З М І С Т

Любченко К. М. Експертні системи в практичній медицині……. 3
Марковский А.П., Зюзя А.А., Шерстюк В.Д. Получение булевых преобразований специальных классов для построения эффективных алгоритмов защиты информации….. 7
Зайченко Ю.П., Мурга Н.А. Применение систем на нечёткой логике к задаче медицинской диагностики……... 14
Абу Усбах А.Н., Мельник А.П., Пономарчук Д.С. Об одном подходе к обнаружению ошибок передачи данных на основе систем остаточных классов…… 23
Салапатов В.І. Синтаксичний аналіз із розподілом лексем на групи…….. 29
Марковский А.П.,Порхун Е.В., Мнацаканов А.В. Хеш-память с ограниченным временем поиска по ключу……. 33
Пустоваров В.І. Побудова баз знань для сумісного створення програм і обладнання та їх формальної верифікації……. 41
Томашевський В.М., ДіденкоД.Г. Порівняння результатів роботи систем імітаційного моделювання OPENGPSS та GPSS/PC…… 47
Амонс О.А., Янов Ю.О., Безпалий І.О. Кластеризація документів на основі статистичної близькості термів……... 54
Аленин О.И. Система управления виртуальными кластерами…….. 62
Баня Е.Н., Селиванов В.Л. Об особенностях построения систем счисления разных классов 67
Сергієнко А.М. Проектування обчислювачів з регістровими затримками……... 73
Павлов О.А., Місюра О.Б., Мельников О.В., Щербатенко О.В., Михайлов В.В. Загальна схема планування та управління складними об'єктами , що мають мережне представлення технологічних процесів й обмежені ресурси…… 77
Ткаченко С. В. Практическое применение МППО на примере разработки торговой стратеги для рынка FOREX…….. 88
Ролик А.И., Тимофеева Ю.С., Турский Н.И. Управление устранением неисправностей в ИТ-системах…….…. 94
Жабин В.И., Макаров В.В. Использование таблиц функций для быстрого вычисления многочленов……... 108
Жабина В.В. Повышение эффективности параллельной обработки данных на уровне операций в потоковых системах …….. 112
Кулаков А.Ю. Способ повышения эффективности GRID систем на базе сетей MPLS….. 117
Дорогой Я.Ю., Яшин В.Е. Программный комплекс для симуляции многопоточных нейронных сетей……...……. 122
Стіренко С.Г., Шаурін Д.О. Підвищення ефективності роботи IT інфраструктури на основі технології віртуалізації……. 127
Иларионов Р. Программный драйвер связи 3D сканирующего устройства с системами компьютерного проектирования…….. 134
Серая О.В.,Каткова Т.И., Бачкир Л.В. Оценивание состояния с использованием нечеткой регрессии……. 140
Томашевський В.М., Дмитрик І.М. Аналіз моделей навчання та контролю знань…….. 146
Талаєв О.К, Дуднік А.С., Березовський О.М. Моделювання бездротових комп'ютерних мереж в залежності від алгоритму управління та розподілу трафіку…….. 152

Відомості про авторів

Абу Усбах О.Н.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”. E-mail: aliksey@mail.ru
Аленин О.И.	зам. директора центра суперкомпьютерных вычислений НТУУ ”КПИ”
Амонс О.А.	к.т.н., старший викладач кафедри АУТС НТУУ ”КПІ”
Баня Е.Н.	к.т.н., доцент кафедры АСОИУ НТУУ ”КПИ”
Безпалий І.О.	студент, кафедри АУТС НТУУ ”КПІ”
Березовський О.М.	аспірант кафедри ОТ НТУУ ”КПІ”
Діденко Д.Г.	аспірант кафедри АСОІУ НТУУ “КПІ”
Дмитрик І.М.	аспирантка Национального транспортного униерситета
Дорогой Я.Ю.	ассистент кафедры АУТС НТУУ ”КПИ”
Дуднік А.С.	аспірант кафедри ОТ НТУУ ”КПІ”
Жабин В.И.	д-р техн. наук, профессор кафедры ВТ НТУУ «КПИ»
Жабина В. В.	аспирант кафедры СКС НТУУ ”КПІ”
Зайченко Ю.П.	д-р техн. наук, профессор кафедры ММСА НТУУ ”КПИ” УНК «ИПСА»
Зюзя О. А.	аспірант кафедри ОТ НТУУ ”КПІ”
Иларионов Р.	к.т.н., доцент, Болгария
Каткова Т.И.	к.п.н., доцент кафедры математики и математических методов Бердянского университета менеджмента и бизнеса. E-mail: 777-kit@ukr.net
Любченко К.М.	ст. викладач кафедри інтелектуальних і інформаційних систем ЧНУ ім. Б. Хмельницького. E-mail: lkn@ukr.net
Макаров В.В.	к.т.н., начальник сектора АНТК «Антонов»
Марковський О.П.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”. E-mail: markovskyy@mail.ru
Мельник О.П.	аспірант кафедри ОТ НТУУ ”КПІ”
Мельников О.В	к.т.н., ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Михайлов В.В.	к.т.н., ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Місюра О.Б.	к.т.н., ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Мнацаканов А.В.	аспірант кафедри ОТ НТУУ ”КПІ”
Мурга Н.А.	студент кафедри ММСА НТУУ ”КПІ” УНК «ІПСА»
Павлов А.А.	д.т.н., проф. каф. АСОІУ НТУУ ”КПІ”
Пономарчук Д.С.	студент кафедри ОТ НТУУ ”КПІ”
Порхун Е.В.	студент кафедри ОТ НТУУ ”КПІ”
Пустоваров В.І.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”
Ролик А.И.	к.т.н., доцент кафедры АУТС НТУУ ”КПІ”
Салапатов В.І.	к.т.н., доцент кафедры кібернетики, ЧНУ
Селиванов В.Л.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”
Серая О.В.	к.т.н., доцент кафедры экономической кибернетики и маркетингового менеджмента НТУ «ХПИ». E-mail: Seraya@kpi.kharkov.ua
Сергієнко А.М.	к.т.н., ст. наук. спів., кафедра ОТ НТУУ ”КПІ”
Стіренко С.Г.	к.т.н., доцент кафедри ОТ НТУУ ”КПІ”
Талаєв О.К.	ст. викладач кафедри ОТ НТУУ ”КПІ”
Тимофеева Ю.С.	магистр, кафедра АУТС НТУУ ”КПІ”
Ткаченко С.В.	аспирант НТУУ ”КПІ” УНК «ИПСА» E-mail: tkachenko_sv@ukr.net
Томашевський В.М.	д-р техн. наук, професор кафедри АСОІУ НТУУ “КПІ”
Турский Н.И.	магистр, кафедра АУТС НТУУ ”КПИ”
Шаурін Д.О.	студент кафедри ОТ НТУУ ”КПІ”.
Шерстюк В.Д.	студентка кафедри ОТ НТУУ ”КПІ”.
Щербатенко О.В.	к.т.н., ст. наук. спів., кафедра АСОІУ НТУУ ”КПІ”
Янов Ю.О.	аспирант кафедры АУТС НТУУ ”КПІ”
Яшин В.Е.	студент кафедры АУТС НТУУ ”КПІ”

УДК 004

ЛЮБЧЕНКО К.М.

ЕКСПЕРТНІ СИСТЕМИ В ПРАКТИЧНІЙ МЕДИЦИНІ

Одними з перших експертних систем є медичні. У статті міститься огляд в історичній ретроспективі цього класу експертних систем, і визначаються деякі перспективи їх застосування і подальшого розвитку в практичній медицині.

One of the first expert systems was medical. In article the review in a historical retrospective show of this class of expert systems contains, and some prospects of their application and the subsequent development in applied medicine are defined.

Експертні системи є одним з найпоширеніших типів систем штучного інтелекту. Вони розроблялися як науково-дослідні інструментальні засоби з 1960-х років і розглядалися як штучний інтелект спеціального типу, призначеного для успішного вирішення складних задач у вузькій предметній галузі, такий як медична діагностика захворювань [2, 28]. Найбільш широке поширення експертних систем в різних областях людської діяльності почалося на початку 80-х років XX століття.

Оскільки одними з перших експертних систем, які мають попит і зараз, є медичні, то основною метою даної статті є огляд в історичній ретроспективі цього класу експертних систем і визначення перспектив його подальшого розвитку.

Сформульована мета передбачає розгляд наступних питань:

- аналіз поняття "Експертна система";
- приклади медичних діагностичних експертних систем, їх класифікація;
- вимоги сучасного підходу до оцінки стану здоров'я людини;
- постановка завдання створення експертної системи для біорезонансної діагностики.

Приведемо декілька класичних визначень експертних систем і проаналізуємо їх.

Розробник медичної експертної системи MYCIN Э. Фейгенбаум із Станфордського університету визначив поняття експертної системи як "... інтелектуальної комп'ютерної програми, в якій використовуються знання і процедури логічного виводу для вирішення завдань, досить важких для того, щоб вимагати для свого вирішення значного обсягу експертних знань людини" [2, 33].

П. Джексон в [3, 19] дає наступне формальне визначення: "Експертна система – це програма для комп'ютера, яка оперує із знаннями в певній предметній галузі з метою вироблення рекомендацій або вирішення проблем".

Експертна система – це комп'ютерна програма, яка в деякій галузі виявляє ступінь пізнань рівнозначну ступеню пізнання людини-експерта. Зазвичай ця галузь строго обмежена [4, 406].

Експертні системи – це складні програмні комплекси, що акумулюють знання фахівців в конкретних предметних галузях і тиражують цей емпіричний досвід для консультацій менш кваліфікованих користувачів [1, 39].

Автори [6, 32] відзначають: реалізація концепції про те, що ефективність комп'ютерної програми при вирішенні задач залежить від знань, якими вона володіє, а не лише від формалізмів і схем виводу, які вона використовує, привела до розвитку спеціалізованих програмних систем, кожна з яких є "експертом" в деякій вузькій предметній галузі – ці програми отримали назву експертних систем.

Основні етапи процесу розробки експертної системи представлені на рис. 1 [2, 41].

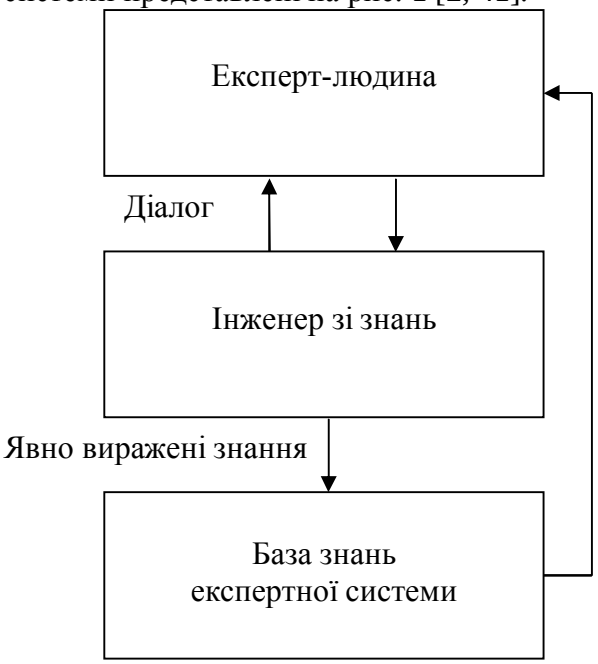


Рис. 1. Процес розробки експертної системи

Таким чином, можна зробити висновок про те, що експертні системи відрізняються від інших систем штучного інтелекту наступними основними ознаками:

1) орієнтованість на вузькоспеціалізовану предметну галузь;
2) вони розробляються для вирішення задач, які підходять для людини-експерта;
3) здатність пояснювати свої рішення зрозумілим користувачеві образом.

Ці ознаки в тій чи іншій мірі простежуються в медичних експертних системах, перші суттєві розробки яких відносяться до початку 1970-х років. Застосування експертних систем в медицині найефективніше при вирішенні задач діагностики, інтерпретації даних, прогнозуванні перебігу захворювань і ускладнень, моніторингу перебігу захворювання і планування лікувально-діагностичного процесу. В наш час побудова ефективних експертних систем різного призначення, у тому числі, медичних систем, залишається завданням актуальним і перспективним. Пройдений лише перший відрізок довгого шляху, труднощі подолання якого обумовлені, з одного боку, надзвичайною складністю об'єкту, що моделюється з використанням експертних систем – мозку людини, а з іншого боку високою складністю завдань, вирішення яких покладається на медичні експертні системи [6, 8].

Одна з перших діагностичних експертних систем MYCIN призначена для роботи в галузі діагностики і лікування зараження крові і медичних інфекцій. На підставі симптомів система видає декілька діагнозів із зазначенням відповідних коефіцієнтів визначеності і пропонує курс лікування виявленої інфекції. База знань MYCIN містить близько 500 правил, розроблених за допомогою групи з інфекційних захворювань Станфордського університету.

На основі обчислених коефіцієнтів визначеності для кожного діагнозу користувачеві пред'являється список можливих результатів.

Успіх MYCIN спричинив створення на її основі оболонки EMYCIN і цілого ряду нових експертних систем. Наприклад, NEOMYCIN, в якій була спроба використовувати більш абстрактний підхід до вирішення медичних проблем, ніж у прототипі, допомагає лікарям діагностувати і лікувати менінгіт і схожі з ним захворювання. При розробці системи було надано значної уваги моделюванню того методу вирішення проблеми, якій властивий лікареві (когнітивне моделювання). Тому в даному програмному продукті знання представлені в такий спосіб, який полегшує пояснення і навчання. Ключова відмінність між MYCIN і NEOMYCIN – це явне відокремлення діагностичної процедури від знань про хвороби.

Експертна система INTERNIST, яка розроблена в Піттсбургському університеті, діагностує декілька сотень різноманітних хвороб на рівні кваліфікованого лікаря. Складні діагнози внутрішніх захворювань, що виводяться системою, визначаються на підставі історії хвороби, симптомів і результатів лабораторних досліджень. Система спирається в своїх рішеннях на набір профілів захворювань (більше 500), що описуються більш, ніж 3500 проявами хвороби і які містять факти, що зустрічаються у зв'язку з кожним із захворювань. Перший варіант системи називається INTERNIST-I, другий – CADUCEUS.

На окрему увагу заслуговує експертна діагностична система по невідкладних станах у дітей ДИН, яка розроблена в Московському НДІ педіатрії і дитячої хірургії.

Як відмічено в [7], база знань ДИН містить описи 34 синдромів, які включають 84 стани. Для системи це список діагностичних припущень-гіпотез. База експертних знань лікаря-реаніматолога налічує більше 1000 діагностичних критеріїв і висновків про динаміку розвитку невідкладного стану.

База знань про синдроми в системі містить декларативну і процедурну інформацію. Декларативні знання описують синдроми (клінічна картина, додаткові синдроми), а процедурні вказують на те, як використовувати знання в процесі діагностики.

Механізм логічного виводу ДИН формує висновок на основі запрограмованої логіки лікаря-реаніматолога. Основою його роботи є змішана стратегія: використовується і прямий, і зворотний ланцюжок міркувань.

Відзначимо, що переважна більшість медичних експертних систем орієнтована на роботу з окремими органами або системами органів. Прокласифікуємо найбільш відомі діагностуючі експертні системи за цим критерієм:

1) порушення кислотно-лужної і водно-сольової рівноваги у пацієнтів, що визначаються на основі застосування знань про захворю-

вання і симптоми, які викликаються ними (ABEL);
2) захворювання крові (MYCIN, HEME, порушення здатності згущуватися крові: AI/COAG, CLOT);
3) захворювання нирок (AI/MM, DIALYSIS THERAPY ADVISOR, EEG ANALYSIS SYSTEM);
4) ревматичні захворювання (AI/RHEUM, ARAMIS);
5) захворювання серцево-судинної системи (ANGY, ANNA, DIAGNOSER, DIGITALIS ADVISOR, GALEN, HEART IMAGE INTERPRETER, HT-ATTENDING, MECS-AI, MI, МОДИС);
6) захворювання щитовидної залози (MECS-AI, THYROID MODEL);
7) порушення в роботі нервової системи (BLUE BOX, HEADMED, NEUREX);
8) захворювання очей (CASNET/GLAUCOMA, MEDICO, OCULAR HERPES MODEL, PEC);
9) порушення функцій легенів (CENTAUR, PUFF, WHEEZE, ТАРТА);
10) захворювання, пов'язані з болем в грудях (EMERGE, MED1);
11) захворювання печінки (MDX, PATREC, RADEX).

Таблиця 1

сегменти	значення	інтерпретація
1, 2	0-20	4 стадія функц. нед.
	21-30	3 стадія функц. нед.
	31-40	2 стадія функц. нед.
	41-49	1 стадія функц. нед.
	50-65	норма
	66-90	функц. напр.
	91-100	істинна гіперфункція
3-30	0-20	4 стадія функц. нед.
	21-30	3 стадія функц. нед.
	31-50	2 стадія функц. нед.
	51-77	1 стадія функц. нед.
	78-86	норма
	87-90	функц. напр.
	91-100	істинна гіперфункція

Більшість медичних експертних систем застосовуються лише при нозологічній формі захворювання.

Сучасна медицина вимагає частішого застосування методів діагностики, що дозволяють оцінити стан організму на ранній стадії захворювання або навіть до його початку, коли розвитку недуги ще можна запобігти. Для ефективного підтримання здоров'я людини надзвичайно важливо вчасно виявити зміни в організмі людини, які потенційно можуть призвести до виникнення хвороб [5].

Одним із сучасних методів діагностики, що дозволяє оцінити стан здоров'я людини, є метод біорезонансної діагностики організму Р. Фолля, методологія якого побудована на інтеграції класичної китайської акупунктури і сучасних досягнень в області електрофізіології, гомеопатії, імунології і інших розділів медицини. Зокрема, апаратно-програмний комплекс АТМ Express дозволяє отримати уявлення про об'єктивний стан організму, особливостях перебігу патології у даного пацієнта [8]. Для оцінки цього стану програма АТМ Express Test реєструє частотні характеристики хвилевих процесів в організмі людини згідно сегментарним зонам шкіряної проекції органів і систем. У результаті цього формується відповідна діаграма з 30 показників, значення яких інтерпретуються у відповідності з таблицею 1.

Отримані дані характеризують функціональний стан організму пацієнта [8]:

- інтенсивність і збалансованість фізіологічних процесів в органах і системах;
- міра ендогенних і екзогенних інтоксикацій;
- наявність радіаційного навантаження;
- можливі пошкодження мікробіологічних циклів на рівні крові, лімфи або в кишечнику;
- дозволяють визначити локалізацію і стадію хронічних процесів;
- візуалізують спроможність таких енергетичних структур як аура і чакри.

Для традиційної оцінки цих показників і подальших досліджень по постановці діагнозу пацієнта необхідний висококваліфікований лікар. Проте, для успішної постановки діагнозу необхідні значні витрати часу і т.д. Тому актуальним є задача розробки експертної системи, яка б в автоматичному режимі проводила оцінку отриманих даних за наступними основними критеріями:

- наявність феномену падіння стрілки, які свідчить про скупчення в тканинах вільних радикалів, що виникають при хронічних процесах, локальних метаболічних порушеннях;
- наявність феномену падіння стрілки, які свідчить про скупчення в тканинах вільних радикалів, що виникають при хронічних процесах, локальних метаболічних порушеннях

- рівень діаграм, який порівнюється з отриманими даними відносно рівня нормальної функції;
- наявність бокової диференціальної різниці в симетричних відведеннях, що свідчить про однобічний процес;
- лабільність функціонального стану.

Експертна система, що розробляється, дозволяє провести як нозологічну, так і донозологічну діагностику стану здоров'я людини.

Перелік посилань

1. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский – СПб: Питер, 2000. – 384 с.
2. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование, 4-е издание.: Пер. с англ. – М.: ООО "И.Д. Вильямс", 2007. – 1152 с.
3. Джексон П. Введение в экспертные системы.: Пер. с англ.: Уч. пос. – М.: Издательский дом "Вильямс", 2001. – 624 с.
4. Ин Ц., Соломон Д. Использование Турбо-Пролога: Пер. с англ. – М.: Мир, 1993. – 608 с.
5. Кузьмук В. В., Супруненко О. О. Роль інформаційних технологій у донозологічній оцінці стану здоров’я людини // Тезисы и доклады международной конференции "Интегративная медицина" (24-25 мая 2008 р.). – К.: Алтимед, 2008. – С. 108-109.
6. Продеус А. Н., Захрабова Е. Н. Экспертные системы в медицине / А. Н. Продеус, Е. Н. Захрабова. – К.: ВЕК +, 1998. – 320 с.
7. Таперова Л. Н. Дин – экспертная диагностическая система по неотложным состояниям / Л. Н. Таперова, В. Веприцкая, Б. А. Кобринский // Программные продукты и системы. – 1995. – № 1. – С. 30-32.
8. Филюнова Е. Г. АТМ Express: Биорезонансная диагностика и терапия (ПБРТ): методические рекомендации / Филюнова Е. Г., Сиряковская Е. И., Демъянцева И. В. – К.: Алтимед, 2007. – 27 с.

УДК 004.93(015.7)

МАРКОВСКИЙ А.П.,
ЗЮЗЯ А.А.,
ШЕРСТЮК В.Д.

ПОЛУЧЕНИЕ БУЛЕВЫХ ПРЕОБРАЗОВАНИЙ СПЕЦИАЛЬНЫХ КЛАССОВ ДЛЯ ПОСТРОЕНИЯ ЭФФЕКТИВНЫХ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ

В статье представлены результаты исследований нелинейных булевых преобразований, обратные к которым обладают свойством неоднозначности и их использования в криптографических алгоритмах. Предложен новый метод получения булевых функциональных преобразований этого класса. Метод оперирует с процедурной формой представления булевых преобразований. Это позволяет строить преобразования от сотен булевых переменных. Булевы преобразования такого класса могут быть использованы для ускорения идентификации пользователей на основе концепции нулевых знаний. Установлены зависимости между временем построения преобразований и параметрами процедурной формы.

This paper presents investigation of nonlinear Boolean transformations inverse for which are ambiguous and its application in cryptographical algorithms. A new method for designing such class of Boolean transformations is suggested. The method deals with the procedure form representation of Boolean transformations. It allowed to buil Boolean transformatiom from hundreds Boolean variables. Boolean transformation on such class can be use for accelerate of user identification based on “zero-knoledge” conseption. The relationship between transformation building time and procedure form parameters is established.

Введение

Динамичное расширение и углубление процессов информационной интеграции требует адекватного совершенствования методов и средств защиты данных и контроля над правами доступа к ним.

К настоящему времени в основе большей часть систем защиты информации лежит использование криптографических механизмов. В свою очередь, такие механизмы базируются на математических преобразованиях, обладающих специфическими свойствами. Эти преобразования относятся к различным областям современной математики, но наиболее часто к теории чисел, конечных полей и булевой алгебре. Булевы преобразования обеспечивают существенно более высокую производительность реализации функций защиты информации в сравнении с другими преобразованиями.

Расширение использования средств защиты данных, в том числе в системах реального времени, требует радикального повышения скорости реализации вычислений, связанных с информационной безопасностью. Это определяет важность расширения возможностей использования для защиты информации именно булевых преобразований. Такое расширение может быть достигнуто путем получение преобразований со специальными свойствами, важными для защиты данных.

Таким образом, проблема расширения возможностей использования булевых функциональных преобразований в свете требований повышению скорости реализации функций защиты информации является актуальной.

Анализ проблемы получения необратимых преобразований, обратные к которым неоднозначны

Эффективность алгоритмов защиты информации определяется двумя факторами: уровнем защищенности и объемом вычислительных ресурсов, затрачиваемых на реализацию функций защиты.

В основе всех криптографических алгоритмов защиты информации лежит аналитически неразрешимая математическая задача [1]. В большинстве случаев практического использования, такие задачи имеют вид необратимого преобразования $Y=F(X)$, то есть преобразования, для которого определена алгоритмически функция $F(X)$ вычисления в прямом направлении и не существует аналитического способа получения функции $\Phi(Y)$ обратного преобразования $X=\Phi(Y)$ по известной функции $F(X)$. Единственным способом решения таких задач, то есть нахождения значения X для заданного значения Y при известной функции $F(X)$ является перебор значений X. Большая часть аналитически неразрешимых задач, лежащих в основе современных криптографических алгоритмов относится к теории чи-

сел, и булевой алгебре. В частности, к теории чисел относятся задачи дискретного логарифмирования, на основе которых построены большинство алгоритмов несимметричного шифрования (алгоритмов с открытым ключом), в том числе, широко используемые на практике RSA, El-Gamal, EEC, а также алгоритмы цифровой подписи, такие как DSS [1]. В основе достаточно широкого класса криптографических алгоритмов лежит аналитически неразрешимая задача булевой алгебры – отыскание корней систем нелинейных булевых уравнений. К этому классу алгоритмов относятся все алгоритмы блочного симметричного шифрования, такие как DES, IDEA, Rijndael, а также большая часть хеш-алгоритмов, в том числе, наиболее распространенные на практике RC-5, SHA и RIPEMD-160 [2].

Основным достоинством алгоритмов построенных на основе аналитически неразрешимых задач теории чисел является существование нескольких ключей. Так в алгоритмах шифрования RSA, El-Gamal, EEC, ключи, используемые для прямого и обратного преобразований различны. При наличии нескольких ключей, часть из них могут использоваться как открытые, а часть – как закрытые. Это позволяет строить на этой основе существенно более эффективные механизмы защиты информации и организации доступа к информационным ресурсам по сравнению с алгоритмами, имеющими единый ключ. Именно поэтому появление в 1978 г. алгоритма с “открытым” ключом – RSA, в основе которого лежит аналитически необратимое преобразование $F(X) = A^X \bmod M$, связывают с “открытием новой эры в технологии защиты информации” [2]. В теоретическом плане, существование нескольких ключей криптографического преобразования обусловлено тем, что необратимая задача является неоднозначной, то есть, существует, по крайней мере, два ключа X_1 и X_2 для которых выполняется $F(X_1) = F(X_2)$. Основным недостатком алгоритмов защиты информации, в основе которых лежат аналитически неразрешимые задачи теории чисел является низкая скорость их реализации, обусловленная высокой вычислительной сложностью операций модулярного экспоненцирования над числами, разрядность которых составляет тысячи.

Этого недостатка лишены алгоритмы защиты информации, в основе которых лежит аналитически неразрешимая задача булевой алгебры. Рекурсивное вычисление систем булевых функций может быть организовано достаточно эффективно как программными средствами на универсальных процессорах, так и специализированными аппаратными вычислителями. В оценочном плане, скорость криптографической обработки данных алгоритмами на основе булевых функций и на основе модулярной арифметики отличается на 3-4 порядка [3]. Однако, алгоритмы на основе булевых преобразований обладают существенно меньшими функциональными возможностями, которые не позволяют реализовать с их использованием эффективные протоколы защиты информации, подобно алгоритмам на основе неразрешимых задач теории чисел. В частности, использование булевых преобразований не позволяет реализовать базовые для современных технологий защиты информации концепции несимметричного шифрования данных, цифровой подписи сообщений, идентификации на основе схемы “нулевого знания”. Одним из факторов, обуславливающих ограниченные функциональные возможности алгоритмов на основе булевых преобразований является однозначность последних [3].

Одним из наиболее перспективных направлений совершенствования технологии защиты данных является расширение функциональных возможностей алгоритмов, основанных на булевой алгебре [2]. Это позволит строить на основе таких алгоритмов эффективные протоколы защиты данных в сетях, реализация которых требует существенно меньших вычислительных ресурсов и, соответственно, может выполняться на порядки быстрее, чем при использовании алгоритмов, основанных на модулярной арифметике.

В рамках реализации этого направления совершенствования технологии защиты информации важным представляется разработка методологии получения необратимых булевых преобразований, обладающих свойством неоднозначности [3]. Такие преобразования, в частности, могут быть эффективно использованы для создания эффективных протоколов идентификации удаленных абонентов многопользовательских систем на основе концепции “нулевых знаний” [4].

Целью исследований является разработка метода получения булевых функциональных преобразований, обладающих свойствами не-

обратимости и неоднозначности обратного преобразования.

Метод получения преобразований специальных классов в процедурной форме

Для достижения поставленной цели необходимо сформировать булево функциональное преобразование $F(X)$, определенное на множестве 2^n значений n-битового бинарного вектора $X=\{x_1,x_2,\ldots,x_n\}$, $\forall i\in\{1,\ldots,n\}$: $x_i\in\{0,1\}$ результатом которого является n-битовый выходной бинарный вектор $Y=F(X)$, $Y=\{y_1,y_2,\ldots,y_n\}$, $\forall i\in\{1,\ldots,n\}$: $y_i\in\{0,1\}$. Каждую i-тую бинарную компоненту y_i выходного вектора Y можно рассматривать, как значение булевой функции $f_i(X)$, определенной на множестве значений X. Исходя из этого, можно говорить, что преобразование F(X) состоит из n булевых функций $f_1(X),f_2(X),\ldots,f_n(X)$: $F(X)=\{f_1(X),f_2(X),\ldots,f_n(X),\}$.

Для того, чтобы синтезируемое преобразование $F(X)$ обладало свойством необратимости, необходимо, чтобы каждая из булевых функций $f_1(X),f_2(X),\ldots,f_n(X)$, составляющих преобразование $F(X)$ должна быть нелинейной и зависеть от каждой из n входных переменных $X=\{x_1,x_2,\ldots,x_n\}$.

Для того, чтобы преобразование $F(X)$ обладало свойством неоднозначности, необходимо, чтобы существовало множество Ω входных векторов, на которых преобразование $F(X)$ принимает одинаковое значение: $\forall Q,G\in\Omega,\ Q\neq G$: $F(Q)=F(G)=U$.

Как известно, булевы функциональные преобразования могут быть представлены в следующих трех формах:

- в виде таблиц истинности;
- в алгебраических нормальных формах
- в процедурном виде, то есть в виде алгоритма вычисления выходного вектора Y по значению входного вектора X.

Применительно к задачам защиты информации, преимущественно, используется третья из указанных форм представления булевых функциональных преобразований, поскольку число переменных, на которых определены такие преобразования составляет сотни. Исходя из этого, предлагается формировать преобразование в процедурном виде.

В качестве базовой процедурной схемы вычисления преобразования $F(X)$ предлагается структура, показанная на рис.1.

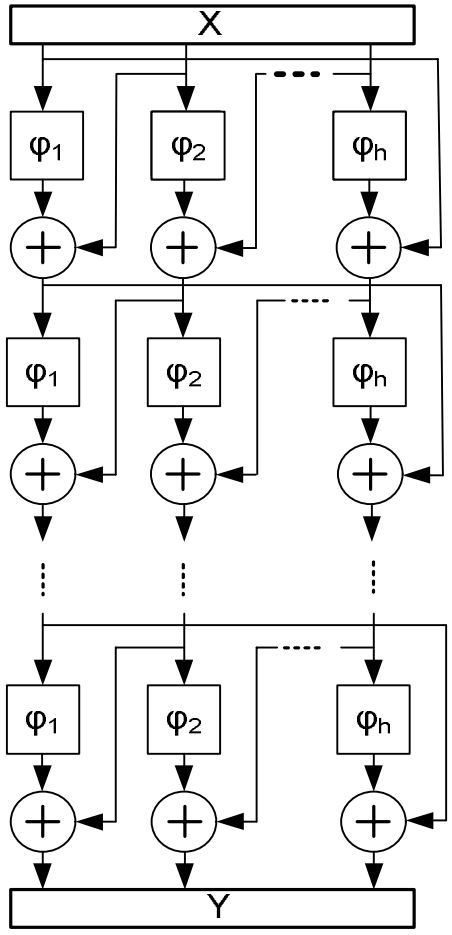


Рис.1. Структура процедурной формы вычисления булевых преобразований

Структура предполагает организацию вычисления $F(X)$ с разбиением данных на k-разрядные фрагменты. Если полагать, что n и k являются степенями 2, то число h фрагментов определяется как $h = n/k$. Подобно большинству структур, используемых для реализации булевых преобразований алгоритмов защиты информации, структура, показанная на рис.1, состоит из перемежающихся слоев нелинейного преобразования и перемешивания фрагментов. Нелинейное преобразование осуществляется с использованием h табличных булевых функциональных преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$. Каждое j-тое, $j\in\{1,\ldots,h\}$, фрагментарное преобразование $\varphi_j(V)$ определено на множестве 2^k всех возможных значений k-битового вектора $V=\{v_1,v_2,\ldots,v_k\}$, $\forall l\in\{1,\ldots,k\}$: $v_l\in\{0,1\}$. Для перемешивания результатов обработки отдельных фрагментов используется суммирование по модулю 2 результатов нелинейных преобразований смежных фрагментов. Для того, чтобы каждый бит выходного вектора Y зависел от каждого из битов входного

вектора X, число повторяющихся циклов в структуре преобразования должно быть не меньше h – числа фрагментов.

Если обозначить через Z_{qj} k-битовый вектор, являющийся j-тым фрагментом n–битового кода $W_q=\{Z_{q1},Z_{q2},\ldots,Z_{qh}\}$ формируемого на выходе q-того цикла, при этом $W_0 = X$, $W_h = Y$, то процедура вычисления функции, представленной на рис.1 аналитически может быть описана в виде:

$$\begin{aligned}
&\forall j \in \{1,...,h\} : Z_{0,j} = \{x_{(j-1)\cdot k+1}, x_{(j-1)\cdot k+2},...,x_{j\cdot k}\};\\
&\forall q \in \{1,...,h\}:\\
&\forall g \in \{1,...,g-1\} : Z_{qg} = \varphi_j(Z_{q-1,g}) \oplus Z_{q-1,g+1}\\
&Z_{qh} = \varphi_h(Z_{q-1,h}) \oplus Z_{q-1,1}
\end{aligned} \qquad (1)$$

Ниже через $W_q(Q)$ обозначено значение промежуточного кода на выходе q-го цикла для значения входного вектора $X=Q$. Соответственно, через $Z_{qj}(Q)$ обозначено код j-того фрагмента вектора $W_q(Q)$.

Задачей формирования неоднозначного и необратимого булевого преобразования $F(X)$ в рамках структуры, показанной на рис.1, является получение фрагментарных булевых функциональных преобразований – $\varphi_1,\varphi_2,\ldots\varphi_h$, которые для заранее выбранного множества Ω входных бинарных векторов $\Omega =\{X_1,X_2,\ldots,X_m\}$ обеспечивают единое значение результата, вычисляемого в соответствии с (2.10). Каждое j-тое, $j\in\{1,\ldots,h\}$, фрагментарное булево преобразование $\varphi_j(V)$ состоит из k булевых функций $\varphi_j =\{\phi_{j1}(V),\phi_{j2}(V),\ldots,\phi_{jk}(V)\}$ и может иметь результатом значения, принадлежащие интервалу от 0 до 2^k. Для решения этой задачи предлагается специальный метод, который состоит в последовательном определении значений фрагментарных функциональных преобразований в процессе преобразования входных векторов из множества Ω. Метод использует множество Θ, в которое включается для каждого из фрагментарных функциональных преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$ список наборов их аргументов, на которых значение преобразования определяются случайным образом в процессе вычисления значения текущего входного вектора X_t.

Предлагаемый метод сводится к выполнению следующей последовательности действий:

1. Значения всех h фрагментарных преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$ на всех 2^k возможных булевых наборах значений их аргументов принимаются равными –1, что соответствует неопределенному значению. Множество $\Omega=\varnothing$.
2. Выбирается произвольный n-битовый вектор X_1, который включается во множество Ω.
3. Вычисляется значение $U = F(X_1)$ в соответствием с (2.10). При этом, на каждом q–том из h слоев процедуры, для каждого j-го фрагмента (j=1,…h), если для предшествующего фрагмента $z_{q-1,j}(X_1)$ значение фрагментарного функционального преобразования не определено, то есть $\varphi_j(z_{q-1,j}(X_1))$=-1, то оно определяется как случайное число из интервала от 0 до 2^k-1.
4. Выбирается произвольно вектор X_t. Положить $W_0(X_t)=X_t$. Множество Θ полагается пустым: $\Theta=\varnothing$. Случайным образом генерируется номер N_c слоя ($N_c=\{1,\ldots,h\}$), на котором осуществляется уравнивание (“слияние”) промежуточных значений вектора X_t и одного из векторов множества Ω с порядковым номером меньшим t. Установить номер q текущего слоя в единицу: q=1. Если N_c=1, то переход на п.7.
5. Если q<N_c-1, то вычисление промежуточных значений для каждого j–го фрагмента осуществляется следующим образом: в случае $\varphi_j(z_{q-1,j})\neq$-1, то вычисление производится в соответствии с (1). Если значение фрагментарного функционального преобразования φ_j на наборе $z_{q-1,j}$ не определено: $\varphi_j(z_{q-1,j})$= -1, то случайным образом формируется принадлежащее интервалу от 0 до 2^k-1 значение φ_j на наборе $z_{q-1,j}$, после чего вычисление осуществляется согласно (1). При этом номер j доопределяемого преобразования φ_j вместе с набором $z_{q-1,j}$ заносятся в множество Θ: $\Theta=\Theta\cup <j,z_{q-1,j}>$. После вычисления таким образом всех h фрагментов промежуточных результатов q-слоя для кода X_t осуществляется переход к следующему слою, путем присваивания $q=q$+1 и возврата на повторное выполнение пп. 5.
6. Если $q=N_c$-1, и $\varphi_j(z_{q-1,j})$= -1, то значение преобразования φ_j на наборе $z_{q-1,j}$, выбирается таким образом, чтобы на наборе $z_{q,j} = \varphi_j(z_{q-1,j})\oplus z_{q-1,j+1}$ значение преобразования φ_j было не определено, то есть выполнялось: $\varphi_j(z_{q,j})$= -1. При этом номер j доопределяемого преобразования φ_j вместе с набором $z_{q-1,j}$ заносятся в множество Θ: $\Theta=\Theta\cup <j,z_{q-1,j}>$. Если $\varphi_j(z_{q-1,j}) \neq$ -1, то вычисление $z_{q,j}$ осуществляется в соответствии с (2.10). После вычисления таким образом всех h фрагментов промежуточных результатов q-слоя для кода X_t осуществляется

переход к следующему слою, путем присваивания $q=q+1$.

7. Для каждого j-го фрагмента проверяется выполнение условие: $\varphi_j(z_{q-1,j}) = -1$, если это условие выполняется, то случайным образом $d \in \{1,\ldots,t-1\}$, после чего осуществляется переход на пп.8. Для каждого j-го фрагмента проверяется выполнение условие: $\varphi_j(z_{q-1,j}) \neq -1$, но при этом существует такое $d \in \{1,\ldots,t-1\}$, что $\varphi_j(z_{q-1,j}) \oplus z_{q-1,j+1} = z_{qj}(X_d)$. Если это условие выполняется, то осуществляется переход на пп.8. Если означенные выше условия не выполняются, то проведенный подбор значений фрагментарных функциональных преобразований для кода X_t является конфликтным. Все определенные при преобразовании X_t значения фрагментарных булевых преобразований вновь считаются неопределенными: $\forall <j,z> \in \Theta$: $\varphi_j(z) = -1$. Возврат на пп.4.

8. Для каждого j-го фрагмента выполняется следующее: если $\varphi_j(z_{q-1,j}) = -1$, то соответствующее значение фрагментарного преобразования на наборе определяется как: $\varphi_j(z_{q-1,j}(X_t)) = z_{q-1,j+1}(X_t) \oplus z_{q-1,j}(X_d)$, если $j<h$ и $\varphi_j(z_{q-1,j}(X_t)) = z_{q-1,1,}(X_t) \oplus z_{q-1,j+1}(X_d)$, если $j=h$.

9. Если $t < m$, то увеличивается на единицу номер t текущего вектора X из множества Ω: $t = t+1$. Возвращение на пп. 4.

10. Для всех h фрагментарных функциональных булевых преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$, на наборах, которые не определены раньше, установить значения, которые генерируется, как случайные целые числа, принадлежащие интервалу от 0 до 2^k-1: $\forall j \in \{1,\ldots,h\}$, $\forall z \in \{0,\ldots,2^k-1\}$: $\varphi_j(z) = -1$ определить : $\varphi_j(z) = \text{Random}(0,2^k-1)$.

Результатом работы приведенной выше последовательности действий является h таблиц фрагментарных булевых преобразований $\varphi_1,\varphi_2,\ldots\varphi_h$, которые полностью задают, в совокупности со структурой, показанной на рис.1, процедуру вычисления преобразования $F(X)$.

Вероятность того, что результатом преобразования над случайным входным кодом X_e будет получено значение, совпадающее с U равно 2^{-n}. Учитывая, что численное значение n на практике составляет сотни, то вполне очевидно, что вероятность подбора идентифицирующего кода внешним по отношению к системе лицом, при использовании предложенного преобразования, практически равна нулю.

Использование разработанного метода получения нелинейных и неоднозначных булевых преобразований иллюстрируется следующим примером. Пусть n – разрядность кодов на входе и выходе преобразования $F(X)$ равна 16, то есть $n=16$. Пусть, далее, обрабатываемые преобразованием $F(X)$ коды разделяются на 4 фрагмента ($h=4$), каждый по 4 разряда ($k=n/h=4$). Это означает, что процедура преобразования состоит из 4-х слоев. Фрагментарные табличные булевы преобразования $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ определены на множестве 16-ти возможных значений 4-х булевых переменных и каждое из низ содержит 4 булевых функции, так, что каждое из упомянутых преобразований может принимать значения от 0 до 15-ти. Пусть множество Ω состоит из 3-х 16-битовых ключей, которые могут быть представлены с использованием 16-ричной системы в следующем виде: X_1=5E96h, X_2 = B1C6h, X_3 = 7A48h. Динамика заполнения таблиц истинности фрагментарных булевых функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ в процессе обработки ключей, принадлежащих множеству Ω отражена в таблице 1. Незаполненные клетки таблицы соответствуют неопределенному значению фрагментарных булевых преобразований на соответствующих наборах входных переменных.

Результатом работы предложенного метода являются таблицы истинности фрагментарных булевых преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$.

После обработки всех 3-х кодов X_1,X_2,X_3 в значения фрагментарных функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ определены на 64% наборов значений их входных переменных. Соответственно, на 36% наборов в соответствии с пп.10 изложенного выше метода, значения функций определяются случайным образом.

Табл.1. Пример заполнения таблиц фрагментарных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$

X	Таблицы значений фрагментарных булевых преобразований							
	После обработки X_1				В окончательном виде			
	φ_1	φ_2	φ_3	φ_4	φ_1	φ_2	φ_3	φ_4
0			11		12	4	11	8
1					3	14	9	2
2					0	0	6	7
3	4				4	8	4	11
4	8				8	12	6	5
5	10				10	7	3	8
6				2	3	10	7	2
7			12	15	13	15	12	15
8					12	7	15	6
9	3		6		3	9	6	8
10		15			5	15	15	10
11		13		14	15	13	10	14
12			12		8	9	12	11
13		6		0	5	6	10	0
14		2			8	2	12	9
15					0	3	15	3

Формирование фрагментарных булевых функциональных преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ в рамках изложенного метода выполняется с использованием проб: если для при обработке очередного входного вектора $X\in\Omega$, в процессе которого используется случайное определение значений преобразований $\varphi_1,\varphi_2,\varphi_3,\varphi_4$ на определенных наборах, не удается избежать конфликта с ранее определенными значениями, то, согласно пп.7 осуществляется возврат на повторную обработку вектора X. Таким образом, основным фактором, от которого зависит время формирования нелинейного функционального преобразования является количество проб заполнения таблиц фрагментарных функциональных преобразований.
при обработке вектора окажется успешной определяется следующим выражением:

$$P_g = (1-\frac{h^2\cdot g^2}{2^{2\cdot(k+1)}})^h \qquad (2)$$

Соответственно, среднее число t_g проб, затрачиваемых для заполнения таблиц фрагментарных преобразований при обработке g–го входного вектора определяется как: $t_g = 1/P_g$. Тогда, среднее число T_m проб, затрачиваемых для заполнения таблиц фрагментарных преобразований при обработке всех m входных кодов, составляющих множество Ω, определяется как сумма среднего числа проб, требующихся для обработки каждого из m входных кодов:

$$T_m = \sum_{j=1}^{m}\frac{1}{(1-\frac{h^2\cdot j^2}{2^{2\cdot(k+1)}})^h} \qquad (3)$$

Общий объем V_T памяти таблиц фрагментарных нелинейных булевых преобразований определяется в виде:

Исходя из сказанного, важным аспектом анализа предложенного метода получения неоднозначного и необратимого функционального булевого преобразования в процедурной форме является оценка зависимости числа проб от ее параметров.

