{(X - a_k)^2} - \lambda - \lambda_k = 0, \tag{28}$$

$$-\frac{a_{ik} - a_{ijk}}{(X - a_j)^2} - \lambda - \lambda_j = 0, \tag{29}$$

$$-\frac{a_{kj}+a_{ijk}}{(X-a_i)^2}-\lambda-\lambda_i=0, \tag{30}$$

$$-\frac{a_{ij}+a_{ijk}}{(X-a_k)^2}-\frac{a_{ik}+a_{ijk}}{(X-a_j)^2}+\frac{a_{kj}+a_{ijk}}{(X-a_i)^2}+\lambda=0. \tag{31}$$

Из (23) видно, что

$$\frac{1}{X-a_k}>0,\ \lambda_{ij}\geq 0, \tag{32}$$

следовательно,

$$\lambda<0. \tag{33}$$

Из (25) и (33) следует, что

$$\lambda_{kj}>0,\ a_{kj}=0. \tag{34}$$

Из (30) и (34) следует, что

$$\lambda_i>0,\ a_i=0. \tag{35}$$

Предположим, будто бы

$$a_j>0,\ \lambda_j=0. \tag{36}$$

Тогда с учетом (21) выражение (27) примет вид

$$\lambda_{ij}-\frac{a_j}{X(X-a_j)}-\lambda_{ijk}=0. \tag{37}$$

Так как

$$\lambda_{ijk}\geq 0, \tag{38}$$

то из (37) следует

$$\lambda_{ij}>0,\ a_{ijk}=0. \tag{39}$$

Из (29) и (37) выводим

$$\lambda=-\frac{a_{ik}+a_{ijk}}{(X-a_j)^2}. \tag{40}$$

Из (31) в соответствии с (34), (39), (40) получаем

$$\frac{a_{ijk}}{(X-a_k)^2}-\frac{a_{ijk}}{X^2}=0 \tag{41}$$

или

$$a_{ijk}a_k=0, \tag{42}$$

если $a_{ijk}=0$, то из (28) и (37) следует

$$\lambda_k=-\lambda>0. \tag{43}$$

Тогда из (42) получается, что

$$a_k=0. \tag{44}$$

Таким образом, имеем

$$a_k=a_i=0,\ a_{kj}=a_{ij}=0. \tag{45}$$

При учете же (15) и (45) выражение (16) примет вид

$$\frac{a_{ij}+a_{ijk}}{X-a_k}+\frac{a_{ik}+a_{ijk}}{X-a_j}-\frac{a_{jk}+a_{ijk}}{X-a_i}=\frac{a_{ik}+a_{ijk}}{X-a_j}=1. \tag{46}$$

Итак, функция (18) имеет безусловный максимум, равный единице. Исходя из (46) можно заключить, что выполняется неравенство (7). Следовательно, множество $X = \{X_i\}$, $i = \overline{1,n}$ с определенным выше расстоянием d_{ij} образует метрическое пространство.

Выводы. Предложены новые меры близости, отражающие естественные соотношения между сравниваемыми изделиями. Они характеризуются простой и ясной геометрической интерпретацией, а использование их обеспечивает исключительно четкое разделение. Эти метрики прошли апробацию в задачах планирования приборостроительного производства. Приведенные метрики могут быть использованы при анализе и синтезе структур сложных систем различной природы (технических, экономических, социологических и др.).

Список литературы: 1. *Боннер Р.Е.* Некоторые методы классификации. В кн.: Автоматический анализ сложных изображений. М.: Мир, 1969. 273 с. **2**. *Салыга В. И., Федоров А. А*. Модель текущей специализации в задаче распределения квартальной программы //Электротехническая промышленность. 1977. Вып. 8 (454). С. 23-25 **3.** *Федоров А.А., Федоров М. А.* Об одной мере близости экономических объектов, описываемых числовым вектором // Вестник ХГПИ. **4**. *Федоров А. А*. Об одной мере близости объектов в признаковом пространстве // АСУ. Харьков, ХАИ. 1979. Вып. 2. С. 125-127. **5.** *Гаибова М.А.* Многокритериальная оптимизация инвестиционных проектов развития промышленных предприятий. Самара: ГУ, 2004. 137с. **6.** *Иваниенко В.В.* Управление эффективностью использования ресурсов производства. Харьков: Изд. ХНЭУ, 2005. 368 с.

Поступила в редколлегию 16.06.2010

Федоров Андрей Алексеевич, канд. техн. наук, доцент кафедры организации производства и управления персоналом НТУ «ХПИ». Научные интересы: разработка моделей производственных процессов, проблемы классификации. Адрес: Украина, 61002, Харьков, ул.Фрунзе, 21, тел. 707-68-56.

Лопухин Юрий Владимирович, ст. преподаватель кафедры АПВТ ХНУРЭ. Научные интересы: проектирование программного обеспечения, автоматизации проектирования цифровых устройств. Адрес: Украина, 61166, Харьков, пр.Ленина, 14, тел. 70-21-326.

Скобликов Алексей Юрьевич, асп. НИПКИ «Молния» НТУ «ХПИ» . Научные интересы: телекоммуникация. Адрес: Украина, 61002, Харьков, ул.Фрунзе, 21, тел. 707-68-56.

УДК 519.6

В.И. ГРИЦЮК

АНАЛИЗ МОДИФИЦИРОВАННЫХ РЕКУРРЕНТНЫХ АЛГОРИТМОВ ДЛЯ ИДЕНТИФИКАЦИИ СИСТЕМ, ИЗМЕНЯЮЩИХСЯ ВО ВРЕМЕНИ

Приводится анализ модифицированных рекуррентных алгоритмов наименьших квадратов для оценки параметров, изменяющихся во времени. Исследуются их свойства сходимости. Алгоритм с экспоненциальным забыванием и восстановлением пригоден для отслеживания параметров, изменяющихся во времени, имеет тот же порядок сложности, что и стандартный рекуррентный алгоритм наименьших квадратов, но улучшенные свойства сходимости. Представлены результаты моделирования, которые демонстрируют способность приведенных алгоритмов отслеживать изменяющиеся во времени параметры.

1. Введение

Адаптивное регулирование вызывает необходимость построения новых, исходящих из метода наименьших квадратов, рекуррентных алгоритмов, которые отслеживают изменяющиеся во времени параметры и справляются с влиянием неизмеряемых помех и немоделированной динамики.

Поэтому *актуальным* является построение алгоритмов, обладающих указанными свойствами, которое концентрируется на решении проблемы - при сохранении глобальной сходимости во время-инвариантном случае обеспечить неисчезающие элементы ковариационной матрицы P_k [1,2].

Цель исследования – сравнение модифицированных рекуррентных алгоритмов, их свойств сходимости, анализ результатов моделирования, подтверждающих свойства приведенных алгоритмов

2. Сходимость

Сравнивается алгоритм с постоянным следом (ПСА) и алгоритм с экспоненциальным забыванием и восстановлением (ЭЗВА). В первом при заданном следе определяется переменный фактор забывания:

$$\hat{\beta}_k = \hat{\beta}_{k-1} + \frac{\alpha_T P_{k-1}\phi_k}{1+\phi_k^T P_{k-1}\varphi_k}(y_k - \phi_k^T\hat{\beta}_{k-1}), \quad (1)$$

$$R_k = P_{k-1} - \frac{\alpha_T P_{k-1}\phi_k\phi_k^T P_{k-1}}{1+\phi_k^T P_{k-1}\phi_k}, \quad (2)$$

$$P_k = 1/\bar{\lambda}_k R_k, \quad (3)$$

где $\bar{\lambda}_k = \frac{trR_k}{trP_0}$.

Для исследования сходимости алгоритма рассматривается последовательность квадратов норм ошибок параметров [3]:

$$\tilde{\beta}_k = \beta_k - \hat{\beta}_k, \ \left\|\tilde{\beta}_k\right\|^2_{P_k^{-1}} = \tilde{\beta}_k^T P_k^{-1}\tilde{\beta}_k. \quad (4)$$

Можно доказать, что квадраты норм (4), возникающие из (1)-(3), образуют для каждого $\hat{\beta}_0 \in R^M$ невозрастающую монотонную последовательность:

$$\left\|\tilde{\beta}_k\right\|^2_{P_k^{-1}} - \left\|\tilde{\beta}_k\right\|^2_{P_{k-1}^{-1}} = -\frac{\bar{\lambda}_k\alpha_T\tilde{\beta}_{k-1}^T\phi_k\phi_k^T\tilde{\beta}_{k-1}}{1+\phi_k^T P_{k-1}\phi_k}.$$

В случае переменных во времени параметров, если теряется положительная определенность P_k, $\left\|P_k^{-1}\right\| = \infty$, сходимости может не произойти.