Пусть, обрабатывается g-тый по очереди входной вектор X_g множества Ω. Поскольку при обработке вектора, в среднем, определяется значения фрагментарных преобразований на половине слоев, то среднее число значений одного преобразования, определяемых при обработке одного вектора X составляет $h/2$. Из этого следует, что к концу обработки g–го входного вектора X_g в каждом из h фрагментарных преобразований будет определено, в среднем, $h\cdot g/2$ значений. Из приведенного выше описания метода (пп.7) следует, что конфликтная ситуация возникает, если при обработке j-го фрагмента $z_{q,j}$ промежуточного значения на q-том слое значения соответствующего преобразования φ_j ранее определены как на наборе $z_{q,j}$, так и на наборе $\varphi_j(z_{q,j})\oplus z_{q-1,j+1}$ для $j<h$ или наборе $\varphi_j(z_{q,j})\oplus z_{q-1,1}$ для $j=h$. То есть, для того, чтобы конфликтная ситуация возникла, необходимо, чтобы для двух фиксированных наборов значения функционального преобразования φ_j были ранее определены. Учитывая, что общее число наборов, на которых определяется φ_j равно 2^k, из них, в среднем, на $h\cdot g/2$ значение преобразования φ_j уже определено ранее, то вероятность того, что на двух фиксированных наборах значение φ_j определено ранее составляет $(h\cdot g)^2/2^{2\cdot(k+1)}$. Принимая во внимание, что для повторной обработки вектора достаточно наличие конфликтной ситуации в одном фрагменте, вероятность P_g того, что проба случайного заполнения всех h фрагментарных преобразований $\varphi_1,\varphi_2,\ldots,\varphi_h$

$$V_T = h \cdot 2^{\frac{n}{h}} \quad (4)$$

Для проверки полученной теоретическим путем зависимости времени формирования функционального преобразования в зависимости от параметров процедурной формы и числа m возможных входных векторов были проведены статистические исследования с применением программной модели. Результаты экспериментальных исследований показали, что теоретическая оценка (3) упомянутых зависимостей, в целом, соответствует результатам, полученным при статистическом моделировании. Так, при длине кода доступа n=256 при параметрах процедурной формы h=16 и k=16, ее построение для числа кодов доступа m=4000 требует около 40000 проб, что занимает не более часа работы персонального компьютера. При этом множество Ω содержит около 4000 входных векторов, для которых результат преобразования $F(X)$ одинаков. При этом в процессе построения функционального преобразования заполнено около 47% таблиц фрагментарных преобразований. Остальные наборы таблиц в соответствии с пп.10 заполняются случайным образом.

Выводы

В результате проведенных исследований предложен метод получения процедурной формы нелинейных булевых функциональных преобразований, обратное к которым обладает свойством неоднозначности. Такие преобразования позволяют использовать их в некоторых применениях вместо модулярных операций над большими числами, что обеспечивает значительное (на 2-3 порядка) повышение производительности.

Установлены зависимости между параметрами процедурной формы, временем реализации алгоритма и характеристиками булевых перобразований.

Разработанный метод позволяет получать булевы функциональные преобразования, применение которых обеспечивает повышение скорости реализации алгоритмов защиты информации и средств идентификации удаленных абонентов.

Список литературы

1. Иванов М.А., Криптографические методы защиты информации в компьютерных системах и сетях. М.: ”Кудиз-образ”, 2001.– 368 с.
2. Самофалов К.Г., Марковский А.П., Гаваагийн Улзисайхан, Бардис Н., Метод получения булевых балансных SAC-функций для систем защиты информации. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.–1998.– № 31.– С.131-140.
3. Seberry J., Zhang X., Zheng Y. Nonlinearity and propagation characteristics of balanced Boolean functions.//Information and Computation Academic Press. 1995.–Vol. 119, № 1 –P.1-13.

УДК 681.3.07

ЗАЙЧЕНКО Ю.П.,
МУРГА Н.А.

ПРИМЕНЕНИЕ СИСТЕМ НА НЕЧЁТКОЙ ЛОГИКЕ К ЗАДАЧЕ МЕДИЦИНСКОЙ ДИАГНОСТИКИ

В данной работе проводится исследование возможности применения нечётких нейронных сетей для задачи медицинской диагностики на примере применения сетей для определения раковых образований в гинекологии. Производится исследование возможностей нечёткой нейронной сети Такаги-Сугено-Канга для решения данной задачи. Так же в работе произведена оценка возможности уменьшения количества тестов, необходимых для процесса диагностирования, что и завершает данную статью.

Investigation of neural network use for medical diagnostic problem on cancer detection example in gynaecology is conducted in this work. Investigation of Takagi-Sugeno-Kang's fuzzy neural network opportunities for this problem solving is conducted. Furthermore, posibility of test's (which are necessary for diagnostics process) number decrease is conducted in this work.

Вступление

Существует очень много работ по применению нейронных сетей в задачах экономики, однако применение нейронных сетей к задаче медицинской диагностики авторам не встречалось. В то же время эта задача весьма актуальна. Применение математических моделей вообще к данной задаче предложено в [1]. Однако ещё авторы данной работы указывают на значительную ошибку, выдаваемую данной моделью и указывают на необходимость поиска более совершенной (модели). Целью данной работы будет анализ применения нечёткой нейронной сети Такаги-Сугено-Канга (TSK-сети) ([2]) к данным, полученным в результате проведения колпоскопического исследовани с целью обнаружения опухолей. Данные представляют из себя (для каждого пациента) 32 теста и 1 диагноз (диагноз – агрегированный диагноз нескольких специалистов). 32 теста рассматриваются как входы сети TSK, а диагноз – выход. Вся выборка данных разбивается на обучающую и проверочную при соотношениях 50:50, 60:40, 70:30, 80:20, 90:10 соответственно. На основании этого строится и обучается нечёткая сеть. Работу завершает анализ возможности уменьшения числа тестов, необходимых для эффективной работы системы.

Постановка задачи

Имеется выборка данных по результатам обследования пациентов. Первые 32 столбца выборки – результаты тестов. Последний столбец – диагноз, поставленный специалистами по результатам тестов. То есть, имеем для каждого случая (пациента) 32 входа и 1 выход. Выборка данных разбивается на обучающую и проверочную в соотношениях: 50:50, 60:40, 70:30, 80:20, 90:10.

В данной работе необходимо провести анализ возможности применения сети TSK для данных обучающих выборок и выполнять проверку эффективности функционирования сети на соответственных проверочных выборках при различном числе правил нечёткого вывода. По результатам экспериментов выделить закономерности и сделать выводы.

Далее, для найденного оптимального соотношения обучающей и проверочной выборки и числа правил произвести последовательное уменьшение тестов, участвующих в диагностике.

1. Применение нечёткой нейронной сети TSK

Ниже приводится описание сети, изложенное в [2].

Обобщённую схему вывода в модели TSK при использовании M правил и N переменных x_j можно представить в следующем виде:

$$R_1: \ if \quad x_1 \in A_1^{(1)}; x_2 \in A_2^{(1)}; ...; x_n \in A_n^{(1)} \quad then \quad y_1 = p_{10} + \sum_{j=1}^{N} p_{1j} x_j ;$$

$$R_M: \ if \quad x_1 \in A_1^{(M)}; x_2 \in A_2^{(M)}; ...; x_n \in A_n^{(M)} \quad then \quad y_M = p_{M0} + \sum_{j=1}^{N} p_{Mj} x_j ,$$

где $A_i^{(k)}$ – значение лингвистической переменной x_i для правила R_k с функцией принадлежности $\mu_A^{(k)}(x_i), i=\overline{1,N}; k=\overline{1,M}$.

В нечёткой сети TSK пересечение условий правила R_k определяется функцией принадлежности в форме произведения, то есть:

$$\mu_A^{(k)}(x)=\prod_{j=1}^{N}\mu_A^{(k)}(x_j). \qquad (1)$$

При M правилах вывода композиция выходных результатов сети определяется по следующей формуле (аналогично выводу Сугено):

$$y(x)=\frac{\sum_{k=1}^{M}\omega_k y_k(x)}{\sum_{k=1}^{M}\omega_k}. \qquad (2)$$

где $y_k = p_{k0}+\sum_{j=1}^{N} p_{kj}x_j$. Присутствующие в этом выражении веса ω_k интерпретируются как степень выполнения условий правила: $\omega_k=\mu_A^{(k)}(x)$, которые задаются формулами (1).

Для того чтобы сеть (далее будет иногда заменяться название нечёткая нейронная сеть TSK на, просто, сеть) смогла выполнять функцию, которая от неё ожидается, необходимо её обучить.

Это обучение подразумевает, во-первых, задание количества правил, которые будут использованы сетью TSK (аналогично, сеть и сеть TSK в изложении будут равнозначны). Во-вторых, необходимо задать начальные функции принадлежности (ФП) для каждого входа и каждого правила. В-третьих, задаться видом «то»-части нечётких правил. После выполнения данных действий можно приступать, собственно, к обучению сети.

Возникает немаловажный вопрос: как задать количество правил и ФП входов?

Здесь на помощь приходит ещё один механизм, называемый кластеризацией.

Ниже приведено описание кластеризации, предложенное в [2].

Алгоритм самоорганизации относит вектор x к соответственному кластеру данных, которые представляются центром c_i, используя соревновательное обучение.

В данной работе будет интересен такой алгоритм самоорганизации, как алгоритм разностного группирования.

Для его описания используется всё та же работа [2].

Алгоритм разностного группирования – это модификация алгоритма пикового группирования, в котором векторы, подлежащие кластеризации x_j, рассматриваются как потенциальные центры кластеров. Пиковая функция $D(x_i)$ задаётся формулой:

$$D(x_i)=\sum_{j=1}^{N}\exp\left\{-\frac{\|x_i-x_j\|^{2b}}{\left(\frac{r_a}{2}\right)^2}\right\}. \qquad (3)$$

где значение коэффициента r_a определяет сферу соседства. На значения $D(x_i)$ значительно влияют только x_j, которые находятся в пределах данной сферы.

При большой плотности точек вокруг x_i значение функции $D(x_i)$ большое. После расчёта значений пиковой функции для каждой точки x_i, выбирается вектор x, для которого мера плотности $D(x)$ окажется самой большой. Именно эта точка и становится первым центром c_1.

Выбор следующего центра c_2 возможен после исключения предыдущего центра и всех точек, лежащих в его окрестности.

Пиковая функция переопределяется следующим образом:

$$D_{new}(x_i)=D(x_i)-D(c_1)\cdot\exp\left\{-\frac{\|x_i-c_1\|^{2b}}{\left(\frac{r_b}{2}\right)^2}\right\}. \qquad (4)$$

При новом определении функции D коэффициенты r_b обозначают новые значения константы, которая задаёт сферу соседства очередного центра. Обычно придерживаются условия, что $r_b \geq r_a$.

После модификации значения пиковой функции ищется новая точка x, для которой $D_{new}(x)\to\max$. Она становится новым центром.

Процесс поиска очередного центра возобновляется после исключения всех компонент,

которые отвечают уже отобранным точкам. Инициализация завершается в момент фиксации всех центров, которые предусмотрены начальными условиями.

В соответствии с описанным алгоритмом происходит самоорганизация множества векторов x, которая состоит в нахождении оптимальных значений центров, которые представляют множество данных с минимальной погрешностью.

Если мы имеем дело с множеством учебных данных в виде пар векторов $(x_i \quad d_i)$, так как это имеет место при обучении с учителем, то для нахождения центров, которые отвечают множеству векторов d_i, достаточно сформировать расширенный вектор: $[x_i, d_i] \to x_i$.

Процесс группирования, который проводится с использованием расширенных векторов x_i, позволяет определить также и расширенные версии центров c_i.

С учётом того, что размерность каждого нового центра равна сумме размерностей векторов x и d, то в описании этого центра можно выделить часть p, соответствующую вектору x (первые N компонент) и остаток q, который отвечает вектору d. Таким образом, можно получить центры как входных переменных, так и ожидаемых выходных значений $c_i = [p_i, q_i], i = \overline{1, K}$.

Теперь необходимо на вопрос: как именно может помочь данный алгоритм для решения поставленной проблемы?

Ответ очевиден. Количество полученных кластеров после его применения и будет искомым количеством правил. Значения компонент центров кластеров будут определять центры ФП. Дисперсии ФП (берутся ФП гауссовского вида) задаются некоторыми начальными значениями, которые в дальнейшем будут корректироваться.

«Если»-часть правил построена.

Рассматривается «то»-часть.

А здесь возникает два сценария дальнейших действий: считать функцию в «то»-части либо линейной, либо константой.

Необходимо разобраться с каждым из этих случаев в отдельности.

Для начала, константа. Как задать данную константу для каждого правила? А данная задача уже решена. Выше описано применение алгоритма разностного группирования для обучения с учителем. Так вот, размерность вектора d для данного случая равна единице. Выходит, для каждого i-го правила q_i и будет искомой константой.

Сеть можно «доучить» (подкорректировать дисперсии), используя метод Back Propagation, с которым можно ознакомиться в [2].

Далее, линейная функция. Для её построения можно применить метод наименьших квадратов (МНК далее). Необходимо рассмотреть, как это можно сделать, однако, вместе с алгоритмом, который будет использоваться для обучения сети такого вида – гибридным алгоритмом обучения нечётких нейронных сетей.

Приводится выдержка из его описания, предложенного в работе [2].

В гибридном алгоритме параметры, которые подлежат адаптации, разделяются на две группы. Первая из них состоит из линейных параметров p_{kj} – «то»-части нечётких правил, а вторая – из параметров нелинейных ФП «если»-части. Уточнение параметров происходит в два этапа.

На первом этапе при фиксации отдельных значений параметров функций принадлежности (в первом цикле – это значения, которые получены путём инициализации), решая систему линейных уравнений, рассчитываются линейные параметры p_{kj} полинома TSK. При известных значениях ФП зависимость для выхода можно представить в виде линейной формы относительно параметров p_{kj}:

$$y = \sum_{k=1}^{M} \omega'_k \left(p_{k0} + \sum_{j=1}^{N} p_{kj} x_j \right). \qquad (5)$$

где

$$\omega'_k = \frac{\prod_{j=1}^{N} \mu_A^{(k)}(x_j)}{\sum_{r=1}^{M} \prod_{j=1}^{N} \mu_A^{(r)}(x_j)}, \; k = \overline{1, M}. \qquad (6)$$

При размерности обучающей выборки $L\left(x^{(l)}, d^{(l)}\right)$, $(l = \overline{1, L})$ и замене выходного сигнала сети ожидаемым значением $d^{(l)}$, получаем систему из L линейных уравнений вида:

$$\begin{bmatrix} \omega'_{11} & \omega'_{11}\cdot x_1^{(1)} & \dots & \omega'_{11}\cdot x_N^{(1)} & \dots & \omega'_{1M} & \omega'_{1M}\cdot x_1^{(1)} & \dots & \omega'_{1M}\cdot x_N^{(1)} \\ \omega'_{21} & \omega'_{21}\cdot x_1^{(2)} & \dots & \omega'_{21}\cdot x_N^{(2)} & \dots & \omega'_{2M} & \omega'_{2M}\cdot x_1^{(2)} & \dots & \omega'_{2M}\cdot x_N^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \omega'_{L1} & \omega'_{L1}\cdot x_1^{(L)} & \dots & \omega'_{L1}\cdot x_N^{(L)} & \dots & \omega'_{LM} & \omega'_{LM}\cdot x_1^{(L)} & \dots & \omega'_{LM}\cdot x_N^{(L)} \end{bmatrix} \bullet \begin{bmatrix} p_{10} \\ p_{11} \\ \dots \\ p_{1N} \\ \dots \\ p_{M0} \\ p_{M1} \\ \dots \\ p_{MN} \end{bmatrix} = \begin{bmatrix} d^{(1)} \\ d^{(2)} \\ \dots \\ d^{(L)} \end{bmatrix} \quad (7)$$

где ω'_{lj} означает уровень активации (вес) условия i-го правила при предъявлении l-го входного вектора $x^{(l)}$. Это выражение можно записать в матричном виде:

$$Ap = d \quad (7)$$

Размерность матрицы A равна $L\cdot(N+1)\cdot M$. При этом количество рядов L обычно бывает значительно больше количества столбцов $(N+1)\cdot M$. Для отыскания p применим метод наименьших квадратов.

На втором этапе после фиксации линейных параметров p_{kj} рассчитываются фактические выходные сигналы $y^{(l)}, l=\overline{1,L}$, для чего используется линейная зависимость:

$$y = Ap \quad (8)$$

После этого рассчитывается вектор ошибки $\varepsilon = y - d$ и критерий:

$$E = \frac{1}{2}\sum_{l=1}^{L}\left(y\left(x^{(l)}\right) - d^{(l)}\right)^2 \quad (9)$$

Сигналы ошибок распространяются через сеть в обратном направлении соответственно к методу Back Propagation, прямо к первому слою сети (ФП), где могут быть рассчитаны компоненты вектора градиента целевой функции относительно параметров ФП. После вычисления вектора градиента делается шаг спуска градиентным методом.

После уточнения нелинейных параметров снова запускается процесс адаптации линейных параметров функции TSK (первый этап) и не нелинейных (второй этап). Этот цикл повторяется до тех пор, пока стабилизируются все параметры процесса.

2. Уменьшение количества входных переменных для диагностики

Теперь необходимо ответить ещё на один вопрос: действительно для диагностики нужны все входные переменные или их число можно уменьшить.

Необходимо определиться с тем, по какому признаку будут исключаться тесты. Здесь ответ, с одной стороны, прост и очевиден, а с другой – очень сложен. Прост он тем, что порядок исключения будет определяться величиной корреляции результатов тестов. Сложность возникает в том, что если есть пара элементов, для которых коэффициент корреляции равен 0,98001 и пара элементов, для которых коэффициент корреляции равен 0,98, то сказать точно между какими элементами больше зависимость – нельзя, так как последующие эксперименты могут существенно изменить картину. Однако в данной работе считается, что чем больше корреляция, тем строго больше зависимость между элементами и рассмотрение данной проблемы оставлено для последующих работ.

Строится корреляционная матрица для всех тестов – A, где a_{ij} – коэффициент корреляции между тестами i и j.

Далее, допускается, что найден максимальный элемент в матрице A – a_{ij}.

Выбирается из i-го и j-го (произвольно) какой тест оставляется для рассмотрения, а второй – исключается.

Допускается, что исключается тест под номером i.

Из матрицы A вычёркивается строка и столбец с индексом i.

Далее, необходимо определиться с критерием, не выполнение которого будет свидетельствовать о том, что текущее количество тестов недостаточно для постановки диагноза.

Допустима ошибка диагностики 0,005. Следовательно, на проверочной выборке должно выполнятся условие

$\max_i \left| y_i - \overline{y_i} \right| < 0{,}005$, где y_i – значение, поставленное врачами; $\overline{y_i}$ – диагноз, выданный системой. Принимается данное выражение за вышеуказанный критерий и обозначается как K.

3. Результаты экспериментов

Теперь можно перейти к практическому рассмотрению указанных выше вопросов.

Прежде всего, производится, как было сказано ранее, кластеризация и, одновременно, строится сеть с константой в «то»-части.

Все данные, которые имеются (193 пациента) отображаются в единичный гиперкуб. $r_a = r_b$. И в таблице будет обозначено RI (range influence).

Количество полученных правил в таблице будет обозначено RQ (rules quantity).

Критерием эффективности работы алгоритма берётся $RMSE = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(y_i - \overline{y_i}\right)^2}$.

Дисперсии для гаусовских функций принадлежности для каждого входа будут свои и одинаковые.

Результаты экспериментов заносятся в нижеприведённые таблицы. Отметим отдельно, что строка «Об:Пров» обозначает отношение обучающей выборки к проверочной.

Таблица 1

Об:Пров	50:50			60:40		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	64	$1{,}8315\cdot10^{-17}$	0,0332	64	$1{,}5072\cdot10^{-17}$	0,0414
0,2	64	$2{,}6832\cdot10^{-17}$	0,0312	64	$3{,}1898\cdot10^{-17}$	0,0389
0,3	64	$3{,}2746\cdot10^{-17}$	0,0297	64	$3{,}7296\cdot10^{-17}$	0,0371
0,4	64	$5{,}9620\cdot10^{-17}$	0,0282	64	$4{,}5869\cdot10^{-17}$	0,0351
0,5	62	$6{,}0560\cdot10^{-17}$	0,0277	61	$5{,}4587\cdot10^{-17}$	0,036
0,6	55	$2{,}3649\cdot10^{-16}$	0,0623	57	$3{,}6669\cdot10^{-16}$	0,0387
0,7	47	$2{,}1801\cdot10^{-16}$	0,0552	47	$1{,}6546\cdot10^{-16}$	0,047
0,8	32	$4{,}3250\cdot10^{-16}$	0,0376	33	$2{,}2483\cdot10^{-16}$	0,0637
0,9	21	$2{,}7109\cdot10^{-16}$	0,0504	27	$4{,}2174\cdot10^{-16}$	0,0776
1	19	$6{,}3821\cdot10^{-16}$	0,0549	20	$7{,}7099\cdot10^{-16}$	0,0912

Таблица 2

Об:Пров	70:30			80:20		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	70	$1{,}4754\cdot10^{-17}$	0,0435	89	$1{,}2581\cdot10^{-17}$	0,0331
0,2	70	$2{,}1681\cdot10^{-17}$	0,0418	89	$2{,}0689\cdot10^{-17}$	0,0302
0,3	70	$3{,}3746\cdot10^{-17}$	0,0388	89	$2{,}9218\cdot10^{-17}$	0,0293
0,4	69	$4{,}7760\cdot10^{-17}$	0,0365	88	$4{,}1680\cdot10^{-17}$	0,0286
0,5	68	$5{,}2207\cdot10^{-17}$	0,0358	85	$5{,}0317\cdot10^{-17}$	0,0269
0,6	64	$5{,}6841\cdot10^{-17}$	0,0386	74	$8{,}3585\cdot10^{-17}$	0,0290
0,7	51	$2{,}0565\cdot10^{-16}$	0,0534	57	$2{,}3107\cdot10^{-16}$	0,047
0,8	37	$4{,}0681\cdot10^{-16}$	0,0967	34	$6{,}5636\cdot10^{-16}$	0,0956
0,9	28	$4{,}5956\cdot10^{-15}$	0,1212	25	$9{,}5112\cdot10^{-16}$	0,0865
1	6	$1{,}2788\cdot10^{-15}$	0,1691	5	$2{,}3033\cdot10^{-15}$	0,1637

Учебная выборка – 90%, проверочная – 10%.

Таблица 3

RI	RQ	RMSE train	RMSE test
0,1	96	$1{,}1710\cdot10^{-17}$	$2{,}9216\cdot10^{-18}$
0,2	96	$2{,}1647\cdot10^{-17}$	$5{,}9195\cdot10^{-17}$
0,3	96	$2{,}6863\cdot10^{-17}$	$8{,}1233\cdot10^{-17}$
0,4	96	$3{,}3784\cdot10^{-17}$	$9{,}6381\cdot10^{-17}$
0,5	92	$4{,}4979\cdot10^{-17}$	$1{,}3867\cdot10^{-16}$
0,6	80	$1{,}6441\cdot10^{-16}$	$3{,}0583\cdot10^{-16}$
0,7	61	$2{,}2497\cdot10^{-16}$	$4{,}5655\cdot10^{-16}$
0,8	44	$5{,}5951\cdot10^{-16}$	$1{,}3725\cdot10^{-15}$
0,9	31	$1{,}0292\cdot10^{-15}$	$3{,}4575\cdot10^{-15}$
1	5	$1{,}7631\cdot10^{-15}$	$3{,}3432\cdot10^{-15}$

Таким образом, видно, что данный алгоритм позволяет построить очень хорошую модель диагностирования. Также видно, что качество диагностики возрастает ростом числа правил (падением индекса влияния). Однако, как видно из графиков (Рис. 2, Рис. 3), этот рост монотонным в общем случае назвать нельзя.

Если же характеризовать зависимость «рост качества диагностики – рост объёма обучающей выборки», то следует более внимательно присмотреться к графикам:

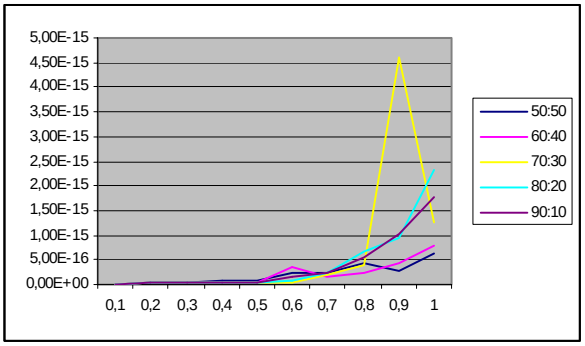


Рис. 1 Зависимость RI – RMSE train для таблиц 1 – 3

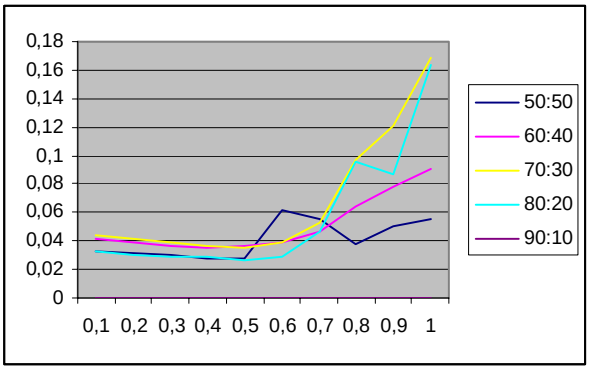


Рис. 2 Зависимость RI – RMSE test для таблиц 1 – 3

Как видно из графиков на рис. 1, для обучающей выборки при $RI = 0{,}1 \div 0{,}5$ будет точно наблюдаться прямая зависимость между качеством диагностики и числом правил нечёткого вывода, а для остальных *RI* четкой зависимости не наблюдается; а для проверочной (рис. 2) – возрастание ошибки до соотношения выборок 7:3 и дальнейшее стремительное падение оной.

Далее рассматривается случай, когда функция в «то»-части линейна.

Задаются следующие начальные условия. Правила и функции принадлежности берутся из предыдущих экспериментов (кластеризации). Обозначения в таблицах аналогичны (хотя RI уже не актуален). Обучение будет проводиться гибридным методом.

Таблица 4

Об:Пров	**50:50**			**60:40**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	64	$2{,}3976 \cdot 10^{-8}$	0,033091	64	$1{,}7003 \cdot 10^{-8}$	0,041256
0,2	64	$2{,}3981 \cdot 10^{-8}$	0,031066	64	$1{,}7008 \cdot 10^{-8}$	0,038731
0,3	64	$2{,}411 \cdot 10^{-8}$	0,029487	64	$1{,}7095 \cdot 10^{-8}$	0,036763
0,4	64	$2{,}5551 \cdot 10^{-8}$	0,027863	64	$1{,}8073 \cdot 10^{-8}$	0,034738
0,5	62	$2{,}235 \cdot 10^{-6}$	0,027419	61	$1{,}6779 \cdot 10^{-6}$	0,035122
0,6	55	$5{,}51 \cdot 10^{-6}$	0,028531	57	$3{,}931 \cdot 10^{-6}$	0,03407
0,7	47	$8{,}112 \cdot 10^{-6}$	0,030635	47	$6{,}314 \cdot 10^{-6}$	0,043796
0,8	32	$8{,}5395 \cdot 10^{-6}$	0,03862	33	$7{,}1297 \cdot 10^{-6}$	0,041018
0,9	21	$1{,}9289 \cdot 10^{-5}$	0,038845	27	$1{,}0663 \cdot 10^{-5}$	0,046861
1	19	$3{,}6898 \cdot 10^{-5}$	0,045584	20	$2{,}8545 \cdot 10^{-5}$	0,047587

Таблица 5

Об:Пров	**70:30**			**80:20**		
RI	**RQ**	**RMSE train**	**RMSE test**	**RQ**	**RMSE train**	**RMSE test**
0,1	70	$1{,}4761 \cdot 10^{-8}$	0,043158	89	$1{,}4916 \cdot 10^{-8}$	0,033651
0,2	70	$1{,}4767 \cdot 10^{-8}$	0,041357	89	$1{,}4919 \cdot 10^{-8}$	0,030443
0,3	70	$1{,}4817 \cdot 10^{-8}$	0,038156	89	$1{,}4985 \cdot 10^{-8}$	0,0293
0,4	69	$5{,}7658 \cdot 10^{-8}$	0,035841	88	$5{,}5153 \cdot 10^{-8}$	0,0288
0,5	68	$1{,}3224 \cdot 10^{-6}$	0,035571	85	$6{,}0647 \cdot 10^{-8}$	0,0291
0,6	64	$3{,}4702 \cdot 10^{-6}$	0,043127	74	$3{,}7547 \cdot 10^{-6}$	0,031
0,7	51	$4{,}9418 \cdot 10^{-6}$	0,047605	57	$6{,}8552 \cdot 10^{-6}$	0,0359
0,8	37	$7{,}3412 \cdot 10^{-6}$	0,048571	34	$1{,}0559 \cdot 10^{-5}$	0,0148
0,9	28	$1{,}0643 \cdot 10^{-5}$	0,046314	25	$1{,}7685 \cdot 10^{-5}$	0,0376
1	6	0,00022524	0,095813	5	$6{,}9307 \cdot 10^{-4}$	0,1178

Обучающая выборка – 90%, проверочная – 10%.

Таблица 6

RI	RQ	RMSE train	RMSE test
0,1	96	$1{,}3748\cdot10^{-8}$	$7{,}3358\cdot10^{-8}$
0,2	96	$1{,}3750\cdot10^{-8}$	$7{,}3361\cdot10^{-8}$
0,3	96	$1{,}3806\cdot10^{-8}$	$7{,}3317\cdot10^{-8}$
0,4	96	$1{,}4470\cdot10^{-8}$	$7{,}3233\cdot10^{-8}$
0,5	92	$5{,}4135\cdot10^{-7}$	$1{,}3909\cdot10^{-6}$
0,6	80	$3{,}2039\cdot10^{-6}$	$9{,}8092\cdot10^{-6}$
0,7	61	$6{,}0106\cdot10^{-6}$	$2{,}4068\cdot10^{-5}$
0,8	44	$9{,}8820\cdot10^{-6}$	$3{,}0978\cdot10^{-5}$
0,9	31	$1{,}6437\cdot10^{-5}$	$6{,}3682\cdot10^{-5}$
1	5	$7{,}0728\cdot10^{-4}$	0,003

Как видно, тенденции здесь аналогичны экспериментам с алгоритмом разностного группирования. Наблюдается зависимость роста качества диагностики с ростом числа правил. Что касается зависимости между объёмом выборки и качеством диагностики, то можно сказать следующее.

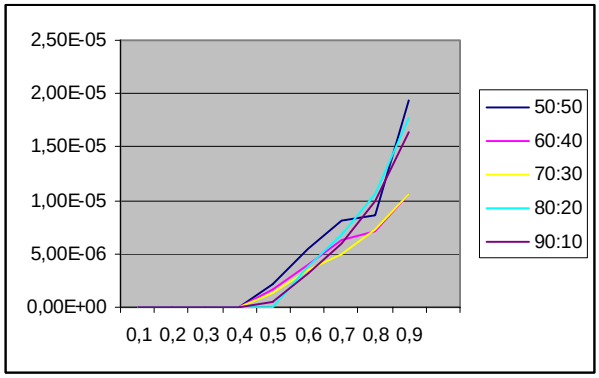


Рис. 3 Зависимость RI – RMSE train для таблиц 4 – 6

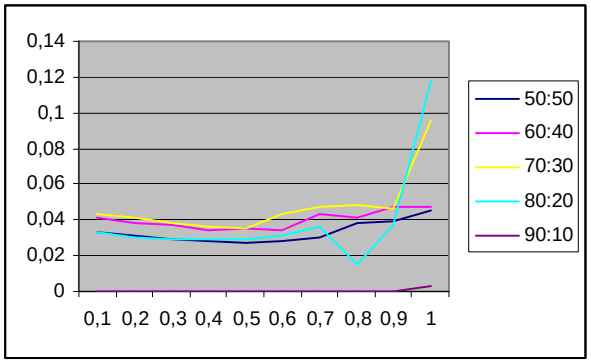


Рис. 4 Зависимость RI – RMSE test для таблиц 4 – 6

На обучающей выборке (рис. 3) эта зависимость не имеет постоянный характер и колеблется. Для проверочной выборки (рис. 4) наблюдается возрастание ошибки с ростом объёма обучающей выборки до соотношения 7:3 и дальнейшее стремительное падение ошибки.

4. Сравнительный анализ алгоритмов

Красноречивее всех описаний говорят результаты, зафиксированные в таблицах 1-3 и 4-6. При функциях-константах в «то»-части правил, полученных после кластеризации система выдавала гораздо лучшие результаты и быстрее шли вычисления, чем линейные функции, получаемые с помощью МНК (см. описание гибридного алгоритма). Необходимо отметить также, что применение для этой же задачи чёткой нейронной сети Back Propagation, рассмотренное в [3], показало, что она по параметру качество диагностики уступает нечёткой сети TSK в обоих случаях. А что касается параметра скорость обучения – уступает случаю функции-константы в «то»-части нечётких правил, но превосходит случай линейной функции.

Однако, как показывают эксперименты, у алгоритмов есть и общие свойства. Из опытов видно (таблицы 1-6), что с ростом числа правил растёт точность диагностики, как для алгоритма разностного группирования (функция-константа), так и гибридного алгоритма (линейная функция). Для обоих алгоритмов наблюдается зависимость падения точности диагностики на проверочной выборке до соотношения выборок 7:3 и дальнейший стремительный рост точности. Однако, что касается обучающей выборки, то по поводу точности диагностики можно сказать, что с ростом объёма растёт точность диагностики для случая функции-константы в «то»-части правил при $RI = 0{,}1 \div 0{,}5$, а про второй случай – чёткой закономерности не наблюдается.

5. Уменьшение числа тестов, необходимых для постановки диагноза

К диагностическим данным применяются инструкции (пункт 2), предложенные ранее

для решения вопроса о количестве тестов, необходимых для диагностики.

Берётся отношение обучающей выборки к проверочной 9:1. Для формирования системы диагностики применяется алгоритм разностного группирования. Обозначается через n – число тестов, которые используются для диагностики. RI берётся равным 0,1. K – критерий, приведённый во втором пункте.

Результаты занесены в таблицу 7.

Таблица 7

n	RQ	RMSE train	RMSE test	K
2	76	$9{,}1466\cdot10^{-15}$	$2{,}5951\cdot10^{-14}$	$3{,}6710\cdot10^{-13}$
3	88	$9{,}8243\cdot10^{-16}$	$3{,}5033\cdot10^{-15}$	$6{,}1173\cdot10^{-14}$
4	95	$2{,}9052\cdot10^{-17}$	$7{,}9815\cdot10^{-17}$	$7{,}7716\cdot10^{-16}$
5	95	$3{,}2477\cdot10^{-17}$	$7{,}6187\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
6	96	$2{,}1861\cdot10^{-17}$	$6{,}4230\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
7	96	$2{,}0654\cdot10^{-17}$	$6{,}5948\cdot10^{-17}$	$7{,}77168\cdot10^{-16}$
8	96	$3{,}5415\cdot10^{-17}$	$6{,}5741\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
9	96	$1{,}7497\cdot10^{-17}$	$3{,}9496\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
10	96	$1{,}7743\cdot10^{-17}$	$6{,}2081\cdot10^{-17}$	$8{,}8818\cdot10^{-16}$
11	96	$1{,}5131\cdot10^{-17}$	$3{,}8456\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
12	96	$2{,}0499\cdot10^{-17}$	$5{,}4912\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
13	96	$1{,}6495\cdot10^{-17}$	$6{,}60624\cdot10^{-17}$	$6{,}6613\cdot10^{-16}$
14	96	$1{,}8812\cdot10^{-17}$	$4{,}3088\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
15	96	$1{,}5806\cdot10^{-17}$	$3{,}1263\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
16	96	$1{,}4932\cdot10^{-17}$	$3{,}5029\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
17	96	$1{,}7960\cdot10^{-17}$	$3{,}9873\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
18	96	$1{,}4720\cdot10^{-17}$	$3{,}3978\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
19	96	$1{,}7069\cdot10^{-17}$	$3{,}5090\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
20	96	$1{,}2195\cdot10^{-17}$	$3{,}0116\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
21	96	$1{,}4985\cdot10^{-17}$	$3{,}0080\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
22	96	$1{,}4824\cdot10^{-17}$	$3{,}5931\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
23	96	$1{,}5059\cdot10^{-17}$	$2{,}8923\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
24	96	$1{,}3239\cdot10^{-17}$	$3{,}2038\cdot10^{-17}$	$4{,}4409\cdot10^{-16}$
25	96	$1{,}2587\cdot10^{-17}$	$2{,}3005\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
26	96	$1{,}5864\cdot10^{-17}$	$2{,}2263\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
27	96	$1{,}3964\cdot10^{-17}$	$2{,}9947\cdot10^{-17}$	$3{,}3307\cdot10^{-16}$
28	96	$1{,}6132\cdot10^{-17}$	$1{,}7530\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
29	96	$1{,}2920\cdot10^{-17}$	$1{,}4681\cdot10^{-17}$	$2{,}2204\cdot10^{-16}$
30	96	$1{,}2898\cdot10^{-17}$	$2{,}9216\cdot10^{-18}$	$5{,}5511\cdot10^{-17}$
31	96	$1{,}1404\cdot10^{-17}$	$2{,}9216\cdot10^{-18}$	$5{,}5511\cdot10^{-17}$

Таким образом, видно, что во всех опытах значение критерия K удовлетворяется и, следовательно, количество тестов можно значительно уменьшить.

Проводились ещё опыты для различных значений RI с целью добиться наименьшего числа правил при наименьшем числе тестов. Все опыты здесь приведены не будут, но вот несколько примеров:

9 тестов:

Таблица 8

RI	RQ	RMSE train	RMSE check	K
1	2	0,0233	0,0672	0,5364
0,6	11	$5{,}6436\cdot10^{-15}$	$1{,}0704\cdot10^{-14}$	$1{,}2401\cdot10^{-13}$

7 тестов:

Таблица 9

RI	RQ	RMSE train	RMSE check	K
1	3	0,0218	0,0621	0,6588
0,7	6	0,0164	0,0647	0,7306
0,5	32	$5{,}784\cdot10^{-15}$	$1{,}922\cdot10^{-14}$	$1{,}1446\cdot10^{-13}$

Эксперименты показали, что сокращать количество тестов можно. Однако не следует считать, что так же легко их можно уменьшить на практике. Во-первых, врачу необходимо обезопасить себя от поломки оборудования, которое производит тест. Во-вторых, есть заболевания, для которых значения тестов могут быть одинаковые и, поэтому, необходимо проводить дополнительные тесты для определения вида заболевания (в данном случае – действительно ли это рак, или фоновое заболевание). В-третьих, очевидно, что человек не может оперировать одновременно таким большим числом информации, которое, де-факто, требуют от него результаты экспериментов. Естественно в мозгу врача-диагноста происходят несколько отличные процессы от модели. И т.д.

Выводы

В данной работе был проведён анализ применения нечёткой нейронной сети TSK к задаче медицинской. Было проведено ряд вычислительных экспериментов, в результате которых были установлены некоторые факты. Первое, нечёткая нейронная сеть Такаги-Сугено-Канга применима для рассматриваемой задачи. Второе, для задачи, рассматриваемой в работе, применение функции-константы в «то»-части нечётких правил оказалось более эффективным, чем применение линейной функции. Третье, наблюдалась чёткая зависимость роста ошибки на проверочной выборке от соотношения выборок 50:50 до 70:30 в обоих случаях, в то же время на обучающей выборке никакие закономерности не просматривались за исключением случая функции-константы в «то»-части правила и только для $RI = \overline{0{,}1;0{,}5}$. Четвёртое, количество входов системы (тестов) можно значительно сократить. Однако на практике это не стоит делать по указанным в статье причинам. Полученные результаты подтверждают эффективность применения нечёткой нейронной сети TSK к задаче медицинской диагностики, однако ряд рассуждений показывают, что оптимальная система для решения данной задачи не найдена. Поиск данной системы и составит предмет последующих работ.

Список литературы

1. Steve Saggese, Trevor Johnson, Ivy Basco, Yalew Tamrat. Automatic Segmentation of Uterine Cervix for in vivo localization and Identification of Cervical Intraepithelial Neoplasia.– Apogen Technologies 7545 Metropolitan Dr San Diego, CA 92108.
2. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник. – К.: Видавничий Дім «Слово», 2004. – 352 с.
3. Мурга Н.А. Применение нейросетей при колпоскопическом осмотре шейки матки с целью обнаружения кансероподобных образований. Системний аналіз та інформаційні технології: Матеріали X Міжнародної науково-технічної конференції (20-24 травня 2008 р., Київ).–К.:НТУУ «КПІ» 2008.– 235 сторінка.

УДК 004.052.42

АБУ УСБАХ А.Н.,
МЕЛЬНИК А.П.,
ПОНОМАРЧУК Д.С.

ОБ ОДНОМ ПОДХОДЕ К ОБНАРУЖЕНИЮ ОШИБОК ПЕРЕДАЧИ ДАННЫХ НА ОСНОВЕ СИСТЕМ ОСТАТОЧНЫХ КЛАССОВ

Статья посвящена решению проблемы повышения эффективности выявления ошибок передачи данных в каналах со спектральной модуляцией за счет учета природы их возникновения. С целью гарантированного обнаружения канальных ошибок одной и более кратности был предложен подход, использующий модульное представление на основе Китайской теоремы об остатках. Проведенные теоретические исследования предложенного подхода позволили доказать гарантированность обнаружения символьных ошибок кратностью менее или равной числу контрольных символов в рассматриваемом модульном представлении.

The paper is dedicated to solving the efficiency increasing problem for data transmission error detection in spectrum modulation channel by properties such errors appearance accounted. For the guaranteed errors detection in one and more channel symbols the approach based on Chinese Reminder Theorem has been proposed. In the course of the theoretical researches of proposed approach the guaranteed error detection has been proved for errors quantity less or equal to number of the check symbols in modular representation.

Введение

Развитие и интеграция средств передачи цифровых данных играет важную роль на современном этапе развития компьютерных и информационных технологий. Передача данных в цифровом виде находит применение в таких областях как мобильная связь (EDGE, CDMA2000, 1xEV-DO), цифровое телевидение (DVB-T/H/C/S/S2, ATSC), телекоммуникационных системах беспроводной передачи данных (Wi-Fi, Wi-MAX, Bluetooth), компьютерные сети и др [1]. Характерной особенностью современных средств передачи данных является использование различных видов спектральной модуляции (фазовой-PSK, квадратурно-амплитудной-QAM) для повышения пропускной способности канала. При этом одно изменение фазы/амплитуды может нести в себе более одного информационного бита – канальный символ. Например: QPSK – канальный символ 2 бита, 8-PSK – 3 бита, 16-,32-,64-,128- и 256-QAM – 4,5,6,7 и 8 бит соответственно.

Рост пропускной способности каналов передачи цифровых данных осложняется помехами при передаче и требует адекватного развития средств обнаружения ошибок. Для значительной части систем передачи цифровых данных исправление ошибок выполняется путем повторной передачи блока. Для таких систем, в отличие от корректирующих кодов, фазы обнаружение ошибок и их исправления разнесены. При использовании спектральной модуляции доминируют многократные битовые искажения данных, поскольку одиночная ошибка передачи канального сигнала может изменить произвольное число бит канального символа. Эту особенность битовых искажений необходимо учитывать при создании средств контроля ошибок в каналах со спектральной модуляцией.

Таким образом, проблема повышения эффективности обнаружения многократных битовых искажений в каналах со спектральной модуляцией является актуальной и имеющей практическую значимость.

Анализ существующих средств обнаружения многократных ошибок

В каналах передачи данных со спектральной модуляцией цифровая информация фактически передается символами, каждый из которых модулируется одним канальным сигналом.

Контролируемый блок B, содержащий m бит: $B=\{b_1,b_2,\ldots,b_m\}$, $b_l\in\{0,1\}$, $l=1,\ldots,m$ можно рассматривать состоящим из $t = m/k$ канальных символов: $B=\{X_1,X_2,\ldots,X_t\}$. Каждый j-тый из этих символов X_j, $j\in\{1,2,\ldots,t\}$ включает k смежных бит $X_j=\{x_1,x_2,\ldots,x_k\}=\{b_{(j-1)\cdot k+1},b_{(j-1)\cdot k+2},\ldots,b_{j\cdot k}\}$ контролируемого блока.

При передаче цифровых данных для обнаружения ошибок наиболее часто используют циклические коды и в частности CRC (Cyclic Redundancy Check – циклическая избыточная

проверка). Как и контрольные суммы, CRC относятся к средствам блокового контроля.

Сущность контроля с использованием CRC состоит в том, что блок $B=\{b_1,b_2,\ldots,b_m\}$ представляется полином $P(B)$ степени $m+k$: $P(B)=b_1\cdot x^k+b_2\cdot x^{k+1}+b_3\cdot x^{k+2}+\ldots+b_{m-1}\cdot x^{k+m-1}+b_m\cdot x^{k+m}$. Контрольный код $R(B)$ вычисляется как остаток от деления полинома $P(B)$на образующий полином $Q(X)$ степени k CRC.

По основному критерию эффективности – надежности обнаружения ошибок, циклические коды превосходят контрольные суммы. Ошибки при передаче блока данных не обнаруживается, если полином $E(X)$, соответствующий вектору ошибки делится на образующий полином $Q(X)$ CRC без остатка. Показано [3], что все полиномы $E(X)$, соответствующие указанным ниже ошибкам не делятся на специальным образом выбранный базовый полином $Q(X)$, а следовательно, они обнаруживаются гарантированно:

1. Все искажения битов $b_1,b_2,\ldots,b_m$ нечетной кратности, если базовый полином $Q(X)$ может быть представлен в виде произведения полиномов: $Q(X)=(x+1)\cdot S(X)$;
2. Все двукратные искажения битов $b_1,b_2,\ldots,b_m$ контролируемого блока, если базовый полином $Q(X)=q_0+q_1\cdot x+q_2\cdot x^2+\ldots+q_k\cdot x^k$ содержит не менее трех ненулевых компонент.
3. Группа ошибок, локализованные в рамках k разрядов.

Для остальных ошибок показано [3], что остаток $R(B)$ представляет собой результат хеширования блока B данных в пространство 2^k всех возможных контрольных кодов. Соответственно, вероятность P_{CRC} того, что эти ошибки не будут обнаружены с использованием CRC с образующим полиномом $Q(X)$ степени k определяется как $P_{CRC}=2^{-k}$.

В каналах со спектральной модуляцией, CRC позволяет гарантированно обнаружить только однократную ошибку передачи модулированного сигнала, поскольку искажаемые при этом биты локализованы в рамках группы из k битовых позиций, а значение k на практике меньше степени образующего полинома CRC. При большей кратности ошибок модулированного сигналов, CRC не гарантирует их обнаружения.

Наряду с высокой надежностью, CRC обладает рядом недостатков, наиболее важным из которых является принципиально последовательный характер вычисления контрольного кода, что обуславливает существование ограничений на скорость выполнения операций, связанных с контролем ошибок. Этот недостаток особенно актуален в современных условиях быстрого роста скоростей передачи данных.

Другим эффективным средством обнаружения ошибок в каналах со спектральной модуляцией являются взвешенные контрольные суммы (Weighed Check Sum – WCS) [4]. В отличие от CRC, использование технологии WCS учитывает особенности возникновения битовых искажений в каналах со спектральной модуляцией и позволяет гарантированно обнаруживать битовые искажения, возникающие в двух символах. Недостатком WCS является то, что она не позволяет гарантированно обнаруживать битовые искажения, вызванные ошибочной передачей более 2-х канальных сигналов. Существенным недостатком WCS является также значительно большее по сравнению с CRC число контрольных разрядов, равное $k\cdot(1+\log_2 t)$.

Таким образом, существующие средства обнаружения многократных ошибок в каналах со спектральной модуляцией не обеспечивают эффективное решение обнаружения битовых искажений, вызванных ошибочной передачей более 2-х канальных сигналов.

Целью работы является создание эффективного способа обнаружения битовых искажений, вызванных многократными ошибками передачи канальных сигналов.

Организация контроля ошибок при представлении передаваемых данных в системах остаточных классов

Для гарантированного обнаружения ошибок в одном и более канальном символе необходимо учитывать символьную структуру передаваемых данных. Для этого предлагается использовать метод, основанный на представлении целых чисел в системах остаточных классов – СОК.

Пусть существует система n взаимнопростых чисел $\{p\}^n=\{p_1,p_2,\ldots,p_n\}$ и число P, равное их произведению:

$$P=\prod_{j=1}^{n}p_j \qquad (1)$$

Тогда, согласно Китайской теореме об остатках [1] любое число M, принадлежащее интервалу $[0…P\text{-}1]$ может быть однозначно представлено множеством остатков от деления на взаимно-простые числа, образующие систему $\{p\}^n$.

$$\forall M \in \{0,..., P-1\} : \{r_1, r_2, ..., r_n\} \Leftrightarrow M, r_j = M \bmod p_j, j = 1,..., n \quad (2)$$

где $\Leftrightarrow$ символ, обозначающий взаимно-однозначное соответствие. представление числа M (2) называется представлением числа M в системе остаточных классов (СОК). Всегда можно выбрать такую систему взаимно-простых чисел, что ее элементы $p_1,…,p_n$ будут удовлетворять условию:

$$\forall j \in \{1,...,n\} : p_j \geq 2^k \quad (3)$$

Пусть число элементов системы $\{p\}^n$ будет больше числа символов в блоке: $n>t$ $\{p\}^t = \{p_1,p_2,…,p_t\} \subseteq \{p\}^n$. Тогда, согласно (2) и (3) последовательность символов $X_1,X_2,…,X_t$, составляющих контролируемый блок B данных можно рассматривать как представление в системе остаточных классов некоторого числа A:

$$B = \{X_1,...,X_t, X_{t+1},...,X_n\} \Leftrightarrow A, \forall i \in \{1,...,t\} : X_i = A \bmod p_i \quad (4)$$

То же число A может быть представлено в системе остаточных классов на основе $\{p\}^n$:

$$\{X_1,...,X_t, X_{t+1},....,X_n\} \Leftrightarrow A, \forall j \in \{1,...,n\} : X_j = A \bmod p_j \quad (5)$$

Изложенные теоретические положения могут быть положены в основу метода контроля ошибок, возникающих в последовательных интерфейсах со спектральной модуляцией.

Сущность предлагаемого метода состоит в следующем:

1. Выбирается система $\{p\}^n$ взаимно-простых чисел, в зависимости от типа спектральной модуляции определяется число k бит, модулируемых одном сигналом, протоколом определяется число t пересылаемых символов в блоке.

2. Согласно (5) передаваемый блок B данных представляется в выбранной системе $\{p\}^n$ числом A. При этом символы $X_1,X_2,…,X_t$ являются информационной частью передаваемого блока B, а символы $X_t,…,X_n$ образуют контрольные символы. Обозначим через T передаваемый блок, представляющий собой конкатенацию информационного блока B и блока Z контрольных символов: $T = B \parallel Z$.

3. Информационная и контрольная части блока передаются приемнику.

4. Принятый блок представляет собой посимвольную сумму переданного блока T и n-символьного вектора ошибки $E=\{e_1,e_2,…,e_n\}$, где e_j – k-битовый символ: $e_j = \{\xi_{1j}, \xi_{2j},…,\xi_{kj}\}$, компоненты которого $\xi \in \{0,1\}$:

$$R = \{Y_1,...,Y_n\} = T + E \quad (6)$$

5. Проверка отсутствия ошибок в передаче блока состоит в восстановлении, согласно Китайской теореме об остатках, некоторых чисел C_1 и C_2 [2]:

$$\{Y_1, Y_2,..., Y_t\} \Leftrightarrow C_1, \{Y_1, Y_2,...., Y_n\} \Leftrightarrow C_2 \quad (7)$$

При этом, информация считается переданной без ошибок, если выполняется равенство указанных чисел C_1 и C_2: $C_1=C_2$.

Можно показать, что предложенный метод контроля гарантирует обнаружение ошибок в передаче до $(n\text{-}t)$ символов, если выполняются следующие условия:

а). Система $\{p\}^n$ является монотонно возрастающей последовательностью.

б). Произведение модулей $p_{t+1},p_{t+2},…,p_n$ контрольных символов не сравнимо с единицей по любому из модулей символов данных блока B:

$$\prod_{i=t+1}^{n} p_i \bmod p_j \neq 1, \forall j \in \{1,...,t\} \quad (8)$$

Действительно, ошибки в передаче канальных символов могут вызвать искажения битов символов информационного блока B, контрольного блока Z, информационного и контрольного блоков одновременно. Это требует доказательства для ряда частных случаев.

Частный случай А: Искажению подверглись символы только контрольного блока. В этом случае, поскольку, ошибки не затронули символов информационного блока B, то справедливо следующее:

$$\{X_1, X_2,..., X_t\} = \{Y_1, Y_2,..., Y_t\} : C_1 = A \quad (9)$$

В соответствии с Китайкой теоремой остатках, представление в системе остаточных классов однозначно, следовательно, двум различным множествам остатков соответствуют различные целые числа:

$$\{X_1,..., X_n\} \Leftrightarrow A, \{X_1,..., X_t, Y_{t+1},...., Y_n\} \Leftrightarrow C_2 \neq A \quad (10)$$

Исходя из (9) и (10) справедливо следующее: $C_1 = A \neq C_2$. Из этого следует, что $C_1 \neq C_2$,

то есть ошибка гарантированно обнаруживается, если число неверно переданных канальных сигналов не превышает *n-t*.

Частный случай B: ошибки имеют место как при передаче информационных символов, так и при передаче символов контрольного блока. В этом случае, не нарушая общности, можно рассматривать вариант локализации ошибок, при котором ошибки произошли при передаче *d* последних символов и *q* последних символов контрольного блока.

$$A \Leftrightarrow \{X_1,..., X_t\}$$
$$A \Leftrightarrow \{X_1,....X_n\}$$
$$C_1 \Leftrightarrow \{X_1,..., X_{t-d}, X_{t-d+1}+e_{t-d+1},..., X_t+e_t\} \quad (11)$$
$$C_2 \Leftrightarrow \{X_1,..., X_{t-d}, X_{t-d+1}+e_{t-d+1},...., X_t+e_t,$$
$$X_{t+1},..., X_{n-q}, X_{n-q+1}+e_{n-q+1}, X_n+e_n\}$$

Согласно Китайской теореме об остатках справедливо следующее:

$$M = \sum_{i=1}^{n} c_i, g_i, r_i \bmod P, \{r_1.r_2,..., r_n\} \Leftrightarrow M, \quad (12)$$
$$c_i = P / p_i; g_i \cdot c_i = 1 \bmod p_i$$

Выражения (11) можно представить, с учетом (12), в следующем виде:

$$A = (\sum_{i=1}^{t} c_i \cdot g_i \cdot X_i) \bmod P^t$$
$$A = (\sum_{i=1}^{t} c_i' \cdot g_i' \cdot X_i) \bmod P \quad (13)$$
$$C_1 = (\sum_{i=1}^{t} c_i \cdot g_i \cdot X_i + \sum_{j=t-d+1}^{t} c_j \cdot g_j \cdot e_j) \bmod P^t$$

где $P^t = \prod_{i=1}^{t} p_i$, $P = P^t \cdot \prod_{j=t+1}^{n} p_j = P^t \cdot P^{C_1} \cdot P^{C_2}$,

$P^{C_1} = \prod_{l=t+1}^{n-q} p_l$ – произведение модулей, соответствующих символам контрольного кода, которые переданы без ошибок.

$P^{C_2} = \prod_{h=n-q+1}^{n} p_h$ – произведение модулей, соответствующих символам контрольного кода, переданных с ошибками.

Если предположить, что ошибки не обнаруживаются, то тогда верно:

$$C_1 = C_2 \Rightarrow C_1 - A = C_2 - A \quad (14)$$

Подставив (13) в (14) получим следующее:

$$(\sum_{j=t-d+1}^{t} c_j \cdot g_j \cdot e_j) \bmod P^t =$$
$$= (\sum_{j=t-d+1}^{t} c_j' \cdot g_j' \cdot e_j + \sum_{l=n-q+1}^{n} c_l' \cdot g_l' \cdot e_l) \bmod P \quad (15)$$

Согласно (13) и (14) справедливо:

$$P = P^t \cdot P^{C_1} \cdot P^{C_2}$$
$$c_j' = c_j \cdot P^{C_1} \cdot P^{C_2}, \forall j \in \{1,...t\}$$
$$c_l' = P^t \cdot P^{C_1} \cdot \frac{P^{C_2}}{p_l} = P^t \cdot P^{C_1} \cdot P^{C_2/l}, \forall l \in \{t+1,...n\} \quad (16)$$

где $P^{C_2/l}$ – произведение модулей, соответствующих символам контрольного блока, переданным с ошибками, за исключением *l*-того символа.

Пусть произведение модулей символов информационного блока, при передаче которых не произошло ошибок, равно P^{tr}, а произведение модулей символов информационного блока, при передаче которых произошли ошибки равно - P^{te}:

$$P^t = P^{tr} \cdot P^{te}$$
$$P = P^{tr} \cdot P^{te} \cdot P^{C_1} \cdot P^{C_2}$$
$$c_j = P^{tr} \cdot \frac{P^{te}}{p_j} = P^{tr} \cdot P^{te/j} \quad (17)$$

где $P^{te/j}$ – произведение модулей, соответствующих символам информационного блока, при передаче которых произошли ошибки, кроме *j*–го.

С учетом ранее полученных выражений (15) – (17) справедливо следующее:

$$(P^{tr} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j \cdot e_j) \bmod P^{tr} \cdot P^{te} =$$
$$= (P^{C_1} \cdot (P^{C_2} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j' \cdot e_j +$$
$$+ P^{te} \cdot \sum_{l=n-q+1}^{n} P^{C_2/l} \cdot g_l' \cdot e_l)) \bmod P^{tr} P^{te\bullet} \cdot P^{C_1} \cdot P^{C_2} \quad (18)$$

В выражении (18) левая и правая части могут быть сокращены на P^{tr}, поскольку оба модуля и подмодульных выражения делятся на P^{tr}:

$$(\sum P^{te/j} \cdot g_j \cdot e_j) \bmod P^{te} =$$

$$= (P^{C_1} \cdot (P^{C_2} \cdot \sum_{j=t-d+1}^{t} P^{te/j} \cdot g_j' \cdot e_j + \quad (19)$$

$$+ P^{te} \sum P^{te/l} \cdot g_l' \cdot e_l)) \bmod P^{te} P^{C_1} P^{C_2}$$

Согласно свойству сравнений в выражении (19) правая часть делится на P^{C1}, поскольку и модуль и подмодульное выражение делятся на P^{C1}. Следовательно, и левая часть выражения (19) делится на P^{C1}.

Одним из возможных решений уравнения (19) является нулевое значение левой и правой части. Множители как левой, так и правой частей (19) не равны нулю, поскольку из (12) следует, что:

$$\begin{aligned} g_j &= 1..p_j - 1 \\ e_j &= 1..p_j - 1 \end{aligned} \quad (20)$$

Следовательно, равенство нулю левой и правой частей уравнения (19) выполняется при справедливости следующей системы:

$$\begin{aligned} g_j \cdot e_j &= p_j, \forall j \in \{t-d+1,...,t\} \\ g_j' \cdot e_j &= p_j, \forall j \in \{t-d+1,...,t\} \\ g_l' \cdot e_l &= p_l, \forall l \in \{n-q+1,...,t\} \end{aligned} \quad (21)$$

Из (21) следует: $g_j = g_j', \forall j \in \{t-d+1,...,t\}$. Из этого очевидным является следующее: $c_j \bmod p_j = c_j' \bmod p_j = c_j \cdot P^{C_1} \bmod p_j$. Из последнего следует, что $P^{C_1} \bmod p_j = 1$. Однако это противоречит условию (7). Следовательно, левая и правая части уравнения (19) не могут быть равными нулю. Левая часть выражения (19) не может превышать модуль, равный P^{te}. Достаточным условием того, чтобы выражение (19) не выполнялось является условие:

$$P^{te} < P^{C_1} \quad (22)$$

По условию а) последовательность модулей монотонно возрастает, следовательно, неравенство (22) будет иметь место, если число модулей в произведении левой части (22) не превышает число модулей в произведении правой части (22). Таким образом, будут гарантированно обнаружены ошибки передачи канальных сигналов, кратность которых не превышает $u<n-t$ ошибок в контрольных символах и $n-t-u$ ошибок передачи символов информационного блока. Таким образом, количество h гарантированно обнаруживаемых ошибок передачи символов определяется следующей формулой:

$$h = n - t \quad (23)$$

Важным аспектом эффективности предложенного метода является анализ количества контрольных разрядов, используемых для обнаружения битовых искажений, вызванных ошибками передачи канальных сигналов.

Следует учитывать, что в предложенном методе разрядность контролируемых символов и символов контроля может быть различной, при условии, что обе указанные величины кратны разрядности канального символа. Это условие обеспечивает локализацию ошибок – ошибка в одном канальном символе приведет к ошибке только в одном символе данных или контроля. В простейшем случае контролируемый символ совпадает с канальным, а разрядность контрольных символов кратна разрядности канальных символов.

Проведенный анализ показал, что предложенный метод имеет ряд ограничений накладываемых необходимостью иметь монотонно возрастающую последовательность взаимно простых модулей удовлетворяющих (8). Объективно, длина такой последовательности, а значит, и размер контролируемого блока данных непосредственно зависит от разрядности символов контроля, которая определяет верхнюю границу для модулей. Все члены последовательности должны быть $p_j < 2^{\alpha}$, $j=1..n$, где α – разрядность символов контроля. Также влияет на длину последовательности модулей разрядность контролируемых символов данных, поскольку минимальный член последовательности должен быть больше любого контролируемого символа, т.е. все члены последовательности должны быть $p_j \geq 2^k$, $j=1..n$. Значение h определяет какое число модулей будет использовано для функций контроля, следовательно уменьшает число модулей для данных. Таким образом, например, при h=2, k=4, α=8 последовательность модулей будет иметь 53 члена, из которых 2 старших будут соответствовать контрольным разрядам (16 бит контроля), следовательно, разрядам данных будет сопоставлен 51 модуль. Это позволяет контролировать блок данных длиной 51*4=204 бита. Следует отметить, что рост α приводит к экспоненциальному увеличению длинны контролируемых данных. Так, при увеличении α в два раза (32 бита контроля)

можно контролировать блок данных длиной 26156 бит. Другим способом увеличения размера контролируемого блока является использование разрядности контролируемых символов кратной разрядности канальных символов. Однако при этом рост будет линейным.

Таким образом, предложенный метод обнаружения ошибок в модемных интерфейсах со спектральной модуляцией эффективен при относительно больших значениях α и k. Учитывая, что с развитием технологии передачи цифровых данных значение k увеличивается, перспективность предложенного метода обнаружения ошибок в перспективе увеличивается. Не взирая на то, что предложенный метод не позволяет контролировать блоки данных произвольной длины, он, в отличие от CRC, обеспечивает гарантированное обнаружение ошибок и может адаптироваться под качество канала передачи путем использования большего или меньшего числа контрольных разрядов.

Выводы

Предложен метод контроля ошибок в каналах со спектральной модуляцией, основанный на представлении информационных и проверочных символов в СОК.

Метод позволяет повысить эффективность обнаружения ошибок в каналах со спектральной модуляций благодаря тому, что учитывается символьная природа таких ошибок. Предложенный метод позволяет гарантированно обнаруживать пачки битовых ошибок любой длины, а также адаптировать процедуру контроля под качество линии передачи.

Метод может использоваться при модификации существующих и разработке новых протоколов передачи цифровых данных в различных отраслях – мобильной связи, цифровом телевещании, беспроводной передачи и др.

Наиболее перспективным направлением развития предложенного подхода является разработка методов коррекции ошибок с использованием представления в СОК.

Список литературы

1. Анісімов А.В. Алгоритмічна теорія великих чисел. –К.: Академперіодика.–2001.–153 с.
2. Бєйкер А. Введение в теорию чисел. Мн.: Вышєйш. шк., –1999.–340 с.
3. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. М.: Издательский дом ”Вильямс”, 2004.– 1104 с.
4. Самофалов К.Г., Марковский А.П., Мулки Яссин Ахмед Ал Бадайнех. Обнаружение и исправление ошибок передачи данных с использованием взвешенных контрольных сумм // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 3(14).–К.,НАУ.– 2008.– С.121-128

УДК 004.415.2.043

САЛАПАТОВ В.І.

СИНТАКСИЧНИЙ АНАЛІЗ ІЗ РОЗПОДІЛОМ ЛЕКСЕМ НА ГРУПИ

Традиційний розбір операторів мов програмування із розбиттям їх на окремі лексеми як рівноправні одиниці мови достатньо ускладнює синтаксичний аналіз. Природні мови спілкування мають при їхньому сприйнятті відокремлення підмету, присудку та інших частин мови для усвідомлення сенсу речення. Дещо подібне пропонується на етапі лексичного розбору операторів мов програмування. Всі лексеми при цьому розбиваються на три групи: лексеми-об'єкти, лексеми дії та інші. Особливість мов програмування полягає у обов'язковій наявності пари лексем: лексеми-об'єкту та лексеми дії. Причому з цієї пари починаються всі оператори і обов'язковим елементом у цій парі завжди є лексема дії. Тому, якщо при лексичному аналізі з'являється лексема-об'єкт, то доречним є одночасний пошук відповідної лексеми дії. Такий підхід дозволяє значно спростити синтаксичний аналіз операторів мови та прискорити його виконання.

The traditional analysis of program language operators if with laying out of them on separate lexemes as equal in rights units of language complicates enough the syntactic analysis. The natural languages of intercourse are had at their perception of separation to the subject, to the predicate and other parts of language for the awareness of sense of suggestion. Something similar is offered on the stage of lexical analysis of program language operators. All lexemes are here divided into three groups: lexemes-objects, lexemes of action at al. A feature as programming if consists in the obligatory presence of pair of lexemes: lexeme-object and lexeme of action. Thus from this pair all operators begin and in this pair it is always the lexeme of action is a mandatory member. Therefore, if a lexeme-object appear at the lexical analysis, the simultaneous search of the proper lexeme of action is appropriate. Such approach allows considerably simplify the syntactic analysis of operators of language and accelelerate his implementation.

Ефективність засобів компіляції

Потреба в нових мовах програмування та створенні для них систем компіляції ставить задачу щодо їхньої ефективної та швидкої розробки. Сучасні системи компіляції ще й досі породжують надто надмірні коди програм. Тому методи створення ефективних систем компіляції містять великий резерв щодо підвищення продуктивності роботи комп'ютерів і на даний час залишаються вельми актуальними. Важливим компонентом систем компіляції є синтаксичний аналізатор, який перетворює текст програми у внутрішнє уявлення. Математичною моделлю аналізатора є граматика мови програмування. Контекстно вільна граматика описує всі оператори мови і складається з низки правил, у правій частині яких можуть бути присутніми нетермінали. У мовах програмування такими нетерміналами є такі поняття, як вираз, оператор або інструкція та окремі фрази від часток операторів.

Створення сучасних синтаксичних аналізаторів пов'язано з двома різновидами аналізу – згори донизу та знизу догори, а точніше їхніми різновидами LL-аналізу та LR-аналізу [1, 2]. Останні два алгоритми передбачають посимвольний перегляд тексту програми з кроком уперед. При цьому всі елементи оператора мають оброблятися за єдиним алгоритмом. Це вносить певні складнощі, оскільки нетермінали, які мають бути присутніми у певних місцях правил граматики, описуються за своїми власними правилами. Ці правила мають вигляд $G = f(T, N, P, N_s)$, де N_s – стартовий символ (спеціальний нетермінал), який визначає тип оператора. T являє собою множину терміналів (в мові програмування це константи, ідентифікатори, ключові слова, символи пунктуації і т.і.). N – це нетермінали (в нашому випадку такі поняття, як вирази, оператори та окремі частки операторів). В зв'язку з цим доречно розглядати граматику кожного нетерміналу та його обробку окремо. Так, якщо згідно з граматикою у певному місці оператора має бути вираз, то обробку оператора мови в цій частині треба виконувати згідно граматики виразу, тобто окремою підпрограмою. При цьому ознакою кінця виразу є або ключове слово з конструкції оператора, або спеціальні знаки-розподільники.

Поєднання в одному аналізі розбору оператора, розбору виразу та інших частин оператора ускладнює та уповільнює процес компіляції в цілому. Тому розробка нових ефективних засо-

бів синтаксичного аналізу у мовах програмування на даний час є все ще актуальною.

Огляд методів синтаксичного аналізу

З моменту появи компіляторів і до даного часу алгоритми синтаксичного аналізу майже не змінились. При цьому всі лексеми мови розбираються однаково згідно одного з обраних алгоритмів [1, 2, 3]. Але одні лексеми вимагають виконання певних дій (оператори, знаки арифметичних операцій, фразові ключові слова), а інші являють собою дані, над якими виконуються дії. Синтаксичний розбір операторів за конкретним алгоритмом згідно граматики кожного разу вносить певні корективи у всі попередні алгоритми синтаксичного розбору. Тому кожного разу, при створенні нової мови програмування і, відповідно, нового компілятора приходиться відтворювати такий алгоритм заново. Хоча на даний час створено багато засобів по створенню систем компіляції (компілятори компіляторів) [2], але за своєю ефективністю вони значно поступаються тим, що заново створюються. На жаль, в цьому компоненті за останні роки майже нічого не змінилось. Тому виникла нагальна потреба створити такий алгоритм аналізатора, який складався б із загальної частини, яка присутня в усіх компіляторах та спеціальної частини, де б можна було врахувати особливості конкретного компілятора. До загальної частини можна віднести обробку ключових слів, констант, змінних, знаків пунктуації та арифметичних дій. Задача полягає у розробці такого універсального алгоритму синтаксичного аналізатора, який би мав можливість налаштовуватись на конкретний алгоритм згідно граматики мови та не вносив би надмірність у машинний код.

Розподіл лексем

Огляд існуючих мов програмування показав, що всі лексеми, які являють собою термінальні символи, можна розділити на дві великі групи: лексеми-об'єкти та лексеми дії. Перші описують данні, а другі – дії над цими даними так само, як це має місце у різних системах: алгебрі, геометрії, базах даних і т.і. Таким чином, $L \rightarrow L_d + L_a$, де L – множина всіх лексем, L_d – лексеми даних, L_a – лексеми дії. Припустимість тих чи інших дій задається правилами граматики. Лексеми, які визначаються за допомогою символів, можуть бути визначені через таблиці символів, ключові слова – за допомогою таблиці ключових слів. У правій частині правил граматики для операторів мов програмування присутні нетермінальні символи, які, у свою чергу, можуть бути описані окремими правилами. До таких нетерміналів відносяться вирази та оператори мови. Вирази описуються граматикою, яка співпадає з правилами арифметичний дій, тобто існує пріоритетність виконання операторів, урахування дужок, які змінюють пріоритетність дій, та деякі інші. Вирази є одним з основних компонентів мов програмування. В залежності від типу результату у виразах припускаються дані певного типу та відповідно дії, що припустимі для обробки цих даних. Те саме виникає і при обробці операторів. За ключовими словами операторів згідно граматик розміщуються вирази. При цьому ознаками кінця виразу можуть бути як ключові слова операторів, так і спеціальні символи.

Лексеми даних – L_d обробляються лексичним аналізатором під час їх появи. Тип результату визначається припустимими терміналами дії – L_a. Ці лексеми визначають порядок їхньої обробки згідно пріоритетів. Це в значній мірі спрощує опис правил граматики. Ключові слова операторів обробляються в останню чергу, тому вони мають найнижчий пріоритет. Результатом їх обробки є породження гілок дерева поточного оператора. Треба зазначити, що даний алгоритм аналізу є модифікацією алгоритму "знизу догори", тобто від кінцівок дерева до його кореня. Стартовий символ N_s, тобто тип оператора мови, розпізнається через першу або другу лексему, за один крок роботи лексичного аналізатора, про що йдеться далі.

Модифікований стековий алгоритм

У виразах усі дії, за деяким виключенням, є бінарними, тобто вони вимагають наявності двох операндів. Через це доцільно розпізнавати пари лексем у такому порядку: лексема-об'єкт, лексема дії. Лексема-об'єкт у нашому випадку являє собою дані (для виразів це константи, змінні, масиви), а лексема дії вказує на обробку даних. Опис лексем-об'єктів у мовах програмування в основному стандартний, відмінності пов'язані, головним чином, з ідентифікаторами. Лексеми дії також мають стандартний за семантикою набір, що присутній в усіх мовах програмування, та спеціальний набір тільки для конкретної мови.

Унарними є такі дії, як унарний мінус, дужки та деякі інші. Для унарного мінуса можна ство-

рити фіктивний операнд – нуль, а для дужок – пустий операнд.

Пріоритетність дій у виразах є найбільш зрозумілою і визначає порядок виконання цих дій. Якщо пріоритет поточної лексеми дії не перевищує пріоритет попередньої, то виконується обробка попередньої лексеми дії. Інакше обробка відкладається у стек. Таким чином, маємо порядок обробки лексем дії, що представлений нижче у табл. 1.

Табл. 1

Порівняння пріоритету поточної операції з попередньою, перевірка стеку	**Дія**
не перевищує	обробка попередньої операції з поточною лексемою-об'єктом
перевищує	запис у стек попередньої пари: лексема-об'єкт, лексема дії
стек непустий, права дужка або кінець виразу	читання зі стеку пари лексем та їх обробка
стек пустий, кінець виразу	кінець обробки виразу

У цьому алгоритмі необхідно передбачити змінні під *попередню* лексему-об'єкт та лексему дії, а також під *поточну* лексему-об'єкт та лексему дії. В разі запису в стек чергової пари лексем змінні з *попередніх* лексеми-об'єкту та лексеми дії записуються в стек, на їхнє місце переписуються відповідні *поточні* лексеми. Таким чином вивільняються місця під чергові нові *поточні* лексеми з виразу при продовженні обробки виразу.

В разі запису в стек лівої дужки вона записується в стек з "пустим" операндом і продовжується обробка виразу за даним алгоритмом пріоритету. Коли зустрічається права дужка, здійснюється так би мовити зворотний хід у стеку, тобто обробка лексем у стеку з наступним їх виштовхуванням звідти. Ознакою кінця виразу є або спеціальний знак, або ключове слово відповідного оператора згідно його граматики. Кінцем обробки виразу є наявність пустого стеку та досягнення в обробці виразу кінця оператора. Обробка передбачає побудову гілок синтаксичного дерева розбору оператора.

Відсутність паритету дужок одразу ж виявляється через стек як помилка. В разі невідповідності типів операндів стандартний алгоритм має передбачати в разі необхідності перетворення типів (за допомогою спеціальних підпрограм). Тип результату виразу має відповідати опису оператора.

Якщо за основу взяти цей алгоритм, то опис граматики зводиться до опису ключових слів (стандартних та нестандартних) та їхніх семантик як стандартних, так і нестандартних.

Лексеми дії можуть являти собою як спеціальні символи, так і ключові слова, але в обох випадках їхній код складається з коду пріоритету та власно коду лексеми дії, що породжує певний тип обробки.

Ключові слова операторів мови та окремих фраз операторів повинні також визначати пріоритет обробки та відповідну семантику. Так, наприклад семантика ключового слова IF умовного оператора породжує команди порівняння та відповідного умовного переходу.

Всі ключові слова можна звести в одну таблицю та застосувати їх кодування, щоб відрізнити лексеми дії виразу від лексем дії операторів.

Синтаксичний аналіз операторів з таким розподілом лексем доречно розбити на дві частини: обробку виразів та обробку операторів. При цьому обробку виразів необхідно виділити в окрему підпрограму, що породжує дерево розбору саме виразу, а фразові ключові слова операторів мови обробляти окремо. Такий підхід дозволяє достатньо легко вносити зміни синтаксису у мови програмування. Причому окремо в обробку виразів і окремо в обробку операторів.

Синтаксис лексем-об'єктів може відрізнятись в різних мовах і охоплює ідентифікатори, константи (цілого, дійсного, символьного та інших типів) і визначається у кожному випадку окремо. Їхній опис не складає великих труднощів щодо конкретної мови.

Лексеми дії крім стандартних дій повинні передбачати ще й нестандартні згідно правил граматики. Визначення тієї чи іншої дії легко задається в таблиці ключових слів спеціальним кодом, що являє собою посилання і забезпечує перехід на відповідну частину програми синтаксичного аналізатора. Структура елементу таблиці ключових слів має наступний вигляд.

мнемоніка MNEM	тип нетерміналу TYP	код CODE	пріоритет PRIOR

Рис. 1

Такий термінал розпізнається шляхом пошуку у таблиці ключових слів за мнемонікою MNEM.Тип нетерміналу TYP визначає, до якого типу нетерміналу відноситься дана лексема –

до виразу чи до оператора. Це значною мірою спрощує перевірку правильності синтаксису оператора і дозволяє швидко виявити синтаксичні помилки. Код CODE разом з пріоритетом PRIOR визначають послідовність обробки та власно перехід на відповідний алгоритм обробки згідно коду. Таким чином, визначаючи нову лексему дії як термінал, для її обробки необхідно додати відповідний програмний фрагмент.

До стандартних дій по обробці операторів таких, як умовний оператор (IF), оператори циклу (FOR, WHILE, REPEAT), оператори введення/виведення (READ, WRITE) [3, 4] можна додавати й нестандартні оператори (наприклад побітової обробки). Для цього необхідно визначити правило обробки та додати підпрограми, які забезпечують синтаксичний розбір таких нестандартних операторів. Звернення до цих операторів забезпечується через посилання, яке подано як код CODE у таблиці ключових слів.

Ключові слова в синтаксисі операторів визначають їхню послідовність в тій чи іншій конструкції оператора мови і їхня обробка виконується у самому кінці, тобто з найнижчим пріоритетом. Як бачимо, пріоритет використовується, в основному, при обробці виразів і досить зручно вписується у загальний алгоритм синтаксичного аналізу. Причому ключові слова операторів у нашому випадку використовуються також як ознаки кінця виразів у мовних конструкціях.

Запропонований алгоритм синтаксичного розбору покладає на лексичний аналізатор розпізнавання не однієї лексеми, а пари лексем: лексеми-об'єкту та лексеми дії і повертати їх синтаксичному аналізатору для подальшого розбору. При цьому на першому кроці аналізу оператора лексичний аналізатор розпізнає стартовий символ N_s – тип оператора мови та подальшій алгоритм аналізу згідно правил граматики.

За даним алгоритмом синтаксичного аналізу було створено кросс-компілятор мови С для мікроконтролера AVR, в якому реалізовані методи машинно-залежної оптимізації коду програм.

Висновки

Запропонований алгоритм синтаксичного розбору дозволяє легко вносити корективи у граматику і, відповідно, у алгоритми обробки виразів та як власно операторів, так і сполучених операторів. За допомогою коду лексеми забезпечується посилання на підпрограму обробки, а за допомогою її пріоритету – порядок обробки у синтаксичному у дереві розбору. Оскільки аналіз здійснюється через розбір пари лексем, то швидкість обробки операторів вища, ніж за класичними алгоритмами через направлений перебір варіантів. При цьому дещо ускладнюється алгоритм лексичного аналізатора, що суттєво не впливає на загальну швидкість синтаксичного аналізу.

Другою перевагою даного методу є простота опису правил синтаксичного розбору операторів та виразів.

Перелік посилань

1. Хопкрофт Джон Э., Мотвани Раджив, Ульман Джефри Д. Введение в теорию автоматов, языков и вычислений. – М.: Издательский дом "Вильямс", 2002. – 528 с.
2. Ахо Альфред В., Сети Равви, Ульман Джефри Д. Компиляторы: принципы, технологии и инструменты. – М.: Издательский дом "Вильямс", 2003. – 768 с.
3. Себеста Роберт У. Основные концепции языков программирования. – М.: Издательский дом "Вильямс", 2001.– 672с.
4. Бен Ари М. Языки программирования. Практический сравнительный анализ. – М; Мир, 2000. – 366 с.

УДК 004.074.32

МАРКОВСКИЙ А.П.,
ПОРХУН Е.В.,
МНАЦАКАНОВ А.В.

ХЕШ-ПАМЯТЬ С ОГРАНИЧЕННЫМ ВРЕМЕНЕМ ПОИСКА ПО КЛЮЧУ

В статье предложена новая организация хеш-поиска, которая предполагает хранение ключа по одному из двух хеш-адресов. Это позволяет ограничить время поиска двумя обращениями к памяти. Предложена процедура рекурсивной записи ключей в память. Получены аналитические оценки вероятности коллизий. Проанализированы возможности использования предложенной организации хеш-поиска для статических и динамических массивов данных.

In article the new hash-searching organization such that keys may storing in one of dual hash-address has been proposed. It is allowed to limit the searching time by dual memory access. The procedure of recursion recording keys into the hash-memory has been developed. The analytical evaluation of collision probability has been obtained. The possibilities of proposed organization for hash-searching in static and dynamic arrays of data has been analyzed

Введение

Поиск по ключу традиционно является одной из базовых операций обработки информации. В последнее десятилетие многократно выросли объемы информационных массивов, в которых выполняется поиск. Расширение использования информационных технологий для управления процессами, протекающими в реальном времени имеет следствием ужесточение требований к времени выполения поиска по ключу.

Указанные факторы снижают эффективность использования в современных условиях традиционных технологий поиска, для которых время поиска зависит от объема поискового массива. В этих условиях создаются предпосылки расширения применения хеш-поиска — потенциально наиболее быстрой технологии поиска, для которой время доступа не зависит от объема данных, в которых выполняется поиск. До последнего времени широкое применение хеш-памяти сдерживалось присущими ей недостатками: наличием коллизий и избыточностью требуемой памяти. Успехи технологии изготовления интегральных микросхем привели к резкому удешевлению памяти и, тем самым, создали предпосылки к снижению значимости второго из указанных недостатков хеш-адресации.

Наличие коллизий, обусловленных совпадением хеш-адресов различных ключей, требует специальных механизмов для их разрешения. Это усложняет работу с хеш-памятью и обуславливает случайный характер времени поиска. Вместе с тем, существует достаточно широкий круг применений, прежде всего, связанных с обработкой информации в реальном времени, для которых важным является гарантированное время поиска.

Исходя из этого, на современном этапе развития информационных технологии, проблема ограничения времени поиска по ключу при использовании хеш-адресации является актуальной и значимой для практики.

Анализ известных технологий разрешения коллизий

Сущность хеш-адресации состоит в том, что адрес, по которому хранится информация в хеш-памяти, функционально зависит от этой информации. Для ключа X его хеш-адрес A определяется в результате выполнения над ключом хеш-преобразования $h(X)$: $A=h(X)$. В большинстве случаев, число ключей, хранимых в хеш-памяти меньше общего числа ее ячеек, так, что хеш-памяти присуща избыточность использования объема накопителя. Основной характеристикой избыточности является коэффициент загрузки – α, определяемый как отношение числа m хранящихся в памяти ключей к общему числу M ячеек хеш-памяти: $\alpha=m/M$ [1]. Поскольку разрядность ключей всегда больше разрядности хещ-адреса, то при их размещении в хеш-памяти возникает явление коллизии – ситуации, при которой несколько различных ключей имеют одинаковый хеш-адрес.

В современной технологии хеш-адресации целесообразно выделять два типа хеш-поиска: в динамическом и статическом массивах ключей. В первом случае коллизии неизбежны и для их

разрешения используются специальные средства. Для статических массивов ключей можно путем подбора хеш-функции добиться отсутствие коллизий. Такая хеш-адресация постоянных массивов ключей называется perfect или совершенной [2].

Эффективность динамической хеш-памяти характеризуется зависимостью скорости поиска от коэффициента α ее загрузки [1]. В свою очередь, скорость хеш-поиска характеризуется средним s_{ave} и максимальным s_{max} числом обращений к памяти, необходимыми для выполнения поиска. Необходимость разрешения коллизий увеличивает число обращений к памяти. Наиболее распространенным на практике способом разрешения коллизий является пробинг – то есть запись ключа X в первую свободную ячейку, адресуемую совокупностью кодов, получаемых в результате ряда хеш-преобразований: $h_1(X)$, $h_2(X)$, $h_3(X)$,… В простейших случаях функции ряда зависят от друг друга. Так, при линейном пробинге, каждый следующий адрес на единицу больше предыдущего: $h_i(X) = (h_1(X)+i) \bmod M$. Хотя такие функции просто вычисляются, при их использовании имеет место недостаток – вторичная группировка записей, которая существенно замедляет процесс поиска. При использовании специальных генераторов независимых хеш-функций [3] достигается теоретически наименьшее среднее число s_{ave} обращений к памяти для разрешения коллизий, которое определяется формулой:

$$s_{ave} = \frac{1}{1-\alpha} \qquad (1)$$

Основным недостатком известных технологий пробинга является то, что они не ограничивают максимальное число s_{max} обращений к памяти в процессе разрешения коллизий. Так, с вероятностью $p_s = \alpha^{s+1}$ при поиске может возникнуть необходимость в более, чем s обращениях к памяти.

Для ряда важных практических применений, связанных с обработкой информации в реальном времени, доминирующее значение имеет ограничение значения s_{max}.

Для совершенной хеш-адресации (то есть без коллизий) постоянных массивов ключей, основным критерием эффективности является зависимость времени подбора хеш-преобразования, не образующего коллизий от коэффициента α загрузки памяти.

Известно ряд технологий выполнения такого подбора [1,2,4], наиболее известными из которых являются методы Cichelli R.J. и Czech Z.J. Все они используют перебор функций хеш-преобразования. Для этих методов, среднее число проб T_α, требующееся для поиска perfect хеш-преобразования определяется следующей формулой [4]:

$$T_\alpha = e^m \cdot (1-\alpha)^{m\cdot(\frac{1-\alpha}{\alpha})+\frac{1}{2}} \qquad (2)$$

Как следует из формулы (2), основным недостатком известных методов получения совершенных хеш-преобразований для постоянных массивов ключей является большое число требуемых проб.

Целью исследований является ограничение числа требуемых обращений к хеш-памяти в процессе разрешения коллизий для динамических массивов ключей и сокращение времени получения хеш-преобразования с ограниченной длиной цепочки коллизий для постоянных массивов ключей.

Концепция хеш-памяти с ограничением коллизий

Для решения задачи ограничения времени хеш-поиска предлагается концепция хранения ключей только по одному из двух возможных хеш-адресов. Это позволяет ограничить длину цепочки возникающих коллизий и обеспечить поиск по ключу не более, чем за два обращения к накопителю.

Предлагаемая концепция предполагает организацию хеш-памяти в виде двух банков (или таблиц), обозначенных так T_1 (левый банк памяти) и T_2 (правый банк памяти), каждый из которых содержит $M/2$ ячеек памяти. Для каждого из ключей определяется один возможный хеш-адрес в банке T_1 и один возможный хеш-адрес в банке T_2. Для адресации ключа X в банках памяти используются две хеш-функции $h_1(X)$ для T_1 и $h_2(X)$ для T_2, такие, что пара значений $h_1(X)$ и $h_2(X)$ однозначно определяют ключ X.

Хранение ключа X в левом банке памяти является приоритетным. Соответственно, запись кода ключа X вначале выполняется в ячейку, адресуемую $h_1(X)$ левого банка T_1. Если она уже занята, то запись выполняется в ячейку с адресом $h_2(X)$ правого банка памяти T_2.

Поскольку коды хеш-адресов $h_1(X)$ и $h_2(X)$ ключа X однозначно определяют код самого

ключа X, то в рамках предлагаемой концепции можно существенно сократить объем используемой памяти. Для этого, при размещении ключа X в ячейку $h_1(X)$ левого банка памяти T_1, в ней реально записывается код $h_2(X)$, который вместе с адресом $h_1(X)$ однозначно определяет значение ключа X. Аналогично, при размещении ключа X в правом банке T_2 в его ячейку с адресом $h_2(X)$ записывается код $h_1(X)$. Кроме указанных кодов, которые вместе с хеш-адресом однозначно определяют ключ, в ячейках хранится теговый бит занятости.

Поиск ключа в предлагаемой хеш-памяти не более, чем за два обращения к накопителю, достигается за счет того, что часть ключа, которая вместе с адресом её ячейки однозначно определяет ключ X, может храниться только в одной из двух "доступных для хранения" ячеек: по адресу $h_1(X)$ в банке T_1, либо по адресу h_2(X) в банке T_2.

Алгоритм поиска по ключу X состоит в следующем:

1. Для заданного ключа X вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$

2. Считывание кода из ячейки, адресуемой h_1(X) левого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

3. Сравнение считанного кода с $h_2(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом.

4. Считывание кода из ячейки, адресуемой h_2(X) правого банка памяти. Если теговый бит считанного кода равен нулю (ячейка свободна), то поиск завершен с отрицательным результатом.

5. Сравнение считанного кода с $h_1(X)$. Если сравниваемые коды совпадают, то поиск завершен с положительным результатом. В противном случае – поиск завершен с отрицательным результатом.

Выполнение условия хранения каждого из ключей в одной из двух ячеек хеш-памяти обеспечивает длину цепочки коллизий не больше двух, Соответственно, состояние хеш-памяти, для которого выполняется это условие, может быть названо состоянием ограниченных коллизий (СОТ).