Для второго типа алгоритма с матрицей ковариаций

$$P_k = \frac{1}{\lambda}P_{k-1} - \frac{\alpha P_{k-1}\phi_k\phi_k^T P_{k-1}}{1+\phi_k^T P_{k-1}\phi_k} + \beta I - \delta P_{k-1}^2$$

выполняются такие условия: 1)экспоненциальное забывание и восстановление, 2)верхняя граница для P, т.е. ненулевая нижняя граница для P^{-1}, 3)верхняя граница для P^{-1}, т. е. ненулевая нижняя граница для P. Свойства сходимости могут быть обобщены для случая 3, изменяющегося во времени. Основанием для этого может служить условие, независимое от механизма генерирования данных.

3. Результаты моделирования

Для примера рассматривается модель скользящего среднего:

$$y_k = au_k + bu_{k-1}, \text{ где } u_k = \frac{1}{2}\left[1 + \operatorname{sgn}(\sin\frac{\pi}{400}k)\right].$$

Параметры таковы:

K	a	b
$[1,789]$	0.1	0.2
$[790,799]$	0.2	0.3

$[800,899]$	0.3	0.4	
$[900,999]$	0.4	0.5	и
$[1000,1099]$	0.5	0.6	
$[1100,1999]$	0.6	0.8	

$$\begin{cases} \beta_0 = [a\,b]^T, \\ \hat{\beta} = [\hat{a}\,\hat{b}]^T, \\ \phi_k = [u_k\,u_{k-1}]^T. \end{cases}$$

Результаты идентификации, использующие оба метода, приведены ниже.

На рис.1 и 2 сравнивается предсказанный выход, полученный с использованием ПСА и ЭЗВА соответственно, с выходом системы.

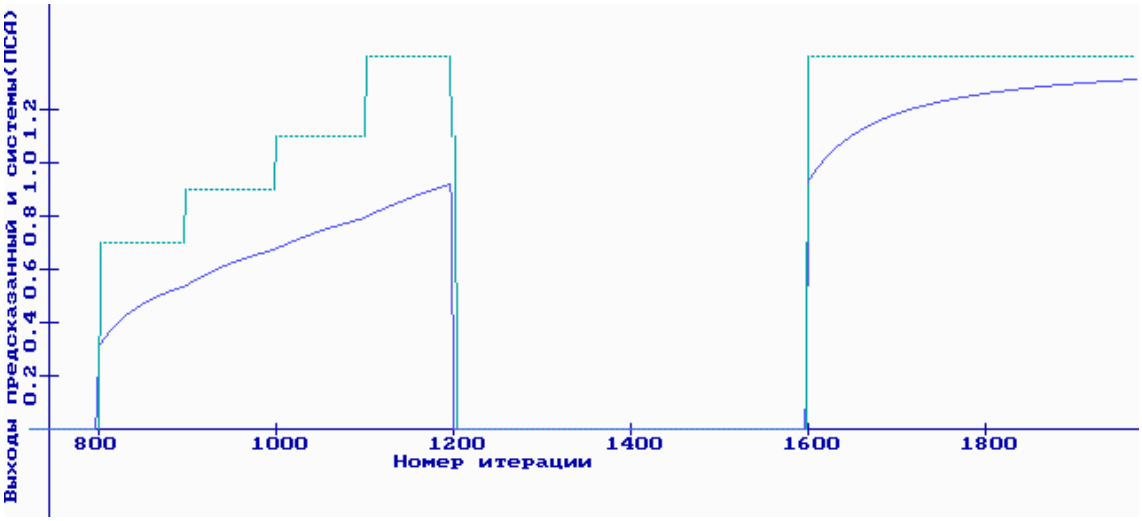


Рис.1. Предсказанный выход с использованием ПСА (____) в сравнении с выходом системы (. . .)

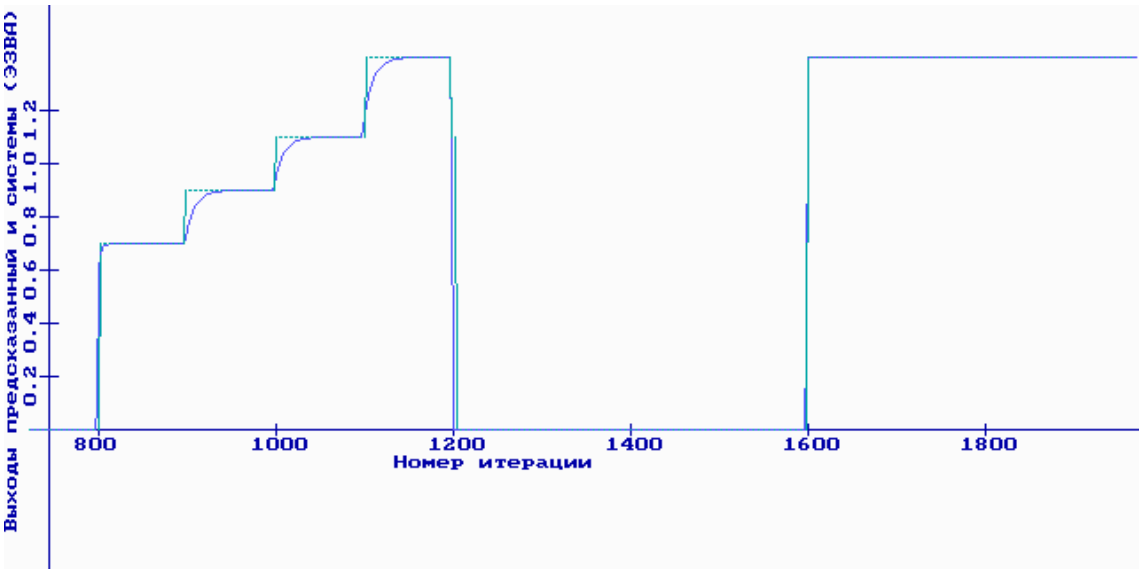


Рис. 2. Предсказанный выход с использованием ЭЗВА (____) в сравнении с выходом системы (. . .)

На рис.3 приводится сравнение корня из средне-квадратической ошибки предсказания $PE(\theta_k)$, полученной с использованием СТ алгоритма и ЭЗВА:

$$PE(\theta_k) \triangleq \left(\frac{1}{k}\sum_{i=1}^{k} e_i^2\right)^{1/2},$$

где $e_i = y_i - \phi_i^T \hat{\beta}_i$.

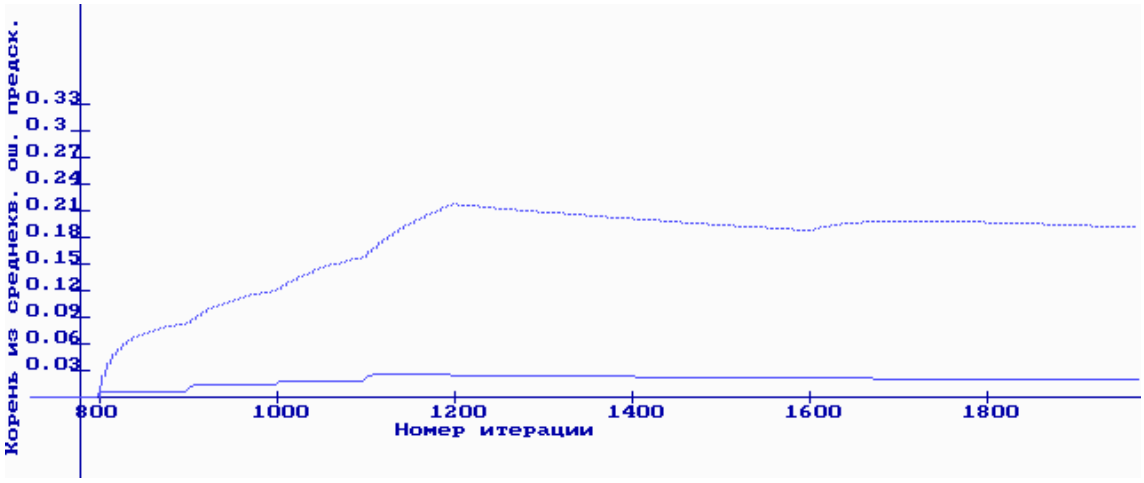


Рис. 3. Сравнение корня из среднеквадратической ошибки предсказания, полученной с использованием СТ алгоритма (. . .) и ЭЗВА (___)

На основе результатов , представленных на рис.1 и 2, можно заключить, что ЭЗВА производит быстрое оценивание (когда $\phi_k \neq 0$),благодаря его свойствам экспоненциального восстановления, и обеспечивает коррекцию быстрее, чем СТА (добавление единичной матрицы к обновленной P приводит к тому, что малое, но существенное усиление достижимо вдоль направления $[1 \quad 1]^T$ при k=900).

Заключение

ЭЗВА имеет тот же порядок сложности, что и рекуррентный алгоритм наименьших квадратов, но обладает улучшенными свойствами сходимости. На основе результатов моделирования демонстрируются свойства приведенных алгоритмов. Для увеличения точности предлагается сочетать этот метод с методом факторизации [2].