Запись ключа X в хеш-памяти всегда осуществляется в одну из двух доступных для его хранения ячеек. Однако, может возникнуть ситуация, при которой обе эти ячейки хеш-памяти заняты другими, ранее записанными ключами.

Схематично такая ситуация показана на рис.1. Здесь через a ,b обозначены ячейки левого банка памяти T_1, а через v,z и w – ячейки правого банка T_2. До записи ключа X_4 в хеш-память записаны: ключ X_1, для которого $h_1(X_1)=a$, $h_2(X_1)=z$, записывается в ячейку a левого банка T_1. Ключ X_2, для которого допустимыми являются ячейки b и w ($h_1(X_2)=b$, $h_2(X_2)=w$) помещается в свободную ячейку b левого банка T_1. Для ключа X_3, ($h_1(X_3)=b$, $h_2(X_3)=v$), допустимыми являются ячейки b и v. Однако ячейка b левого банка T_1 уже занята ключом X_2, поэтому ключ X_3 помещается в ячейку v правого банка T_2. Ситуация после записи трех ключей X_1, X_2 и X_3 показана на рис.1.

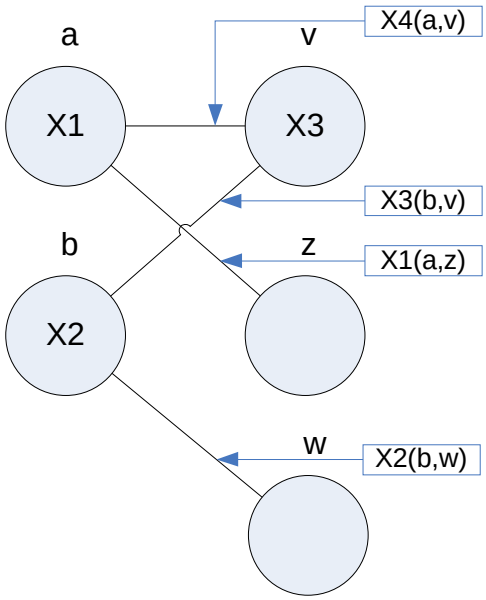


Рис. 1. Запись ключей X_1, X_2, X_3 в память

При записи ключа X_4, для которого $h_1(X_4)=a$ и $h_2(X_4)=v$, возникает ситуация, когда обе допустимых для него ячейки – a в банке T_1 и v в банке T_2 заняты.

Для разрешения такой ситуации предлагается использовать рекурсивное перемещение в памяти ранее записанных ключей с тем, чтобы освободить место для нового ключа.

В рамках рассматриваемого примера, при невозможности записи ключа X_4 производится анализ возможности перемещения ключа X_1, занимающего ячейку a, на которую претендует ключ X_4 в левом банке. Поскольку альтернативная для ключа X_1 ячейка z правого банка T_2 свободна, то ключ X_1 переписывается в нее (фактически из ячейки $a=h_1(X_1)$ считывается код $h_2(X_1)$, адресующий ячейку $z=h_2(X_1)$ правого

банка памяти T_2, которую записывается код $a=h_1(X_1)$). После освобождения ячейки a, в нее помещается ключ X_4 (фактически в ячейку a записывается $h_2(X_4)$). Ситуация после записи ключа X_4 схематично показана на рис.2.

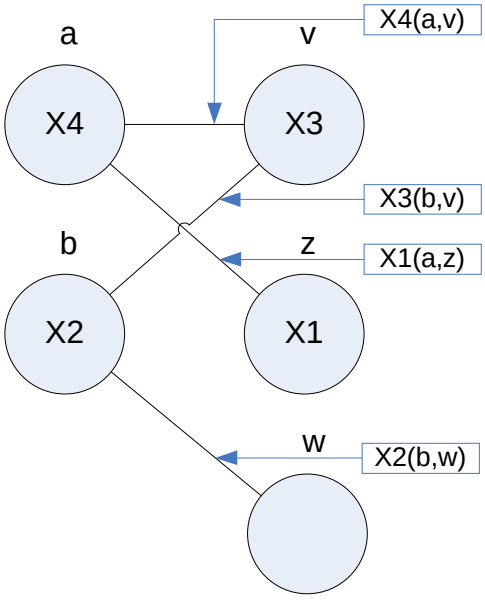


Рис. 2. Запись ключа X_4 в память

Для выполнения рекурсивной записи ключей в хеш-память предлагается формализоВанный алгоритм. Для его описания используется переменная t ($t \in \{1,2\}$), которая соответствует номеру банка памяти, с которым осуществляется работа в текущий момент времени. Для работы алгоритма необходимо организовать стек, в котором размещаются записываемые и перемещаемые в процессе рекурсии ключи. Каждая из ячеек стека содержит два поля L, R для записи адресов в банках хеш-памяти и однобитовое поле nb для сохранения номера банка памяти.

Алгоритм записи ключа X сводится к выполнению последовательности действий:

1. Вычисляются значения хеш-функций $h_1(X)$ и $h_2(X)$, устанавливается значение t=1.

2. Из ячейки по адресу $h_1(X)$ в левом банке памяти T_1 считывается код d ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_2(X)$ и теговый бит β равный единице. Операция записи заканчивается. Иначе переход на пп.4.

3. Из ячейки по адресу $h_2(X)$ правого банка памяти T_1 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Операция записи заканчивается.

4. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля L,R, nb стека. Если код d совпадает с одним из кодов, размещенных в поле R стека, то выполнить запись невозможно.

5. Модифицируется код $h_2(X)$: $h_2(X)=d$. Значение переменной t изменяется: $t = 3-t$.

6. Из ячейки по адресу $h_2(X)$ правого банка памяти T_2 считывается код g ссылки и теговый бит β. Если теговый бит β равен нулю: β=0, то есть ячейка не занята, то в нее записывается код $h_1(X)$ и теговый бит β равный единице. Переход на пп.10.

7. Коды $h_1(X)$, $h_2(X)$ и t помещаются в поля L,R, nb стека. Если код g совпадает с одним из кодов, размещенных в поле L стека, то выполнить запись невозможно.

8. Модифицируется код $h_1(X)$: $h_1(X)=g$. Значение переменной t изменяется: $t = 3-t$.

9. Из ячейки по адресу $h_1(X)$ считывается код d. Переход на пп.5.

10. Из полей L,R и nb стека выталкиваются коды соответственно $h_1(X)$, $h_2(X)$ и t.

11. Если t=1, код $h_2(t)$ и единичный бит β записывается по адресу $h_1(t)$ в левый банк памяти T_1. Переход на пп. 13.

12. Если t=2, код $h_1(t)$ и единичный бит β записывается по адресу $h_2(t)$ в правый банк памяти T_2.

13. Если стек не пуст, значение переменной t изменяется: $t = 3-t$ и возврат на повторное выполнение пп.8. Иначе конец операции записи.

Количество обращений к накопителю при выполнении описанной рекурсивной процедуры записи ключа является случайным числом, зависящим от коэффициента α загрузки памяти. Средне значение числа T_W обращений к памяти при записи определяется в виде:

$$T_W = 2\cdot(1-\alpha)+\sum_{j=1}^{\infty}\alpha^j\cdot(1-\alpha)\cdot(2\cdot j+1) = \frac{2}{1-\alpha}-1 \qquad (3)$$

Анализ формулы (3) показывает, что для используемых на практике значений α: 0.7≤ α ≤ 0.9 время записи по сравнению с обычным пробингом увеличивается практически вдвое.

Важным для исследования эффективности предлагаемой концепции представляется получение оценки вероятности возникновения коллизии, как ситуации, в которой запись нового ключа с сохранением СОК невозможна.

Оценка вероятности возникновения коллизий

Для каждого из ключей формируется два хеш-адреса. Возможна редкая ситуация, когда для ограниченное подмножество Ω из *w* ключей число формируемых хеш-адресов будет меньше *w*. В этой ситуации, размещение ключей подмножества Ω согласно предложенной концепции является принципиально невозможным. То есть, существует возможность возникновения коллизий. Пример такой ситуации схематически показан на рис.3. Кружочками на этом рисунке показаны ячейки левого и правого банков памяти, а дуги – соответствуют ключам подмножества $\Omega=\{X_1,X_2,\ldots,X_6\}$, $h_1(X_1)=a$, $h_2(X_1)=v$, $h_1(X_2)=a$, $h_2(X_1)=z$; $h_1(X_3)=b$, $h_2(X_3)=v$; $h_1(X_4)=b$, $h_2(X_4)=z$; $h_1(X_5)=c$, $h_2(X_5)=v$; $h_1(X_6)=c$, $h_2(X_6)=z$.

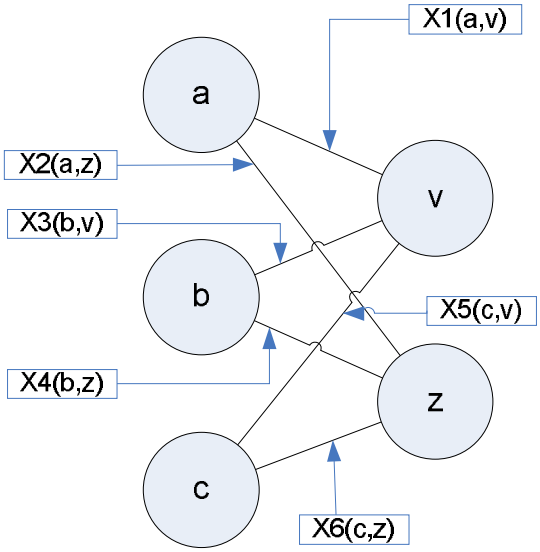


Рис. 3. Пример коллизии при размещении 6-ти ключей в 5-ти ячейках

В ситуации, показанной на рис.3, коллизия получается при размещении 6-ти ключей. При этом, в правом банке занятыми оказываются только две ячейки *v* и **z**, причем существуют ключи, адресующие каждую из упомянутых ячеек и каждую из 3-х ячеек *a*,*b*,*c* левого банка памяти. Общее число вариантов ключей при этом равно 6-ти. Очевидно, что для подмножества Ω, содержащего меньше 6-ти ключей коллизия невозможна. Действительно, если предположить, что в левом банке используется только одна ячейка, то общее число ключей, адресующих эту ячейку и *u* ячеек левого банка памяти не превысит *u*, в то время, как суммарное число ячеек составит *u*+1.

Вместе с тем, возможны и другие варианты коллизий, при которых множество Ω содержит более 6-ти ключей. Один из таких вариантов, при котором множество Ω включает 9 ключей, которые должны быть размещены в 6-ти ячейках, показан на рис.4.

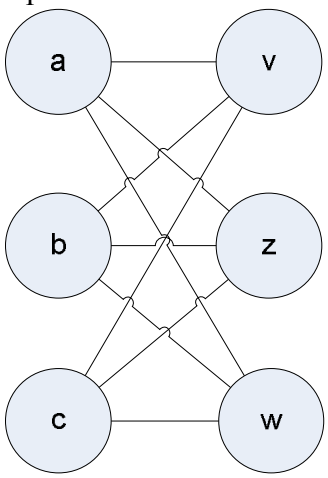


Рис. 4. Пример коллизии при размещении 9-ти ключей в 6-ти ячейках

Важным аспектом исследования эффективности предложенной организации хеш-доступа является получение оценки вероятности возникновения коллизий.

Пусть в хеш-памяти размещается *m* ключей, а общее количество ячеек в обоих банках равно *M*. Аналитическая оценка вероятности p_6 возникновения ситуации, показанной на рис.3, при которой для множества из 6-ти ключей формируется только 5 адресов, может быть выполнена следующим образом.

Пусть фиксируются две ячейки правого банка хеш-памяти, обозначенные как *v* и *z*. Если $\alpha>0.5$, то число адресных ссылок *m* ключей на правый банк памяти превышает количество *M*/2 его ячеек, то есть $m>M/2$. Это означает, что с вероятностью близкой к единице имеется ключ *X*, с адресной ссылкой *v* для правого банка памяти: $h_2(X)=v$. Адресная ссылка этого ключа *X* в левом банке адресует некоторую ячейку a, то есть $h_1(X)=a$. Аналогично, с вероятностью, близкой к единице существует ключ Y, такой, что $h_2(Y)=z$ и $h_1(Y)=b$.

Если выбрать фиксированную ячейку *c* левого банка памяти, то вероятность того, что существует ключ *G*, для которого $h_1(G)=c$ и $h_2(G)=v$ составляет M^{-1}. Вероятность того, что фиксированный ключ имеет ссылки на два фиксированных в левом и правом банках хеш-памяти адреса составляет M^{-2}. Поэтому, если фиксировать три ключа Q_1,Q_2 и Q_3, то вероятность того, что все они имеют заранее определенные адресные ссылки (в частности, для первого Q_1 из них: $h_1(Q_1)=a$ и $h_2(Q_1)=z$; для второго Q_2 : $h_1(Q_2)=b$ и $h_2(Q_2)=v$; а для третьего Q_3 из них: $h_1(Q_3)=c$, $h_2(Q_3)=z$) составляет M^{-6}. Таким

образом, вероятность того, что возникнет ситуация, показанная на рис.3 для трех фиксированных ключей Q_1,Q_2 и Q_3 и трех фиксированных ячеек: *c*, *v*, *z* составляет M^{-7}. Исходя из этого, ситуация, показанная на рис.3 не возникает, если она не имеет места для всех возможных вариантов выбора упомянутых трех ключей и всех возможных вариантов выбора ячеек *c* и *v*, *z*. Количество вариантов выбора трех ключей Q_1,Q_2 и Q_3 равно:

$$\frac{m!}{3!\cdot(m-3)!}=\frac{m\cdot(m-1)\cdot(m-2)}{6}$$

Для больших *m*, приближенно равно $m^3/6$.

Число вариантов выбора ячейки *c* равно *M*, а количество вариантов выбора пары ячеек *v* и *z* для больших *M* приближенно равно $M^2/2$. С учетом этого, вероятность q_6 того, что ситуация, показанная на рис.3 не возникает определяется выражением:

$$q_6=(1-M^{-7})^{\frac{m^3\cdot M^3}{12}}$$

Вероятность p_6 того, что при хеш-адресации множества из *m* ключей для 6-ти из них возникнет коллизийная ситуация, показанная на рис.3 определяется в виде:

$$p_6=1-q_6=1-(1-M^{-7})^{\frac{m^3\cdot M^3}{12}} \qquad (4)$$

Для имеющих место на практике больших значений *m* и *M* формула (4) может быть упрощена путем раскрытия скобок и учета только первых двух членов биномиального ряда, поскольку значения всех остальных составляют не более 5% второго члена ряда:

$$p_6\approx 1-(1-\frac{m^3\cdot M^3}{12}\cdot M^{-7})=\frac{m^3}{M^4\cdot 12}=\frac{\alpha^3}{12\cdot M} \qquad (5)$$

Проведенные экспериментальные исследования, в целом, подтвердили справедливость теоретической оценки (5) для вероятности p_6 коллизий для подмножества из 6-ти ключей.

Так, проведя аналогичные выкладки для ситуации показанной на рис.4, когда коллизия образуется при адресации 9-ти ключей, можно показать, что вероятность p_9 оценивается следующим выражением:

$$p_9\approx\frac{\alpha^3}{36\cdot M^3} \qquad (6)$$

Сравнение выражений (5) и (6) показывает, что для имеющих место на практике больших значений *M*: $p_6>>p_9$. Отсюда можно сделать вывод, что вероятность коллизии, возникающей при размещении в хеш-памяти подмножества ключей, быстро падает с увеличением числа ключей этого подмножества. Это означает, что с точностью, достаточных для оценочных расчетов, вполне можно полагать, что коллизии могут быть образованы только подмножеством из 6-ти ключей, вероятность которой оценивается полученной выше формулой (5).

Оценка эффективности применения для динамических и постоянных ключей

Предложенная концепция хеш-адресации ключей с ограничением адресов их размещения может быть использована для повышения эффективности как динамической, так и статической хеш-памяти.

Важнейшей характеристикой эффективности является время поиска, которое определяется средним s_{ave} и максимальным s_{max} числом обращений к накопителю.

Как уже отмечалось, использование предложенного подхода позволяет принципиально ограничить максимальное время поиска двумя обращениями к памяти. То есть, при использовании предложенного подхода s'_{max}=2. Как указывалось выше, при использовании пробинга, вероятность p_s того, что потребуется не менее *s* обращений к памяти определяется как $p_s=\alpha^{s+1}$. Например, при типичном на практике значении коэффициента загрузки α=0.75 при использовании пробинга с вероятностью 0.01 потребуется для поиска 28 обращений к памяти. То есть, для такой ситуации, применение предложенного подхода позволяет уменьшить максимальное время поиска в 14 раз.

Среднее число s'_{ave} обращений к накопителю зависит от коэффициентов загруженности каждого из банков хеш-памяти и определяется формулой:

$$s'_{ave}=\frac{\alpha_L+2\cdot\alpha_R}{\alpha_L+\alpha_R} \qquad (7)$$

где α_L – коэффициент загрузки левого банка памяти, а α_R – коэффициент загрузки правого банка.

Экспериментально установлено, что при малых значения α (α< 0.7) более загруженным оказывается левый банк хеш-памяти, соответственно, среднее число обращений к накопителю при поиске меньше 1.5. Так, при α=0.6 коэффициент α_L загрузки левого банка памяти состав-

ляет 0.7, в то время как для правого банка α_R = 0.5. Соответственно, при этом, среднее число s'_{ave} обращений к накопителю при поиске, в соответствии с формулой (7) составляет 1.42.

При $\alpha > 0.7$ левый и правый банки памяти заполняются практически одинаково и среднее число обращений к памяти при поиске практически равно 1.5.

Таким образом, сопоставление значений среднего числа обращений к памяти при поиске для предлагаемой концепции s'_{ave} и традиционного пробинга s_{ave} позволяет сделать следующий вывод. При $\alpha > 0.4$ (то есть практически всегда) справедливо $s'_{ave} < s_{ave}$, это означает, что применение предложенной концепции позволяет уменьшить среднее число обращений к памяти при поиске, то есть ускорить поиск. Например, при наиболее типичном значении коэффициента загрузки α=0.75 использование предложенного подхода позволяет в 2.7 раз уменьшить время поиска по сравнению с пробингом.

Следует указать, что повышение скорости поиска достигается за счет замедления процесса записи ключей. Поэтому, реальная эффективность применения предложенного подхода может быть достигнута только для класса задач, в которых интенсивность операций поиска существенно выше интенсивности операций модификации поискового массива.

При реализации предложенной концепции для хеш-адресации динамических массивов ключей необходимо предусмотреть механизм разрешение коллизий – то есть ситуации, при которой существует подмножество Ω ключей, хеш-адреса которых образуют замкнутый класс и их число меньше, чем количество ключей, составляющих множество Ω.

В качестве такого механизма предлагается использовать дополнительную память переполнения малого объема. В такой памяти предполагается размещать ключи, которые не могут быть размещены в банках хеш-памяти по своим хеш-адресам. Для того, чтобы реализовать переход к дополнительной памяти переполнения необходимо в ячейках основной памяти, помимо тегового бита занятости, выделить второй теговый бит перехода к дополнительной памяти.

Применительно к хеш-поиску в постоянных массивах ключей использование предложенного подхода позволяет значительно сократить время получения хеш-преобразования, допускающего размещение ключа в одной из двух альтернативных ячейках хеш-памяти.

Для получения такого хеш-преобразований случайно выбираются две хеш-функции $h_1(X)$ и $h_2(X)$. Массив ключей в соответствии с описанной выше рекурсивной процедурой записывается в банки хеш-памяти. Если в процессе записи возникнет коллизия – то есть невозможность размещения ключей в соответствии с предложенной концепцией, то хеш-функции $h_1(X)$ и $h_2(X)$ меняются, после чего производится повторная попытка размещения заданного массива постоянных ключей. Среднее число T_p выбора пары хеш-функций $h_1(X)$ и $h_2(X)$ исходя из (5) составит:

$$T_p = \sum_{j=1}^{\infty} j \cdot \frac{\alpha^{3\cdot(j-1)}}{12^{j-1} \cdot M^{j-1}} \cdot (1 - \frac{\alpha^3}{12 \cdot M}) = \\ = \frac{1}{1 - \frac{\alpha^3}{12 \cdot M}} \qquad (8)$$

Сопоставление формул (2) и (8) показывает, что применение предложенного подхода к хеш-адресации постоянных массивов ключей позволяет на несколько порядков ускорить подбор подходящего хеш-преобразований. Однако, необходимо иметь ввиду, что указанный значительный выигрыш во времени формирования хеш-преобразования достигнут за счет того, что при использовании предложенного подхода, в отличие от совершенной хеш-адресации, допускается два (а не одно) обращений к накопителю в процессе поиска, то есть за счет снижения скорости поиска в полтора раза.

Принимая во внимание, что число обращений к памяти, требующее для записи одного ключа определяется формулой (3), общее число N_w обращений к памяти, требующееся для размещения постоянного массива ключей в хеш-памяти определяется следующей формулой:

$$N_w = \frac{m}{1 - \frac{\alpha^3}{12 \cdot M}} \cdot (\frac{2}{1-\alpha} - 1) \qquad (9)$$

Анализ формулы (9) показывает, что использование предложенного подхода для хеш-адресации постоянных массивов ключей эффективнее существующих методов совершенной хеш-адресации при больших значениях коэффициента α загрузки памяти и при большом числе размещаемых в хеш-памяти ключей. Практическая значимость предлагаемого подхода для совершенной хеш-адресации состоит в том, что при его использовании может быть

получено решение задачи, недостижимое при применении известных методов, а именно: нахождение хеш-преобразований, которые гарантируют размещение ключей большого постоянного массива ключ по одному из двух альтернативных адресов, с коэффициентом использования памяти, близким к единице.

Выводы

В результате проведенных исследований предложена концепция организации хеш-памяти с ограниченным временем поиска по ключу. Разработаны технологические аспекты этой концепции, проведен теоретический и экспериментальный анализ ее эффективности.

Основной фактор эффективности предложенной концепции состоит в ограничении числа обращений в памяти при поиске за счет усложнения процедуры записи ключей. Доказано, что это позволяет в 2-3 раза уменьшить среднее время поиска по ключу и на порядок – максимальное время в сравнении с использованием пробинга. Это достигается за счет соответствующего увеличения времени записи ключа. Поэтому использование предложенного подхода эффективно при условии когда интенсивность операций поиска по ключу превышает интенсивность операций модификации поискового массива.

Исследованиями показано, что предложенный подход может быть использован не только для повышения скорости поиска в динамических массивах ключей, но и для многократного ускорения в сравнении с известными методами, подбора эффективных хеш-преобразований для постоянных массивов ключей.

Список литературы

1. Кохонен Т. Ассоциативная память.-М.:Мир,1980.- 198 С.
2. Марковский А.П., Гаваагийн Улзисайхан, Бардис Николас. Об одном подходе к повышению эффективности и уровня защищенности систем хранения информации на основе хеш-памяти//Вісник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка.- 1998,- № 31,- С.14-23.
3. Салех Ибрагим Аль-Омар. Использование генераторов булевых функций для повышения эффективности хеш-памяти. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.- 2003.- № 40.- С.131-140.
4. Czech Z.J., Havas G., Majevski B.S. An Optimal algorithm for generating minimal perfect hash functions.//Information processing letters. – 1997. – Vol.43.- № 5. - P.257-264.

УДК 681.3.06

ПУСТОВАРОВ В.І.

ПОБУДОВА БАЗ ЗНАНЬ ДЛЯ СУМІСНОГО СТВОРЕННЯ ПРОГРАМ І ОБЛАДНАННЯ ТА ЇХ ФОРМАЛЬНОЇ ВЕРИФІКАЦІЇ

Розглянуто підхід до комплексного розв'язання задач аналітичної підготовки аналізу і синтезу програмного забезпечення. Ключові елементи такого підходу складають формати подання заголовків специфікацій і змісту ресурсів розв'язання задач (РРЗ) і база знань, що систематизує пошук аналогів, вибір близьких компонентів, використання правил перетворень та механізми декомпозиції РРЗ. Показано, що застосування таких баз знань дозволяє максимально використати попередні розробки і гранично наблизити значення цільових семантичних змінних до вимог обмежень і критеріїв щодо характеристик обробки.

The approach to the complex task decision for analytical support of software analysis and synthesis is considered. Key elements of such approach use representation formats for specification headers contents of task decision resources (TDR) and knowledge base for analogues search, a choice of best components, use of transformation rules and decomposition mechanisms for TDR. It is shown, that application of such knowledge bases allows using the most of the previous software, and making values of target semantic variables extremely close to restriction requirements and criteria of processing speed.

Вступ. Традиційна побудова систем автоматизованого проектування (САП) програмних і апаратних ресурсів розв'язання задач (РРЗ) спирається на табличну організацію [1] вбудованих структур даних. Більшість цих таблиць, що настроюються при реалізації мови та трансляції з вхідних мов зберігають актуальність і в сучасних засобах оптимізації, синтезу та верифікації [3, 6, 7]. Перспективним підходом до комп'ютерної автоматизації сумісного проектування програм і апаратури прискорення обробки є включення баз знань (БЗ) аналітичного супроводження рішень [2] з безпосереднім використанням робочого середовища табличних і кубічних сховищ реляційних СУБД до складу САП РРЗ.

Фактично всі правила і механізми обробки можна задати комплексами відношень, накопиченими в таблицях і кубах. Головне призначення таблиць і правил БЗ в САП РРЗ – забезпечити підготовку проектних рішень на базі аналізу наявних РРЗ і вимог до розробок [1, 3]. Інформаційно-аналітичну базу (ІАБ) САП РРЗ складають комплекси табличних і кубічних сховищ для: **граматичного аналізу** об'яв, описів і реалізацій проблемних галузей досліджень (ПГД) і РРЗ; **визначення базових характеристик** якості РРЗ; **збереження** попередньо розроблених **РРЗ** разом з характеристиками їх виконання; **збереження даних про верифікацію РРЗ, дії з верифікації та осіб**, відповідальних за її виконання; **розпізнавання** проблемних областей і РРЗ за їх специфікаціями, реалізаціями та іменуваннями; **вибір** РРЗ, що найкраще відповідають заданим критеріям.

Головна проблема використання реляційних СУБД полягає у складності обробки правил, декларацій виразів і операторів комп'ютерних мов. Одним з шляхів підвищення рівня автоматизації проектування в САП РРЗ є використання полів даних спеціального аналітичного типу [4]. Примірники таких даних визначають елементи програмних модулів і їх специфікацій з накопиченням бажаних і фактичних характеристик модулів програм і моделей апаратури. В різних САП РРЗ підготовка і реалізація проектних рішень стосуються задач аналізу і синтезу об'єктів ПГД, тощо.

Визначення сховищ реляційної БД для задач вхідного синтаксичного і лексичного аналізу. Лексичний аналіз (ЛА) та синтаксичний аналіз (СА) використовують ІАБ з сукупністю таблиць і списків лексем, шаблонів, правил та кубів підсумкових даних. Основні сховища даних для граматичної обробки вхідних мов включають ***інформаційні таблиці*** $\boldsymbol{r}_{sk}$, $\boldsymbol{r}_{tk}$, $\boldsymbol{r}_{rk}$, $\boldsymbol{r}_{kk}$, $\boldsymbol{r}_{dk}$, $\boldsymbol{r}_{lk}$, $\boldsymbol{r}_{mk}$, ***координатні таблиці*** $\boldsymbol{a}_{dk}$, $\boldsymbol{a}_{sk}$, $\boldsymbol{a}_{tk}$, $\boldsymbol{a}_{ek}$, $\boldsymbol{a}_{rk}$, $\boldsymbol{a}_{lk}$, $\boldsymbol{a}_{ck}$, $\boldsymbol{a}_{fk}(x_{k1},\ldots,x_{ki_{max}}, \xi_{k1},\ldots, \xi_{kj_{max}}, a_{k1},\ldots, a_{kn_{max}})$ та ***куби*** $\prod_k \boldsymbol{a}_{fk}(\sigma_{k1}(\xi_{k1}), \ldots, \sigma_{kj}(\xi_{kj_{max}}))|x_{k1}, \ldots, x_{ki_{max}}, \xi_{k1},\ldots, \xi_{kj_{max}}$, де k – номери координат; $x_{k1}, \ldots, x_{ki_{max}}$ – поля ідентифікації; $\xi_{k1},\ldots, \xi_{kj_{max}}$ – поля семантично істотних даних; $a_{k1},\ldots,a_{kn_{max}}$ – поля додаткових даних обліку. Ключова частина структур основних таблиць та куба підсумків граматичного аналізу показана на Рис. 1. Фактично вона охоплює всі використані таблиці граматичного аналізу і при наявності таблиць

r_{tk}, r_{rk}, r_{kk}, r_{lk}, r_{mk}, a_{dk}, a_{tk}, a_{ek}, a_{rk}, a_{ck}, a_{vk} для кожної мови, реалізованої в САП РРЗ, забезпечує обробку кодів, складених на різних мовах програмування і моделювання апаратури, перетворених в єдину форму внутрішнього подання.

Рис. 1. Узагальнена інформаційно-логічна структура БЗ перетворень парсера САП

Визначення структури сховищ БЗ внутрішніх даних семантичної обробки задач. Семантична обробка використовує ІАБ, показану на Рис. 2, з сукупністю таблиць r_{cm}, r_{sm}, r_{tm}, r_{lpm}, r_{dm}, r_{lrm}, r_{lm}, r_{fm}, що включають специфікації даних, цілей, шаблонів, правил і шляхів розв'язання користувача і стандарти вбудованої семантичної обробки мов САП та РРЗ. Кубічне сховище використовується для накопичення оцінок ефективності і статистик виконання окремих РРЗ, спрямованих виключно на розв'язання внутрішніх задач ПГД. Координат-

ні таблиці куба ПГД $\boldsymbol{a}_{tm}$, $\boldsymbol{a}_{pm}$, $\boldsymbol{a}_{sm}$, $\boldsymbol{a}_{lm}$, $\boldsymbol{a}_{cm}$, $\boldsymbol{a}_{vm}$, $\boldsymbol{a}_{dm}$, $\boldsymbol{a}_{dvm}$ визначають обробку задач підтримки прийняття рішень, якщо такі розв'язуються в рамках цієї ж ПГД.

Статистики і підсумки, накопичені в кубі ПГД, використовуються на етапі проектування для пошуку найкращої реалізації всіх РРЗ.

Рис. 2. Узагальнена інформаційно-логічна структура БЗ семантичних перетворень САП РРЗ

Визначення структури сховищ БЗ даних процесу проектування РРЗ. За аналогією з етапами аналізу моделей програм побудуємо схему ІАБ для процесу проектування. Проектування РРЗ використовує ІАБ, показану на Рис. 3, з сукупністю таблиць $\boldsymbol{r}_{en}$, $\boldsymbol{r}_{pn}$, $\boldsymbol{r}_{sn}$, $\boldsymbol{r}_{tn}$, $\boldsymbol{r}_{rn}$, $\boldsymbol{r}_{an}$, $\boldsymbol{r}_{dn}$, $\boldsymbol{r}_{in}$, що зберігають дані про проекти РРЗ, плани і стан їх реалізації, шаблонів, правил та РРЗ спроектованих і в процесі проектування. Кубічне сховище використовується для нако-

пичення оцінок стану створення проектів ПГД і окремих РРЗ. Координатні таблиці куба проектів $\boldsymbol{a}_{pn}$, $\boldsymbol{a}_{zn}$, $\boldsymbol{a}_{sn}$, $\boldsymbol{a}_{nn}$, $\boldsymbol{a}_{cn}$, $\boldsymbol{a}_{dn}$, $\boldsymbol{a}_{mn}$, $\boldsymbol{a}_{fn}$ задають напрямки підтримки прийняття рішень при реалізації проектів в одній ПГД.

Рис. 3. Узагальнена інформаційно-логічна структура БЗ процесів проектування САП РРЗ

Оскільки весь процес створення РРЗ пов'язаний зі способами подання специфікацій, програм, моделей і процесів доведення [7] важливо використати узагальнений механізм формування аналітичних моделей функціональності обчислень і всіх можливих оцінок їх складності. Такий механізм створюється на базі узагальненого абстрактного типу аналітичного даних [4], в якому використовуються спеціальні канонічні

форми зберігання [4, 6] для полегшення зіставлень варіантів рішень програм і доведень.

Вимоги до канонічних форм подання кодів програм та протоколів верифікації [5] спираються на відношення лінійного порядку $\preccurlyeq$ [6] на базі кодів типів різних термінальних і нетермінальних вузлів УСГ. Термінальні вузли включають поля образів ідентифікації змінних або внутрішню форму подання константних даних:

- ***впорядковуюча нумерація кодів типів*** термінальних та нетермінальних вузлів УСГ;
- ***впорядкування або нумерація змінних*** через запам'ятовування імен або ідентифікаторів, заданих за будь-якою системою довільних евристичних правил [6];
- ***впорядкування константних даних*** за їх типами і значеннями у внутрішній формі;
- ***можливість встановлення відношення канонічного лінійного порядку*** для комутативних операцій серед виразів та операторів;
- ***можливість побудови формул різних оцінок складності*** для визначення точності, надійності та ефективності варіантів РРЗ;
- ***можливість підрахунку значень різних оцінок складності*** для конкретизованих варіантів РРЗ;
- ***можливість порівняння аналітичних і числових*** оцінок складності для вибору кращого варіанта реалізації РРЗ.

Додатково розглянемо можливості створення РРЗ для конструктивних задач. Звичайно вони в технічних ПГД вони обмежуються розрахунковими задачами аналізу і синтезу з наступною перевіркою відповідності розрахунків гранично припустимим значенням. В задачах аналізу обробляються дані про навантаження конкретизованого технічного об'єкта на рівні його моделі. В задачах синтезу визначаються параметри елементів технічних об'єктів на основі граничних значень їх навантажень. При розв'язанні базових класів конструктивних задач використовують наступні процедури розв'язання часткових задач та задач проблемного рівня САП РРЗ:

- ***часткове підбиття підсумків в задачах аналізу поточного стану та рішень*** з використанням технологій накопичення та вибірки підсумків в гіперкубах [2];
- ***оцінка навантажень на елементи об'єктів***, що виконується шляхом прямих обчислень або розв'язання рівнянь операціями перетворень даних аналітичного типу;
- ***підготовка рішень та формування значень параметрів елементів об'єктів*** розрахунковими формулами, одержаними через зворотні перетворення аналітичних даних;
- ***оцінки обмежень навантажень і параметрів об'єктів*** для оцінки працездатності, надійності і ефективності проектних рішень.

Таким чином, показані 4 класи задач створюють основу реалізації комплексу РРЗ для будь-якої ПГД, в тому числі і для побудови САП. Для задач оцінок варіантів проектів визначають гіперкуби $\prod_k \boldsymbol{r}_{fk}$, таблиці $\boldsymbol{r}_{ck}$ та прості або підсумкові змінні оцінок ξ_j та $\sigma_{kj}(\xi_{kj\max})$, які входять до критеріїв рішень ПГД.

Умови коректного розв'язання задач САП зі створення надійних і ефективних РРЗ. Важливо, щоб БЗ дозволила знайти найкращі РРЗ, що розв'язують близькі задачі, і автоматизованим шляхом побудувати необхідний залишок РРЗ, що проектується. Контроль точності, надійності, ефективності та своєчасності розв'язання забезпечується специфікаціями РРЗ вторинного рівня.

Доказові перетворення виразів специфікацій зв'язків виконуються вбудованими операціями аналітичного типу даних з контролем вторинних цілей і критеріїв, заданих специфікаціями ПГД і РРЗ [4]. Для перетворень важливо визначити шлях формування прирощення тексту коду P_{pn} при побудові модифікованого РРЗ і оцінки прирощень потрібних елементів критеріїв. Спочатку треба визначити різницю між специфікацією S_{pn} і P_{pn}, а потім трансформувати її в код додатку до базового коду РРЗ для формування результату $\Delta P_{pn} = f(y, S_{pn} - P_{pn})$. При відсутності базового коду перетворення синтезу формуються як послідовність прирощень, що формуються з первинної специфікації з врахуванням заданих критеріїв.

Визначення загальних правил проектування РРЗ та настроювання роботи САП. З одного боку в шлях просування при синтезі кодів визначається або вручну, або автоматизованою підказкою прирощення. Якщо при автоматизованому синтезі необхідно зробити, що за попереднім доведенням відповідає специфікації, то при ручному синтезі необхідно формально довести коректність перетворення.

З іншого боку треба автоматизувати генерацію різних варіантів повний перебір яких є NP-складною задачею [6]. Однак перший ліпший варіант програми можна побудувати через перетворення, що послідовно наближують ство-

рення повного коду РРЗ, а потім удосконалювати його результатами адекватних перетворень, які покращують значення критеріїв.

Кількість можливих шляхів n_c перетворення визначається кількістю n_k відомих методів потенційно точного або наближеного розв'язання задач, занесених до БЗ САП РРЗ. Ці кількісні оцінки визначаються кількістю і розмірністю вхідних $n_i(N_{ik})$, вихідних $n_o(N_{ok})$ та робочих змінних $n_w(N_{wk})$, сформованих аналітичними методами підготовки розв'язання задач

$$n_k = n_k(n_i(N_{ik}), n_w(N_{wk}), n_o(N_{ok})). \quad (1)$$

Як правило, кількість ефективних варіантів розв'язання найбільш досліджених задач не перевищує десятків, а для одержання технічно ефективного результату можна обмежитися першою десяткою методів.

Вибір порядку створення задач та підзадач в САП РРЗ. Звичайно після первинного визначення типів даних (координат) будь-якого супероб'єкта ПГД доцільно створити набір типових комплексів задач для розв'язання в цій галузі. Розмірність комплексів задач визначається кількістю та розмірністю характеристик конкретизацій об'єктів, включених до моделі конкретної комплексної ситуації в ПГД. Кількість задач n_t обмежується кількістю об'єктів та агрегатів даних, які можуть бути вхідними та вихідними в підпрограмах. Кількість задач буде істотно менше кількості попарних комбінацій об'єктів, як вхідних та вихідних даних. Таким чином, n_t залежить від кількості та структурованості даних в об'єкти для їх використання в задачах як вхідних і вихідних даних.

Таким чином, для доказової розробки, верифікації і оперативного настроювання РРЗ вдосконалені САП повинні виконувати наступну послідовність дій, пов'язаних з вмістом вбудованої БЗ супроводження проектів та аналітичних перетворень.

1. Визначити базову структуру типів даних (координат) ПГД та комплексів об'єктів для розв'язання задач аналізу та синтезу.

2. Визначити цільові змінні та вторинні критерії в специфікаціях окремих задач ПГД.

3. Організувати адекватні перетворення за правилами перетворень аналітичних даних і методами розв'язання задач, занесеними до БЗ семантичних перетворень, показаної на рис.2.

4. Організувати пошук аналогів та супроводження реалізації проектів з використанням БЗ процесів проектування, показаної на рис. 3.

Процес доказової підготовки та настроювання РРЗ повинен супроводжуватись аналітичним контролем функціональності і продуктивності модулів, критичних за часом виконання [4].

Висновки

Таким чином, для підвищення рівня автоматизації САП РРЗ доцільно істотно розширити вбудовану БЗ до блоків управління обробкою, а також проміжні підсумки можуть використовуватися для ефективної комбінації фрагментів кодів. Ключовим елементом організації таких БЗ є використання абстрактного типу аналітичних даних. Кількість задач аналізу і синтезу в конкретному комплексі ПГД істотно обмежена кількістю істотних груп об'єктів та їх характеристик. Накопичення даних про ефективні реалізації РРЗ в БЗ є спеціальним варіантом задачі вибору ефективних рішень за допомогою БЗ, реалізованих в рамках реляційних СУБД [2].

Перелік посилань

1. Ахо А., Сети Р., Ульман Дж. Компиляторы: принципы, технологии, инструменты: Пер. с англ. – М.: Издательский дом Вильямс, 2001. – 768 с.
2. Бэлсон Д., Гокмен М. Ингрем Дж. Внутренний мир Oracle8. Проектирование и настройка: – К.: Изд-во "ДиаСофт". – 2000. – 800 с.
3. Лисков Б., Гатэг Дж. Использование абстракций и спецификаций при разработке программ. Пер. с англ. М.: Мир, 1989. 424 с.
4. Пустоваров В.І. Супроводження контролю відповідності програм і моделей формальним специфікаціям задач // Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка – К.: «Век+». 2007, 47, с. 269-279.
5. Hehner E.C.R. Practical theory of programming. Springer-Verlag, New York, 1993 – 243 p.
6. Metzger R.C., Zhaofang W. Automatic algorithm recognition and replacement: a new approach to program optimization / The MIT Press, Cambridge, 2000. 219 p.
7. Woodcock J., Davies J. Using Z. Specification, Refinement, and Proof. C.A.R. Hoare Series editor, 1995 – 390 p.

УДК 004.94

ТОМАШЕВСЬКИЙ В.М.,
ДІДЕНКО Д.Г.

ПОРІВНЯННЯ РЕЗУЛЬТАТІВ РОБОТИ СИСТЕМ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ OPENGPSS ТА GPSS/PC

В статье рассмотрены различные аспекты проведения экспериментов в дискретно-событийных системах OpenGPSS и GPSS/PC, сравниваются их качественные и количественные результаты работы. Предложен подход разделения эксперимента на независимые части для параллельного выполнения на узлах кластера с последующей сборкой результатов, который приводит к повышению производительности системы или уменьшению времени моделирования и при этом не влияет на правильность конечных результатов.

The report deals with questions of computing experiment in distributed discrete-event simulation systems OpenGPSS and GPSS/PC, their high-quality and quantitative job performances are compared. The problems of experiment distribution by independent part, deployment this part on cluster node, parallel execution and result assembles and here does not influence on the rightness of end-point also were laboured in the report.

Вступ

Імітаційне моделювання є одним з важливих методів аналізу різних складних систем. Разом зі збільшенням продуктивності та складності сучасних комп'ютерних систем постають нові вимоги до систем імітаційного моделювання – прискорення отримання результатів моделювання. Для цього можна використовувати розподілені системи імітаційного моделювання, які розглянуті в роботах [1, 2].

Існує багато реалізацій розподілених систем і систем тиражування експериментів, наприклад, SPEEDES [3], PARASOL [4] або Triad.Net [5]. Нажаль ці системи не охоплюють розробників, які працюють з поширеною мовою імітаційного моделювання GPSS [6]. Тому доцільно підвищення швидкості моделювання вже побудованих GPSS моделей. Останнім часом була розроблена нова розподілена дискретно-подійна система імітаційного моделювання OpenGPSS [7], яка доступна в режимі онлайн і використовує мову імітаційного моделювання GPSS. Але постає проблема довіри до результатів роботи нової системи моделювання.

Постановка задачі

Цілі дослідження: необхідно якісно та кількісно порівняти результати роботи систем імітаційного моделювання GPSS/PC та OpenGPSS. Еталоном при порівнянні може виступати широко відома прикладна програма GPSS/PC версії 2.0 компанії Minuteman Software (http://www.minitemansoftware.com/), яку багато років використовують розробники моделей і викладачі вузів.

Проведення обчислювального експерименту в послідовних системах моделювання

Розглянемо типовий обчислювальний експеримент мовою GPSS, який моделює роботу одноканальної системи масового обслуговування (СМО) з чергою (табл. 1).

Табл. 1. Текст тестової GPSS-програми

№	Команди GPSS
100	GENERATE 10,5
110	QUEUE QUE1
120	SEIZE PRIB1
130	DEPART QUE1
140	ADVANCE 15,5
150	RELEASE PRIB1
160	TERMINATE
170	GENERATE 5000
400	TERMINATE 1
500	START 1

Проведемо комп'ютерні прогони тестової GPSS-програми в двох системах моделювання. Для накопичення статистично-значимих результатів в системі GPSS/PC, кожний експеримент проводиться 3 рази, і в звіті відображаються лише середні арифметичні показники в таблицях результатів.

Система OpenGPSS може працювати як в звичайному послідовному режимі (використовується лише один вузол моделювання), так і в розподіленому режимі, як описано в роботі [8] (використовується два або більше вузла моделювання). Спочатку будемо використовувати систему OpenGPSS всього з одним вузлом моделювання. У зведеній табл. 2 наведена порівняльна інформація про роботу обох систем.

Табл. 2. Порівняння результатів послідовного моделювання

Показник	Значення	GPSS/PC 2.0	OpenGPSS (один вузол)
Модельний час		5000	5000
Інформація про пристрій PRIB1			
ENTRIES	Кількість входів в пристрій	338	334
UTIL (%)	Коефіцієнт використання пристрою	0,997	0,997
AVE.TIME	Середній час зайняття пристрою	14,75	15,0
Інформація про чергу QUE1			
MAX.	Максимальна довжина черги	169	161
CONTENT	Поточна довжина черги	168	161
ENTRIES	Кількість входів в чергу	506	495
ENTR (0)	Кількість входів з нульовим часом чекання	2	3
AVE.CON	Середня довжина черги	80,85	81,0
AVE.TIME	Середній час перебування всіх транзактів в черзі	798,89	816,0
AVE.(0)	Середній час знаходження транзактів в черзі крім нульових входів	802,06	821,0

Результати моделювання дещо відрізняються, тому що в кожній системі моделювання використовуються різні датчики псевдо-випадкових чисел (ДПЧ).