Список литературы: 1. *Halwass M.* “Least squares”- Modifikationen zur schatzung zeitvarianter parameter /Messen Stenern Regeln, 1990. 33. N1. P. 8-14. **2.** *Грицюк В. И.* Рекуррентная факторизованная идентификация динамических объектов // Прогр. и аннот. докл. Международной школы. Проектирование автоматизированных систем контроля и управления сложными объектами.Харьков, 1992. С. 10. **3.** *Googwin G. C., Hill D. J., Palaniswami M.* A perspective on convergence of adaptive control algorithms . Automatica. 20.1984. 5.P. 519-532.

Поступила в редколлегию 23.05.2010

Грицюк Вера Ильинична, канд. техн. наук, доцент кафедры СТ ХНУРЭ. Научные интересы: стохастические системы управления. Хобби : музыка, литература. Адрес:Украина, 61166, Харьков, пр. Ленина, 14, тел. 702 -10-06.

УДК 378.147

Н.А. БОНДАРЕНКО, В.И. ШЕХОВЦОВА, Е.А.ЧАСОВСКАЯ

МОДЕЛИРОВАНИЕ РАБОТЫ ФАЙЛОВОЙ СИСТЕМЫ В ОПЕРАЦИОННОЙ СИСТЕМЕ WINDOWS

Проводится анализ основных принципов работы файловой системы операционной системы Windows. Строится объектная модель работы файловой системы. Разрабатываются классы – TFAT («Таблица размещения файлов») и TFST («Таблица свободного пространства»), которые используются для составления программы моделирования реакции файловой системы на запросы прикладных программ по добавлению и исключению файлов.

Введение

Известно, что основное назначение компьютеров заключается в обработке и хранении данных. Поэтому одним из самых главных объектов в нем есть файлы. Файлы отличаются один от другого именами, размерами, структурой и носителем. Для работы с файлами в современных языках программирования предусмотрены разнообразные операции – открытие, закрытие, чтение, запись, создание, исключение и др. Все эти операции выполняются через *файловую систему*, которая является составной частью операционной системы (ОС).

Исторически первой ОС для персональных компьютеров фирмы IBM (или совместимых с ними) была система MS DOS (дисковая операционная система фирмы Microsoft). Именно в рамках этой ОС была разработана файловая система, которая и до сих пор работает в современных компьютерах. Файловая система новой ОС фирмы Microsoft – Windows работает на тех же принципах и алгоритмах, что и в MS DOS (за исключением ОС Windows NT).

Современные файловые системы строятся на принципах быстродействия, удобства и безопасности данных. Они имеют специальные средства защиты файлов, предотвращения их повреждения, возобновления поврежденных файлов, форматирования и дефрагментации дисков.

Современные программисты должны хорошо понимать принципы работы файловой системы в целях ее эффективного использования для рациональной организации своих файлов и повышения быстродействия прикладных программ.

Эта работа посвящена моделированию основных принципов работы файловой системы в операционной системе Windows. Конечно, данная работа не может охватить все аспекты работы файловой системы. В то же время она может дать представление о том, как работает файловая система.

1. Анализ задачи и разработка объектной модели

1.1. Анализ основных принципов работы файловой системы

Как известно, с аппаратной точки зрения жесткий диск является магнитным диском, на поверхности которого сохраняются определенным образом намагниченные участки. Диск вращается на большой скорости над устройством считывания/записи (головкой), которое превращает электромагнитные импульсы от намагниченных участков в двоичные данные. Последние попадают в буфер данных и становятся доступными сначала для файловой системы, а через нее – для прикладной программы. Запись данных происходит в обратном направлении.

Наименьший такой участок (т.е. наименьшее количество данных, которые считываются или записываются на жесткий диск) называется *кластером*. Таким образом, с логической точки зрения жесткий диск можно рассматривать как последовательную совокупность кластеров. В зависимости от емкости диска и аппаратных особенностей размер кластера составляет от 512 байт до 2 Кбайт.

Каждый кластер имеет свой адрес, т.е. свой порядковый номер в последовательности кластеров. Таким образом, при вращении диска над устройством считывания/записи (головкой) последовательно проходят кластеры от первого до последнего.

Для осуществления операции считывания/записи адрес кластера предварительно размещается в регистр адреса. Когда над головкой проходит кластер именно с этим адресом (номером), происходит считывание/запись из буфера данных. При этом сам жесткий диск (его аппаратные средства) не «знает», где и какие файлы на нем размещены. Его функция – запись и считывание данных из кластеров.

Таким образом, главная задача файловой системы (ФС) заключается в обеспечении записи/чтения файлов на диск по запросам программ, т.е. в размещении файлов в кластерах. Например, при образовании нового файла с определенным именем, ФС отвела ему кластеры из 100-го по 500-й и записала в них соответствующие данные. При запросе на чтение файла с этим именем ФС должна обеспечить считывание данных именно из этих кластеров.

Эта задача решается созданием специальной таблицы в составе файловой системы, а именно – так называемой таблицы размещения файлов, или FAT (от англ. File Allocation Table). Каждая строка этой таблицы содержит информацию об имени файла, номер кластера, с которого начинается файл, количество кластеров, занимающих файл (длина файла), и дополнительную информацию (дата образования файла, дата последней модификации, владелец файла и пр.). Приблизительный вид структуры показан в табл. 1.

Таблица 1. Приблизительный вид структуры FAT

Имя файла	Номер кластера, с которого начинается файл	Количество кластеров, которые занимает файл	Дополнительная информация

Но при этом появляются некоторые проблемы. Во-первых, при такой организации FAT каждый новый файл должен начинаться с первого незанятого (свободного) кластера и занимать непрерывный участок кластеров. После исключения любого файла участок, который он занимал, больше не может быть занятым. Во-вторых, если не использовать дисковое пространство, которое освобождается после исключения файлов, диск через очень короткое время будет переполнен. Дело в том, что сама операционная система в процессе работы постоянно записывает, считывает и изымает временные системные файлы независимо от пользователя. Например, операции копирования в буфер (Ctrl+C) и вставки из буфера (Ctrl+V) в действительности происходят через запись соответствующих данных на диск, которые изымаются после закрытия прикладной программы. То же происходит и у многих современных программ, например, MS Word, Excel, PowerPoint, Photoshop и др.

Поэтому проблема использования дискового пространства, которое освобождается после исключения файлов, является очень острой. Она может быть решена за счет другой структуры FAT, а также за счет введения в состав файловой системы новой таблицы – таблицы свободного пространства, или FST (от англ. Free Space Table).

Новая структура FAT также содержит имя файла и дополнительную информацию. Но вместо номера кластера, с которого начинается файл и их количества (т.е. информации об одном непрерывном участке) FAT будет содержать информацию о совокупности участков, в которых будет размещен файл (табл. 2).

Таблица 2. Новая структура FAT

Имя файла	Дополнительная информация	(N1, L1)	. . .	(Nk, Lk)

Здесь (N1, L1) – номер кластера, с которого начинается первый участок и количество кластеров, который он занимает; (Nk, Lk) – номер кластера, с которого начинается к-й (последний) участок, и количество кластеров, который он занимает.

Таким образом, файлы размещаются не в одном непрерывном участке кластеров, а в совокупности таких участков.

Информация о свободных участках кластеров содержится в таблице свободного пространства (FST), которая имеет простую структуру:

(A1, L1)
. . .
(An, Ln)

где (A1, L1) – номер и длина первого свободного участка кластеров;

(An, Ln) – номер и длина последнего свободного участка кластеров.

Взаимодействие между FAT и FST происходит таким способом. После запроса программы на размещение какого-либо файла определенной длины ФС записывает в FAT его имя и дополнительную информацию. Дальше она последовательно берет из FST информацию о свободных участках кластеров и записывает ее в FAT, пока их совокупная длина не будет равняться длине файла. При этом информация об участках, которые отведены под файл, изымается из FST.

При запросе на исключение файла информация о его размещении возвращается к FST. При этом смежные участки объединяются, а сама FST сортируеться по увеличению номеров кластеров. Соответствующая строка изымается из FAT.

Такой подход позволяет рационально использовать дисковое пространство, но и имеет определенный недостаток – снижение скорости при работе с файлами, поскольку теперь данные считываються/записываються не одним непрерывным блоком, а несколькими фрагментами, которые разбросаны по отдельным участкам дискового пространства. Поэтому данная ситуация получила название *фрагментации* диска. Чтобы ускорить работу с файлами, необходимо периодически проводить операцию *дефрагментации*, при которой файловая система физически перемещает файлы так, чтобы каждый из них занимал один непрерывный участок, а свободное пространство также было одним участком.

1.2. Разработка объектной модели

Разработка объектной модели заключается в описании классов (их свойств и методов), которые будут использованы при созданнии программы моделирования работы файловой системы.

На основании проведенного в п. 1.1 анализа работы файловой системы нами предложены два новых класса – FAT и FST. Рассмотрим их детальнее.

Класс FAT. Свойства класса: Count – количество занятых строк FAT. Режим доступа – только чтение. Это свойство автоматически изменяется при добавлении и изъятии файлов. Поэтому режим доступа к этому свойству – только чтение.

Методы класса:

1. AddFile(FN:string;Len:integer) – добавление файла к FAT. Эта процедура имеет два параметра: FN – имя файла и Len – длина файла (количество кластеров, которые занимает файл).

2. DeleteFile(FN:string) – исключение файла. Эта функция имеет один параметр FN – имя файла, который должен быть изъят из FAT. Функция возвращает список участков, которые занимал файл.

3. GetFileInfo(i:integer; out FN:string; out C:chain) – получение информации о файле из FAT. Эта процедура имеет один входной параметр – номер файла в FAT. Исходные параметры: FN – имя файла и C – список участков, которые занимает файл.

Класс FST. Свойства класса: Count – количество занятых строк FST. Режим доступа – только чтение. Это свойство автоматически изменяется при добавлении и изъятии участков из FST. Поэтому режим доступа к этому свойству – только чтение.

Методы класса:

1. GetFreeSpace(L:integer) – выделение участков под файл длиной L. Эта функция возвращает значение в виде списка участков, в которых будет размещен файл. При этом данные участки изымаются из FST.

2. AddFreeSpace(C:space) – возвращение свободных участков к FST. Эта процедура имеет один параметр C – список участков, которые были высвобождены при изъятии файла. Она соответствующим образом коррегирует FST (сортирует по адресу и совмещает смежные участки).

3. GetFreeSpaceInfo(i:integer; out addr:integer; out len:integer) – получение информации об *i-м* свободном участке в FST. Исходные параметры этой процедуры: addr – адрес *i-го* свободного участка; len – длина этого участка.

4. GetTotalFree – получение информации о количестве свободного пространства на диске. Эта функция не имеет входных параметров и возвращает общее количество свободных кластеров на диске.

Такие основные свойства и методы, которые дают возможность для разработки программы моделирования работы файловой системы.

Таким образом, на основании анализа файловой системы разработана объектная модель, с помощью которой можно разработать программу моделирования ее работы. В этой модели предложены классы FAT и FST, определены их свойства и методы.

2. Программная реализация объектной модели

В соответствии с объектной моделью было выполнено программирование свойств и методов классов FAT и FST средствами языка Delphi 7. Описание и реализация этих классов содержится в одном модуле.

Таким образом, выполнена программная реализация классов TFAT и TFST, с помощью которых будет разработана программа моделирования работы файловой системы.

Соответствующий программный код приведен в Приложении 1.

3. Разработка программы моделирования файловой системы

3.1. Разработка программы моделирования

Программа моделирования работы файловой системы состоит из формы. Содержание моделирования заключается в том, что с помощью полей «Имя файла» и «Длина» будем задавать произвольное имя файла и его длину. Кнопка «ПРИБАВИТЬ ФАЙЛ» моделирует реакцию файловой системы на запрос по образованию нового файла, а именно:

1. Выполняется сравнение длины файла и общего свободного пространства на диске.
2. Из таблицы FST выбираются свободные участки кластеров для размещения файла.
3. Выбранные участки изымаются из FST и добавляются к FAT с соответствующей коррекцией этих таблиц.

Все изменения в этих таблицах отображаются в полях FAT и FST.

Аналогичным способом отслеживается реакция файловой системы на запрос по исключению файлов.

В приложении 2 приводится текст программы моделирования файловой системы на языке Delphi 7.

3.2. Результаты моделирования

В начале моделирования работы файловой системы условно принято, что дисковое пространство состоит из свободных 16000 кластеров, номер первого кластера равняется 1, таблица FAT пустая. Для улучшения отслеживания действий файловой системы будем считать, что дополнительная информация о файлах отсутствует.

После запуска программы моделирования были отображены начальные состояния FAT и FST в соответствии с этими предположениями.

Здесь запись (1 16000) означает участок, который начинается из кластера №1 и имеет длину 16000 кластеров.

Дальше была выполнена проверка реакции файловой системы на добавление и исключение файлов. В соответствующие поля было введено произвольное имя файла «ABC» и длина 100 и нажата кнопка «ПРИБАВИТЬ ФАЙЛ». Получен такой результат.

1. В FAT появилась строка с именем файла и участком, какой он будет занимать.
2. Файлу отведен один участок в соответствии с его длиной.
3. Свободное пространство сократилось и теперь начинается с 101-го кластера и составляет 15900 кластеров (т.е. 1600–100=15900).

Дальше аналогичным способом были прибавлены еще три файла разной длины.

Был проверен запрос на исключение первого файла («ABC»).

Для этого в списке FAT был выделен файл «ABC» и нажата кнопка «ИЗЪЯТЬ ФАЙЛ». Таким образом, полученный результат отвечает ожидаемому.

Дальше был изъят «Файл_3». После исключения этого файла в программе моделирования также получен ожидаемый результат.

Таким образом, после исключения файлов начинается фрагментация свободного пространства.

Было также проверено добавление файла в этих условиях. Прибавлен файл «XXX» длиной 450 кластеров. Свободные участки должны быть занятыми частями нового файла.

Результат, полученный в программе моделирования, полностью отвечает ожиданиям.

Таким образом, разработанная программа моделирования работы файловой системы работает в соответствии с ожиданием и может быть использована в учебных целях.

С этой целью в программе для быстрого заполнения диска предусмотрена кнопка «ТЕСТ», которая моделирует автоматическое добавление 20 файлов с именами «FILE_1» ... «FILE_20» определенной длины. Дальше в произвольном порядке было изъято несколько файлов и прибавлен файл «Aaa» длиной 5000 кластеров. Таким образом, разработана программа моделирования файловой системы в операционной системе Windows. Показана реакция файловой системы на запросы прикладных программ на добавление и исключение файлов. Полученные результаты подтверждают правильность работы этой программы.

Заключение

Результаты выполненной работы следующие:

1. Проведен анализ основных принципов работы файловой системы операционной системы Windows.
2. На основании этого анализа построена объектная модель работы файловой системы.
3. Средствами языка Delphi 7 разработаны классы – TFAT («Таблица размещения файлов») и TFST («Таблица свободного пространства»), которые были использованы для разработки программы моделирования.
4. Проведено тестирование разработанной программы с использованием новых классов на примере моделирования реакции файловой системы на запросы прикладных программ по добавлению и исключению файлов.
5. Полученные результаты демонстрируют правильность работы новых классов и программы моделирования.
6. Данная разработка может быть полезной при преподавании дисциплин, связанных с вычислительной техникой, системным и прикладным программированием.

Список литературы: **1.** *Культин Н.* Программирование в Delphi 5. 2000. Спб.: Питер. 464 с. **2.** *Бондаренко М. А.* Основи інформаційних технологій та програмування, Х.: ФОП Павленко О.Г.. 2010. 600 с.

Поступила в редколлегию 06.02.2010

Бондаренко Николай Андреевич, канд. техн. наук, профессор кафедры информатики и компьютерных технологий Украинской инженерно-педагогической академии. Научные интересы: проектирование технических систем. Адрес: Украина, 61145, Харьков, ул. Университетская, 16, тел. 773-79-17.

Шеховцова Виктория Ивановна, ассистент кафедри информатики и компьютерных технологий Украинской инженерно-педагогической академии. Научные интересы: проектирование информационных систем. Адрес: Украина, 61145, Харьков, ул. Университетская, 16, тел. 773-79-17.