Розподілені обчислювальні експерименти

Система OpenGPSS використовує автоматичне розподілення GPSS-програм. Далі розглянемо чи впливає це на результати моделювання.

Для того, щоб система OpenGPSS почала працювати в розподіленому режимі, модифікуємо тестову GPSS-програму – додаємо команди керування імітаційним експериментом та отримаємо модифіковану GPSS-програму (табл. 3). Тобто замість того щоб запускати програму моделювання три рази як раніше, будемо виконувати комп'ютерні прогони за допомогою команд керування START, CLEAR та RMULT. Проміжні результати статистики для пристрою PRIB1 та черги QUE1 будемо зберігати у файлі RES.TXT командою RESULT.

Табл. 3. Текст модифікованої GPSS-програми

№	Команди GPSS
100	GENERATE 10,5
110	QUEUE QUE1
120	SEIZE PRIB1
130	DEPART QUE1
140	ADVANCE 15,5
150	RELEASE PRIB1
160	TERMINATE
170	GENERATE 5000
180	SAVEVALUE MSV01,FR$PRIB1
190	SAVEVALUE MSV02,FC$PRIB1
200	SAVEVALUE MSV03,FT$PRIB1
210	SAVEVALUE MSV04,Q$QUE1
220	SAVEVALUE MSV05,QA$QUE1
230	SAVEVALUE MSV06,QM$QUE1
240	SAVEVALUE MSV07,QC$QUE1
250	SAVEVALUE MSV08,QZ$QUE1
260	SAVEVALUE MSV09,QT$QUE1
270	SAVEVALUE MSV10,QX$QUE1
400	TERMINATE 1
410	CLEAR
420	RMULT 1,2,8,5,8,2,8
500	START 1
510	RESULT RES.TXT,MSV01,01;FR$PRIB1
520	RESULT RES.TXT,MSV02,02;FC$PRIB1
530	RESULT RES.TXT,MSV03,03;FT$PRIB1
540	RESULT RES.TXT,MSV04,04;Q$QUE1
550	RESULT RES.TXT,MSV05,05;QA$QUE1
560	RESULT RES.TXT,MSV06,06;QM$QUE1
570	RESULT RES.TXT,MSV07,07;QC$QUE1
580	RESULT RES.TXT,MSV08,08;QZ$QUE1
590	RESULT RES.TXT,MSV09,09;QT$QUE1
600	RESULT RES.TXT,MSV10,10;QX$QUE1
610	CLEAR
620	RMULT 5,1,6,6,7,1,1
630	START 1
640	RESULT RES.TXT,MSV01,01;FR$PRIB1
650	RESULT RES.TXT,MSV02,02;FC$PRIB1
660	RESULT RES.TXT,MSV03,03;FT$PRIB1
670	RESULT RES.TXT,MSV04,04;Q$QUE1
680	RESULT RES.TXT,MSV05,05;QA$QUE1
690	RESULT RES.TXT,MSV06,06;QM$QUE1
700	RESULT RES.TXT,MSV07,07;QC$QUE1
710	RESULT RES.TXT,MSV08,08;QZ$QUE1
720	RESULT RES.TXT,MSV09,09;QT$QUE1
730	RESULT RES.TXT,MSV10,10;QX$QUE1
740	CLEAR
750	RMULT 2,2,5,2,8,9,3
760	START 1
770	RESULT RES.TXT,MSV01,01;FR$PRIB1
780	RESULT RES.TXT,MSV02,02;FC$PRIB1
790	RESULT RES.TXT,MSV03,03;FT$PRIB1
800	RESULT RES.TXT,MSV04,04;Q$QUE1

810	RESULT RES.TXT,MSV05,05;QA$QUE1
820	RESULT RES.TXT,MSV06,06;QM$QUE1
830	RESULT RES.TXT,MSV07,07;QC$QUE1
840	RESULT RES.TXT,MSV08,08;QZ$QUE1
850	RESULT RES.TXT,MSV09,09;QT$QUE1
860	RESULT RES.TXT,MSV10,10;QX$QUE1

За допомогою команди RMULT встановлюються початкові значення для всіх датчиків псевдо випадкових чисел. Команда START запускає обчислювальний експеримент. У GPSS-програмі можливо декілька разів використовувати команду START, для отримання проміжних результатів моделювання. Команда CLEAR очищає всю статистику системи і всі видаляє транзакти з моделі. Після виконання команди CLEAR уся система, окрім ДПЧ, знаходиться у початковому стані. Команда RESULT заносить результати моделювання у тимчасовий буфер результатів, для збереження значень і наступного аналізу.

Введемо визначення, які необхідні для подальшого розгляду матеріалу статті.

Нехай *сегмент обчислювального експерименту* (СОЕ) – це послідовність команд керування, яка починається командою CLEAR або RMULT, містить не більше однієї команди CLEAR, RMULT, START і закінчується командою START або RESULT. Сегменти обчислювального експерименту розділяються на два типи: залежні і незалежні.

Будемо називати *залежним сегментом обчислювального експерименту* (ЗСОЕ) такі СОЕ, які не містять команду RMULT або не містять команду CLEAR. Дані в таких сегментах залежать від передісторії станів модельованої системи.

Незалежні сегменти обчислювального експерименту (НСОЕ) – це такі СОЕ, які обов'язково складаються з послідовності команд CLEAR, RMULT, START. Саме послідовність команд CLEAR, RMULT заставляє систему "забути" свої попередні стани, тобто сегмент не залежить від передісторії станів системи.

Будемо називати *кадром сегмента обчислювального експерименту* (КСОЕ) – послідовність СОЕ, яка починається з одного НСОЕ, та включає в себе всі ЗСОЕ, які залежать від цього початкового сегмента.

Припустимо, що S_{ij} – j-а імітаційна модель, яка виконується на i-му вузлі, $i=\overline{1,n}$, $j=\overline{1,n_i^j}$, де n_i^j – кількість моделей на i-му вузлі. У кожної моделі i-го вузла є свій модельний час, який позначимо $T_i=\{t_{i1},t_{i2},..,t_{in_i^t}\}$.

Наявність послідовності команд CLEAR, RMULT, START, дозволяє виділити незалежні СОЕ, множину яких позначимо $U_{ij}=\{u_{ij1},u_{ij2},..,u_{ijn_i^u}\}$. Усі інші СОЕ відносимо до залежних, множина – $C_{ij}=\{c_{ij1},c_{ij2},..,c_{ijn_i^c}\}$. Зауважимо, що перший СОЕ в моделі завжди незалежний, тому він належить множині $U_{ij}=\{u_{ij1},u_{ij2},..,u_{ijn_i^u}\}$.

Побудуємо кадр сегмента обчислювального експерименту як послідовність СОЕ, що починається з одного НСОЕ, і включає в себе всі ЗСОЕ, які залежать від цього початкового сегмента:

$$K_{ijm}=\{u_{ijm}\mid\forall m: u_{ijm}\in U_{ij}\}\cup\{c_{if}\mid c_{if}\in C_{ij},$$

$$c_{if}\ залежить_від_u_{im}\},i=\overline{1,n},\ j=\overline{1,n_i^j},\ m=\overline{1,n_i^u}.$$

З побудови КСОЕ випливає незалежність кадрів один від одного.

Нехай K_{ij} – множина КСОЕ на i-му вузлі для j-ої моделі:

$$K_i=\bigcup_{n_i^u}^{j=1}K_{ij},$$

а результати проведення всіх СОЕ з КСОЕ позначимо R_{ij}:

$$R_{ij}=\bigcup_{n_i^k}^{m=1}R_{ijm}.$$

Об'єднання результатів по всім КСОЕ j-ої моделі дає загальний результат:

$$R_i=\bigcup_{n_i^u}^{j=1}R_{ij}.$$

Тобто проведення одного КСОЕ дає один і тільки один результат з множини R_i.

Нехай A_i = { A_i^{AgSim}, $A_i^{Ag\,\mathrm{Re}\,p}$, A_i^{AgSpl}, A_i^{AgSnc}, A_i^{AgTrf}, A_i^{AgPwr}, A_i^{AgUsr}, A_i^{AgGbr} } – множина активних агентів i-го сервера, $i=\overline{1,n}$, де n – кількість серверів імітаційного моделювання в кластері [9].

Введемо матрицю $\mathbf{B}_{iz}$ розмірністю $n\times n$, яка вказує чи є зв'язок між i-м та z-м сервером:

$$\mathbf{B}_{iz}=\begin{cases}1, & якщо\ існує\ зв'язок\\ & між\ i-м\ та\ z-м\ сервером;\\ 0, & в\ іншому\ випадку;\end{cases},$$

$$i=\overline{1,n},\ z=\overline{1,n}.$$

Стан *i*-го вузла опишемо як $P_i(A_i, S_i, T_i, K_i, R_i)$, а стан кластера моделювання – $P = \bigcup_{n}^{i=1} P_i(A_i, S_i, T_i, K_i, R_i)$.

Незалежність КСОЕ дозволяє проводити тиражування – одночасне виконання КСОЕ на різних вузлах кластера імітаційного моделювання, з подальшим об'єднанням результатів.

Система OpenGPSS автоматично виділяє множину КСОЕ, які можуть одночасно виконуватись на різних вузлах кластера. Після запуску *j*-ої GPSS-моделі агент користувача A_i^{AgUsr}, який виконує функцію компілятора [10] і переводить GPSS-текст у внутрішній формат та виділяє кадри СОЕ:

$$A_i^{AgUsr} : P_i(S_i, K_i) \to P_i(S_i \cup \{s_{ij}\}, K_i \cup \{k_{ij}\}).$$

Агент реплікації AgRep [11] копіює GPSS-модель та всі КСОЕ на інші вузли:

$$A_i^{Ag\,Re\,p} : P_i(S_i, K_i) \to$$
$$\to P_z(S_z \cup S_i, K_z \cup K_i), \forall i, z : \mathbf{B}_{iz} = 1,$$

тому кожний вузол повністю зберігає інформацію про модель та експеримент для запобігання втрат даних у разі вимкнення одного з вузлів.

У кластері відсутнє централізоване керування запуском КСОЕ – кожний вузол випадковим чином визначає КСОЕ із множини ще необроблених сегментів. Це дозволяє уникати конфліктів блокування: жоден вузол не чекає на результати обробки ОЕ іншими вузлами – вони діють за оптимістичним планом – продовжують виконувати моделювання над сегментами. Цей підхід відрізняється від підходу з динамічним розподілом на підмножини вузлів кластера і визначення локального вузла-координатора за допомогою одного з алгоритмів голосування, які наведені в [12] і мають меншу масштабованість (канал зв'язку координатора може бути перевантаженим) та надійність (при відмові координатора необхідно чекати переголосування).

Агент імітаційного моделювання AgSim виконує КСОЕ на *i*-му вузлі:

$$A_i^{AgSim} : P_i(\{t_{i1}, t_{i2}, .., t_{ij}, .., t_{in_j^t}\}) \to$$
$$\to P_i(\{t_{i1}, t_{i2}, .., \grave{t}_{ij}, .., t_{in_j^t}\}),$$
$$\text{де } t_{ij} <= \grave{t}_{ij}.$$

Агент реплікації AgRep, реалізований мовою PL/SQL [13, 14], періодично копіює таблиці результатів REPORT и REPORT_DETAIL на інші вузли:

$$A_i^{Ag\,Re\,p} : P_z(R_z) \to P_i(R_i \cup R_z), \forall i, z : \mathbf{B}_{iz} = 1.$$

Потім AgRep виконує *збирання результатів* – об'єднання результатів всіх СОЕ для подальшого виконання та аналізу загальних результатів і завершує ОЕ.

Розглянемо приклад. Розділимо модифіковану GPSS-програму з табл. 3 на сегменти, отримаємо три СОЕ, які наведені в таблиці 4. СОЕ1 – незалежний, тому що це перший СОЕ та його виконання ні від чого не залежить; СОЕ2 та СОЕ3 – також незалежні, тому що виконуються команди CLEAR, RMULT, які встановлюють "початковий" стан системи.

Табл. 4. Уривок тексту GPSS-програми з сегментами

№	Команди GPSS	Сегменти	Кадри сегментів
	…	COE1 (HCOE)	KCOE1
410	CLEAR		
420	RMULT 1,2,8,5,8,2,8		
500	START 1		
510	RESULT RES.TXT,MSV01,01;FR$PRIB1		
520	RESULT RES.TXT,MSV02,02;FC$PRIB1		
530	RESULT RES.TXT,MSV03,03;FT$PRIB1		
540	RESULT RES.TXT,MSV04,04;Q$QUE1		
550	RESULT RES.TXT,MSV05,05;QA$QUE1		
560	RESULT RES.TXT,MSV06,06;QM$QUE1		
570	RESULT RES.TXT,MSV07,07;QC$QUE1		
580	RESULT RES.TXT,MSV08,08;QZ$QUE1		
590	RESULT RES.TXT,MSV09,09;QT$QUE1		
600	RESULT RES.TXT,MSV10,10;QX$QUE1		
610	CLEAR	COE2 (HCOE)	KCOE2

620	RMULT 5,1,6,6,7,1,1		
630	START 1		
640	RESULT RES.TXT,MSV01,01;FR$PRIB1		
650	RESULT RES.TXT,MSV02,02;FC$PRIB1		
660	RESULT RES.TXT,MSV03,03;FT$PRIB1		
670	RESULT RES.TXT,MSV04,04;Q$QUE1		
680	RESULT RES.TXT,MSV05,05;QA$QUE1		
690	RESULT RES.TXT,MSV06,06;QM$QUE1		
700	RESULT RES.TXT,MSV07,07;QC$QUE1		
710	RESULT RES.TXT,MSV08,08;QZ$QUE1		
720	RESULT RES.TXT,MSV09,09;QT$QUE1		
730	RESULT RES.TXT,MSV10,10;QX$QUE1		
740	CLEAR	СОЕ3 (НСОЕ)	
750	RMULT 2,2,5,2,8,9,3		
760	START 1		
770	RESULT RES.TXT,MSV01,01;FR$PRIB1		
780	RESULT RES.TXT,MSV02,02;FC$PRIB1		
790	RESULT RES.TXT,MSV03,03;FT$PRIB1		
800	RESULT RES.TXT,MSV04,04;Q$QUE1		
810	RESULT RES.TXT,MSV05,05;QA$QUE1		
820	RESULT RES.TXT,MSV06,06;QM$QUE1		
830	RESULT RES.TXT,MSV07,07;QC$QUE1		
840	RESULT RES.TXT,MSV08,08;QZ$QUE1		
850	RESULT RES.TXT,MSV09,09;QT$QUE1		
860	RESULT RES.TXT,MSV10,10;QX$QUE1		
		Неявне автоматичне збирання результатів	

З прикладу видно, що початкові значення всіх ДПЧ, які використовувались раніше обов'язково повинні бути перевизначені командою RMULT.

Проведемо з GPSS-програмою, яка наведена в табл. 4, обчислювальний експеримент для кількості вузлів кластера один, два та три вузла. Незалежні СОЕ можуть безпечно виконуватись на різних вузлах кластера імітаційного моделювання з подальшим збиранням результатів, що приводить до зменшення часу на проведення імітаційного експерименту. Якщо кластер складається з одного вузла, то всі СОЕ виконуються на ньому. При наявності двох вузлів: на одному вузлі виконуються два сегмента, а третій НСОЕ виконується на другому вузлі. Для трьох вузлів – кожен сегмент виконується на окремому вузлі. Середнє арифметичне результатів наведене в табл. 5. Система GPSS/PC не може працювати в розподіленому режимі, тому для неї наведена інформація при послідовному проведенні експерименту. Як видно із табл. 5 результати моделювання знову приблизно однакові.

Табл. 5. Порівняння результатів розподіленого моделювання

Показник	**Розшифровка**	**GPSS/PC 2.0**	**OpenGPSS (один вузол)**	**OpenGPSS (два вузла[1])**	**OpenGPSS (три вузла[2])**
Астрономічний час, с	Астрономічний час моделювання, секунди	4	8	9	6
Модельний час	Модельний час	5000	5000	5000	5000
FR$PRIB1	Коефіцієнт використання пристрою (в тисячних долях)	997	997	998	998
FC$PRIB1	Число входів в пристрій	338	331	341	338

[1] Використовується середнє арифметичне по двом вузлам.
[2] Використовується середнє арифметичне по трьом вузлам.

FT$PRIB1	Середній час використання пристрою	14	15	14	14
Q$QUE1	Довжина черги	168	161	162	158
QA$QUE1	Середня довжина черги	80	79	82	77
QM$QUE1	Максимальна довжина черги	169	162	162	159
QC$QUE1	Загальне число входів в чергу	506	492	503	596
QZ$QUE1	Число нульових входів в чергу	2	1	1	1
QT$QUE1	Середній час знаходження транзакта в черзі (нульові входи включно)	798	811	819	776
QX$QUE1	Середній час знаходження транзакта в черзі (без нульових входів)	802	813	820	777

Порівняння швидкості моделювання

Розглянемо, як кількість транзактів в моделі впливає на швидкість моделювання, для чого в нашій останній GPSS-програмі у рядку 170 замість блоку «GENERATE 5000» будемо використовувати «GENERATE 5000», «GENERATE 10000», «GENERATE 20000»... «START 90000». Усі експерименти будемо проводити для різної кількості вузлів кластера. Результати проведення експериментів відображені на рис. 1. По осі абсцис відкладені значення модельного часу, а по осі ординат астрономічний час в секундах.

На рис. 1 астрономічний час моделювання для системи з двома вузлами «приблизно» такий самий як і для одного вузла, тому що GPSS-програма містить три (непарне число) сегмента, із-за чого один з вузлів «чекає» другий вузол поки той послідовно моделює два сегменти. Швидкість збільшується але не в два рази. На тренд, що відображає час для трьох вузлів, на початку моделювання впливають накладні витрати, але для модельного часу 90 тис. од. система моделювання OpenGPSS випереджає GPSS/PC.

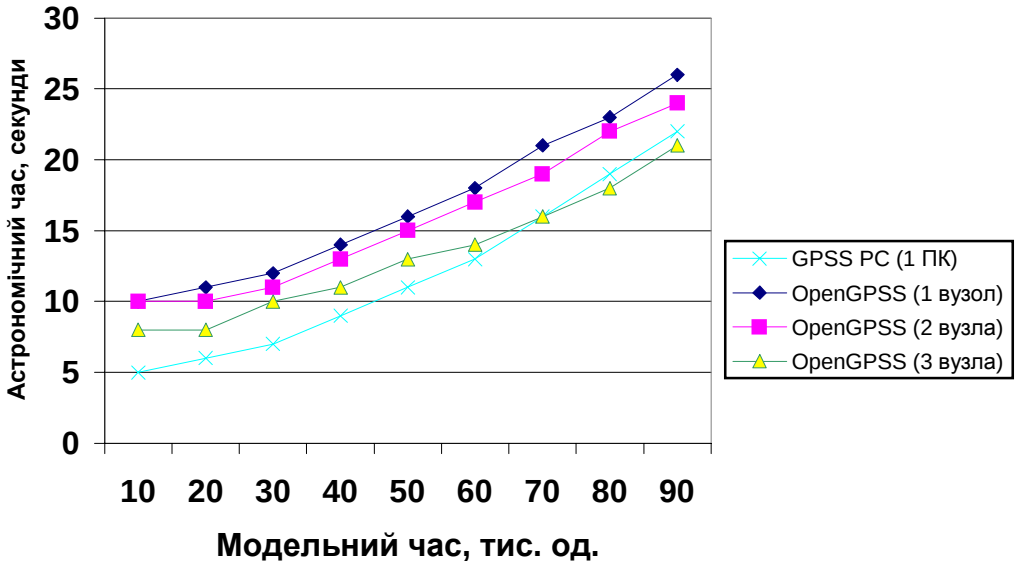


Рис.1. Залежність астрономічного часу моделювання від кількості вузлів кластера та модельного часу

Висновки

1. Системи GPSS та OpenGPSS на однакових GPSS-програмах видають приблизно однакові результати при послідовному і розподіленому режимі проведення експерименту. Для отримання більш точних результатів необхідно збільшити кількість прогонів моделі.

2. Запропонований підхід розділення експерименту на незалежні частини для подальшого паралельного виконання на вузлах кластера і збирання результатів призводить до підвищен-

ня продуктивності системи (зменшення часу моделювання), що помітно на кластері з трьома або більше вузлами.

3. Отримані результати підтверджують можливість прозорого автоматичного розділення експерименту на незалежні частини для прискорення швидкості моделювання.

Перспективи досліджень

Оптимізація режимів роботи кластера для підвищення швидкості моделювання.

Побудова та дослідження кластера з великою кількістю вузлів (16-32).

Побудова розподіленої дискретно-подійної системи імітаційної моделювання з оптимістичною синхронізацією модельного часу у вузлах кластера.

Перелік посилань

1. Richard M. Fujimoto. Parallel and Distributed Simulation Systems. Wiley, 2000.
2. Замятина Е.Б. Современные теории имитационного моделирования: Специальный курс. – Пермь: ПГУ, 2007. – 119 с.
3. SPEEDES. http://www.speedes.com.
4. Mascarenhas E., Knop F., Vernon R. ParaSol: A multithreaded system for parallel simulation based on mobile threads. Winter Simulation Conference, 1995.
5. А.И. Миков, Е.Б. Замятина, А.Н. Фирсов. Инструментальные средства удалённого параллельного моделирования. В книге Proceedings of XXII International Conference “Knowledge-Dialogue-Solution”.– FOI-COMMERCE, Sofia, 2006, pp. 280-287.
6. Шрайбер Томас Дж. Моделирование с использованием GPSS. – М.: Машиностроение, 1980. – 593 с.
7. Киевский центр имитационного моделирования. http://www.simulation.kiev.ua.
8. Діденко Д.Г. Реалізація тиражування обчислювального експерименту в розподіленій системі моделювання OpenGPSS. – Наукові вісті – К. 2007. – № 5. С. 49-53.
9. Томашевский В.Н., Диденко Д.Г. Агентная архитектура распределенной дискретно-событийной системы имитационного моделирования OpenGPSS. Системні дослідження та інформаційні технології. № 4, 2006. – К.: ВПК “Політехніка”, 2006. С.123–133.
10. Ахо Альфред, Сети Рави, Ульман Джеффри. Компиляторы: принципы, технологии, инструменты. Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – 768 с.
11. Діденко Д.Г. Агент реплікації в розподіленій дискретно-подійній системі імітаційного моделювання OPENGPSS. Матеріали міжнародної наукової конференції “Інтелектуальні системи прийняття рішень та прикладні аспекти інформаційних технологій”, 2006. С. 264–266.
12. Таненбаум Э., Стеен ван М. Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877 с.
13. Кайт Том. Oracle для профессионалов. Книга 2. Расширение возможностей и защита: Пер. с англ. – М.: ДиасофтЮП, 2003. – 848 с.
14. Кайт Том. Oracle для профессионалов. Книга 1. Архитектура и основные особенности: Пер. с англ. – М.: ДиасофтЮП, 2003. – 672 с.

УДК 004.93(015.7)

АМОНС О.А.,
ЯНОВ Ю.О.,
БЕЗПАЛИЙ І.О.

КЛАСТЕРИЗАЦІЯ ДОКУМЕНТІВ НА ОСНОВІ СТАТИСТИЧНОЇ БЛИЗЬКОСТІ ТЕРМІВ

У статті описано підхід до кластеризації колекції документів з невідомою наперед кількістю кластерів. В основу підходу покладено метод, оснований на статистиці появи ключових термів. Запропоновано модифікацію методу знаходження матриці подібності на основі схожості косинуса. Для аналізу якості й знаходження граничних значень алгоритму використана модифікація функції конкурентної подібності. Підхід реалізований у вигляді прикладного застосування сервера SmartBase. Наведені результати експериментальних досліджень запропонованого підходу до кластеризації інформації з використанням часто вживаного текстового корпусу підтверджують працездатність запропонованих рішень.

In the given work the approach to clustering of documents collections with unknown quantity of clusters is described. A method of finding matrix of similarity is improved. The method is based on the statistics of key terms occurrence in documents. For quality analysis and finding of limiting values of algorithm, there was used a function of competitive similarity improving. The approach is realized as the application server SmartBase's application. Implementation details and results of the process are shown. Russian text set is used.

Вступ

Зі швидким ростом всесвітньої павутини, впровадженням у міністерствах, відомствах і великих організаціях різноманітних засобів підтримки інформаційно-аналітичної діяльності, насамперед інформаційних систем, систем електронного документообігу, інформаційні бази інформаційно-телекомунікаційних систем спеціального призначення (ІТС СП) набувають суттєвих розмірів. Вони стрімко розширюються переважно за рахунок неструктурованих даних, що породжує проблему швидкого орієнтування в них. Дійсно, відсутність можливості отримувати актуальну і повну інформацію з конкретної теми, яка цікавить користувача, перетворює в непотріб велику частину накопичених ресурсів і робить марними зусилля фахівців. Оскільки безпосередній пошук і анналіз інформації за заданою темою досліджень вимагає все більших трудовитрат, багато рішень приймаються на основі неповного бачення проблеми. Використання засобів для автоматичної систематизації і класифікації текстової інформації дозволяє скоротити час на пошук потрібної інформації, забезпечити її повноту і тим самим підвищити якість дослідження та швидкість реагування на зовнішні зміни. Одним із засобів для підвищення ефективності роботи фахівців в умовах функціонування ІТС СП є інструменти кластеризації текстів, які підлягають аналізу. Стаття присвячена аналізу існуючих методів і засобів пошуку текстової інформації, розробленню дієвого алгоритму кластеризації текстової інформації і його дослідженню на наявному текстовому корпусі.

1. Пошук документів і кластеризація

Пошук документів є однією з найуживаніших операцій оброблення інформації. Дійсно, перш ніж документи можуть бути опрацьованими різними процедурами, їх необхідно знайти серед величезного числа подібних. Якщо пошук документів буде не досить ефективним, то й загальні процедури оброблення інформації, представленої у вигляді документів, не будуть ефективними. Якщо історію пошуку документів можна розпочинати від зародження бібліотечних інформаційних систем [1], то період його інтенсивних досліджень започаткований формуванням Глобальної мережі і її перетворенням у практичний інструмент підтримки різних видів діяльності людини. Саме тоді з'явилися відомі пошуковики, ефективність яких швидко зростала під впливом нагальних потреб [2]. Останні події зі світу пошуку документів свідчать, що для збільшення його ефективності застосовуються не лише моделі, традиційні для морфологічного і синтаксичного рівнів мовної системи, але й семантичні моделі [3].

Хоч сутність проблеми пошуку документів розуміють як користувачі, так і фахівці в галузі

інформаційних технологій (ІТ), лише останні розуміють її складність як науково-практичної проблеми. І джерело цієї складності не стільки у необхідності застосування серйозних моделей і методів лінгвістики, скільки у необхідності такої їх реалізації засобами ІТ, яка задовольняє високий рівень вимог до пошуку, насамперед його адекватності і швидкості. Для кращого розуміння проблем пошуку наведемо необхідні визначення. Під адекватністю розуміють відповідність результатів пошуку інформаційним потребам користувача з погляду визначених цілей і створених на їх основі запитів. Швидкість розглядається у контексті ефективності алгоритму і можливого часу його виконання. Сам пошук інформації становить собою процес виявлення в деякій множині (колекції) документів (текстів) усіх таких з них, які присвячені певній темі, задовольняють заздалегідь визначеній умові пошуку (запиту) чи містять необхідні (що відповідають інформаційній потребі) факти, відомості, дані. Кластеризація – це процес створення кластерів. Згідно з розповсюдженним визначенням, "кластери – це неперервні області (якогось) простору з більш високою щільністю елементів, відділені від інших таких ж областей з відносно низькою щільністю елементів". Під кластеризацією документів розуміють процес поділу всієї колекції на групи, у середині яких знаходяться близькі за тематикою документи, а в різних групах, навпаки, далекі.

Хоч існують різноманітні підходи до побудови пошуковиків, для найефективніших з них загалом властиве застосування, крім зазначених моделей і методів лінгвістики, технологій кластеризації та індексування. І якщо індексування швидше пов'язане з технологічними аспектами, то кластеризація – це традиційна для ІТ наукова проблема. На сьогодні, як засіб для впорядкування текстової інформації, у пошукових системах застосовують різні методи кластерного аналізу. Це, насамперед, статистичні класифікатори на основі ймовірносних методів. Найбільш відомими з них є сім'я Байесових алгоритмів. Наступною групою є класифікатори, що використовують методи на основі штучних нейронних мереж, наприклад алгоритми ART, SOM [4]. І останню групу складають класифікатори, що базуються на функціях схожості, насамперед k-means, Forel, FRiS Cluster [5,6,11].

Одними із найуживаніших є методи, засновані на метриці близькості. У цьому випадку документи представляються у вигляді вектору ознак у просторі ознак, тобто набором ключових слів. Є декілька підходів до його формування. В найпростішому випадку кожна ознака відповідає присутності в тексті одної із словоформ, що зустрічається у текстовій колекції. Величину кожного елемента вектора можуть підраховувати по різному: наприклад, прирівнювати одиниці, якщо ознака зустрічається у даному тексті, чи нулю у іншому разі; вона може бути рівною кількості входжень його у документ, нормованою до кількості ознак; чи також враховувати частоту появи ознаки у всьому текстовій колекції. Таке представлення має суттєвий недолік – простір ознак має велику розмірність, більша частина його елементів є надмірними, навіть шкідливими. Для подолання цієї проблеми використовуються методи зменшення розмірності простору ознак. Це, насамперед, виділення лем слова, нормальних форм, видалення стоп-слів, використання синонімічних груп тощо. Але в цьому випадку є ймовірність не використати значущу інформацію.

На наступному кроці підраховується матриця близькості між векторами документів. І виконується власне кластеризація. Більш детальну інформацію з описом особливостей існуючих алгоритмів можна знайти в працях [7,8].

Крім загальних проблем, для всіх методів кластеризації текстів постає проблема відображення змісту кластеру, на основі якого до нього приєднується той чи інший текстовий документ. Це необхідно для зручного використання результатів людиною. Найбільш розповсюджений підхід до рішення цієї проблеми складається у використанні представлення кластеру у вигляді набору найбільш важливих слів.

2. Постановка проблеми

Проблема полягає у розробленні текстового кластеризатора і дослідженні його властивостей. Загальну модель текстового кластеризатора можна представити багатоосновною алгебраїчною системою такого вигляду:

$$R = < T, C, D, B, R, S > \qquad (1)$$

де:

$T = \{T_1, T_2, ..., T_1, ..., T_n\}$ – множина текстів, що підлягають класифікації (колекція);

$T_l = \{t_1, t_2, ..., t_j, ...\}$ – множина термів, з яких складається *l*-й документ;

$C = \{C_1, C_2, ..., C_k\}$ – множина класів-рубрик (кластерів), де *k* – кількість кластерів;

$D = \{D_1, D_2, ..., D_m\}$ – множина описів, кожний з яких має певну внутрішню структуру, де *m* – кількість описів;

$B = \{b_1, b_2, ..., b_k\}$ – множина еталонних зразків (стовпів), $i = 1,2,\ldots, k$;

$R \subset C \times D$ – відношення між кластерами і описами, яке має таку властивість: $\forall C_i \in C \; \exists \; D_j \in D : (C_i, D_j) \in R$, причому кожному кластеру відповідає єдиний опис;

S – сигнатура, яка включає такі операції:

S_1: $T \rightarrow C$ – операція кластеризації, яка полягає у виконанні перетворень над текстами, після яких, або робиться висновок про належність документа T_l зі структурою D_l до класу C_i, або висновок про створення нового кластеру C_j, до якого можна буде віднести даний текстовий документ. Будемо вимагати, щоб жодний текст не міг відноситись до декількох кластерів одночасно;

S_2: $C \times C \rightarrow C$ – теоретико-множинна операція перетину кластерів;

S_3: $C \times C \rightarrow C$ – теоретико-множинна операція об'єднання кластерів.

3. Загальний опис підходу до розв'язання проблеми

Підхід, реалізований авторами, полягає у використанні векторної моделі текстових документів, згідно з якою кожний текстовий документ представляється у вигляді вектору зважених ознак і належність документів до кластерів визначається на основі міри близькості відповідних векторів.

Досить логічно кластеризація виконується у таких чотири основних етапи:

Етап 1. Це підготовчий етап, який полягає у переході від множини текстів *T* до множини T^* їх векторів.

Етап 2. На цьому етапі відбувається перехід до множини описів документів з урахуванням ваги ознак. Будемо використовувати статистичні міри ваги, які добре зарекомендували себе у пошуку документів, насамперед міри, що характеризують частку деякого терму документа у загальній кількості термів та їх появи в документах всього набору [8].

Етап 3. Будуємо матрицю близькості між документами на основі популярної функції схожості косинуса, фізичним змістом якої є косинус кута між векторами.

Етап 4. На цьому етапі виконується власне кластеризація.

Розглянемо наведені етапи більш детально.

1) Підготовчий етап. Для кожного документу T_l множини текстів *T* виділяємо множину значимих слів, які приводяться до нормальних форм. Будемо традиційно називати ці слова ключовими термами і позначати t_q. Як і в багатьох інших підходах, ми розглядаємо в якості ключових ті терми, частота яких у даному тексті істотно перевищує деяку середню частоту. Крім того ми будемо виключати стоп-слова, такі як прийменники, сполучники тощо.

2) Від колекції текстів переходимо до множини описів текстів. Знаходимо вагу кожного терму. Для цього кожному терму t_i документу T_l ставимо у відповідність статистичну міру w_{li}, що характеризує відношення кількості входжень цього терму у документ до загальної кількості термів та враховує частоту появи терму в документах всієї колекції:

$$w_{li} = -\log(p(t_i) \cdot f_t(T_l, t_i)) \qquad (2)$$

$$f_t(T_l, t_i) = \frac{f_r(T_l, t_i)}{f_r(T_l, t_i) + 1 + \frac{d(T_l)}{350}},$$

$$p(t_i) = 1 - e^{-1.5 \cdot \frac{f_c(t_i)}{n}}$$

де

$f_r(T_l, t_i)$ – число входжень терма t_i у документ T_l;

$d(T_l)$ – кількість термів у документі T_l;

$f_c(t_i)$ – число входжень терма t_i в колекцію.

3) Будуємо матрицю близькості між документами на основі популярної функції схожості косинуса, фізичним змістом якої є косинус кута між векторами:

$$sim(T_1, T_2) = \frac{\sum_i w_{1i} w_{2i}}{\sqrt{\sum_i w_{1i}^2 \sum_i w_{2i}^2}}, \qquad (3)$$

де w_{1i}, w_{2i} - вага терма ti в документах T1,T2 відповідно.

У виразі (3) враховуються лише терми, що входять одночасно як в документ T_1, так і в до-

кумент T_2. Але цій оцінці притаманні певні недоліки. Продемонструємо їх на прикладі. Розглянемо два тексти:

a) "Президент обратился с представлением в Конституционный суд с просьбой растолковать конституционный механизм замены отдельных членов правительства."
b) "Как Президент выражаю почет и благодарность всем ветеранам трагической афганской войны и украинским воинам-миротворцам за отвагу, доблесть, верность присяге и чести"

У цих текстах є лише одне спільне ключове слово "Президент" і воно вживається лише одного разу. І оскільки кількість ключових термів у кожному з документів приблизно рівна, то використання виразів (2) і (3) дає оцінку близькості $\mathrm{sim}(T_1,T_2) \sim 1$, що не повною мірою характеризує дійсний зв'язок цих текстів.

Цього недоліку можна позбутися використовуючи для оцінки близькості такий вираз:

$$sim(T_1,T_2) = \frac{\sum_i w_{1i} w_{2i}}{\sqrt{\sum_i w_{1i}^2 \sum_i w_{2i}^2}} \cdot \sum_i \frac{w1_{1i} + w2_{2i}}{w1_{1i} + w2_{2i}}, \qquad (4)$$

$w1_{1i}, w2_{1i}$ - вага термів, що одночасно присутні в обох документах.

4) Власне кластеризація. Будемо виконувати кластеризацію на основі FRiS-функцій [10]. Розглядаємо колекцію T із n документів, що розбита на k угрупувань. Кожне угрупування i описане одним еталонним об'єктом (стовпом, центроїдом) b_i. Для будь-якого документа $T_1 \in T$ можна знайти відстань $r(T_1,b_i)$ до найближчого стовпа угрупування i. Тоді $r1(T_1,b_i) = \min_i r(T_1,b_i)$ - близькість до найближчого стовпа i, а $r2(T_1,b_i) = \min_{i \neq i^*} r(T,b_i)$ - близькість до найближчого конкурента.

FRiS-функція [10], модифікована для використання матриці близькості між документами, має такий вигляд:

$$F(T,B) = \left(\frac{r1(T,b) - r2(T,b)}{r1(T,b) + r2(T,b)} \right) \qquad (5)$$

Вона є мірою подібності об'єкту T зі стовпом b_i у конкуренції з іншими стовпами.

Знайшовши середнє значення FRiS-функції по усій вибірці, за допомогою виразу (6) отримаємо величину $F(B)$, що характеризує наскільки повно набір стовпів характеризує колекцію:

$$F(B) = (1/m) \sum_{Ti \in T} F(T_i,b) \qquad (6)$$

У роботі також будемо використовувати редуковану FRiS-функцію, у якій стовп конкурента рівновіддалений від кожного об'єкта на відстань $r2^*$. Тоді вирази (5), (6) набудуть такого вигляду:

$$F^*(T,B) = \left(\frac{r1(T,b) - r2^*(T,b)}{r1(T,b) + r2^*(T,b)} \right) \qquad (7)$$

$$F(B) = (1/m) \sum_{Ti \in T} F^*(T_i,b) \qquad (8)$$

4. Опис алгоритму кластеризації

Деталізуючи наведений узагальнений опис, кластеризацію будемо виконувати за допомогою наступного алгоритму:

Крок 1. При k=1 створюємо новий кластер C_{first} і всі документи включаємо до нього.

Крок 2. У кластері, отриманому на попередньому етапі, призначаємо стовпом довільно вибраний документ t і для нього по формулі (8) обчислюємо середню редуковану FRiS-функцію $F^*(B)$.

Крок 3. Крок 1 повторюється при призначенні стовпами всіх m об'єктів кластеру почергово. Першим стовпом b_1 обирається документ T^*_1, для якого величина F^* виявляється максимальною.

Крок 4. Призначаємо другим стовпом довільно обраний документ і підраховуємо для нього середню редуковану FRiS-функцію.

Крок 5. Крок 4 повторюється при призначенні стовпами всіх m документів кластера почергово, крім існуючого стовпа b_1. Другим стовпом b_2 вибирається документ T^*_2, для якого величина F^* виявляється максимальною.

Крок 6. Замість одного кластеру C_{first} створюємо два нових з центроїдами b_1 та b_2.

Крок 7. Після того, як були знайдені два нових стовпи, уся колекція розподіляється за наступним правилом: документ відноситься до того кластеру, для якого близькість $r1$ до найближчого центроїду максимальна.

Крок 8. Якщо при k=1 у перший кластер C_{first} входили всі m документів, то тепер при k=2 документи розподіляються між двома новими кластерами. При цьому може вийти так, що для опису кластера C_1 найкращим виявиться не стовп b_1, а якийсь інший документ із цього кластера. Для покращення місця розташування

стовпа b_1 виконаємо наступну процедуру. Почергово для кожного кластера, у нашому випадку для C_1,C_2, призначаємо внутрішні документи на роль стовпа, і за допомогою виразу (6) обчислюємо середню FRiS-функцію $F(T_{1i},b_i)$. Центроїдом вибирається той документ, що забезпечує максимальну величину FRiS-функції. Аналогічно у кластері C_2 визначається нове положення стовпа b_2 на основі максимуму функції $F(T_{2i},b_i)$. На цьому етапі замість редукованої FRiS-функції використовуємо звичайну, що відображає процес конкуренції між реальними стовпами.

Крок 9. Для подальшого розбиття сукупності документів, необхідно вибрати кластер з мінімальним загальним відхиленням від середнього значення сумісної близькості документів. Підрахунок її виконуємо за допомогою виразу (9)

$$E_k = \sqrt{\sum_{i=1}^{nk} (sim(T_i, b_k) - middleSim_k)^2},$$
$$middleSim_k = \frac{1}{n_k} \sum_{i=1}^{nk} sim(T_i, b_k) \quad , \qquad (9)$$

де n_k – кількість документів у кластері.

Кластер з мінімальною сумісною близькістю E_k вибираємо для подальшої роботи.

Крок 10. Процес продовжується, переходячи на крок 2, тобто у вибраному кластері шукаємо 2 стовпа за допомогою описаного вище алгоритму, створюємо 2 нових кластера, перерозподіляємо всі документи між всіма стовпами, і уточнюємо місце розташування центроїда. Процес повторюється доти, поки колекція документів не поділиться на k кластерів.

5. Програмна реалізація

Реалізація і дослідження розробленого алгоритму виконується у рамках розробки системи прийняття рішень на платформі SmartBase.

Для виконання досліджень була використана колекція документів з Інтернет-ресурсу аналітичної газети «Дзеркало тижня» за 3 періоди. Вона включає новини з 26 різних рубрик. Рубрики використовувались як еталони у подальшому дослідженні.

Для стемінгу, тобто виділення нормальної форми слова та виявлення належності його до тієї або іншої частини мови, було використано вдосконалений алгоритм SnowBall.

У виконаних експериментах використовувався список стоп-слів. Він зчитувався з файлу stopword.xml, створеного на основі списку Штейнфельдта, а також з переліку слів, вільно розповсюджуваного компанією "Яндекс" продукту Yandex.Server-FREE-020-3.8.3. Список включає 279 часто вживаних слів російської мови.

Для видалення HTML-тегів була розроблена власна бібліотека HTML Parser, що надає потрібну функціональність.

Програмний інтерфейс складається з web-застосування, яке надає можливість зчитувати новини з сайту http://www.zn.ua/ за певний період чи з локального файлу даних. Вхідними даними для алгоритму є максимальна кількість кластерів k, $k \in (1,...,n)$, де n – кількість документів у колекції. У алгоритмі кластеризації використовується лише тіло документу (основний текст), заголовок і додаткова інформація не враховуються.

Експерименти проводились на двопроцесорному сервері Intel(R) Xeon (TM) CPU 2,40 GHz 2,40 GHz 1GB RAM. Результати передавались клієнту по http протоколу. Характеритики клієнта наступні: Intel(R) Core(TM) 2 Duo CPU 2,33 GHz 2GB RAM.

6. Методика проведення експерименту

Кожний документ із колекції методом експертної оцінки було віднесено до певних рубрик (наприклад, людина, внутрішня політика, міжнародна політика, право, тощо). Максимальна кількість таких рубрик 26. Але для кожної конкретної колекції документів рубрик може бути менше. На основі використання цих даних були створені еталонні кластери.

Існує декілька характеристик оцінки якості роботи текстового класифікатора. Найбільше поширення одержали точність (P) і повнота (R). Вони також використовуються при оцінці якості пошуку за запитом, наприклад, у пошукових машинах мережі Інтернет.

Під правильною і неправильною рубрикацією будемо розуміти випадки, коли класифікатор приписує аналізований документ деякій рубриці, що розцінюється деяким експертом як вірне і невірне рішення відповідно. Під невірнім відсіюванням документу розуміємо випадок, коли класифікатор не приписує документ рубриці, що, на думку експерта, невірно.