Часовская Елена Александровна, магистрант Украинской инженерно-педагогической академии. Научные интересы: проектирование информационных систем. Адрес: Украина, 61145, Харьков, ул. Університетська, 16, тел. 773-79-17.

```
unit CLASSFST;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms
 Dialogs, StdCtrls, DateUtils;
type space = record
 addr:integer; // Адрес кластера
 len:integer; // Длина участка
end;
type chain=record
 ASP:array[1..100] of space; // Цепочка участков
 Len:integer;
end;
type fal=record // Запись в таблице файлов
 FileName:string; // Имя файла
 Alloc:chain; // Размещение файла
end;

 { Определение класса «Таблица свободного
пространства»}
type
TFST = class
private
 { Внутренние переменные}
 ifSTCount: integer; // Для хранения количества
цепочек в таблице
 aFST: array [1..1000] of space; // Таблица сво-
бодного пространства

 { Процедуры и функции доступа к свойствам}
 function GetCount: integer; // Только чтение
public
 { Свойства класса}
 { Количество цепочек}
 property Count: integer read GetCount;

 { Методы класса}
 { Запрос на свободное пространство}
 function GetFreeSpace(L:integer):chain;
 { Возвращение свободного пространства к таб-
лице}
 procedure AddFreeSpace(C:space);
 procedure GetFreeSpaceInfo(i:integer; out
addr:integer; out len:integer);
 function GetTotalFree():integer;
end;
 { Определение класса «Таблица размещения
файлов»}
type
TFAT = class
private
 { Внутренние переменные}
 ifATCount: integer; // Для хранения количества
файлов в таблице
 aFAT: array [1..1000] of fal; // Таблица разме-
щения файлов

 { Процедуры и функции доступа к свойствам}
 function GetCount: integer; // Только чтение
public
 { Свойства класса}
 { Количество файлов}
 property Count: integer read GetCount;

 { Методы класса}
 { Исключение файла}
 function DeleteFile(FN:string):chain;
 { Добавление нового файла}
 procedure AddFile(FN:string;C:chain);
 { Получение информации о размещении файла}
 procedure GetFileInfo(i:integer; out FN:string; out
C:chain);
end;

implementation
 { Реализация свойств класса FST}
 { Чтение количества цепочек}
function TFST.GetCount: integer;
begin
 Result:= ifSTCount;
end;

 { Реализация методов класса FST}
 { Запрос на свободное пространство}
function TFST.GetFreeSpace(L:integer):chain;
var
 C:chain;
 i,k,sum:integer;
 Rest:integer;
begin
 Rest:=L;
 // Проверка количества свободного простран-
ства
 sum:=0;
 for i:=1 to ifSTCount do sum:=sum+aFST[i].len;
 if Rest>sum then
 begin
 ShowMessage("НЕТ СВОБОДНОГО ПРО-
СТРАНСТВА ДЛЯ РАЗМЕЩЕНИЯ ФАЙЛА");
 C.Len:=0;
 Result:=C;
 exit;
 end;
 for i:=1 to ifSTCount do
 begin
 if Rest>=aFST[i].len then // Свободная область
меньше за остальной файл
 begin
 C.Len:=i; // Адрес участка
 C.ASP[i].addr:= aFST[i].addr; // Длина участка
 C.ASP[i].len:= aFST[i].len;
 Rest:=Rest-aFST[i].len; // Остальной файл
 aFST[i].addr:=0; // Признак занятости участка
 aFst[i].len:=0;
 end
 else // Свободная область большая за остальной
файл
 begin
 C.Len:=i;
 C.ASP[i].addr:= aFST[i].addr;
 C.ASP[i].len:=rest;
 // Корегування FST
 aFST[i].addr:=aFST[i].addr+Rest; // Новый адрес
участка
 aFST[i].len:=aFST[i].len-Rest; // Новая длина уча-
стка
 Rest:=0; // Файл размещен
 break;
 end;
 end;
 // Исключение занятых участков
 sum:=0; // Подсчет занятых участков
 i:=1;
```

```
      while и <= ifSTCount do
      begin
      if aFST[i].len=0 then // Участок занят
      begin
      sum:=sum+1; // Подсчет занятых участков
         for k:=i to ifSTCount-1 do
aFST[k]:=aFST[k+1];
      end
      else i:=i+1;
      end;
       ifSTCount:=ifSTCount-sum; // Количество
записей в FST
      result:=C;

     end;

      { Возвращение свободного пространства в
таблицу}
     procedure TFST.AddFreeSpace(C:space);
     var
      i,k,sum:integer;
      SP:space;
     begin
       ifSTCount:=ifSTCount+1;
      aFST[ifSTCount].addr:=C.addr;
       aFST[ifSTCount].len:=C.len;
      // Сортировка FST по адресу
      SP.addr:=aFST[1].addr;
      SP.len:=aFST[1].len;
      for i:=1 to ifSTCount-1 do
      begin
      SP.addr:=aFST[i].addr;
      SP.len:=aFST[i].len;
      for k:=i+1 to ifSTCount do
      begin
      if aFST[k].addr<SP.addr then
      begin
      SP.addr:=aFST[k].addr;
      SP.len:=aFST[k].len;
      aFST[k]:=aFST[i];
      aFST[i]:=SP;
      end;
      end;
      end;
      // Объединение смежных участков
      sum:=0;
      for i:=1 to ifSTCount-1 do
      begin
       if aFST[i].addr+aFST[i].len=aFST[i+1].addr
then // Смежные участки
      begin
      aFST[i].len:=aFST[i].len+aFST[i+1].len; // Но-
вая длина участка
      sum:=sum+1;
      // Исключение смежных участков
         for k:=i+1 to ifSTCount do
aFST[k]:=aFST[k+1];
      end;

      end;
       ifSTCount:=ifSTCount-sum;

     end;
     procedure TFST.GetFreeSpaceInfo(i:integer; out
addr:integer; out len:integer);
     begin
      if (i<=0) or (i>ifSTCount) then
      begin
      addr:=0;
      len:=0;
      Showmessage("ОШИБКА ДОСТУПА К FST");
      exit;
      end;
      addr:=aFST[i].addr;
      len:=aFST[i].len;
     end;
     function TFST.GetTotalFree():integer;
     var
      i,sum:integer;
     begin
      sum:=0;
      for i:=1 to ifSTCount do sum:=sum+aFST[i].len;
      result:=sum;
     end;

     //—

      { Реализация свойств класса FAT}
      { Чтение количества файлов}
     function TFAT.GetCount: integer;
     begin
      Result:= ifATCount;
     end;

      { Реализация методов класса FAT}
      { Исключение файла}
     function TFAT.DeleteFile(FN:string):chain;
     var
      и, к: integer;
      C:chain;
     begin
     // Showmessage("DeleteFile");
      // Поиск файла в FAT
      k:=0;
      for i:=1 to ifATCount do
      begin
      if FN=aFAT[i].FileName then
      begin
      k:=i;
      break;
      end;
      end;
      if k=0 then
      begin
      ShowMessage("ФАЙЛ "+FN+" НЕ НАЙДЕН');
      C.Len:=0;
      result:=C;
      exit;
      end;
      result:=aFAT[k].Alloc;
      // Корегування FAT
      if k=ifATCount then
      begin
      ifATCount:=ifATCount-1;
      exit;
      end;
      for i:=k to ifATCount do
      begin
       aFAT[i]:=aFAT[i+1];
      end;
      ifATCount:=ifATCount-1;

     end;
```

```
{ Добавление нового файла}
procedure TFAT.AddFile(FN:string;C:chain);
var
 i:integer;
begin
 for i:=1 to ifATCount do
 begin
 if aFAT[i].FileName=FN then
 begin
 ShowMessage("ФАЙЛ "+FN+" УЖЕ СУЩЕ-
СТВУЕТ');
 exit;
 end;
 end;
 ifATCount:=ifATCount+1;
 aFAT[ifATCount].FileName:=FN;
 aFAT[ifATCount].Alloc:=C;
end;
procedure TFAT.GetFileInfo(i:integer; out
FN:string; out C:chain);
begin
 if (i<=0) or (i>ifATCount) then
 begin
 FN:=';
 Showmessage("ОШИБКА ДОСТУПА К FAT");
 exit;
 end;
 FN:=aFAT[i].FileName;
 C:= aFAT[i].Alloc;
end;

end.
```

Приложение 2. Программа моделирования файловой системы

```
unit Unit1;
interface
uses
 Windows, Messages, SysUtils, Variants,
Classes, Graphics, Controls, Forms
 Dialogs, CLASSFST, StdCtrls;

type
 TForm1 = class(TForm)
 Edit1: TEdit;
 Label1: TLabel;
 Edit2: TEdit;
 Label2: TLabel;
 Button1: TButton;
 Button2: TButton;
 ListBox1: TListBox;
 Label3: TLabel;
 Label4: TLabel;
 ListBox2: TListBox;
 Button3: TButton;
 Label5: TLabel;
 procedure FormCreate(Sender: TObject);
 procedure Button1Click(Sender: TObject);
 procedure Button2Click(Sender: TObject);
 procedure SHOWFAT;
 procedure SHOWFST;
 procedure Button3Click(Sender: TObject);
 procedure ListBox1Click(Sender: TObject);
 private
 { Private declarations }
 public
 { Public declarations }
 end;
var
 Form1: TForm1;
 FST:TFST;
 FAT:TFAT;
 C:chain;
 SP:space;
implementation
{$R *.dfm}
procedure TForm1.FormCreate(Sender:
TObject);
begin
 { Образование таблиц}
 FST:=TFST.Create;
 FAT:=TFAT.Create;
 { Начальное образование свободного простран-
ства}
 SP.addr:=1; // Адрес свободного участка
 SP.len:=16000; // Длина участка
 FST.AddFreeSpace(SP);
 SHOWFAT;
 SHOWFST;
end;

procedure TForm1.Button1Click(Sender:
TObject);
// Добавление к файлу
var
 L:integer;
begin
 L:=StrToInt(edit2.text);
 C:=FST.GetFreeSpace(L);
 if C.Len=0 then exit;
 FAT.AddFile(edit1.text, C);
 SHOWFAT;
 SHOWFST;
end;

procedure TForm1.Button2Click(Sender:
TObject);
// Исключение файла
var
 C:chain;
 SP:space;
 и: integer;
begin
 C:=FAT.DeleteFile(edit1.Text);
 for i:=1 to C.Len do
 begin
 SP:=C.ASP[i];
 FST.AddFreeSpace(SP);
 end;
 SHOWFAT;
 SHOWFST;
end;

procedure TForm1.ShowFAT;
// Отображение состояния FAT
var
 i,k:integer;
 FN, S:string;
 C:chain;
 SP:space;
```

```
 T:TObject;
begin
 listBox1.Clear;
 for i:=1 to FAT.Count do
 begin
 S:='';
 FAT.GetFileInfo(i,FN,C);
 for k:=1 to C.Len do
 begin
 s:=s+'('+inttostr(C.ASP[k].addr)+' ';
 s:=s+inttostr(C.ASP[k].len)+') ';
 end;
 S:=FN+' '+s;
 listbox1.AddItem(S,T);
 end;
end;

procedure TForm1.ShowFST;
// Отображение состояния FST
var
 i,k:integer;
 addr,len:integer;
 S:string;
 T:TObject;
begin
 listBox2.Clear;
 for i:=1 to FST.Count do
 begin
 S:='';
 FST.GetFreeSpaceInfo(i,addr,len);
 S:='('+IntToStr(addr)+' '+IntToStr(len)+') ';
 listbox2.AddItem(S,T);
 end;
                    label15.Caption:='РАЗМ:
'+IntToStr(FST.GetTotalFree);
end;


procedure TForm1.Button3Click(Sender: TObject);
// Автоматическое тестирование
var
 i:integer;
begin
 for i:=1 to 20 do
 begin
 C:=FST.GetFreeSpace(i*50);
 if C.Len=0 then exit;
 FAT.AddFile('FILE_'+IntToStr(i), C);
 end;
 SHOWFAT;
 SHOWFST;
end;

procedure TForm1.ListBox1Click(Sender: TObject);
var
 s:string;
 k:integer;
 C:chain;
begin
 k:=ListBox1.ItemIndex+1;
 FAT.GetFileInfo(к,s,C);
 edit1.Text:=s;
end;
end.
```