Для аналізу отриманих результатів використовувалися графіки точності/повноти. Незважаючи на широку поширеність і популярність метрик точності і повноти у задачах інформа-

ційного пошуку, стосовно задачі розбиття документів на кластери їх застосування вимагає деяких уточнень. Введемо наступні позначення:

C_{ei} – i-й нетривіальний (утримуючий більше одного документа) еталонний (складеними експертами) кластер;

C_j – j-й нетривіальний текстовий (побудованою системою) кластер;

$$M_{ij} = \begin{cases} |C_{ei} \cap C_j|, при |C_{ei} \cap C_j| > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$I_{ij} = \begin{cases} 1, при |C_{ei} \cap C_j| > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$U_j = \begin{cases} \sum_i I_{ij}, при \sum_i I_{ij} > 1; \\ 0, у \quad іншому \quad випадку; \end{cases}$$

$$K_i = \begin{cases} \sum_j I_{ij}, при \sum_j I_{ij} > 1; \\ 1, у \quad іншому \quad випадку \end{cases}$$

Тоді точність є відношенням різниці потужності перетину документів в еталонних і в тестових кластерах і кількості повторно використаних (при побудові даного перетину) еталонних кластерів до ефективної кількості документів у тестових кластерах:

$$P = \frac{\sum_{ij} M_{ij} - \sum_j U_j}{\sum_i |C_{ei}| \cdot K_i}$$

Повнота виражається відношенням різниці потужності перетину документів в еталонних і в тестових кластерах і кількості повторно використаних (при побудові даного перетину) еталонних кластерів до кількості документів в еталонних кластерах:

$$R = \frac{\sum_{ij} M_{ij} - \sum_j U_j}{\sum_j |C_j|}$$

7. Результати експериментальних досліджень

Дослідження проводились на трьох колекціях документів, за різний проміжок часу:

- за період з 29.12.2007 по 26.04.2008, 1515 документів;
- за період з 26.04.2008 по 27.00.2008, 1543 документів;
- за період з 31.05.2008 по 5.06.2008, 418 документів;

У тестуванні брало участь 2 алгоритми: розроблений авторами алгоритм Cluster і алгоритм FRiS Cluster [11] (далі FRiSCluster). Останній обраний за результатами досліджень, наведених у праці [9], як кращий з існуючих алгоритмів цього класу. Обидва алгоритми оперують поняттям центра кластеру. Після закінчення своєї роботи вони надають користувачу не тільки результуючий розподіл, але і еталонні зразки - стовпи.

На рисунках 1 - 3 наведені результати оцінки якості кластеризації за показником точності/повноти для різної кількості документів у колекції новин, від 100 до 1600. Кожний документ було представлено за допомогою двох десятків найбільш інформативних термів.

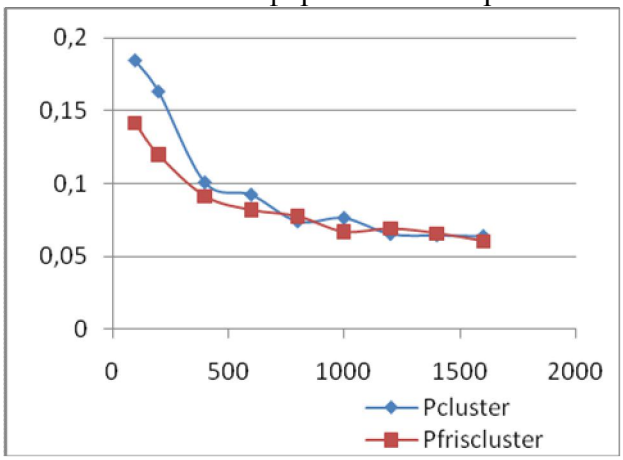


Рис. 1.Порівняння точності алгоритмів

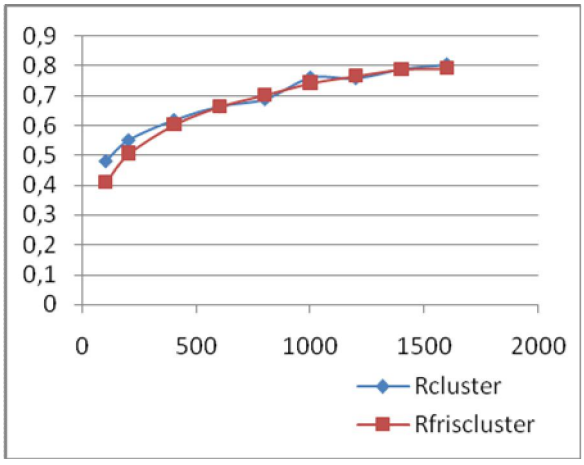


Рис. 2. Порівняння повноти кластеризації

Графіки точності і повноти, зображені на рисунках 1 і 2 відповідно, показують що розроблений алгоритм має співставні з конкурентом, який є одним із найкращих серед відомих алгоритмів, результати. Тільки при невеликій кіль-

кості документів його точність відчутно перевищує точність алгоритму FRiSCluster

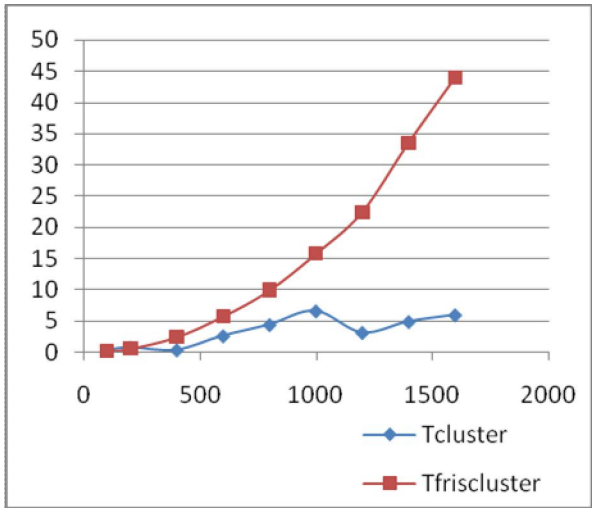


Рис. 3. Порівняння швидкості роботи алгоритмів

На рисунку 3 відображено залежність часу роботи алгоритмів у секундах (ордината) від кількості документів (абсциса). Судячи з графіків, алгоритм Cluster має суттєві переваги, час його роботи менш залежить від збільшення кількості документів у колекції.

Необхідно звернути увагу на невелику точність результатів. Це пояснюється насамперед недосконалістю алгоритму стемінгу, не врахуванням віднесення слова до тієї або іншої частини мови, не врахуванням синонімії, тобто браком засобів виявлення значущих слів та їх відношень.

8. Перспективи досліджень

Для поліпшення показників якості роботи прикладного застосування і розширення сфери його використання видається необхідним виконати такі роботи:

1. Розробка, реалізація і дослідження більш ефективного алгоритму стемінгу.
2. Розробка, реалізація і дослідження моделей представлення семантики і методу кластеризації колекції документів на основі статистичної і семантичної близькості, що дозволить врахувати синонімію та інші відношення термів.
3. Розробка, реалізація і дослідження алгоритму виявлення словосполучень.
4. Розширення кількість мов, з якими працює текстовий кластеризатор.

Висновки

5. Запропоновано підхід до кластеризації колекції документів з невідомою наперед кількістю кластерів на основі статистичних показників, що характеризують кількість входжень ключових термінів у документах і колекції у цілому.
6. Удосконалено метод знаходження матриці подібності на основі схожості косинуса, для аналізу якості й складності якого використана модифікація функції конкурентної подібності.
7. Підхід реалізований у вигляді прикладного застосування сервера SmartBase і виконане його експериментальне дослідження з використанням наявного текстового корпусу. Результати досліджень підтверджують працездатність запропонованих рішень і їх відповідність за показниками точності і повноти відомим розробкам, а за показником швидкості перевищують їх.

Перелік посилань

1. Дж Солтон. Динамические библиотечно-информационные системы. М.: - Мир, 1979.- с.557.
2. Козлов Д.Д. Проблемы применения методов поиска тематических сообществ к задаче тематического информационного поиска в интернет // Труды Всероссийской научной конференции "Методы и средства обработки информации" -М.: Издательский отдел факультета ВМиК МГУ, 2003. - С. 211-215.
3. AllaZaboleeva, Yulia Orlova Computer-aided system of semantic text analysis of a technical specification//Information Technologies and Knowledge. – 2008. – Vol.2. – P.139-145.
4. Vassilis G. Kaburlasos Unified Analysis and Design of ART/SOM Neural Networks. Heidelberg: - Springer Berlin, Volume 4507, 2007.-p 80-93
5. MacQueen J. Some methods for classification and analysis of multivariate observations // Proceedings of the 5th Berkley Symposium on Mathematical Statistic and Probability, University of California Press, 1967, Vol.1, p. 281-297.
6. Peter Grabusts A Study of Clustering Algorithm Application In RBF Neural Networks. //Information technology and management science. - Riga, - 2001. - 5.serija., p.50-57.
7. Корунова Н. В., Кластеризация документов проектного репозитария на основе нейронной сети Кохонена // Труды конф. «Нечеткие системы и мягкие вычисления». – М. - 2008. – С. 77-86.

8. Киселев М. В., Пивоваров В. С., Шмулевич М. М. Метод кластеризации текстов, учитывающий совместную встречаемость ключевых терминов, и его применение к анализу тематической структуры новостного потока, а также ее динамики// Автоматическая обработка веб-данных. - М., 2005. - С. 412-435.
9. Красильников П.В. Воспроизведение лучших результатов ad hoc поиска семинара РОМИП. - М.: IMAT, 2007. – 220с.
10. Борисова И.А., Дюбанов В.В., Загоруйко Н.Г., Кутненко О.А. Использование FRiS-функции для построения решающего правила и выбора признаков // Материалы Всероссийской конференции с международным участие «Знания – Онтологии – Теории» (ЗОНТ–07), Новосибирск, 2007 г. – Т. 2. – С. 67-76.
11. Борисова И.А., Загоруйко Н.Г., Функции конкурентного сходства в задаче таксономии //Материалы Всероссийской конференции с международным участие «Знания – Онтологии – Теории» (ЗОНТ–07), Новосибирск, 2007 г. – Т. 2. – С. 77-86.

УДК 62-82:658.512.011.56

АЛЕНИН О.И.

СИСТЕМА УПРАВЛЕНИЯ ВИРТУАЛЬНЫМИ КЛАСТЕРАМИ

Рассмотрен способ повышения эффективности использования кластерных систем с помощью виртуализации. Сделан краткий обзор типов виртуализации, выбран наиболее подходящий. Предложена структура системы управления виртуальными кластерами в рамках физического кластера.

The way to increase cluster system efficiency using virtualization is considered. The short review of virtualization types is made, and the most suitable is chosen. The structure of management system for virtual clusters over a physical cluster is proposed.

В настоящее время в связи с расширением круга научно – исследовательских задач требуются значительные вычислительные мощности для расчетов, как правило, большие, чем вычислительные мощности соответствующих организаций. При этом вычислительные ресурсы других организаций, доступные удаленно, не удовлетворяют требованиям задач из-за отсутствия на них необходимого специального программного обеспечения [1]. Часто это программное обеспечение может работать только на определенной платформе.

В настоящее время для решения этой задачи используются различные системы виртуализации. В большинстве случаев они позволяют виртуализировать только вычислительные узлы [2]. Вторая проблема заключается в том, что при этом основное внимание уделяется виртуализации вычислительных ресурсов, а виртуализация дискового пространства и сети выполняется неэффективно или не выполняется вообще. С переходом на Грид технологии возникает новая проблема, связанная с тем, что хотя грид может предоставить доступ к большому числу разнообразных ресурсов, часто эти ресурсы не соответствуют требованиям конкретных приложений или сервисов[3]. В вычислительной среде, где программное обеспечение развивается быстро, это несоответствие может привести к недоиспользованию ресурсов, недовольству пользователей и необходимости затрачивать большие усилия на преодоление несоответствия между ресурсами и приложениями. Эти проблемы могут быть решены путем создания кластеров виртуальных машин, с соответствующим набором программного обеспечения, необходимого для запуска приложений. Анализ подобного показывает эффективность его использования в грид – приложениях. При использовании виртуальных машин для запуска приложений потери производительности из-за использования виртуализации не превышают 5%. В работе [4] показано также, что учитывая время на создание и развертывание виртуальной среды в планировщике, а не оставляя этот процесс пользователю, можно достичь значительно большей эффективности использования ресурсов и более точного соответствия требованиям ко времени выполнения заданий. Учет в планировщике времени на передачу и развертывание образов виртуальных машин имеет два преимущества: во-первых, передачу образа можно запланировать и выполнить заранее, во-вторых, можно кешировать часто используемые образы виртуальных машин.

В статье [5] описан способ выделения ресурсов с помощью виртуальных машин, в котором предлагается модифицировать и расширить функции существующих планировщиков, таких как PBS и Sun Grid Engine, что также приводит к повышению эффективности использования ресурсов. Современные кластерные системы используются для разных типов задач. Некоторые задачи требуют большого объема ресурсов, но не накладывают строгих ограничений на время выполнения, в то время как другие задачи должны выполняться с максимальным приоритетом в заданный момент времени [6]. Использование виртуализации предоставляет возможность приостанавливать и возобновлять выполнение задач, переносить задачи вместе с виртуальными машинами на другие физические узлы, а также предоставлять вычислительные ресурсы (виртуальные машины) с предустановленным программным обеспечением, необходимым для выполнения данной задачи.

Есть несколько факторов, оказывающих существенное влияние на эффективность при-

менения виртуализации для систем с кластерной архитектурой:

- используемый тип виртуализации;
- выбор между виртуализацией отдельных ресурсов и построением виртуальных кластеров;
- эффективность работы системы управления ресурсами;
- класс задач, решаемых на кластере.

Для выполнения параллельных прикладных программ, изначально рассчитанных на работу в системах с кластерной архитектурой, целесообразно виртуализировать не отдельные ресурсы кластера, а создать на основе физических ресурсов виртуальный кластер, наиболее точно соответствующий требованиям задач. Для запуска на кластере множества не параллельных программ виртуализация отдельных ресурсов может оказаться предпочтительнее из-за отсутствия накладных расходов на систему управления виртуальными кластерами. Однако, большинство программ, выполняемых на кластерных системах, являются параллельными, и здесь будет рассматриваться только этот вариант.

Типы виртуализации

Существуют такие типы виртуализации: *эмуляция аппаратуры*, *полная виртуализация*, *паравиртуализация*, *виртуализация уровня операционной системы* и *виртуализация уровня приложений*.

При использовании *эмуляции аппаратуры* полностью эмулируется архитектура целевой машины, фактически происходит интерпретация команд гостевого процессора на хост-процессоре. Это самый медленный тип виртуализации, программы выполняются в сотни раз медленнее, чем на физической машине. Примеры систем – Bochs, QEMU.

Полная виртуализация – самый популярный способ виртуализации, предполагает использование программного обеспечения, получившего название «гипервизор», суть которого заключается в создании уровня абстракции между виртуальными серверами и базовым аппаратным обеспечением. Примерами коммерческих решений, в которых реализован данный подход, могут служить программные продукты VMware и Microsoft Virtual PC, а KVM (Kernel Virtual Machine) – это свободно распространяемое решение для ОС Linux. Гипервизор перехватывает команды центрального процессора и служит посредником для доступа к аппаратным контроллерам и периферии. В результате полная виртуализация позволяет установить на виртуальный сервер практически любую операционную систему без каких-либо изменений, причем сама ОС ничего не будет знать о том, что она работает в виртуализованной среде. Основной недостаток данного подхода связан с накладными расходами, которые несет процессор в связи с работой гипервизора. Эти накладные расходы невелики, но ощутимы [7]. В полностью виртуализованной среде гипервизор взаимодействует непосредственно с аппаратным обеспечением и серверами в качестве хостовой операционной системы. Операционные системы, работающие на виртуальных серверах, которыми управляет гипервизор, называют гостевыми.

Полная виртуализация предполагает серьезное использование ресурсов процессора, обусловленное наличием гипервизора, управляющего различными виртуальными серверами и обеспечивающего независимость этих серверов друг от друга. Уменьшить эту нагрузку можно, например, модифицировав каждую операционную систему таким образом, чтобы она «знала» о том, что она работает в виртуализованной среде, и могла взаимодействовать с гипервизором. Такой подход называют *паравиртуализацией*. Примером свободно распространяемой реализации технологии паравиртуализации может служить Xen. Прежде чем операционная система сможет работать в качестве виртуального сервера в гипервизоре Xen, в нее необходимо внести определенные изменения на уровне ядра.

Существует еще один способ виртуализации – встроенная поддержка виртуальных серверов на уровне операционной системы. Этот подход называется *виртуализацией на уровне операционной системы* и использован, например, в Solaris Containers. Существует также Virtuozzo/OpenVZ для ОС Linux. При виртуализации на уровне операционной системы не существует отдельного слоя гипервизора. Вместо этого сама хостовая операционная система отвечает за разделение аппаратных ресурсов между несколькими виртуальными серверами и поддержку их независимости друг от друга. Отличие этого подхода от других проявляется, прежде всего, в том, что в этом случае все виртуальные серверы должны работать в одной и той же операционной системе (хотя каждый экземпляр имеет свои соб-

ственные приложения и регистрационные записи пользователей). Виртуализация на уровне операционной системы теряет в гибкости, но производительность близка к производительности физического сервера. Кроме того, системой, которая использует одну стандартную ОС для всех виртуальных серверов, намного проще управлять, чем более гетерогенной средой.

Виртуализация прикладных приложений включает в себя рабочую среду для локально выполняемого приложения, использующего локальные ресурсы. Виртуализируемое приложение запускается в небольшом виртуальном окружении, которое включает в себя ключи реестра, файлы и другие компоненты, необходимые для запуска и работы приложения. Такая виртуальная среда работает как прослойка между приложением и операционной системой, что позволяет избежать конфликтов между приложениями.

Каждый тип виртуализации обладает своими достоинствами и недостатками. При применении виртуализации для повышения эффективности использования кластерных систем к выбранному типу виртуализации предъявляются такие основные требования:

- невысокие накладные расходы на виртуализацию;
- возможность запуска в гостевой среде операционных систем с предустановленным программным обеспечением.

Следовательно, нецелесообразно использовать эмуляцию аппаратуры и невозможно использовать виртуализацию уровня приложений. Использование паравиртуализации целесообразно только в тех случаях, когда модификация гостевой ОС не представляет трудностей. Наиболее приемлемым является использование виртуализации уровня ОС, если гостевая ОС близка к базовой (например, различные версии Linux), и полной виртуализации для запуска других гостевых ОС (например, запуск Windows на Linux).

Система управления виртуальными кластерами

Как уже было сказано, одним из факторов, существенно влияющих на эффективность использования виртуализации в кластерных системах, является система эффективная работа системы управления виртуальными кластерами. Она управляет виртуальными узлами, сетями, дисковыми ресурсами, осуществляет создание и удаление виртуальных кластеров, мониторинг состояния узлов и т.д.

Для возможности одновременной работы нескольких виртуальных кластеров с разным набором прикладного ПО на одном физическом кластере, необходимо, чтобы с точки зрения гостевых ОС виртуальные кластера были изолированными друг от друга, использовали разные дисковые и сетевые ресурсы. Такая схема показана на рис. 1

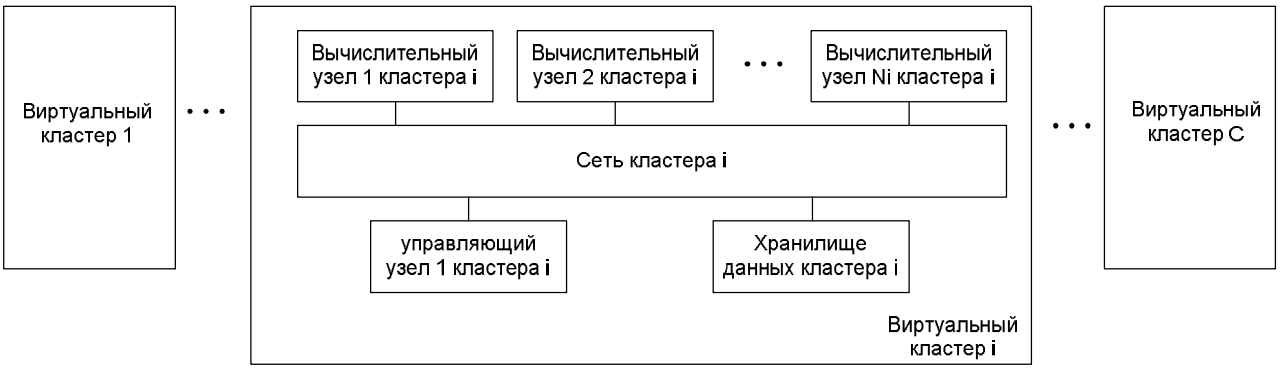


Рис.1 Виртуальные кластера

В соответствии с такой схемой, система управления виртуальными кластерами должна содержать блоки, обеспечивающие управление вычислительными, сетевыми и дисковыми ресурсами. На рис. 2 показана схема системы управления виртуальными кластерами и взаимодействие ее с другими системами, обеспечивающими работу физического кластера, такими как система управления ресурсами физического кластера, интерфейс загрузки заданий, система хранения.

Систему можно разделить на несколько блоков. Блок управления вычислительными ресурсами отвечает за создание, удаление и запуск виртуальных узлов на физических узлах. Блок управления системой хранения от-

вечает за распределение доступного дискового пространства, создание и удаление виртуальных дисковых ресурсов. Блок управления сетевыми службами осуществляет создание, удаление и настройку виртуальных интерфейсов, распределение адресов, управляет сетевыми сервисами, необходимыми для загрузки виртуальных узлов. Блок обработки очереди заданий связан непосредственно с интерфейсом загрузки заданий (или с системой управления заданиями физического кластера, если физические ресурсы виртуализируются не полностью, и есть возможность запуска заданий в базовой ОС физических узлов). База данных с информацией о ресурсах содержит все записи, необходимые для работы остальных блоков.

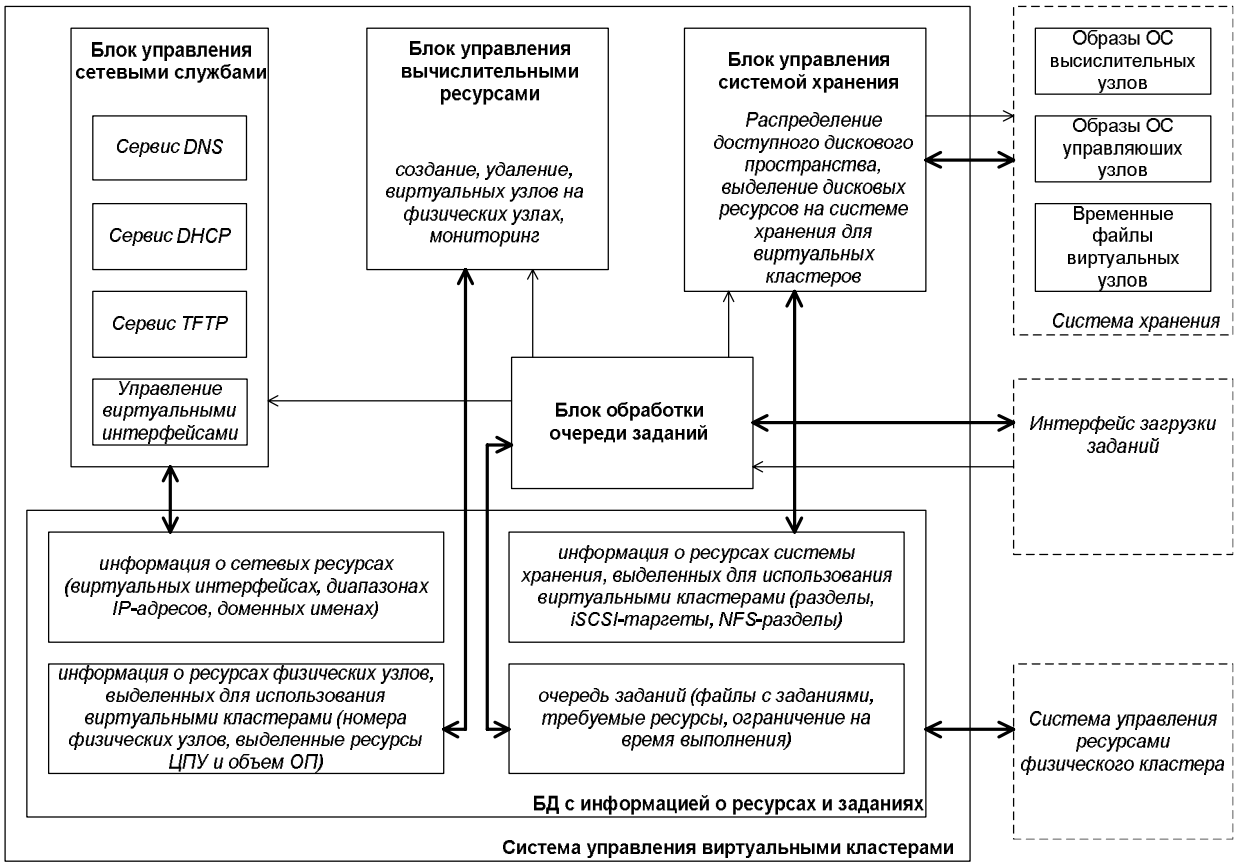


Рис.2 Система управления виртуальными кластерами

Создание/удаление виртуальных кластеров происходит таким образом (рис 3).

Через интерфейс пользователя в блок обработки очереди заданий поступает задание, состоящее из образа виртуального управляющего и вычислительного узлов с предустановленным прикладным программным обеспечением, которое необходимо для решения задачи пользователя, и дополнительных параметров. В дополнительных параметрах задания могут быть указаны такие как необходимый размер оперативной памяти, производительность процессора, количество узлов, требуемое дисковое пространство для хранения временных файлов, требования к сети обмена данными между узлами и т.д. Образы ОС виртуальных узлов передаются на систему хранения и сохраняются там. В базе данных сохраняются ссылки на месторасположение образов ОС узлов в системе хранения с привязкой к данному заданию. При наличии свободных ресурсов в соответствии с дисциплиной обслуживания выбирается задание из очереди. После этого выполняется выделение ресурсов, и загрузка виртуальных узлов на виртуальных машинах в физических узлах. Образ ОС поступает с системы хранения. После загрузки выполняется автоматическая настройка загруженных виртуальных узлов, с помощью протокола DHCP задаются настройки сети, выполняется монтирование каталогов или виртуальных дисков для записи временных файлов.

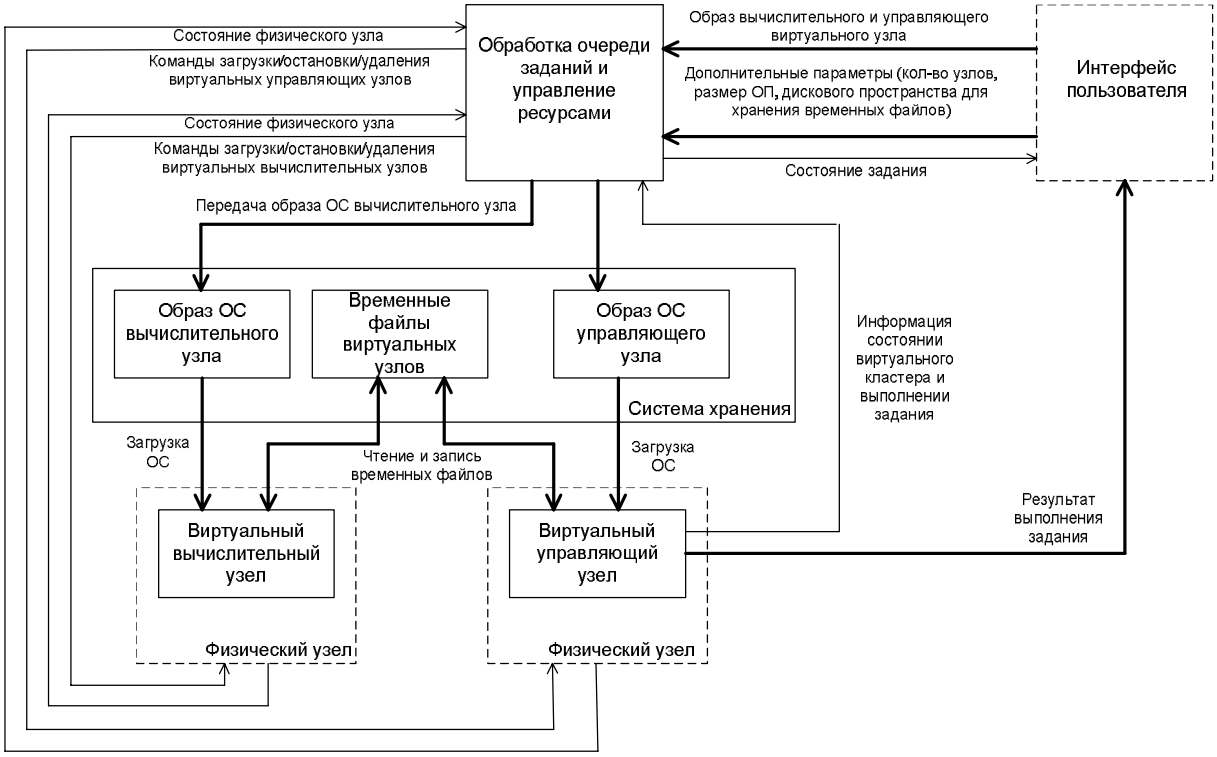


Рис.3 Схема обмена данными и управления при создании и удалении виртуальных кластеров

В процессе работы осуществляется мониторинг состояния физических узлов. При выходе из строя физического узла виртуальные узлы, работавшие на нем, переносятся на работоспособные физические узлы. Это осуществимо потому, что образы ОС и промежуточные результаты работы прикладных программ сохраняются не на локальных дисках физических узлов, а в системе хранения.

После завершения выполнения прикладных программ с помощью интерфейса пользователя подается команда удаления виртуального кластера. Это может выполняться как вручную, так и автоматически при передаче результатов выполнения прикладной программы. При удалении виртуального кластера выполняется размонтирование подключенных дисковых ресурсов, удаление виртуальных сетевых интерфейсов, остановка соответствующих виртуальных машин на физических узлах, и в базе данных о ресурсах занятые ресурсы помечаются как свободные.

Список литературы

1. Keahey, K., T. Freeman, J. Lauret, D. Olson. Virtual Workspaces for Scientific Applications, SciDAC 2007 Conference, Boston, MA. June 2007.
2. NAKADA, H., YOKOI, T., EBARA, T., TANIMURA, Y., OGAWA, H., AND SEKIGUCHI, S. The design and implementation of a virtual cluster management system. In Proceedings of the first IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management, 2007.
3. Foster, I., T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. Virtual Clusters for Grid Communities, CCGRID 2006, Singapore. May 2006.
4. Overhead Matters: A Model for Virtual Resource Management, Sotomayor, B., K. Keahey, I. Foster. VTDC 2006, Tampa, FL. November 2006.
5. Enabling Cost-Effective Resource Leases with Virtual Machines, Sotomayor, B., K. Keahey, I. Foster, T. Freeman. HPDC 2007 Hot Topics session, Monterey Bay, CA. June 2007
6. Combining Batch Execution and Leasing Using Virtual Machines, Sotomayor, B., K. Keahey, I. Foster. HPDC 2008, Boston. June 2008.
7. Jones T. An overview of virtualization methods, architectures, and implementations, IBM, 2006, http://www.ibm.com/developerworks/linux/library/l-linuxvirt.

УДК 004.22

БАНЯ Е.Н.,
СЕЛИВАНОВ В.Л.

ОБ ОСОБЕННОСТЯХ ПОСТРОЕНИЯ СИСТЕМ СЧИСЛЕНИЯ РАЗНЫХ КЛАССОВ

Класс систем счисления целесообразно определять порядком вычисления символов. В системах счисления класса А, к которым относятся позиционные системы, порядок вычисления символов зависим и последовательный. В системах класса В, к которым относятся системы счисления остаточных классов, порядок вычисления символов параллелен и независим. Дана классификация систем счисления.

The class of scales of notation should be defined by the order of symbol's calculation. In A class scales of notation, to which positional scales of notation referred to, the order of symbol's calculation is dependent and successive. In scales of notation of B class, to which scales of notation of vestigial classes referred to, the order of symbol's calculation is parallel and dependent. The classification of scale of notation is given.

История развития вычислительной техники непрерывно связана с разработокой и внедрением все более новых принципов представления и кодирования числовой информации. Количество систем счисления, которые используются в цифровой вычислительной технике, непрерывно растет. На ряду с двоичной и двоично-десятичными системами счисления известны примеры примененииия троичной, двоично-пятеричной, систем счисления, систем счисления с отрицательным, комплексным, иррациональными основаниями, системы счисления остаточных классов. В настоящее время нет единой классификации систем счисления, четкого разделения на классы. В научно-технической литературе встречаются противоречивые и неоднозначные трактовки базовых характеристик и параметров систем счисления, что значительно усложняет разработку теории проектирования преобразователей форм представления информации. Цель данной работы – устранение противоречий, имеющихся неточностей в определениях и терминологии, разработке классификация систем счисления.

Построение произвольной системы счисления следует начать с выбора базовых параметров, характеризующих данную систему, и разработки определенных правил, позволяющих представить любую величину в этой системе.

Под системой счисления будем понимать такой способ изображения множества чисел с помощью ограниченного набора символов, составляющих ее алфавит, при котором эти символы (элементы алфавита) располагаются в установленном порядке, занимая определенные места (позиции).

В качестве базовых параметров произвольной систем счисления выбирают:

1. Максимальную длину последовательности (общее количество позиций (разрядов)) $-(n+1)$
2. Номер позиции (разряда) – i $(i=\overline{0,n})$
3. Допустимое для данной позиции количество символов l_i
4. Возможные значения символа в i – позиции $a_i^j[\,j=\overline{1,l_i}\,]$
5. Набор символов в i-позиции – $A_i \in [a_i^1, a_i^2, ..., a_i^{l_i}]$
6. Полный набор символов, применяемых в системе счисления составляет алфавит систем счисления – А.
7. Значение (мера) i-й позиции (количественное значение каждой единицы i – позиции или количественный эквивалент i – позиции) - Q_i
8. Базис системы счисления (набор выбранных мер) - $B \in [Q_0, Q_1, ..., Q_n, Q_{n+1}]$
9. Диапазон представления чисел – Q
10. Основание, характеризующее i – позицию – p_i
11. Набор оснований $p \in [p_0, p_1, ..., p_n]$

В произвольной системе счисления целое положительное число N изображается последовательностью символов

$$[N] = a_n^j a_{n-1}^j ... a_2^j a_1^j a_0^j,$$

где $[N]$ – представление числа в этой системе счисления.

Такая запись означает, что величина числа может быть определена по формуле:

$$[N] \equiv \sum_{i=0}^{n} a_i^j Q_i \bmod Q$$

Следовательно, в любой системе счисления величина числа зависит как от значений символов a_i^j, так и от количественных мер каждой позиции Q_i.

Порядок вычисления символов a_i^j определяется выбранным классом систем счисления. Символы могут вычисляться либо последовательно во времени, начиная со старшего a_n^j или с младшего символа a_0^j, либо одновременно(параллельно) и независимо. Следует заметить, что такие базовые параметры, как базис B, набор оснований $p \in [p_0, p_1, ..., p_n]$ связана между собой с помощью определенных математических соотношений, вид которых зависит от класса системы счисления. В каждой системе счисления должны выполняться также определенные ограничения, накладываемые на выбор мер Q_i, оснований p_i, количества l_i и набора символов A_i в каждой позиции, обусловленные требованиями обеспечения однозначного и непрерывного представления величин из заданного диапазона Q. Под непрерывным представлением понимают возможность представления всех чисел диапазона Q с фиксированной дискретностью (так, на пример, для целых чисел эта дискретность равна 1).

В системах счисления, применяемых в ЭВМ алфавит А представляет собой конечный целочисленный набор.

Построение систем счисления класса А, которые принято называть позиционными ППС, начинают с выбора допустимого для каждой позиции количества символов l_i, формирования алфавита А и вычисления количественных мер Q_i, называемых в этом случае весами разрядов, т.е. нахождения базиса В. Таким образом, для позиционных систем счисления первичными параметрами являются базис В (набор всех мер Q_i), допустимое для каждой позиции количество символов l_i и алфавит А.

В системах счисления, в которых $Q_0 = I$ и $a_i^1 = 0$, для удовлетворения требований обеспечения непрерывности представление величин веса последующих разрядов должны выбираться, исходя из следующего условия

$$Q_i \le \sum_{s=0}^{i-1} Q_s (l_s - 1) + 1 \quad \forall \mathrm{i} = \overline{0, \mathrm{n}}.$$

Выбрав веса, рассчитывают основания, характеризующие каждую позицию по формуле:

$$p_i = \frac{Q_{i+1}}{Q_i}$$

Если все количественные меры, входящие в базис, равны между собой, т.е. $|Q_i| = Q_0$, получаем непозиционную систему счисления МПС, выступающая как частный случай позиционной системы. В этих системах счисления все основания $|p_i| = 1$. В качестве примера рассмотрим непозиционную систему счисления, называемую унарной или единичной, в которой для записи числа применяется всего один символ. В этой системе все

$Q_i = 1, p_i = 1, l_i = 1, j = 1, a_i = 1 \quad \mathrm{A} \in [1], \mathrm{Q} = \mathrm{n} + 1.$

К этому же классу относят и “римскую” систему счисления, в которой для обозначения чисел 1,5,10,50,100,500,1000 используются заглавные буквы I,V,X,..,C,D,M. В “римской” системе $Q_i = 1$ при $\mathrm{a_i^j} \ge \mathrm{a_{i-1}^j}$ и $Q_i = -1$ при $\mathrm{a_i^j} \lhd \mathrm{a_{i-1}^j}$ т.е. $p_i = \pm 1 \quad \mathrm{A_i} \in [I, V, X, L, C, D, M]$

Особенностью позиционных систем состоит в том, что в них базис обязательно включает не все равные между собой количественные меры (веса). Систему счисления, в которой вес каждого следующего разряда не меньше, чем веса всех предыдущих разрядов, будем называть упорядоченной. Очевидно, что в упорядоченных системах счисления обязательно все основания $p_i \ge 1$.

Позиционная система счисления, в которой основания всех разрядов оказались одинаковыми т.е. $p_i = p$ для всех $i = \overline{o, n}$ называется однородной.

В однородной системе счисления, имеющей $Q_o = I$, вес i – разряда вычисляется по формуле

$Q_i = p_i$.

В связи с тем, что в однородной системе используется только одно основание, удобно такую систему называть по значению её основания. Например, систему счисления с основанием $p = 2$ – двоичной, с основанием $p = 3$ – троичной и т.д. Если в позиционной системе счисления веса выбраны таким образом, что не все основания оказались одинако-

вым, получают систему счисления которую принято называть неоднородной. В неоднородной системе счисления вес i-разряда связан с основаниями всех предыдущих разрядов следующим соотношением:

$$Q_i = Q_0 \prod_{k=0}^{i-1} p_k$$

К неоднородным системам счисления, имеющим целочисленные основания, можно отнести систему измерения времени , систему счисления, в которой в качестве основания выбран набор взаимно простых чисел.

В позиционных системах как однородных, так и неоднородной, могут использоваться не только целые, но дробные и иррациональные основания. Так, например, известны примеры использования однородных систем счисления с дробными основаниями, равными $p = 1 + 2^{1-\gamma}$, где γ -может выбраться, равным 2,3 и т.д., иррациональным основанием $p = \sqrt[\gamma]{2}$ для γ =2,3,…; с основанием р, которое равно числу “золотой” S-пропорции, определяемой выражением

$$p = \lim_{n\to\infty} \frac{\varphi_s(n)}{\varphi_s(n-1)}$$,где $\varphi_s(n) - n$ -ое S-число Фибоначчи.

В так называемой факториальной системе счисления, в которой веса $Q_i = (i+1)!$, все основания представляют собой также целые числа, так как

$$p_i = \frac{(i+2)!}{(i+1)!} = (i+2) \text{ для всех } i = \overline{0,n}$$

Можно привести большое число примеров неоднороднных систем счисления с иррациональным основанием. Например, система счисления, в которой веса представляют собой ряд последовательных натуральных чисел, т.е. $Q_i = i+1$, имеет основания, равные

$$p_i = \frac{i+2}{i+1}.$$

Система счисления, в которой вес младшего разряда $Q_i = 1, p_0 = 2$, а веса остальных разрядов представляемых собой четные числа, имеет основания $p_i = 1 + \frac{1}{i}$ для $i = \overline{1,n}$.

Система счисления, в которой веса Q_i равны числам Фибоначчи, т.е

$$Q_i = \varphi_s(i) = \begin{cases} 0, i < 0, \\ 1, i = 0, \\ \varphi_s(i-1) + \varphi_s(i-s-1), i > 0 \end{cases}$$ где

S=1,2,3,…

Тогда $p_i = \frac{\varphi_s(i+1)}{\varphi_s(i)}$

Если выбрано S=1, веса разрядов такой неоднородной системы оказываются равными 1,2,3,5,8,13, и т.д., если выбрано S=2, веса разрядов соответственно равны 1,1,2,3,4,6,9,13 и т.д.

Если базис позиционной системы счисления найден, можно определить в каком количестве присутствует каждая из выбранных мер Q_i, начиная с наибольшего из весов Q_{n+1}. В результате первого деления получаем частное Q_n^j и остаток r_n:

$$N = a_n^j Q_{n+1} + r_n, \text{ где } 0 \le r_n \le Q_{n+1}.$$

Затем делим остаток r_n на следующий вес Q_n

$$r_n = a_{n+1}^j Q_n + r_{n-1}, \text{ где } 0 \le r_{n-1} \le Q_n$$

или $N = a_n^j Q_{n+1} + a_{n-1}^j Q_n + r_{n-1}$

Далее остаток r_{n-1} делим на Q_{n-1} и т.д. Процесс деления продолжаем до тех пор, пока не будет найден последний остаток $r_o < Q_1$. В результате такого деления получили представление числа N в виде последовательности символов a_i^j, начиная со старшего символа a_n^j. Нетрудно убедиться в том, что выполняя последовательное деление исходного числа N получающихся в результате деления частных на основание p_i, начиная с основания p_0, можно определить символы a_i^j, начиная с младшего символа a_0^j.

Заметим, что в каждом разряде количество целочисленных значений l_i зависит от основания данного разряда. Если все p_i - целые числа и количество символов $l_i = p_i$, такая система счисления является однозначной (не избыточной), а при $a_i^1 = 0$ ее называют натуральной.

Если $a_i^1 \ne 0$, при построении системы счисления необходимо оговаривать, какие конкретно символы выбраны для изображения чисел. Системы счисления могут иметь не

только все положительные цифры, но и все отрицательные. Система счисления с нечетным натуральным основанием $p = 2l + 1$ и цифрами $a_i \in [-l, -l+1, ..., -1, 0, 1, .., l-1, l]$ называют симметричными. Такие системы счисления позволяют представить любое целое число как положительное, так и отрицательное. Примером симметричной системы счисления может служить троичная система с цифрами [-1,0,1]. Если p_i – целые, а количество символов l_i выбрано бо́льшим, чем p_i т.е. $l_i > p_i$, система счисления будет неоднозначной. В таких системах счисления одна и та же величина может быть представлена различными последовательностями символов. Например двоичная система счисления с набором символов (-1,0,1). Для дробных и иррациональных значений p_i допустимое количество символов l_i выбирается с округлением в большую сторону, т.е $l_i \geq]p_i[$ и так как при этом всегда оказывается что $l_i > p_i$, такие системы счисления будут всегда неоднозначными.

Если для двух однородных чисел счисления с основанием p_1 и p_2 справедливо соотношение $p_1 = p_2^m$, будем называть такие счисления родственными. Так, например, родственными будут двоичная система с четверичной, восьмеричной, шестнадцатеричной, четверичная с шестнадцатеричной, а система счисления с основаниями 2 и $\sqrt{2}$,основаниями $i\sqrt{2}$ и -2, с основаниями 2i и 4, с основаниями -2 и -8 и т.п.

Для неоднородных систем счисления также существует понятие родственных систем при следующем условии, если

$$p_{i,1} = \prod_{j=S1}^{s2} p_{j,2} \text{ для } i = \overline{0,n}$$

$$S_1 = \begin{cases} 0, i = 0 \\ \sum_{j=0}^{i-1} (n_s + 1), i = \overline{1,n} \end{cases}$$

$$S_2 = \sum_{j=0}^{i} (n_s + 1) = 1; j = \sum_{s=0}^{i-1} (n_s + 1) + k, i = \overline{1,n}, k = \overline{0,n}$$

Следует отметить, что $S_2 - S_1 + 1 = n_i + 1$, где $p_{i,1} - i$ – основание первой системы счисления $p_{j,1} - j$ – основание второй системы счисления.

Если в какой-то системе счисления ее символы представляются с помощью цифр другой системы счисления, то такую систему называют системой с кодированным представлением ее цифр. Десятичная система счисления, в которой каждая десятичная цифра представляется тетрадой из двоичных цифр называется двоично-кодированной. Например, так называемая двоично-десятичная система счисления 8421 представляет собой неоднородную систему счисления, у которой используются основания и веса:

$p_{0,2}$ =2; $p_{1,2}$ =2; $p_{2,2}$ =2; $p_{3,2}$ =1,25; $p_{4,2}$ =2; $p_{5,2}$ =2; $p_{6,2}$ =2; $p_{7,2}$ =1,25;

и т.д.

$Q_{0,2} = 1; Q_{1,2} = 2; Q_{2,2} = 4; Q_{3,2} = 8; Q_{4,2} = 10; Q_{5,2} = 20;$

и т.д.