УДК 004.932.75

М.І. ВАСЮХІН, А.М. КАСІМ, В.Д. ГУЛЕВЕЦЬ, О.Л. БОЙКО, Н.М. ЧУКАРІНА

МЕТОДИ СТВОРЕННЯ ДИНАМІЧНИХ ГРАФІЧНИХ ОБРАЗІВ ПРИ ВИРІШЕННІ ЗАДАЧ ВІДОБРАЖЕННЯ ПОТОЧНОЇ ОБСТАНОВКИ НА ТЕРИТОРІЇ АЕРОПОРТУ ТА ПРИЛЕГЛИХ ДО НЬОГО ЗОН

Розглядаються питання вирішення задач відображення поточної обстановки на території аеропорту та прилеглих до нього зон за допомогою її подання у вигляді динамічних графічних образів. Наводиться опис програмних методів представлення повітряної та наземної обстановки в районі руху спостережуваних повітряних об'єктів, які дозволяють поліпшити роботу управління повітряним рухом для кожного класу диспетчерів.

1. Вступ

Основним шляхом удосконалення процесів управління повітряним рухом (УПР) стало створення і впровадження автоматизованих систем і засобів автоматизації, що здійснюють обробку даних і забезпечують їх адекватне представлення диспетчерам [1-3]. Необхідність поліпшення УПР і автоматизації обробки даних була продиктована тим, що традиційні засоби УПР перестали забезпечувати належну якість управління, диспетчери не отримували повного уявлення про динаміку повітряної обстановки і витрачали неприпустимо багато часу на заповнення та обробку даних, необхідних для прийняття рішень по управлінню рухом повітряного судна (ПС). Тимчасова завантаженість диспетчерів за відсутності автоматизованої обробки та представлення даних у реальному часі значно збільшується під час пікових навантажень, при виникненні особливих умов польоту або особливих випадків, коли екіпажу ПС потрібна термінова і ефективна допомога з боку органів УПР.

2. Основна частина

Процес автоматизації сфери УПР можна представити такими етапами.

Перший етап автоматизації процесів УПР ставив за свою мету створення і впровадження засобів малої автоматизації збору, обробки і відображення даних первинного та вторинного радіолокаторів (ВРЛ) для невеликих районних центрів (РЦ) із середньою і низькою інтенсивністю польотів, що забезпечують аналогове відображення відміток ПС разом з додатковою інформацією від відповідачів ВРЛ.

Другий етап автоматизації характеризується реалізацією додаткових функцій планування повітряного руху та ототожнення радіолокаційної і планової інформації. Це дозволило здійснювати кореляцію трека ПС з планом польоту, розрахунок поточного плану польоту по маршруту в зоні відповідальності і, як наслідок, підвищити ефективність представлення даних про прогнозований і поточний повітряний рух.

Третій етап автоматизації характеризується розробкою інтегрованих аеродромно-районних засобів обробки даних, що виконують функції системи безпеки, пов'язані з пошуком і запобіганням конфліктних ситуацій: попередження про небезпечні зближення між рухомими повітряними об'єктами, сигналізація про зниження ПС нижче мінімально безпечної висоти, сигналізація про порушення порядку використання повітряного простору, повідомлення про перебування на літаку терориста тощо. У системах і засобах третього етапу автоматизації повинна реалізуватися нова концепція людино-машинної взаємодії на базі графічного інтерфейсу користувача.

Сучасні геоінформаційні аеронавігаційні комплекси реального часу (ГАК РЧ) являють собою складні ергатичні системи, в яких містяться: засоби збору інформації від об'єкта управління, тобто ПС; обчислювальні засоби, об'єднані в апаратно-програмний комплекс, що проводять обробку інформації в цілях підготовки альтернативних варіантів управлінських рішень та забезпечення показу динаміки керованих аерооб'єктів в режимі реального часу; системи відображення інформації, що сприяють візуалізації динамічних сцен (ДС) на картографічній основі; управлінський персонал (оперативний склад ГАК РЧ), що здійснює процес приймання рішень та передачі їх на керований об'єкт. ГАК РЧ є складовою систем захисту території аеропорту та прилеглих до нього зон. В результаті застосування таких комплексів змінюються зміст і психологічна структура праці управлінського персоналу при вирішенні задач УПР. При цьому все більш важливу роль відіграють процеси сприйняття і переробки інформації, ухвалення відповідальних рішень в умовах обмеженого ліміту часу. Включення людини-оператора у вирішення задач УПР зумовлюється психофізіологічними властивостями людини, що дозволяють їй вирішувати задачі управління, повна автоматизація яких неможлива або технічно нераціональна.

Поданий нижче перелік задач, які повинен вирішувати диспетчер сектора управління району повітряного руху, показує різноманіття і складність ухвалюваних людиною оперативних рішень:

– збір та сприйняття інформації про повітряну обстановку, визначення фактичного польоту ПС і моменту входу його в зону відповідальності диспетчера;

– розробка поточного плану польоту та узгодження його з екіпажем і суміжними пунктами управління (поточний план польоту – безконфліктна просторово-часова траєкторія руху ПС – розробляється на підставі інформації про інші поточні плани польотів та практичного їх виконання, враховуючи обмеження у просторі, фактичну метеообстановку, вимоги регулярності й економічності польотів);

– слідкування за поточною траєкторією польоту, порівняння її з траєкторією поточного плану польоту, визначення відхилень за часом, координатами та інтервалами ешелонування;

– прогнозування повітряної обстановки та поточної траєкторії польоту на певному інтервалі часу та попередження пілота про тенденцію до відхилення;

– визначення можливості подальшого польоту по траєкторії поточного плану і ухвалення рішення про вирішення конфлікту;

– узгодження з пілотом і суміжними пунктами управління заходів по ліквідації відхилень від поточного плану, аж до розробки нового плану польоту;

- прийом на управління ПС від сусідніх секторів управління і передача їх диспетчерам сусідніх секторів, прийом повідомлень від відомчих органів і від навігаційних систем [4].

Всі ці дії виконуються диспетчером по одному ПС. Потім він переходить до вироблення управляючих дій по іншому повітряному об'єкту, що входить в сферу його діяльності. Наступний цикл по попередньому ПС поновлюється через деякий проміжок часу. Час диспетчера розподіляється на збір і обробку інформації про повітряну обстановку, її аналіз, прийняття управлінських рішень й передачу їх екіпажу ПС та взаємодіючим службам. Відносні затрати робочого часу диспетчерами РЦ УПР на обслуговування одного ПС оцінюються (у відсотках від загальних витрат) так: радіозв'язок з екіпажами - 30, обробка і ототожнення радіолокаційної інформації - 24, взаємодія із суміжними диспетчерськими пунктами (ДП) - 20, аналіз повітряної обстановки - 16 й ухвалення рішень - 10.