Другая широко распространенная двоично-десятичная система счисления 2421 представляет собой неоднородную систему счисления с основаниями 2,2,0,5,5; 2,2,0,5,5; и.т.д. обе эти системы счисления являются родственными, так как, по принятому определению:

$$\prod_{j=0}^{3} p_{j,2} = p_1 = 10$$

Построение систем счисления класса В, куда входит система счисления остаточных классов СОК, начинают с выбора основания p_i, которые обязательно являеются целыми, причем, с целью обеспечения однозначности и непрерывности представления набор оснований должен включать p_i, представляющее собой взаимно простые числа.

Выбрав основания, можно рассчитывать величины Q_i:

$$Q_i = m_i \frac{p}{p_i} = m_i \hat{p}_i \text{ , где } m_i = 1,2.. p_i \text{ где}$$

$$|Q_i|_{p_i} \equiv 1, |Q_i|_{p_j} \equiv 0, \forall i \neq j$$

Q_i -ортогональные базисы

т.е. определить ее базис В.

Произвольное число X определяется набором остатков $(x_0, x_1, ..., x_n)$ где $X \equiv x_1 \bmod p_1$ или $X \equiv |x_1|_{p1}$, $X \equiv x_2 \bmod p_2$, $X \equiv x_n \bmod p_n$. Процесс вычисления символов осуществляется независимо и параллельно по каждому основанию p_i.

Таким образом, особенность систем счисления остаточных классов состоит в том, что в них первичными параметрами являются целые, взаимно простые основания. При этом Китайская теорема об остатках гарантирует однозначное представление чисел X в диапазоне[0..P-1], где

$$P = \prod_{i=0}^{n} p_i .$$

Формула перевода из СОК в десятичную систему счисления имеет вид

$$X \equiv | x_1 Q_1 + x_2 Q_2 + ... + x_n Q_n |_P .$$

В литературе утверждается, что СОК является непозиционной символической невзвещенной системой счисления [Л.3]. Пример перевода чисел из СОК в десятичную систему счисления с помощью ортогональных базисов, полностью отрицает это утверждение. Эта система система счисления является представителем совсем другого класса В систем счисления, в которых, еще раз подчеркнем, процесс нахождения символов, в отличии от систем класса А, осуществляется паралельно и независимо.

Двоично-десятичная система счисления получившая название 8421+3, представляет собой смещенную на 3 единицы классическую двоично-десятичную систему 8421. Недопустимо относить ее к “незвешенным” системам счисления, в которых веса отдельных разрядов имееют переменные значения.

Если позиционная система счисления имеет p>>2 (например $p = 2^{16}$) то любой ее символ можно представить с помощью СОК (на пример для $p = 2^{16}$ можно выбрать модули $p_1 = 13, p_2 = 16, p_3 = 17, p_4 = 19$ тогда P=67184> 2^{16})для кодировки этих модулей достаточно иметь 18 двоичных разрядов, что не намного больше 16. Целесообразность такого комбинированного представления объясняется возможностью реализации сумматора, в котором перенос будет распространяться не более чем на 5 разрядов. В результате приведенных выше рассуждений, приведем следующую классификацию систем счисления (Рис. 1).

Выводы

1. Предложено системы счисления разделить на два класса. Класс систем счисления определяется порядком вычисления символов. В системах счисления класса А, к которым относятся позиционные системы, порядок вычисления символов зависим и последовательный. В системах класса В, к которым относятся системы счисления остаточных классов, порядок вычисления символов параллелен и независим.
2. Показано, что к системам класса А относятся непозиционные системы счисления, представляющие собой частный случай позиционных систем счисления.
3. Четко сформулировано понятие неоднозначности систем счисления. Показано, что системы счисления с дробными и иррациональными основаниями всегда неоднозначны.
4. Введено понятие родственных систем счисления, показано, что перевод из одной родственной системы счисления в другую осуществляется перекодировкой символов.

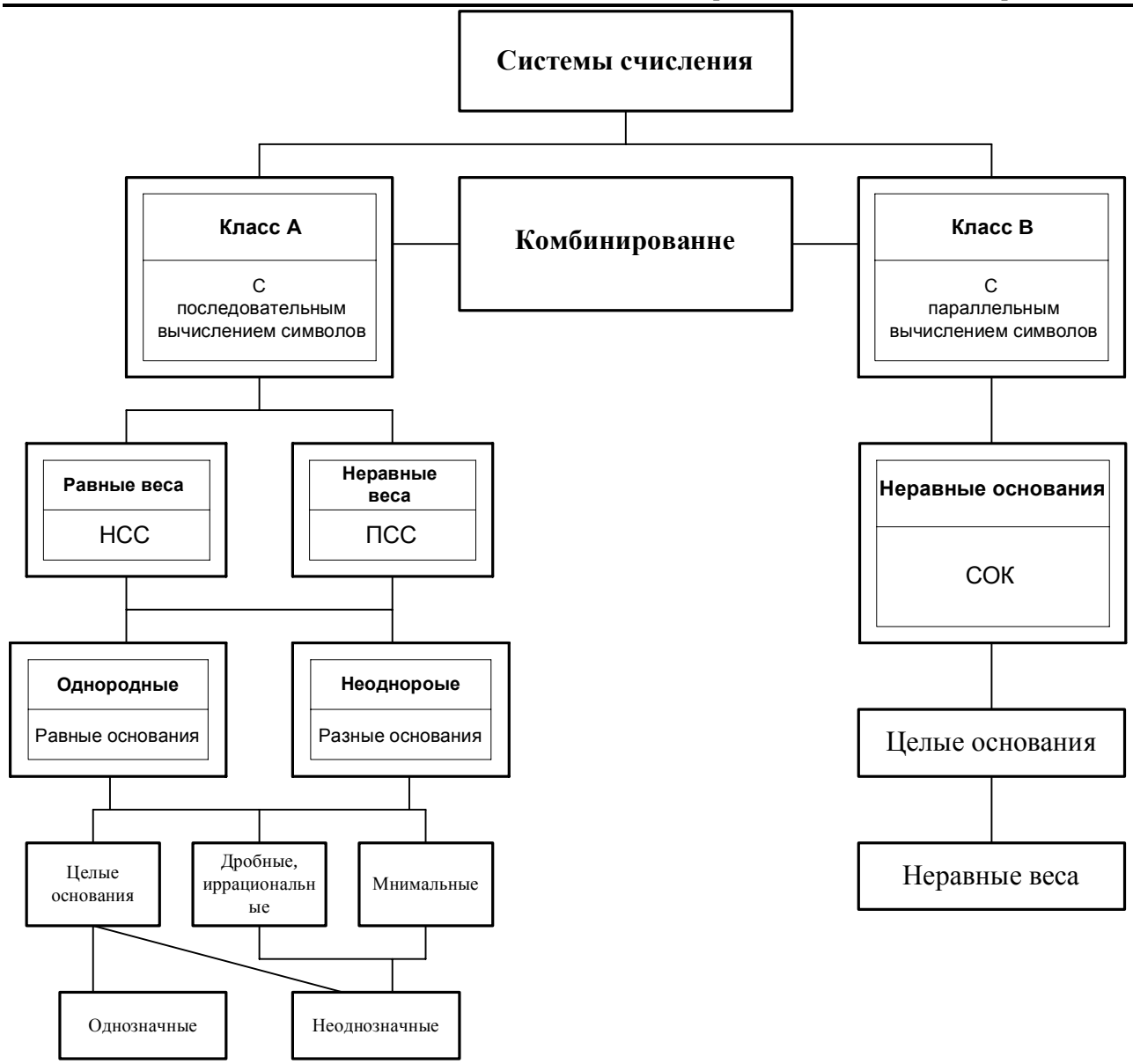


Рис.1.

Список литературы

1. Касаткин В.Н. Новое о системах счисления. – К: Вища школа. С. 82-96.
2. Фомин С.В. Система счисления. – М: Наука. – 1987. – С. 48.
3. Энциклопедия кибернетики. К. Главная редакция УСЭ. – 1974. – 680 с.
4. Лысиков Б.Т. Арифметические и логические основы цифровых автоматов. Минск: В.Ш. – 1980. – 336 с.
5. Стахов А.Н. Введение в алгоритмическую теорию измерений. – М: Сов. радио. –1977. – 287 с.
6. Стахов А.П. Коды золотой пропорции. М: – 1984. – 168 с.
7. Поспелов Д.А. Арифметические основы вычислительной машины дискретного действия. Высшая школа. М: – 1970. – 308 с.
8. Шауман А.М. Основы машинной арифметики. Л: Изд. Ленинград. Университет, – 1978. – 260 с.
9. Корнейчук В.И., Тарасенко В.П. Основы компьютерной арифметики. К. – 2002. – 176с.
10. Е.Н.Баня, В.И.Селиванов. Кодирование знакопеременой информации в цифровых устройствах. Кибернетика №3/ – 1987. С. 81-86.

УДК 681.322

СЕРГІЄНКО А.М.

ПРОЕКТУВАННЯ ОБЧИСЛЮВАЧІВ З РЕГІСТРОВИМИ ЗАТРИМКАМИ

Пропонується методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Показана дієвість методики на прикладі проектування блоку тасування даних.

A method of designing pipelined datapaths which are configured in FPGA is proposed. The method provides the hardware minimization due to the wide utilization of the shift register components. The method is proven at the example of the zigzag scan reordering buffer design.

Програмовані логічні інтегральні схеми (ПЛІС) представляють собою незамінну елементну базу в таких галузях, як цифрова обробка сигналів, телекоммунікації, завдяки можливості організації в них швидкістного виконання алгоритмів в конвейерному режимі. Таке виконання підтримується архітектурою ПЛІС, яка включає велику множину конвейерних блоків множення з додаванням, пам'яті. Але розробка обчислювачів на базі ПЛІС залишається трудомістською, висококваліфікованою роботою. В статті пропонується методика проектування конвейерних обчислювачів, які в повній мірі використовують можливості апаратури сучасних ПЛІС.

В ПЛІС Virtex фірми Xilinx серед бібліотечних компонентів є елемент SRL16, що представляє собою 16-розрядний регістр зсуву з одним входом і одним виходом, який підключається до одного з тригерів цього регістру через багатовходовий мультиплексор, що керується чотирьохрозрядною шиною адреси. Цей елемент є регістровим буфером з регульованою затримкою або просто регістровою затримкою (РЗ) і призначений саме для організації конвейерних обчислень складних потокових алгоритмів. Ресурси РЗ досить великі – кожну другу логічну таблицю ПЛІС можна зконфігурувати як РЗ SRL16. Причому в критерії ефективності складність одного елемента SRL16 оцінюється як складність одного – двох тригерів. Крім того, одна РЗ може замінити собою до 16 регістрів, а також зекономити відповідні програмовані лінії зв′язку. Цим самим можна не тільки зменшити апаратні витрати, але і збільшити швидкодію обчислювача, зменшивши кількість ліній з′єднання з великою затримкою. Тому має сенс розробити методику еффективного використання ресурсів РЗ.

Загальний підхід до розробки функціональної схеми на рівні регістрових передач полягає в наступному. Вибирається множина ресурсів (суматорів, блоків множення, пам′яті і т.і.), складається розклад виконання операцій алгоритму і виконується призначення операцій на ресурси. За цим знаходять множину необхідних регістрів і мережу комутації ресурсів. Такий підхід також застосувють для проектування схеми з РЗ. При цьому ланцюжки регістрів, побудованих в схемі, заміняються на відповідні РЗ. Але ланцюжки регістрів в такій схемі виникають випадково і тому їх виявляється значно менше, ніж можливо і таким чином, РЗ в схемі задіяні нефективно. Крім того, властивість затримки РЗ, яка змінюється динамічно, не використовується.

В роботах [1–3] пропонується метод проектування конвейерних обчислювачів шляхом відображення графа синхронних потоків даних (ГСПД), який представлено в просторі ресурси – час у вигляді конфігурації алгоритму (КА). Метод дає змогу одночасно як складати розклад, мінімізувати кількість процесорних елементів (ПЕ), так і шукати ефективну систему з′єднань між цими ПЕ. Тут під ПЕ розуміється елементарний обчислювач з пам′яттю або без неї, наприклад, суматор, мультиплексор з регістром, РЗ, тощо. Тому має сенс створити методику розробки пристроїв з РЗ на основі цього методу.

На першому етапі синтезу за вказаним методом вершини-оператори однорідного ГСПД разом з дугами розташовуються в трьохвимірному просторі як множини векторів $\boldsymbol{K}_i$ та $\boldsymbol{D}_j$ з урахуванням умов, приведених в [2,3]. При цьому координати вектора $\boldsymbol{K}_i = (s,q,t)^T$ означають номер s ПЕ, де виконується оператор, тип q ПЕ і часову складову t, яка дорівнює номеру такту в періоді виконання алгоритму. Вектори

$\boldsymbol{K}_i$ з однаковою часовою складовою формують один ярус і тому виконуються одночасно. Часова складова $R(\boldsymbol{D}_j)$ вектора $\boldsymbol{D}_j = \boldsymbol{K}_i - \boldsymbol{K}_l$ дорівнює затримці між виконаннями операторів, вершини $\boldsymbol{K}_i$, $\boldsymbol{K}_l$ яких є суміжними. Виконується мінімізація числа ПЕ шляхом виконання вимог $|\boldsymbol{K}_{s,q}| \to L$, тобто число вершин, що відображаються в s-й ПЕ, прямує до L, де L – період виконання алгоритму, тактів. Крім того, при формуванні КА бажано будувати досконалий кістяк ГСПД, як це пропонується в [4].

На другому етапі виконується урівноважування КА, яке полягає в додаванні в дуги графа вершин затримки, поки часові складові усіх векторів $\boldsymbol{D}_j$ не дорівнюватимуть 0 або 1. Після цього виконується оптимізація КА шляхом взаємних перестановок векторів-вершин з одного ярусу з метою мінімізації числа регістрів та числа входів мультиплексорів в результуючій структурі і/або з застосуванням інших стратегій, наприклад, ресинхронізації [1]. Також мінімізується число регістрів склеюванням вершин затримки з одного яруса, які зберігають один і той самий операнд.

На третьому етапі одержана оптимізована КА відображається в граф структури обчислювача шляхом склеювання векторів-вершин з однаковими координатами s,q. КА перетворюється в розклад виконання операторів, використовуючи ту властивість, що часова складова вектору $\boldsymbol{K}_i$ дорівнює моменту виконання оператора безвідносно номера періоду виконання. При цьому можна не будувати структуру і розклад, якщо зразу описати схему обчислювача на мові VHDL [3].

Розглянемо деякий підграф КА, який виконується в РЗ. Цей підграф виконує пересилку операнда x з джерела $\boldsymbol{K}_{i1}$ до споживачів $\boldsymbol{K}_{j1}$, $\boldsymbol{K}_{l1}$ через дуги $\boldsymbol{D}_{j1} = \boldsymbol{K}_{j1} - \boldsymbol{K}_{i1}$, $\boldsymbol{D}_{l1} = \boldsymbol{K}_{l1} - \boldsymbol{K}_{i1}$, а також операнда y з джерела $\boldsymbol{K}_{i2}$ до споживачів $\boldsymbol{K}_{j2}$, $\boldsymbol{K}_{l2}$ через дуги $\boldsymbol{D}_{j2} = \boldsymbol{K}_{j2} - \boldsymbol{K}_{i2}$, $\boldsymbol{D}_{l2} = \boldsymbol{K}_{l2} - \boldsymbol{K}_{i2}$, відповідно (рис.1,а). Для того, щоб підграф виконувався на РЗ, принаймні всі його вхідні вершини повинні мати однакові просторові координати p.

При виконанні алгоритму в РЗ операнди x і y в кожному такті пересилаються на наступний регістр РЗ. Це еквівалентно тому, що в урівноваженій КА ці операнди в кожному такті передаються в наступний ярус і наступний ряд вершин затримки, який відображається в регістр РЗ. Іншими словами, ланцюжки суміжних вершин затримки $\boldsymbol{K}_{Di}$ при рівномірно зростаючих координатах $R(\boldsymbol{K}_{Di})$ розміщуються вздовж паралельних прямих, розташованих під одним кутом до осі ot. Пройшовши ланцюжки з $R(\boldsymbol{D}_{j1})$, $R(\boldsymbol{D}_{l1})$, $R(\boldsymbol{D}_{j2})$, $R(\boldsymbol{D}_{l2})$ вершин затримки, операнди x і y видаються на відповідні вихідні вершини підграфу (рис.1,б). Результуюча схема РЗ показана на рис.1,в.

РЗ має єдиний вихід, тому на підграф КА накладається додаткове обмеження – з вершин затримки, які належать одному ярусу і які відображаються в РЗ, повинно виходити не більше ніж одна дуга, що веде у вихідну вершину. При цій умові вихідний мультиплексор РЗ одночасно підключатиметься тільки до одного регістра РЗ. В іншому разі РЗ повинен мати більше, ніж один вихід або вихідний мультиплексор, як показано пунктиром на рис.1,б,в. Такий підграф повинен бути реалізований на кількох РЗ, як на рис.1,г.

РЗ типу SRL16 має додатковий вхід дозволу синхросерії, керування яким дає змогу загальмувати просування операндів по регістрах РЗ. При використанні цього входу можна зекономити кількість РЗ, якщо величина $R(\boldsymbol{D}_j)$ більше кількості регістрів в РЗ. На рис.2 показано приклад перетворення КА на рис.1,б з метою додаткової затримки на такт операндів, які поступають в вершини $\boldsymbol{K}_{l1}$, $\boldsymbol{K}_{l2}$. Така затримка відповідає векторам $\boldsymbol{D}_j$, які розміщуються паралельно осі ot.

Якщо вершини-джерела операндів мають різні просторові координати s, то на вході РЗ одержується вхідний мультиплексор. Мінімізацію входів таких мультиплексорів можна виконувати згідно з методикою, приведеною в [5].

Таким чином, методика проектування конвейерних обчислювачів з РЗ виглядає як наступна. Начальні дані – КА, період виконання алгоритму L та інші параметри оптимізації. Методика виконується таким самим чином, як це описано в [1], за виключеннями, які описані нижче.

На першому етапі синтезу слід виділити підграфи КА, що відповідають пересилці операндів між ресурсами обчислювача з затримками у часі і/або тасуванням операндів, які передбачається відобразити в окремі РЗ.

На другому етапі треба урівноважити дуги залежностей за допомогою проміжних вершин затримки. Виконати зменшення кількості проміжних вершин затримки для всіх дуг, якщо можливо.

Розмістити вершини затримки на паралельних прямих, що знаходяться під кутом до осі часу або паралельно цій осі таким чином, щоб суміжні вершини затримки відрізнялись по часовій координаті на один такт. Виконати вимоги коректного розміщення вершин, включаючи вимогу реалізації РЗ з одним входом і виходом. У разі неможливості одержати один вхід у РЗ використовують евристику мінімізації числа входів додаткового мультиплексора на вході РЗ згідно з [5], а при неможливості одержати РЗ з одним виходом ланцюжки вершин затримки розщеплюють, щоб вони відображались в додаткові РЗ (див. рис.1,г).

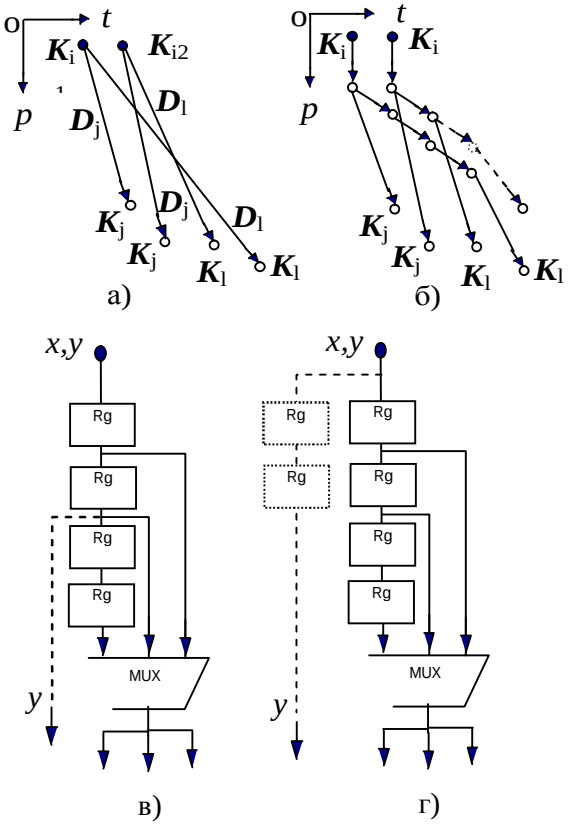


Рис.1. Відображення підграфу КА в РЗ

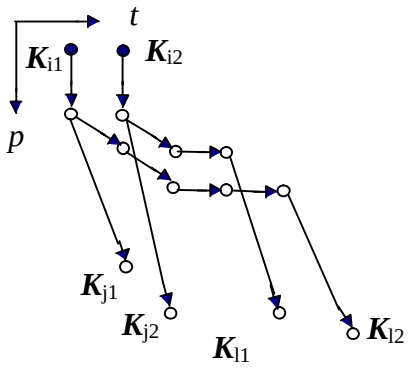


Рис.2. КА, що відповідає РЗ з входом дозволу

Дуги залежностей разом з відповідними вершинами затримки, що інцидентні вершинам-приймачам, відобразити в РЗ. При складанні алгоритму керування обчислювачем якщо в РЗ відображаються тільки дуги, які знаходяться під кутом до осі часу, то операнди в РЗ записуються в кожному такті, а якщо є дуги паралельні цій осі, то у відповідних їм тактах забороняється запис в РЗ.

На третьому етапі обчислювач, описаний на мові VHDL згідно з методикою, представленою в [1], компілюється в конфігурацію ПЛІС, яка вміщує РЗ типу SRL16, які відповідають виділеним підграфам КА.

Розглянемо приклад проектування буфера тасування даних згідно з правилом z-видного обходу, який використовується в кодерах зображення за стандартом H264 [6]. Якщо 16 вхідних даних приходять на вхід такого буфера в натуральному послідовному порядку, то вони виходять з нього в порядку згідно з послідовністю: 0,1,4,8,5,2,3,6,9,12,13,10,7,11,14,15. Як правило, такий буфер будують на основі двохпортової оперативної пам′яті, що призводить до нераціонального використання ресурсів, а також до великої латентної затримки між прийомом вхідних даних і видачею вихідних даних.

Урівноважена КА алгоритму роботи такого буфера, яка підготовлена згідно з одержаною методикою, показана на рис.3. Опис КА на мові VHDL приведено нижче.

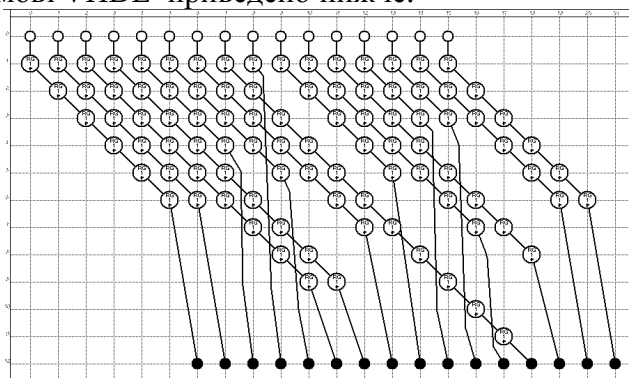


Рис.3. Урівноважена КА буфера тасування

Одержаний буфер при конфігуруванні в ПЛІС Virtex4 фірми Xilinx має мінімальні апаратні витрати – 4 тригери, 9 логічних таблиць і 12 елементів SRL16 – за кількістю розрядів даних. Цей буфер може бути використаний при рекордній тактовій частоті 900 МГц. Такі результати показують дієвість розробленої методики.

Було розроблено ряд конвейерних процесорів швидкого перетворення Фур′є з алгоритмом Винограда. Цей алгоритм характеризується складними перестановками даних і тому мінімальні апаратурні витрати процесорів досягнуті саме завдяки використанню запропонованої методики.

```
entity ZZ4x4 is
   port(CLK : in STD_LOGIC;     – синхросигнал
      START : in STD_LOGIC;     – запуск буфера
      DI : in STD_LOGIC_VECTOR(11 downto 0); – вхідні дані
      DO : out STD_LOGIC_VECTOR(11 downto 0)   ); – вихідні дані
end ZZ4x4;
architecture ZZ4x4 of ZZ4x4 is
   type TARR16 is array (0 to 15) of bit_vector(11 downto 0);
   type TA is array(0 to 15) of natural range 0 to 10;
   signal sr:TARR16;                                  – масив регістрів SRL16
   constant table :TA:=(5,5,3,0,4,8,8,6,4,2,2,6,10,7,5,5); – номери відводів SRL16
   signal fa, addr:natural range 0 to 15;
begin
   FSM:process(CLK,RST)  begin      – лічильник тактів  періоду алгоритму
      if CLK'event and CLK='1' then
         if START='1' then  addr<=0;  else addr<=(addr+1) mod 16; end if;
      end if;
   end process;
   fa<=table(addr);        – перекодування такту в номер відводу SRL16
   SRL16:process(CLK)  begin                  – опис SRL16
      if CLK'event and CLK='1' then
         sr<=DI & sr(0 to 14);                – власне зсув в РЗ
      end if;
   end process;
   DO<= sr(fa);              – видача результату з fa-го відводу  SRL16
end ZZ4x4;
```

Таким чином, запропонована методика проектування конвейерних обчислювачів, зконфігурованих в ПЛІС, яка забезпечує мінімізацію апаратурних витрат за рахунок широкого застосування регістрових затримок. Методика може бути використана для проектування спеціалізованих обчислювачів, в яких використовується буферна пам′ять типу FIFO.

Перелік посилань

1. Сергиенко А.М. VHDL для проектирования вычислительных устройств. – Киев: ДиаСофт. – 2003. – 203 с.
2. Сергієнко А.М. Синтез структур для виконання періодичних алгоритмів з операторами керування // Вістник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. – 2007. – №47. – с.221–227.
3. Каневский Ю.С., Овраменко С.Г., Сергиенко А.М. Отображение регулярных алгоритмов в структуры специализированных процессоров // Электрон. Моделирование.–2002.–Т.24.–№2.–С. 46-59.
4. Сергієнко А.М. Досконалий кістяк графа алгоритму. // Вістник НТУУ “КПІ”. Інформатика, управління та обчислювальна техніка. – 2007. – №46. – с.62–67.
5. Сергиенко А.М., Симоненко В.П. Отображение периодических алгоритмов в программируемые логические интегральные схемы //Электронное моделирование. –Т.29. –2007.–№2.–с.49–62.
6. Richardson I.E.G. H.264 and MPEG-4 Video Compression. Video Coding for Next-generation Multimedia. –Wiley. –2003. –281p.

УДК 519.854.2

ПАВЛОВ О.А.,
МІСЮРА О.Б.,
МЕЛЬНИКОВ О.В.,
ЩЕРБАТЕНКО О.В.,
МИХАЙЛОВ В.В.

ЗАГАЛЬНА СХЕМА ПЛАНУВАННЯ ТА УПРАВЛІННЯ СКЛАДНИМИ ОБ'ЄКТАМИ, ЩО МАЮТЬ МЕРЕЖНЕ ПРЕДСТАВЛЕННЯ ТЕХНОЛОГІЧНИХ ПРОЦЕСІВ Й ОБМЕЖЕНІ РЕСУРСИ

Представлена схема реалізації інформаційної технології планування та управління у системах з мережним представленням технологічних процесів та обмеженими ресурсами (МПТПОР), на основі якої створено комплекси послідовних взаємозв'язаних математичних моделей, сумісних з ієрархією рішень, що приймаються на кожному рівні планування, та системи нових високоефективних взаємозв'язаних алгоритмів розв'язання задач планування в сучасних умовах. Це вперше дозволило розв'язати задачу планування за різними критеріями оптимальності у комплексі.

The realization scheme of the informational technology for planning and management in systems with the network imagination of technological processes and the limited resources (NITPLR) is given. On the basis of the scheme the complexes of sequential interconnected mathematical models which are compatible with the hierarchy of decisions that made on every planning level, and the systems of new highly effective interconnected algorithms for the planning tasks solution in current conditions were created. This has let in the first time to solve the task of planning by different optimality criteria in complex.

Вступ

Постановкам задач планування та управління складними об'єктами та методам їх розв'язання в останні десятиліття приділяється істотна увага з боку багатьох дослідників. Найбільш добре вивчена область планування та управління складними об'єктами – виробниче планування та управління, підходи до якого та отримані результати також мають силу для інших областей – наприклад, в будівництві та в управлінні проектами. Найбільш широке поширення (до 80% усіх підприємств) одержали системи з мережним представленням технологічних процесів та обмеженими ресурсами (МПТПОР). У даній статті розглядається схема реалізації інформАційної технології планування та управління у таких системах.

Загальна постановка задачі планування

Нехай задана множина n комплексів взаємозв'язаних робіт $J = \{J_1, J_2, \ldots, J_n\}$ (комплекс робіт J_i, $i = \overline{1,n}$, надалі називається завданням). На кожній підмножині J_i заданий частковий порядок орієнтованим ациклічним графом. Часткова упорядкованість очевидним образом визначається технологією виконання комплексу робіт. Кожна наступна робота може початись тільки по завершенню попередніх робіт. Вершини графа відповідають роботам, зв'язки вказують на відносини передування. Кінцеві вершини відповідають завершенню виконання завдань. Для кожної вершини j графа відома l_j – *детермінована* тривалість виконання (інтегрований показник, що відображає виділені ресурси – матеріальні, людські, виробничі), для кожної роботи $j^i \in I$ (I – множина кінцевих вершин) задана вага $\omega_i = \omega_j{}^i$; для окремих завдань заданий директивний строк закінчення D_i. Величина ваги визначається потенційною складністю, важливістю і неоднозначністю (для робіт, зв'язаних з необхідністю одержання нового наукового розв'язку) виконання тих робіт, без яких у цілому завдання не може бути виконано. Для виконання робіт застосовується множина обмежених ресурсів. Сукупність ресурсів і виконавців розділена на окремі, досить автономні модулі – *мультиресурси* (мультиресурс – стійка група разом працюючих ресурсів – наприклад, бригада, група однотипного устаткування, однопрофільний підрозділ). Мультиресурси можуть знаходитися як в одній, так і в різних організаціях.

Необхідно побудувати погоджений план виконання комплексів робіт мультиресурсами та розподіл виконання робіт по ресурсах, з

урахуванням наступних критеріїв оптимальності і їх комбінацій:

а) мінімізація сумарного зваженого моменту закінчення виконання завдань (задача 1);

б) мінімізація сумарного часу виконання всіх завдань проекту (задача 2);

в) виконання завдань без порушення директивних строків D_i (планування «точно в строк», задача 3);

г) мінімізація сумарного зваженого моменту закінчення виконання завдань, якщо для деяких завдань $i \in \overline{1,n}$ не можуть бути порушені директивні строки D_i (задача 4);

д) мінімізація сумарного зваженого запізнення виконання завдань відносно директивних строків: $\omega_i \, max(0, C_i - D_i) \to min$, де C_i – момент закінчення виконання комплексу робіт J_i, $i = \overline{1,n}$ (задача 5);

е) мінімізація сумарного зваженого запізнення виконання завдань відносно директивних строків, якщо для деяких завдань $i \in \overline{1,n}$ директивні строки D_i не можуть бути порушені (задача 6); ж) мінімізація сумарного зваженого випередження та запізнення відносно директивних строків: $\omega_i \, |C_i - D_i| \to min$ (задача 7).

У задачах без директивних строків ω_i позначає вагу комплексу робіт, яка дорівнює вазі кінцевої вершини, а в задачах з директивними строками ω_i позначає штраф за відхилення від директивного строку на одиницю часу.

Назвемо *агрегованої роботою* сукупність робіт, виконуваних в одному мультиресурсі в рамках одного заходу в мультиресурс по одному завданню.

Задачі 1–7 розв'язуються при наступних обмеженнях:

- тривалість виконання кожного завдання визначається його критичним шляхом;
- спільні агреговані роботи різних завдань лежать на їх критичних шляхах і виконуються в одному мультиресурсі;
- агрегована робота не передається в інші мультиресурси до її повного завершення.

Задачі 1–7 належать до класу *NP-складних* задач у сильному розумінні.

Трирівнева модель планування та управління складними об'єктами

В основу планування та управління складними об'єктами покладено математичне забезпечення трирівневої моделі планування та управління дрібносерійним виробництвом [1], адаптоване для планування інших класів об'єктів. У відповідності до цієї моделі, побудова розподілу робіт по ресурсах здійснюється в три етапи. Перший етап складається в побудові агрегованої моделі. Якщо які-небудь роботи виконуються в одному мультиресурсі в рамках одного заходу в мультиресурс по одному завданню, то вони агрегуються в одну агреговану роботу. Тривалість виконання агрегованої роботи визначається її критичним шляхом. Для кожного комплексу робіт визначається критичний шлях виконання агрегованих робіт. На основі агрегованої інформації будується граф на критичних шляхах завдань. Вершини отриманого графа – це агреговані роботи, дуги відображають зв'язки між мультиресурсами, що регламентують технологію виконання завдань. Деякі роботи, що належать різним завданням, вимагають виконання в спеціалізованих унікальних мультиресурсах, бажано в рамках одного заходу в мультиресурс. У цьому випадку при Виконанні деяких умов, описаних нижче, формується об'єднана агрегована робота, що на графі зв'язності відображено спільними вершинами. Для визначення пріоритетів завдань при побудові погодженого плану виконання завдань відповідно до критеріїв оптимальності, важливим є розв'язання на першому рівні задачі «Мінімізація сумарного зваженого моменту закінчення виконання завдань одним приладом при відношенні порядку, заданому орієнтованим ациклічним графом» (МЗМ) для випадку, коли вагові коефіцієнти усіх вершин графу зв'язності, крім кінцевих, дорівнюють нулю [2, 3]. У результаті розв'язання цієї задачі формується послідовність виконання агрегованих робіт, у якій підпослідовності агрегованих робіт упорядковані за пріоритетами, що визначають черговість запуску агрегованих робіт на виконання.

Другий етап полягає в побудові погодженого плану виконання завдань з урахуванням зазначених вище критеріїв оптимальності.

Отримані на першому рівні пріоритети агрегованих робіт служать додатковою інформацією, що дозволяє значно підвищити ефективність отриманих розв'язків.

Процедури третього рівня дозволяють у відповідності з побудованим планом для агрегованих робіт побудувати розподіл робіт по

ресурсах (так зване точне планування). На цьому рівні розв'язуються задачі оптимізації за критеріями:

а) Мінімізація сумарного зваженого моменту закінчення виконання завдань (Задача 1);

б) Мінімізація сумарного часу виконання всіх завдань (Задача 2);

в) Мінімізація сумарного зваженого запізнення виконання множини завдань із різними директивними строками (Задача 3);

г) Мінімізація сумарного запізнення виконання множини завдань із загальним директивним строком (Задача 4);

д) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови, якщо для деяких завдань не можуть бути порушені директивні строки (Задача 5);

е) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови, коли для частини завдань заборонені випередження й запізнення відносно директивних строків (Задача 6);

ж) Мінімізація сумарного зваженого запізнення виконання множини завдань за умови мінімізації числа переналагоджень в мультиресурсах (Задача 7);

з) Мінімізація сумарного запізнення виконання множини завдань відносно директивних строків (Задача 8);

и) Мінімізація сумарного зваженого випередження та запізнення відносно директивних строків (Задача 9).

Основні принципи реалізації розв'язання задач планування

Алгоритмічне забезпечення погодженого планування складається з трьох основних блоків, що відповідають трьом рівням загальної схеми планування. Нижче наведені базові принципи алгоритмів розв'язання задач у системі планування. Рівень І. Побудова моделі технологічної та конструкторської агрегації (рис. 1).

При побудові моделі технологічної агрегації виконується агрегація до рівня мультиресурсів і побудова агрегованих робіт. Використовуються наступні принципи агрегування інформації.

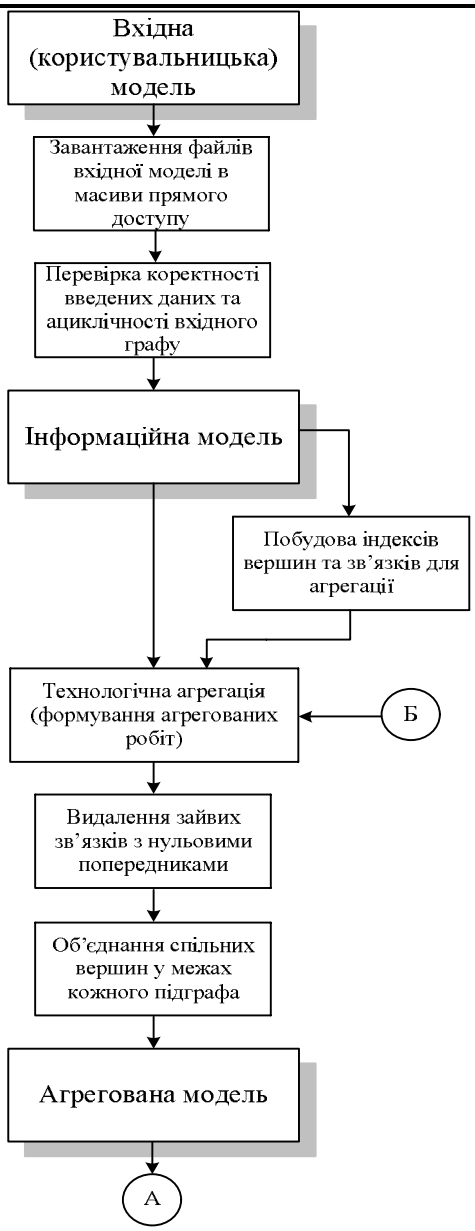


Рис. 1. Рівень І. Побудова агрегованої моделі

Вхідна модель інформації, як правило, представляє собою бази даних з інформацією про операції, взаємозв'язки між ними, групи мультиресурсів, інформацію про завдання та календар роботи (якщо інформація представлена інакше, її треба перетворити до необхідного виду). Вхідний ациклічний орієнтований граф операцій представляється у вигляді ланцюжків технологічних операцій (переліку робіт) і переліку взаємозв'язків проекту (рис. 2). Перетворення вхідного графа в граф агрегованих робіт здійснюється за допомогою процедури первинної агрегації, або агрегації агрегованих робіт. Така агрегація є агрегацією технологічного типу і робить об'єднання суміжних операцій (по технологічних ланцюжках), виконуваних в одному і тому же мультиресурсі по одному завданню, в один елемент, названий агрегованою роботою (рис. 4).

Тривалість виконання агрегованої роботи визначається критичним шляхом у графі виконання робіт, що входять до неї; час переналагодження для агрегованої роботи дорівнює максимальному з часів переналагодження її перших робіт (листів графа). При цьому тривалість роботи *в одному завданні* визначається технологією виконання роботи. Для агрегованої роботи тривалість дорівнює сумі тривалостей робіт усіх завдань у даному мультиресурсі по критичному шляху на одиницю ресурса, тобто

$$L_i = \sum_{j \in CP_i} \frac{Lроб_{ij} \cdot Nроб_i \cdot N_i}{KR_i},$$

де L_i – тривалість i-ї агрегованої роботи; $Nроб_i$ – кількість робіт у завданні (застосовність); $Lроб_{ij}$ – тривалість j-й роботи i-ї агрегованої роботи; N_i – кількість завдань, до яких входить i-а агрегована робота; KR_i – кількість ресурсів у всіх мультиресурсах групи (з розрахунком на паралельне виконання в однотипних мультиресурсах); CP_i – критичний шлях робіт агрегованої роботи; $Lon_i = \max(\sum_{j \in CP_i} Lon_{ij}, 1)$ назвемо тривалістю *агрегованої операції*; $Non_i = \frac{Nроб_i \cdot N_i}{KR_i}$ – кількість агрегованих операцій в агрегованій роботі.

З вищенаведеної формули видно, що фактично тривалість агрегованої роботи виявляється кратною тривалості однієї агрегованої операції, що визначається як сума тривалостей робіт по критичному шляху агрегованої роботи. Агрегована операція – найменша частина агрегованої роботи, що не допускає дроблення на більш дрібні частини.

На рис. 2 штриховою лінією позначені роботи, що об'єднаються. У колі – код (назва) роботи, поруч з ним – код групи мультиресурсів (в дужках – номер заходу). Стрілки позначають входження, числа над стрілками – кількість робіт, що потребують виконання для передачі до попередника (застосовність). На рис. 4 числа зліва від кружків позначають номер агрегованої роботи після перенумерації.

Після одержання первинних агрегованих робіт методом технологічного об'єднання може здійснюватись конструкторське об'єднання агрегованих робіт, що мають однакові коди вершин і мультиресурсів в одному завданні, час запуску яких по критичному шляху завдання відрізняється не більше, ніж на величину максимального з часів їхнього переналагодження. При цьому агреговані роботи стають однією агрегованою роботою із сумарною тривалістю і часом переналагодження, рівному максимальному з цих часів переналагодження (рис. 5).

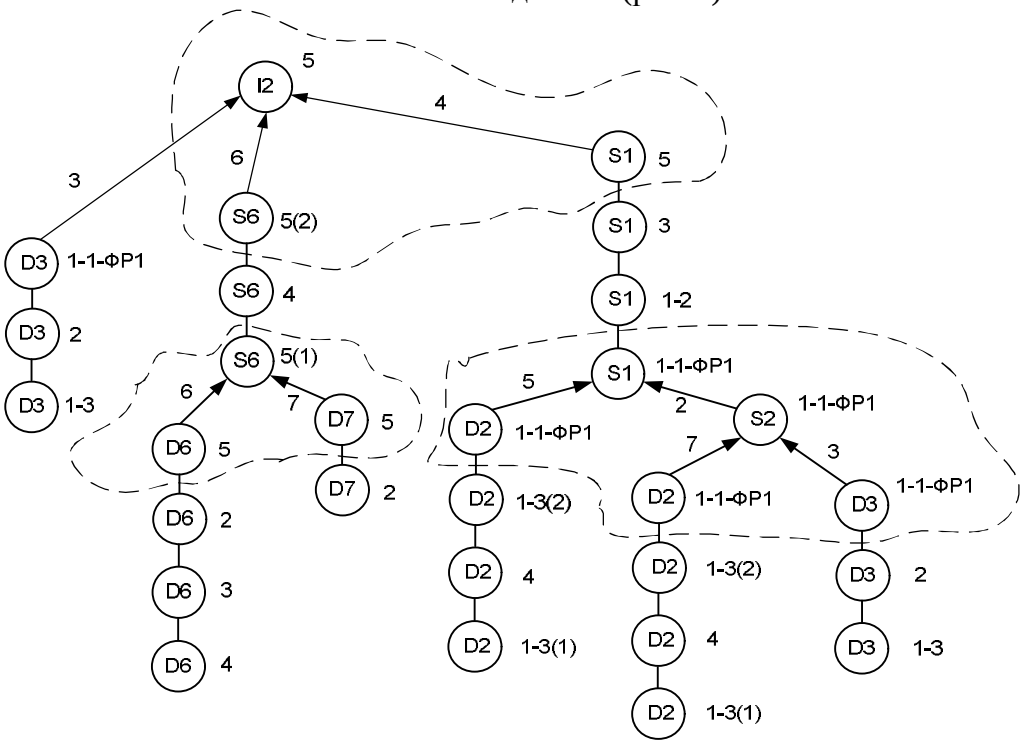


Рис. 2. Приклад вхідного графу виконання робіт та об'єднання в агреговані роботи

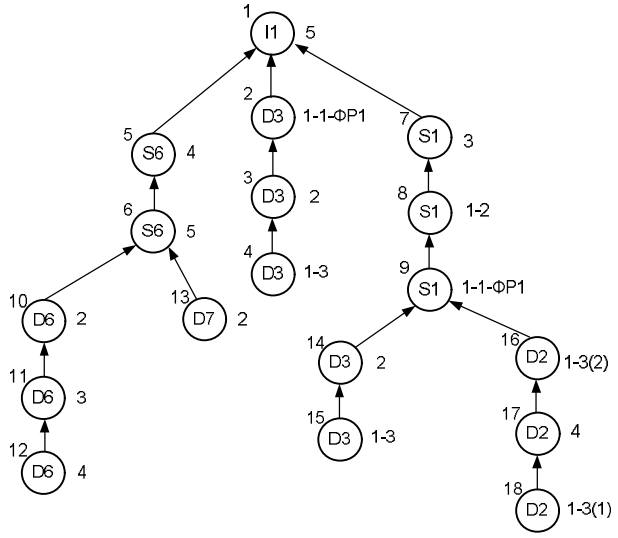


Рис. 4. Граф агрегованих робіт після технологічної агрегації

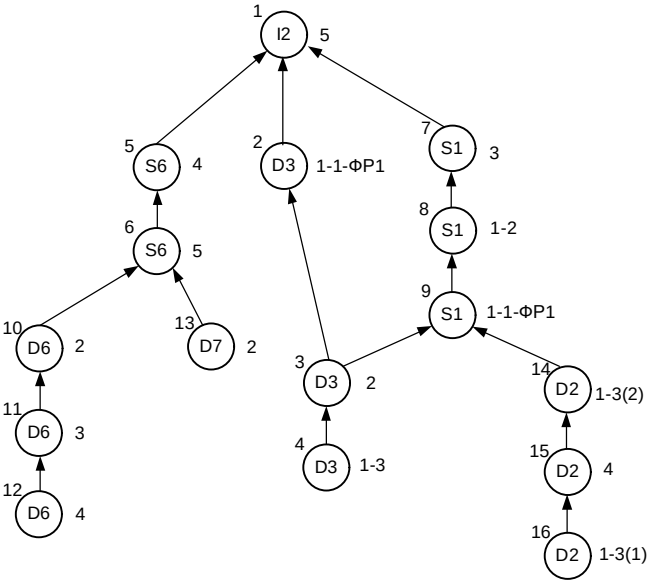


Рис. 5. Граф агрегованих робіт після конструкторської агрегації

Конструкторська агрегація здійснюється рідко, тому що вона звичайно приводить до великих довжин агрегованих робіт, що погано впливає на результат розподілу. Але об'єднання робіт у загальні вершини, що розглядається далі, також є методом конструкторської агрегації.