Із зростанням інтенсивності повітряного руху диспетчер відчуває все більшу нестачу часу на виконання технологічних операцій. Скоротити число ПС, що одночасно знаходяться під управлінням одного диспетчера, можна шляхом розділення повітряного простору на сектори управління. Однак при цьому число узгоджень при прийомі-передачі управління зростає пропорційно квадрату числа секторів, ускладнюється робота екіпажу за рахунок збільшення кількості переходів на зв'язок від одного диспетчера до іншого, скорочується час перебування ПС під управлінням одного диспетчера, що ускладнює саме керування [4], хоча з іншого боку розподіл повітряного простору та аеродромів на райони відповідальності ДП забезпечує ефективний контроль за рухом ПС. За типом виконуваних технологічних задач їх можна класифікувати на ДП «Рулювання», «Старту і посадки», «Круга», «Підходу», «Районного Центру», ДП «Місцевих Повітряних Ліній», а також «Аеродромні Диспетчерські Пункти» (таблиця).

Класифікація диспетчерів за типом виконуваних задач

Диспетчер	Обов'язки
«Аеродромного Диспетчерського Пункту» (АДП)	Контролює готовність екіпажу ПС до виконання польоту, доводить до нього необхідну інформацію, складає добовий план польотів, фіксує початок і закінчення виконання польоту, погоджує виконання плану польотів з іншими службами (наприклад, з АДП іншого аеропорту). Диспетчер АДП не здійснює контроль за фактичною повітряною обстановкою
«Рулювання»	Контролює рух ПС по території аеродрому, видає дозволи на буксування, запуск двигунів, рулювання
«Старту і посадки»	Контролює рух на злітно-посадочній смузі та передпосадковій прямій, керує ПС, що злітають і заходять на посадку, видає дозволи на зліт та посадку
«Круга»	Керує рухом ПС в області повітряного простору від 2 км і нижче та в радіусі 50 км від аеродрому. Видає дозволи на виконання заходу на посадку прилітаючим ПС і вказівки про первинний набір висоти вилітаючим
«Підходу»	Керує рухом ПС в області повітряного простору, обмеженої висотами 2 та 6 км і віддаленням від аеродрому на 90-120 км. Диспетчер «Підходу» вирішує задачі по визначенню черговості заходу на посадку, а також побудови необхідних інтервалів ешелонування
«Районного Центру»	Контролює політ ПС на висотах від 1,5 до 12 км і в рамках встановлених меж в горизонтальній площині
«Місцевих Повітряних Ліній»	Керує польотом ПС від висоти 1,5 км та нижче і в рамках встановлених меж в горизонтальній площині

Контроль за рухом ПС здійснюється від моменту покидання ним стоянки перед зльотом на аеродромі вильоту до зарулювання на стоянку після посадки на аеродромі призначення.

В умовах інтенсивного повітряного руху під керівництвом одного авіадиспетчера може перебувати одночасно 10-20 ПС.

Отже, основною задачею авіадиспетчера є безперервний контроль за повітряною обстановкою і управління повітряним рухом в межах зони його відповідальності. Для виконання цього завдання він використовує радіотехнічні засоби, засоби радіозв'язку з екіпажами ПС, а також електрозв'язку із суміжними секторами та іншими фахівцями. Його робоче місце обладнано моніторами відображення повітряної обстановки, метеообстановки, різними сигнальними табло, довідковою інформацією, засобами зв'язку. Кожне завдання, що вирі-

шується диспетчером, вимагає наявності інформації. До такої інформації, якою повинен володіти диспетчер, відноситься:

– постійна інформація (інструкції, позивні);

– загальноінформаційна (повідомлення про погоду, стан аеродрому);

– конкретно-інформаційна (час підходу літака до зони, дані про стан літака, результати переговорів);

– оперативна (надається диспетчеру для спостереження через засоби відображення реального часу).

На підставі цієї інформації у диспетчера будується просторово-часовий образ повітряної обстановки (її концептуальна модель), на основі якого він ухвалює конкретне управлінське рішення. При цьому необхідно особливо підкреслити, що кожне рішення приймається ним в умовах обмеженого ліміту часу на його вироблення, оскільки обстановка в районі УПР безперервно змінюється, літаки швидко переміщаються в просторі.

Для підвищення рівня сприйняття оперативної інформації тим або іншим диспетчером, а також для прискорення прийняття ним адекватних рішень нами пропонується представляти поточну ситуацію на території аеропорту і прилеглих до нього зон у вигляді ДС, тобто шляхом візуалізації на картографічному фоні лінійно-обертального руху динамічних символів відповідних повітряних об'єктів. При цьому синтез динамічних ефектів (плоскопаралельний рух, обертання і масштабування символу та картографічного фону) на екрані монітора проводиться чергуванням набору образів, який в тій або іншій мірі моделює моменти реального фізичного процесу.

Для досягнення відмічених задач слід враховувати психофізичні особливості сприйняття оком швидкої зміни образів, а також принципи і особливості формування образів на екрані монітора у реальному часі. Одним з ключових моментів моделювання більшості динамічних процесів є встановлення ряду обмежувальних умов на виконання тих або інших елементів цих процесів, причому ці умови звичайно є досить точним відображенням реальних фізичних обмежень. Найпростіший метод імітації складного руху образу, наприклад векторного символу літака, є малювання фігури як контура на наборі вузлових точок з викликом процедур перетворення координат вузлових точок контура. При цьому малювання самої фігури і перетворення координат вузлових точок виконується за простими й швидко працюючими алгоритмами, а процес перемальовування відбувається майже непомітно для очей навіть на не дуже могутніх персональних комп'ютерах.

При моделюванні руху часто виникає необхідність безперервно відстежувати поточне положення об'єкта у випадках, коли воно безпосередньо визначає сам хід процесу. Визначення координат об'єкта звичайно складності не представляє, оскільки завжди відомі координати вузлових точок. Дещо складніше визначати орієнтацію об'єкта.

Для формування ДС поточної обстановки на території аеропорту та прилеглих до нього зон пропонуються такі методи створення динамічних графічних образів.

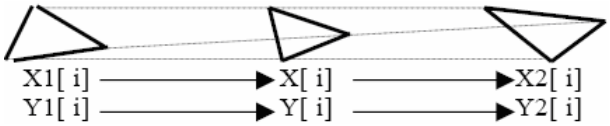
Метод перерисовування графічного об'єкта полягає в багатократному його перемальовуванні із зсувом координат – на поточному кроці по поточних координатах об'єкт малюється, запам'ятовуються старі і визначаються його нові координати, задається затримка (її величина визначає швидкість руху об'єкта). Потім слідує малювання кольором фону по старих координатах і повторення процесу на наступному кроці. Цей метод застосовується для виведення не дуже складних об'єктів невеликого розміру, промальовувати які можна достатньо швидко.

Метод плавної модифікації контурного зображення використовується для анімації образів, що задаються набором координат вузлових точок. Етапи реалізації методу:

1) задається масив координат вузлових точок вихідного (початкового) контурного зображення (X1[1..N], Y1[1..N]). Сполучаючи певним чином ці точки відрізками прямих, одержуємо зображення векторного символу;

2) задається масив координат вузлових точок цільового (кінцевого) контурного зображення (X2[1..N], Y2[1..N]). Кількість точок однакова для обох масивів;

3) плавною модифікацією початкового образу одержуємо цільове зображення. Для цього послідовно знаходяться набори координат X[1..N], Y[1..N] проміжних образів. Кожну і-ту точку проміжного образу вибирають на відрізку прямої між відповідними точками початкового і цільового контурів, тобто між точкою X1[i], Y1[i] і точкою X2[i], Y2[i] (рисунок).



В такий спосіб відрізок ділиться на m частин, де m – кількість проміжних образів, включаючи цільовий.

Проміжні образи перемальовуються, поступово віддаляючись від початкового образу.

У випадку рівномірного розподілу відрізків координати вузлових точок проміжних образів можна розрахувати за формулами:

$$x[i] := x1[i] + (x2[i] - x1[i]) *k/m;$$
$$y[i] := y1[i] + (y2[i] - y1[i]) *k/m;$$

де k – номер проміжного образу; m – кількість розподілів.

Затримка видимості образу визначає швидкість перетворення. Координати точок проміжних образів можна визначати не тільки рівномірним розбиттям прямих ліній між початковим і цільовим зображеннями, але й сполучаючи точки початкового і цільового контурів по кривих лініях з нерівномірним розбиттям [5].

Метод мультиплікації із чергуванням відеосторінок. Для якісних графічних адаптерів можна встановити режими з декількома графічними сторінками. При цьому під час перегляду кадру на одній відеосторінці на іншій організовується процес малювання наступного кадру. Потім сторінки перемикаються, невидима сторінка активізується для малювання і процес повторюється. Перемикання сторінок проводиться достатньо швидко, що важливо при створенні якісних образів, що рухаються. Для складних сцен з великим часом перемальовування кадрів метод чергування відеосторінок виступає єдиним прийнятним.

Метод анімації чергуванням набору образів. Універсальний прийом створення рухомих зображень – послідовне виведення в потрібних місцях екрану наперед створених наборів образів. Звичайно ці образи запам'ятовуються безпосередньо як фрагменти екрану у вигляді прямокутних масивів пікселів. Цей метод найбільш зручний для моделювання руху, який циклічно повторюється, що на практиці зустрічається досить часто (рух літака «зигзагом»). При запам'ятовуванні фрагмента (межі прямокутної області) графічного екрану під нього потрібно відвести пам'ять певного розміру, тобто зарезервувати необхідний масив байтів для зберігання зображення. При цьому спочатку рисунок (символ літака) охоплюється уявним прямокутником за визначеними екранними координатами його лівого верхнього (x1,y1) та правого нижнього (x2,y2) кутів, знаючи які отримуємо число байтів для запам'ятовування прямокутника із зображенням у динамічній пам'яті. Розмір пам'яті, що відводиться для зберігання фрагмента, повинен бути менше 64 Кбайт – одного сегмента даних. Цей розмір можна також обчислити за формулою (x2-x1+1)*(y2-y1+1).

Далі визначається параметр, який міститиме початкову адресу області пам'яті (буфера), що відводиться для зберігання двійкового образу прямокутної ділянки екрану з координатами (x1,y1,x2,y2). Зберігаємо двійковий образ прямокутної області екрану в ОЗП (буфері).

Зчитування графічного фрагмента з динамічної пам'яті здійснюється в точку з координатами лівого верхнього кута прямокутника, у який буде поміщено зображення після його зчитування з динамічної пам'яті. Отже, збережений масив пікселів можна виводити на екран з буфера в позицію, починаючи з верхнього лівого кута області в режимі виведення, який відповідає логічним операціям з бітами, що визначають колір пікселів на екрані та відповідають значенням кольору пікселів, що виводиться з буфера:

Copy – заміна зображення на екрані зображенням з буфера (існуюче зображення замінить копією відповідного фрагмента динамічної пам'яті).

Xor – заміна відбудеться за правилами роботи логічної функції «виключаюче АБО» (результат дорівнює 1, якщо значення бітів різні).

Or – заміна відбудеться за правилами роботи функції «АБО» (результат дорівнює 1, якщо один з бітів рівний 1).

And – заміна відбудеться за правилами роботи функції «І» (результат дорівнює 1, якщо обидва біти рівні 1).

Not – заміна відбудеться за правилами роботи функції «НЕ» (зображення на екрані заміниться інверсним зображенням з буфера).

Бачимо: режим виведення задає спосіб взаємодії пікселів ділянки екрана, куди відбувається зчитування зображення, з пікселами самого зображення.

Колір пікселів зображення з буфера не змінюється, якщо рисунок виводиться в область, яка залита фоном.

Виведення в режимі Xor є зручним для створення зображень, що рухаються, оскільки при першому виведенні одержуємо зображення з буфера, а при другому – відновлюємо зображення на екрані. В буфер поміщають звичайно декілька різних образів, наприклад, символи літака в різних азимутальних напрямках. На екран виводиться перший образ, потім після затримки це виведення повторюють в тому ж місці – відбувається відновлення зображення на екрані. Далі (звичайно в іншій позиції) подібна процедура подвійного виведення проводиться з рештою образів.

Ілюзія руху виникатиме, якщо почергово читати зображення з динамічної області пам'яті, записувати його в певне місце графічного екрана, стирати його після певного часу затримання і, змінюючи координати місцезнаходження зображення в напрямку руху, знову і знову повторювати: записування на екран, затримання, стирання зображення.

Стирання зображення з екрана перед його переміщенням можна здійснити одним із двох способів:

1) Окрім прямокутника з рисунком записати у динамічну пам'ять прямокутник такого ж розміру, що містить фон (ділянку екрана під символом). Після затримання показу зображення зчитати у місці його знаходження прямокутник із фоном і стерти тим самим зображення.

2) Оточити символ більшим прямокутником, сторони якого віддалені від країв символу на L пікселів. Якщо крок зміни координат зображення вибирати меншим за L, то нова копія зображення затиратиме попередню. Цей спосіб є вдалим у випадку, коли колір фону в області руху символу є однаковим.

При створенні складних динамічних двовимірних сцен гарні результати вдається отримати лише за умови достатньо швидкого аналізу умов, що визначають характер модельованого процесу.

Більшість комп'ютерних графічних образів при обмеженості обчислювальних ресурсів частіше за все створюється як набір відрізків прямих ліній, що відображає звичайно каркас модельованого об'єкта. Подібні моделі об'єктів легко і швидко можна відображати і при необхідності трансформувати як для статичних, так і для динамічних сцен. Об'єкти при цьому кодуються як сукупність набору координат вершин і послідовності обходу контура по цих вершинах. Для забезпечення обертального руху таких образів зручно використовувати систему рівнянь, яка описує поворот на кут φ точки (x,y) навколо будь-якої точки з координатами (x_0,y_0):

$$x_1 = x_0 + (x - x_0) \cdot \cos\varphi - (y - y_0) \cdot \sin\varphi;$$

$$y_1 = y_0 + (x - x_0) \cdot \sin\varphi + (y - y_0) \cdot \cos\varphi.$$

Координати (x_1,y_1) задають нове положення точки. При цьому координати точок зламу лінії, яка утворює «літак», заносяться у масив, і кожна пара нових координат обчислюється за наведеною вище системою рівнянь. Слід зазначити, що якщо під час повертання, наступне положення фігури перераховувати на основі координат її точок у попередньому положенні, то через деякий час форма фігури почне спотворюватись. Це відбувається внаслідок накопичення похибок, що виникають у процесі перетворення координат. Щоб цього не відбувалось, необхідно зафіксувати початкове положення фігури у масиві координат точок зламу контуру «літака» і координати точок фігури для кожного нового кута обертання обчислювати за цим початковим положенням. Кількість пар координат точок контурного об'єкта залежить від складності зображення символу.

При побудові символів рухомих об'єктів на екрані монітора необхідно перетворювати розрахункові координати в графічні з дотриманням певних пропорцій, тобто з урахуванням дискретності растрової сітки монітора.

3. Висновки

Показана необхідність формування у реальному часі динамічних сцен поточної обстановки на території аеропорту та в прилеглих до нього зонах, що в першу чергу полегшить роботу кожного класу диспетчерів. Для створення динамічних графічних образів запропоновано методи імітації лінійно-обертального руху складних символів літаків, які представлені у векторному вигляді, на картографічному фоні.

Список літератури: 1. *Агаджанов П.А., Воробьев В.Г., Кузнецов А.А., Маркович Е.Д.* Автоматизация самолетовождения и управления воздушным движением: Учебник для вузов гражданской авиации. М: Транспорт, 1980. 357с. **2.** *Авиационные* автоматизированные комплексы управления и моделирования: Межвузовский сборник научных трудов. К.: КИИГА. 212с. **3.** *Федоров С.М.* Автоматизированное управление самолетами и вертолетами. М.: Транспорт, 1977. 246с. **4.** *Гасов В.М., Соломонов Л.А.* Организация взаимодействия человека с техническими средствами АСУ: В 7 кн. Кн. 1. Практическое пособие. М.: Высшая школа, 1990. 127с. **5.** *Трушин О.В.* Методические указания к лабораторным работам по курсу „Интерактивная машинная графика" для подготовки инженеров по специальности 220200 „Автоматизированные системы обработки информации и управления". Ч. 1. Уфа: Уфимск. гос. авиац. техн. ун-т; 1997. 34с.

Надійшла до редколегії 17.06.2010

Васюхін Михайло Іванович, д-р техн. наук, проф., старший науковий співробітник Інституту кібернетики імені В.М. Глушкова НАН України. Наукові інтереси: інтерактивні геоінформаційні комплекси реального часу. Адреса: Україна, Київ-187, пр. ак. Глушкова, 40, тел. 526-07-73.

Касім Аніса Мохаммадівна, аспірант, молодший науковий співробітник Інституту кібернетики імені В.М. Глушкова НАН України. Наукові інтереси: програмування 2D комп'ютерної графіки, бази картографічних даних, моделювання повітряної обстановки. Адреса: Київ-187, пр. ак. Глушкова, 40, тел. 526-07-73.

Гулевець Вадим Дмитрович, канд. техн. наук, доцент, завідувач кафедри землевпорядних технологій Київського Національного авіаційного університету. Наукові інтереси: автоматизовані інтегровані системи захисту особливо важливих об'єктів. Адреса: Україна, Київ, вул. Гарматна, 1.

Бойко Олена Леонідівна, аспірант, старший викладач кафедри землевпорядних технологій Київського Національного авіаційного університету. Наукові інтереси: бази просторових даних. Адреса: Україна, Київ, вул. Гарматна, 1, тел. 403-16-38.

Чукаріна Наталія Миколаївна, аспірант, асистент кафедри землевпорядних технологій Київського Національного авіаційного університету. Наукові інтереси: бази даних геоінформаційних систем. Адреса: Україна, Київ, вул. Гарматна, 1.