Рівень 2. Побудова плану виконання агрегованих робіт мультиресурсами (рис. 3, 7)

Спочатку здійснюється побудова спеціальних індексних масивів для швидкого доступу до інформації, необхідної для виконання наступних блоків. Побудована інформаційна модель називається алгоритмічною. При індексації визначаються значення окремих реквізитів моделі, що відповідають критерію, обраному користувачем.

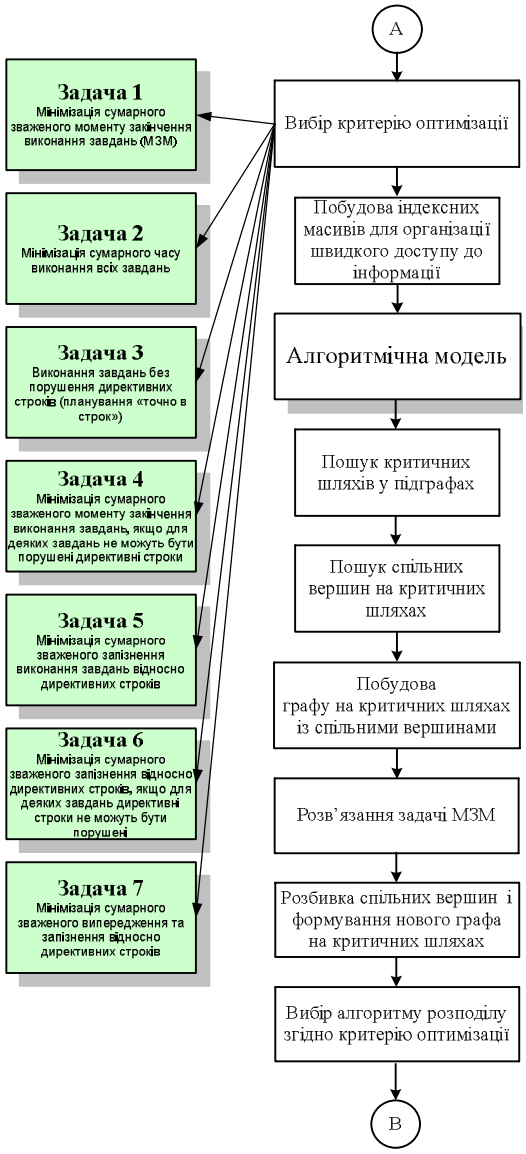


Рис. 3. Рівень II. Побудова плану виконання агрегованих робіт мультиресурсами (ч.1)

На сформованій у результаті перенумерації алгоритмічній моделі здійснюється пошук критичних шляхів завдань. Критичний шлях у графі, де кожна вершина навантажена тривалістю, – це шлях з максимальною тривалістю. Критичні шляхи шукаються за допомогою спрямованого повного перебору вершин графа агрегованих робіт. Для того, щоб був можливий пошук критичного шляху, перед його виконанням перевіряється ациклічність побудованого графа.

Після того, як знайдені критичні шляхи завдань, будується граф на критичних шляхах. Таким чином реалізується агрегування моделі до рівня «одного ресурсу». На графі на критичних шляхах розв'язується задача побудови

оптимального розкладу виконання агрегованих робіт, вважаючи, що це задача теорії розкладів для одного приладу.

Граф на критичних шляхах будується шляхом пошуку та об'єднання спільних вершин, що лежать на критичних шляхах. Ознакою спільної для двох критичних шляхів вершини є збіг кодів вершин і кодів мультиресурсів і факт відмінності тривалостей шляху від вершини до кінцевої вершини не більше, ніж на час налагодження в мультиресурсі. Побудований за такими правилами граф (рис. 6) є графом меншої розмірності, тому що він включає тільки вершини на критичних шляхах. На отриманому графі потім здійснюється побудова оптимальної послідовності.

Особливістю процедур нумерації є те, що при об'єднанні спільних вершин можуть з'явитися однакові ребра. Тому при нумерації передбачається процедура видалення однакових зв'язків. Крім того, потрібно виключити ситуації, коли поєднуються вершини, з яких не всі листові. Тоді потрібно видалити виникаючі при цьому «зайві» нульові зв'язки.

Якщо виконується планування виробничих проектів, то після побудови графа на критичних шляхах можуть бути розраховані розміри партій для кожної агрегованої роботи. Агреговані роботи розбиваються на однакове число партій (при цьому процедура розподілу найбільш проста і швидка, тому що не вимагає перегляду попередників по кожній призначуваній партії і кожному спадкоємцю).

Мінімальне число партій визначається максимальною довжиною партії, рівною довжині зміни, так як не повинна бути порушена умова безперервності виконання партії. Тому мінімальна кількість партій дорівнює цілій частині максимальної з тривалостей агрегованих робіт, поділеній на довжину зміни, плюс одиниця. Нарешті, кількість агрегованих робіт у партії визначається для агрегованої роботи як тривалість агрегованої операції, помножена на кількість мультиресурсів у групі і розділена на кількість партій.

По отриманих масивах графа на критичних шляхах будується оптимальна послідовність агрегованих робіт за допомогою розв'язання задачі МЗМ. Алгоритм розв'язання є оригінальною розробкою лабораторії комбінаторної оптимізації НТУУ «КПІ» під керівництвом д.т.н. проф. О.А.Павлова, що враховує останні досягнення в області проектування ПДС-алгоритмів для важкорозв'язуваних комбінаторних задач [2]. У результаті розв'язання задачі МЗМ отримуємо послідовність виконання агрегованих робіт, розбиту на підпослідовності максимального пріоритету (ПМП).

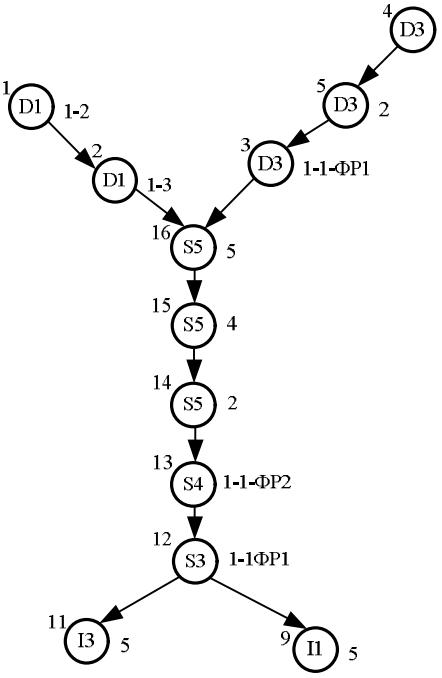


Рис. 6. Граф на критичних шляхах двох завдань (I1 та I3)

Після цього будується новий граф зв'язності, у якому кожна з спільних вершин перетворюється в ланцюжок вершин критичних шляхів завдань зі збереженням відношення передування графа на критичних шляхах. При цьому попередники кожної об'єднаної в графі на критичних шляхах спільної вершини стають попередниками першої (починаючи з листів графа) вершини в ланцюжку, а спадкоємці кожної об'єднаної спільної вершини стають спадкоємцями останньої вершини в ланцюжку.

Наприклад, якщо вершині *i* графа на критичних шляхах відповідає вершина P_i загального графа, а спільні вершини мають номера 3, 6, і їм відповідають по дві вершини загального графа (позначимо їх умовно P_{31}, P_{32}, P_{61}, P_{62}; вони здобувають свої власні номери в даній процедурі), то здійснюється перетворення графа на критичних шляхах, показане на Рис. 8.

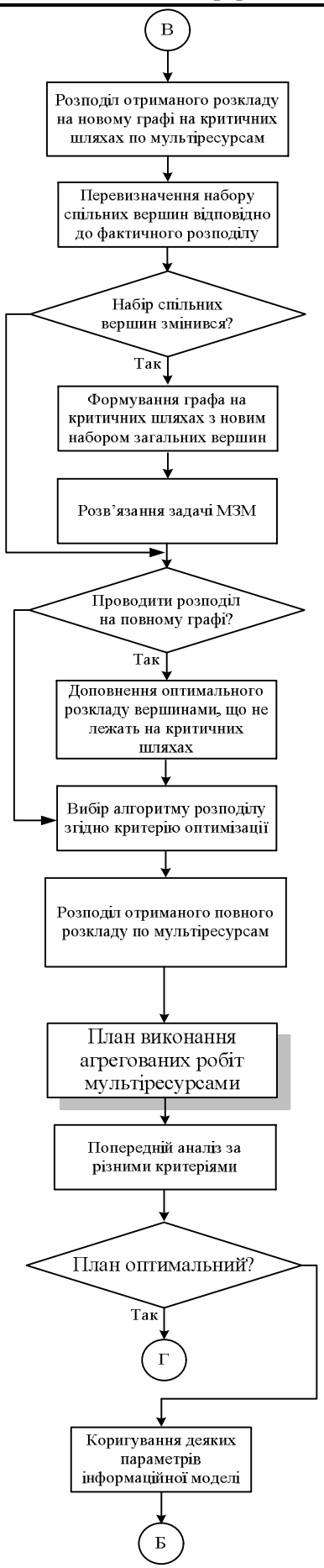


Рис. 7. Рівень II. Побудова плану виконання агрегованих робіт мультиресурсами (ч.2)

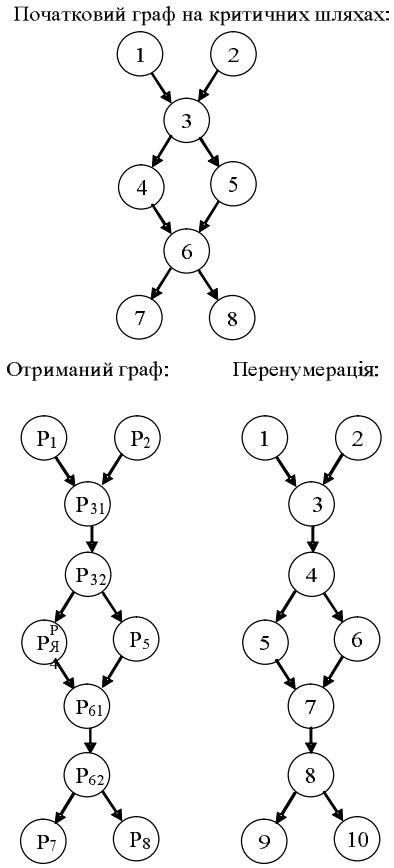


Рис. 8. Приклад перетворення графа в блоці 6

Далі виконується розподіл агрегованих робіт побудованого графу критичних шляхів з розбивкою спільних вершин для уточнення інформації про об'єднання спільних вершин на стадії розподілу. У користувача в діалозі запитуються дата початку планового періоду і дата закінчення планового періоду. За календарем робочих днів визначаються номери робочих днів у плановому періоді. По кожному мультиресурсу розраховується фонд часу мультиресурсу в днях, що дорівнює добутку кількості робочих днів у плановому періоді і коефіцієнта змінності.

Процедура розподілу працює для будь-якого вхідного графа (загальний граф зв'язності агрегованих робіт, граф на критичних шляхах чи граф на критичних шляхах з розбивкою спільних вершин). Для правильної роботи відповідно до обраної підзадачі відкриваються відповідні заданому графу масиви зв'язків і вершин. Це можливо завдяки уніфікації їхніх структур. На даному етапі алгоритму виконується розподіл по графу з розбивкою спільних вершин. При виробничому плануванні вводяться наступні припущення:

- Агреговані роботи не розбиваються на партії: при розбивці на партії неможливо ви-

значити фактичний набір спільних вершин, тому під «партією» в алгоритмі треба розуміти «агрегована робота»;

- У зв'язку з тим, що тривалість нерозбитих на партії агрегованих робіт може перевищувати тривалість робочого дня в деяких мультиресурсах, вважається, що всі мультиресурси працюють цілодобово.

Алгоритми розподілу побудовані з використанням алгоритмів Гіфлера-Томпсона [3]. Під завданнями розуміються або агреговані роботи, або партії робіт, залежно від обраної задачі.

Позначення:

P_t – частковий календарний план з t–1 призначеного завдання;

S_t^k – множина завдань, готових до призначення на k-ий мультиресурс на стадії t;

Sj_k – можливий найбільш ранній момент початку виконання завдання з S_t^k (Sj_k дорівнює моменту завершення попереднього завдання або нулю для завдання без попередників за графом зв'язності);

R_{ik} – i-й інтервал масиву R резервного часу мультиресурсу k, не зайнятий виконанням завдань, призначених на попередніх стадіях;

$R_{ik}^{\text{н}}$ – початок і-го інтервалу;

$R_{ik}^{\text{к}}$ – кінець і-го інтервалу;

$\sigma_{j_k} = \max(R_{ik}^{\text{н}}, Sj_k)$ – момент початку виконання завдання j_k, призначеного на виконання на інтервалі R_i мультиресурсу k;

Cj_k – момент завершення виконання завдання j_k, призначеного на виконання на інтервалі R_i мультиресурсу k: $Cj_k = \sigma_{j_k} + l_{j_k}$.

$\sigma_{j_k}^0$ – мінімально можливий момент початку виконання завдання $j_k \in S_t^k : \sigma_{j_k}^0 = \min\limits_k \min\limits_{j \in S_t^k} \sigma j_k \; \forall k, i: \sigma_{j_k}^0 + l_{j_k} \le R_{ik}^{\text{к}}$.

$C_{j_k}^0$ – мінімально можливий момент завершення виконання завдання $j_k \in S_t^k : C_{j_k}^0 = \min\limits_k \min\limits_{j \in S_t^k} Cj_k \; \forall k, i: R_{ik}^{\text{к}} - l_{j_k} \le R_{ik}^{\text{н}}$.

Розподіл агрегованих робіт заданого графа (оптимальної послідовності) по мультиресурсах здійснюється з виконанням наступних принципів. Виконується стільки ітерацій, скільки партій агрегованих робіт розраховано (без розбивки на партії – 1 ітерація). На кожному кроці призначається на мультиресурс партія агрегованих робіт з мінімальним часом звільнення мультиресурсу і мінімальним номером ПМП. Мінімальний час звільнення мультиресурсі партією агрегованих робіт визначається раннім часом початку виконання партії з урахуванням моменту початку виконання завдання плюс тривалість її виконання (з необхідним налагодженням устаткування). Ранній час початку виконання партії визначається максимальним з часів закінчення всіх його попередників по графу, визначених при призначенні попередника на виконання. При призначенні партії шукається перший вільний проміжок часу в рамках фонду часу мультиресурсу, початок якого не раніш, ніж ранній час початку призначуваної партії, а тривалість не менш тривалості виконання партії з переналагодженням. Резерв шукається в «реальному» часі мультиресурсу. Це означає, що враховується не тільки робочий час мультиресурсу, у якому власне йде виконання, але і часи початку і кінця робочих днів, доробків на початок планового періоду, і час початку роботи всіх мультиресурсів (наприклад, 8-ма година ранку).

Алгоритм 1 виконує mn стадій та будує *компактні* [3] розклади.

1. $t = 1$, $P_1 = \varnothing$, $\sigma_i = \sigma_1$, де σ_i – i-я ПМП, яка включає всіх попередників по графу G. Формуємо множини S_1^k завдань, у яких немає попередніх та потребуючих виконання в k-ому мультиресурсі, $k = \overline{1,m}$.

2. Вибираємо з множини S_1^k для призначення на виконання завдання j_k з мінімальним часом закінчення виконання.

3. Переходимо до наступної стадії. При цьому:

а) поміщаємо завдання j_k в P_t і, таким чином, створюємо P_{t+1};

б) формуємо S_{t+1}^k, $k = \overline{1,m}$, додавши до одного з S_t^k завдання, що безпосередньо слідує за завданням j_k, якщо тільки j_k не останнє в σ_1, а інші S_t^k залишивши без змін;

в) формуємо масив резервів мультиресурсу k;

г) покладемо $t = t + 1$.

4. Якщо залишилися непризначені завдання з ПМП №1, то переходимо к п.2. У противному випадку переходимо до наступної ПМП и к п.2. Якщо призначені на виконання всі завдання – кінець алгоритму.

Наступна модифікація алгоритму 1 дозволяє одержати незатримуючі розклади. ***Алгоритм 2*** заснований на наступній евристиці: у п. 2 алгоритму призначається на мультиресурс завдання з мінімальним номером ПМП та мінімальним часом початку виконання, який визначається як $\sigma_{j_k} = \max(R^{н}_{ik} , Sj_k)$; $\sigma^0_{j_k} = \min\limits_k \min\limits_{j \in S^k_i} \sigma j_k \; \forall k, i: \sigma^0_{j_k} + l_{j_k} \le R^{к}_{ik}$.

Алгоритм 3 може застосовуватись для розподілу завдань при оптимізації за критерієм мінімізації запізнення, тобто без порушення директивних строків завдань. При цьому послідовність розглядається з початку, із завдань більш високого пріоритету. На першому кроці розглядається завдання зі списку кінцевих вершин, що входять до складу ПМП №1 і призначається таким чином, щоб момент закінчення його виконання відповідав директивному строку завдання. На кожному наступному кроці призначається на мультиресурс завдання з максимальним часом запуску в мультиресурсі та мінімальному номері ПМП, у яку воно входить. Максимальний час запуску завдання в мультиресурсі визначається найбільш пізнім часом закінчення виконання завдання мінус тривалість його виконання (з необхідним налагодженням). Найбільш пізній час закінчення виконання завдання визначається мінімальним із часів запуску всіх його спадкоємців по графу, визначених при призначенні попередників на виконання. При призначенні завдання шукається перший вільний проміжок часу в рамках фонду часу мультиресурсу, кінець якого не пізніше, ніж найбільш пізній час закінчення призначуваного завдання, а тривалість не менш тривалості виконання завдання з переналагодженням. Резерв шукається в «реальному» часі мультиресурсу. Якщо необхідний для призначення завдання резерв не знайдений у всіх однотипних мультиресурсах, то це означає, що виконання даного завдання не укладається в його директивний строк, про що користувачу видається повідомлення. Користувач при цьому повинен виключити дане завдання з числа комплексів робіт та перерозподілити план виконання робіт.

Після першого розподілу на графі критичних шляхів, проводиться перевизначення набору спільних вершин відповідно до фактичної інформації про розподіл. Уточнення набору спільних вершин графа на критичних шляхах здійснюється тому, що фактична інформація про розподіл, отриманий на попередньому кроці, визначає більш точну модель графа, чим отримана в порівнянні тривалостей виконання з початку критичному шляху.

Це здійснюється наступним алгоритмом: для кожній спільній вершини перевіряються ознаки об'єднання в графі на критичних шляхах та об'єднання при розподілі, і якщо вони не збігаються, то ознака об'єднання в графі на критичних шляхах встановлюється у значення об'єднання при розподілі, ознака необхідності перерозподілу в «Істина».

Якщо набір спільних вершин не змінився, то модель не має потреби в зміні і, отже, потрібно тільки перерозподілити агреговані роботи з розбивкою на партії (при виробничому плануванні) та доповненням до повного графу (див. наступний крок).

Інакше модель перебудовується і шукається нова оптимальна послідовність виконання агрегованих робіт. Для цього повторно виконуються процедури, починаючи з формування графа на критичних шляхах завдань. Для побудови графа використовується набір спільних вершин, отриманий на попередньому кроці.

У результаті блоку оптимізації отримана припустима оптимальна послідовність, що розбита на підпослідовності спадаючого пріоритету. Така послідовність дає мінімальне значення критерію. Однак вона включає тільки агреговані роботи, що лежать на критичних шляхах завдань, а для розподілу плану виконання робіт необхідно врахувати всі агреговані роботи, необхідні для виконання усіх комплексів робіт. Тому для подальшої роботи алгоритму необхідно доповнити отриману оптимальну послідовність агрегованими роботами завдань, що не лежать на критичних шляхах, з присвоєнням їм відповідного номеру ПМП. Весь надграф кожної з агрегованих робіт, що входить до оптимальній послідовності, буде мати той же номер ПМП, що й ця агрегована робота.

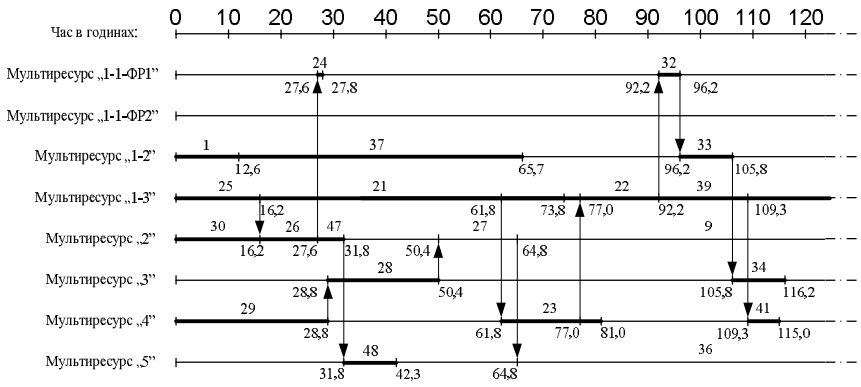


Рис. 9. Результат розподілу плану виконання робіт

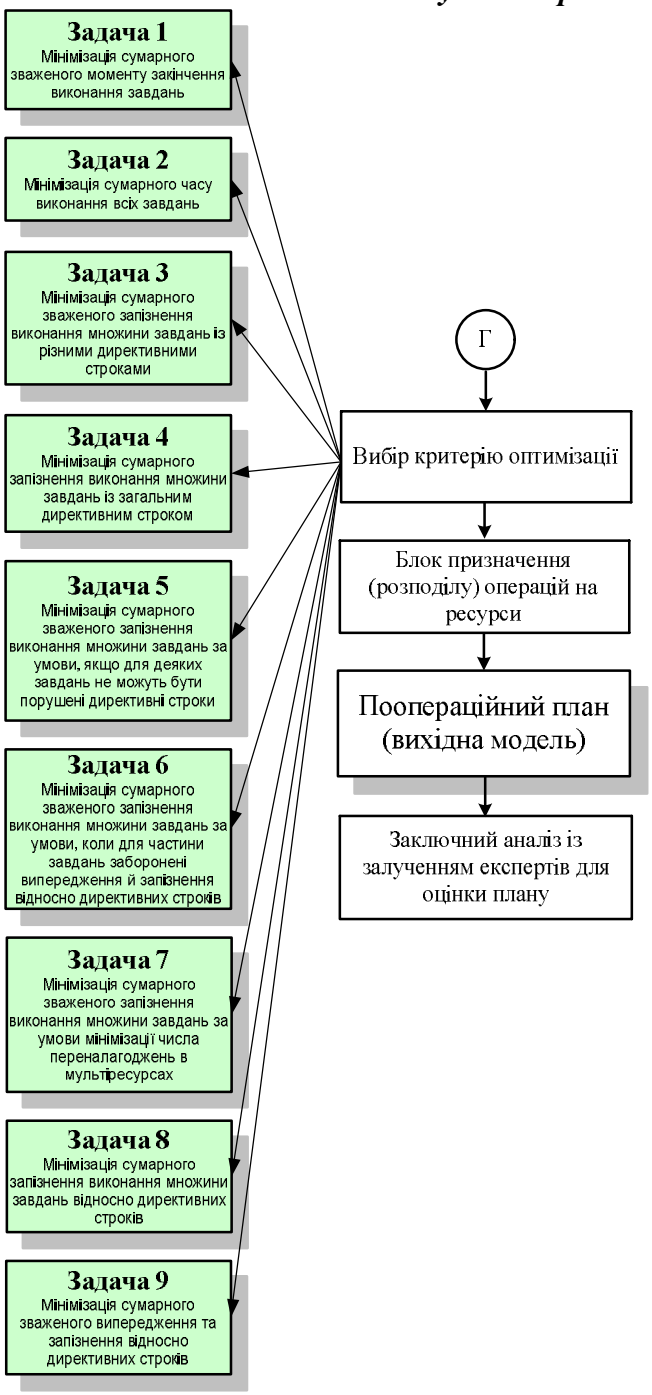


Рис. 10. Рівень ІІІ. Розподіл робіт по ресурсах

Розподіл отриманого розкладу по мультиресурсах з прив’язкою до планового періоду виконується за одним з алгоритмів розподілу, описаних вище, за такими виключеннями: алгоритм виконується по загальному графу зв’язності агрегованих робіт чи графу на критичних шляхах, у залежності обраної підзадачі; у виробничому плануванні виконується розбивка агрегованих робіт на партії (кількість ітерацій дорівнює кількості партій та виконується для партій замість однієї ітерації для агрегованих робіт повної тривалості); мультиресурси мають відповідну позмінну роботу, а не цілодобову; не Виконується перероб’єднання спільних вершин.

Рівень ІІІ. Побудова плану виконання агрегованих операцій мультиресурсами (рис. 10)

Як було сказано раніше, на третьому рівні трирівневої моделі планування здійснюється формування повного плану виконання всіх робіт з розбивкою по ресурсах всіх мультиресурсів і з урахуванням розподілу агрегованих робіт, отриманого на другому рівні. На цьому рівні можуть розв’язуватись оптимізаційні задачі.

Процедури розподілу агрегованих операцій по ресурсах аналогічні алгоритмам розподілу агрегованих робіт по мультиресурсах, описаним вище. Вибір алгоритму розподілу здійснюється залежно від необхідного критерію оптимізації розкладу: для одержання компактних розкладів вибирається Алгоритм 1-АО, для незатримуючих – Алгоритм 2-АО, розкладів без запізнення (для критеріїв, у яких завдання мають директивні строки, що не можна порушувати) – Алгоритм 3-АО. В алгоритмах третього рівня не здійснюється аналіз об’єднання спільних вершин.

Розглянемо на прикладі алгоритм розподілу для одержання компактних розкладів агрегованих операцій (***Алгоритм 1-АО***). Розподіл агрегованих операцій по ресурсах здійснюється за наступними принципами.

Розглядається кожна група однотипних мультиресурсів та здійснюється перебір усіх агрегованих робіт (партій), назначених на виконання у даній групі мультиресурсів. На кожному кроці призначаються на вільне робоче місце агреговані операції з мінімальним часом закінчення, який визначається як час початку виконання партії плюс тривалість виконання агрегованої операції. За побудовою плану виконання агрегованих робіт, усі ресурси будуть мати достатні резерви для виконання операцій, що лежать на критичному шляху операцій агрегованої роботи. За побудовою, кожна агрегована робота містіть ціле число агрегованих операцій рівної тривалості.

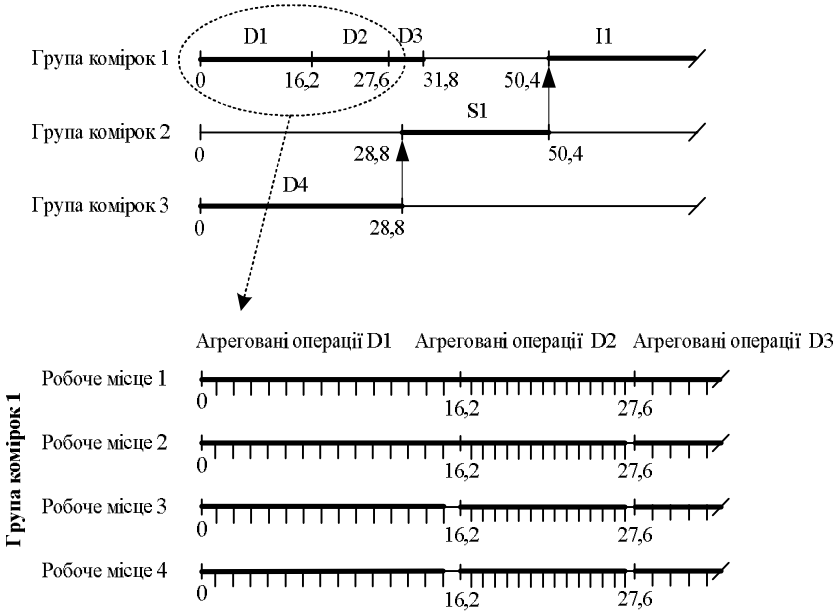


Рис. 11. Приклад розподілу агрегованих операцій по робочих місцях

Схематично процес розподілу агрегованих операцій показаний на рис. 11. Ці резерви можуть бути використані для виконання інших паралельних операцій без переналагодження ресурсу. Як можна бачити, у результаті розподілу, через округлення тривалості агрегованої операції при визначенні тривалості агрегованих робіт, у ресурсах можуть виникати резерви.

Висновки

Загальна схема розв'язання задач в багаторівневій системі планування дрібносерійного виробництва в умовах ринку [1] була адаптована для розв'язання задач планування та управління складними об'єктами з МПТПОР. До таких об'єктів можна віднести широкий клас систем як виробничого, так і невиробничого характеру. Зокрема, на основі створеної моделі авторами вже розроблені комплекси послідовних взаємозв'язаних математичних моделей ієрархічного планування та управління робочим цехом, планування виробництв «на замовлення», планування виробництв дрібносерійного типу; ієрархічного планування виробництв по виготовленню партій, планування та управління проектами. Планується реалізація моделі для будівничих проектів. У статті наведена розширена концептуальна схема інформаційної технології розв'язання задач планування у будь-якому з розглядаємих класів об'єктів управління.

Перелік посилань

1. Павлов О.А., Місюра О.Б., Мельников О.В. Загальна схема розв'язання задач в багаторівневій системі планування дрібносерійного виробництва в умовах ринку / Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка. К.: ВЕК+, 2000.– №33.– С.27-33
2. Основы системного анализа и проектирования АСУ: Учеб. пособие /А.А.Павлов, С.Н.Гриша и др.; Под общ. Ред. А.А.Павлова.– К. : Выща шк.; 1991.– 367 с.
3. Конвей Р.В., Максвелл У.Л. Теория расписаний.– М.: Наука, 1975.– 359 с.

УДК 683.519

ТКАЧЕНКО С. В.

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ МППО НА ПРИМЕРЕ РАЗРАБОТКИ ТОРГОВОЙ СТРАТЕГИИ ДЛЯ РЫНКА FOREX

В настоящей статье рассмотрено практическое применение метода прогнозирования с показателем определенности (МППО) на примере задачи разработки формальной модели торговой стратегии для рынка FOREX. Демонстрируется использование показателя определенности, описана формальная процедура определения шкалы значимости этой качественной оценки прогноза. На примере прогнозирования тренда курса пары валют приведены количественные расчеты эффективности использования показателя определенности.

The subject of the article is an application of the method of prediction with certainty coefficient to a task of developement of FOREX market mechanical trading system. An example of usage of certainty coefficient is demonstrated and formal procedure for definition of its relevance scale is provided. Quantitative results of effectiveness of certainty coefficient usage are provided with regard to prediction of the currency pair trend.

1. Введение

В настоящей статье рассмотрено практическое применение метода прогнозирования с показателем определенности на примере задачи разработки формальной модели торговой стратегии для рынка FOREX. Демонстрируется использование показателя определенности, описана формальная процедура определения шкалы значимости этой качественной оценки прогноза. На примере прогнозирования тренда курса пары валют приведены количественные расчеты эффективности использования показателя определенности.

2. Метод прогнозирования с показателем определенности

Индуктивные методы прогнозирования (такие как нечеткий метод группового учета аргументов (НМГУА), нейронные сети (НС)) не приспособлены для обучения на больших выборках. Как правило для построения модели НМГУА берутся последние N точек имеющихся фактических данных, где N – экспериментально установленный размер окна, дающий в результате аппроксимации наиболее адекватную для прогнозирования модель. В случае превышения N, структура получаемых моделей резко усложняется и ошибка прогноза также быстро возрастает. Таким образом упомянутые методы не способны использовать для обучения всю предысторию временного ряда, и большая часть накопленных знаний не используется. Та же проблема существует и с нейронными сетями, в которых с определенного момента предыдущие знания начинают вытесняться новыми поступающими фактами (эффект забывания).

МППО предлагает способ снять перечисленные ограничения и использовать предысторию для улучшения прогнозирующих характеристик базового метода (НМГУА, НС и т.д.), а также дополнительно предоставляет качественную оценку рассчитанному прогнозу (показатель определенности). В основу метода положено 3 принципа:

1. скользящего окна
2. образной памяти
3. суперпозиции

Принцип скользящего окна позволяет базовому методу обучаться на выборках оптимальной для него длины. За $T-W$ итераций (где T – длина обучающей выборки; W – размер окна, $T > W$) обрабатывается вся выборка. В результате получается некоторое множество N построенных с помощью базового метода интервальных моделей, причем $0 < N \le T - W$. Полученные модели можно рассматривать как шаблоны поведения временного ряда на отдельных его участках (образы). При прогнозировании метод пытается найти в памяти аналогии с поведением временного ряда на последнем временным отрезке (окне), непосредственно предшествующем прогнозируемой точке. Полученное множество моделей, способных аппроксимировать последнее окно в рамках заданной погрешности, составляют множество альтернатив A. Если $A = 0$, то для такого окна строится новая интервальная модель средствами базового метода (единственная альтернатива, $A = 1$). Про-

гноз представляет собой линейную комбинацию прогнозов по интервальным моделям, из числа A. Факторы, влияющие на прогнозное значение и показатель его определенности, включают величину A, накопленную статистику по рейтингам повторяемости и ошибкам прогнозирования интервальных моделей, ошибки аппроксимации моделями текущего окна, критерий несмещенности и др.. т.н. частные показатели. Общий вид формулы для расчета показателя определенности в прогнозируемой точке $i \in (W,T]$ следующий:

$$D_i = \frac{1}{\sum_v d_b} \sum_b d_b D_{bi},$$

где D_{bi} − частный показатель определенности из фиксированного числа B частных показателей определенности;

d_b − вес частного показателя определенности (внешний параметр метода, константа).

Каждый из частных показателей D_{bi}, в свою очередь, вычисляется на основании характеристик найденного множества моделей-альтернатив для окна, непосредственно предшествующего точке i.

$$D_{bi} = \frac{1}{\sum_m r_m} \sum_m r_m D_{mbi},$$

где D_{mbi} − частный показатель определенности, расчитанный для модели-альтернативы m;

r_m − рейтинг повторяемости модели m, отражающий то, как часто модель m попадала во множество альтернатив при расчетах прогнозных значений в предшествующих точках.

Прогнозное значение рассчитывается как взвешенная свертка прогнозов по найденным альтернативам:

$$P_i = \frac{1}{\sum_m q_{mi}} \sum_m q_{mi} P_{mi},$$

где P_{mi} − прогнозируемое значение в точке i, рассчитанное по модели m;

q_{mi} − вес прогнозного значения по модели m, рассчитываемый на основании ее частных показателей определенности:

$$q_{mi} = \frac{1}{\sum_b d_b} \sum_b d_b D_{bmi}.$$

Подробное описание метода можно найти в [1].

3. Постановка задачи торговой стратегии (ТС) в общем виде

Задано множество исходных данных (обучающая выборка): множество векторов входных переменных $\{X_1, X_2, ..., X_i, ..., X_M\}$ и соответствующих им значений выходной переменной $\{y_1, y_2, ..., y_i, ..., y_M\}$, где i − порядковый номер точки наблюдения; M − число точек наблюдения; $X_i = \{x_{i1}, x_{i2}, ..., x_{iN}\}$ − набор лаговых переменных, изменений курса пары валют (приращений) на N предыдущих точках наблюдения; y_i − фактическое изменение курса пары валют в точке i. Требуется на основании наблюдаемых данных построить модель зависимости выходной переменной от вектора входных переменных $Y_i = Y(X_i)$, а также модель управляющих решений $T_i = T(Y_i)$, где T_i − действие, рекомендуемое по отношению к паре валют, которое может принимать значения: –1 (продавать, курс пары валют будет снижаться), 1 (покупать, курс пары валют будет повышаться), 0 (никаких действий). Фактическое значение t_i выражается как:

$$t_i = \begin{cases} -1, \ \textit{если } y_i < 0; \\ 1, \ \textit{если } y_i > 0; \\ 0, \ \textit{иначе.} \end{cases}.$$

4. Роль и место МППО в модели ТС

Используя исходные данные, МППО способен построить модель $Y_i = Y(X_i)$, адекватную наблюдаемым данным. Метод также предоставляет качественную оценку, характеризующую уверенность в сделанном прогнозе, − показатель определенности $D_i = D(X_i)$, условно нормализованный к диапазону $[0;1]$, $D_i \in [0;1]$, но не дается никаких указаний по использованию этой оценки. В зависимости от различных характеристик временного ряда (таких как полнота входных переменных, волатильность и др.) шкала значимости этой качественной оценки будет отличаться. Более того, со временем эта шкала требует пересмотра и корректировки. Для получения формальной модели ТС, нужно установить рабочий уровень показателя определенности (РУПО), − некоторое значение, по достижению которого прогноз будет считаться надежным.

Поиск значения РУПО является отдельной подзадачей, рассматриваемой в разделах **6** и **7**. Пока, допустив наличие определенного РУПО, выразим T следующим образом:

$$T_i = T_i^{МППО} = \begin{cases} -1, \ если\ Y_i < 0 \wedge D_i \geq D^w; \\ 1, \ если\ Y_i > 0 \wedge D_i \geq D^w; \\ 0, \ иначе. \end{cases}, (0.1)$$

где $D^w \in [0;1]$ – значение РУПО.

5. Оценка эффективности ТС: моделирование выигрышей/проигрышей

Будем называть точку, в которой $T_i \neq 0$, стратегической точкой. Рассчитаем эффективность Z_i для предлагаемого действия T_i в такой точке:

$$Z_i = T_i t_i |y_i|. \quad (0.2)$$

Таким образом, эффективность измеряется в курсовых единицах и отражает сумму выигрышей/проигрышей по обоим трендам: восходящему и нисходящему. Эффективность ТС на участке $i = [1..M]$ оценивается как:

$$Z = \sum_i^M Z_i. \quad (0.3)$$

6. Определение РУПО

Зависимость Z от D^w не имеет аналитического выражения, поэтому предлагается разбить диапазон допустимых значений D^w $[0;1]$ на K равных подмножеств, и, принимая верхнюю границу каждого подмножества в качестве РУПО, рассчитать общую эффективность для каждого случая (Z_k) на некоторой обучающей выборке (обычно непосредственно предшествующей рабочему периоду). Так получаем табличную функцию искомой зависимости.

В простейшем случае для работы выбирается РУПО, показавший максимальную эффективность:

$$D^w = D^w_{\max}, \ соответстующий\ Z_{\max} = \max_k Z_k \quad . \quad (0.4)$$

7. Определение стратегических точек

Определение РУПО, само по себе, является частью более сложной задачи, а именно задачи определения стратегических точек (т.е. таких для которых $T_i \neq 0$), решение которой дает модель $T_i = T(Y_i)$.

В некоторых задачах доступны дополнительные качественные оценки, которые также важно учитывать наравне с РУПО. В случае прогнозирования курсов валют, такой оценкой является абсолютная величина прогнозируемого отклонения $|Y_i|$, поскольку она характеризует силу тренда. Прогноз определенного но незначительного отклонения ($|Y_i| < Y^w \wedge D_i > D^w$) при прочих равных условиях более рискован чем прогноз значительного отклонения ($|Y_i| > Y^w \wedge D_i > D^w$). Включим эту оценку в модель T. После адаптации (0.1), модель T принимает конечный вид:

$$T_i = \begin{cases} -1, \ если\ Y_i < 0 \wedge D_i \geq D^w \wedge |Y_i| \geq Y^w; \\ 1, \ если\ Y_i > 0 \wedge D_i \geq D^w \wedge |Y_i| \geq Y^w; \\ 0, \ иначе. \end{cases} .(0.5)$$

Это изменение затрагивает расчет максимальной эффективности $Z_{\max}$ на обучающей выборке, которая теперь является ф-ей двух аргументов: D^w и Y^w. Вместо (0.4) теперь будем использовать:

$$D^w = D^w_{\max}, \ соответствующий\ Z_{\max} = \max_{k,l} Z_{kl}$$
$$Y^w = Y^w_{\max}, \ соответствующий\ Z_{\max} = \max_{k,l} Z_{kl} \quad , \quad (0.6)$$

где k – индекс значения-кандидата D^w;
l – индекс значения-кандидата Y^w.

Как и в случае D^w, для Y^w существует проблема сбалансированности между рискованностью и доходностью, ведь прогнозов с определенностью в районе единицы исключительно мало, а величина прогнозируемого колебания может никогда не превысить заданного Y^w. Рассмотрим пример вида зависимости $Z(D^w_k, Y^w_l)$ построенной на основании обучающей выборки (взято из расчетов проведенного исследования для пары USD/CHF):

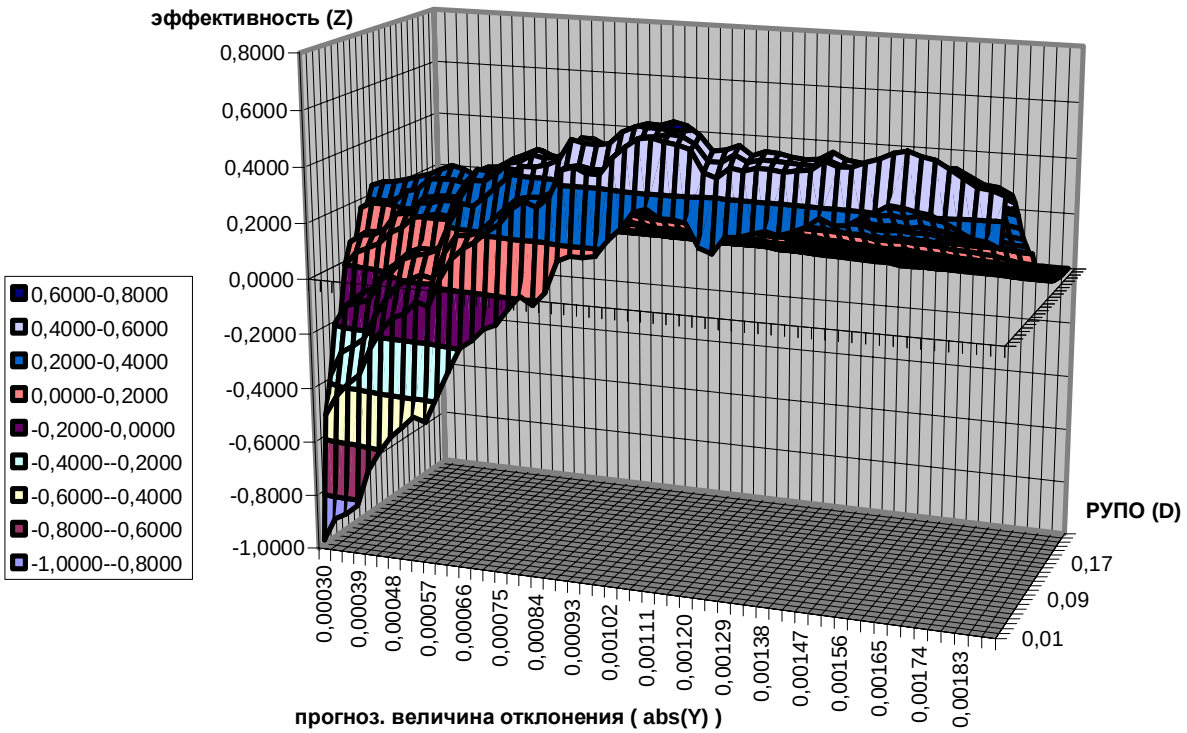


Рис. 1. Пример графика зависимости $Z(D_k^w, Y_l^w)$ ***(проекция 1)***

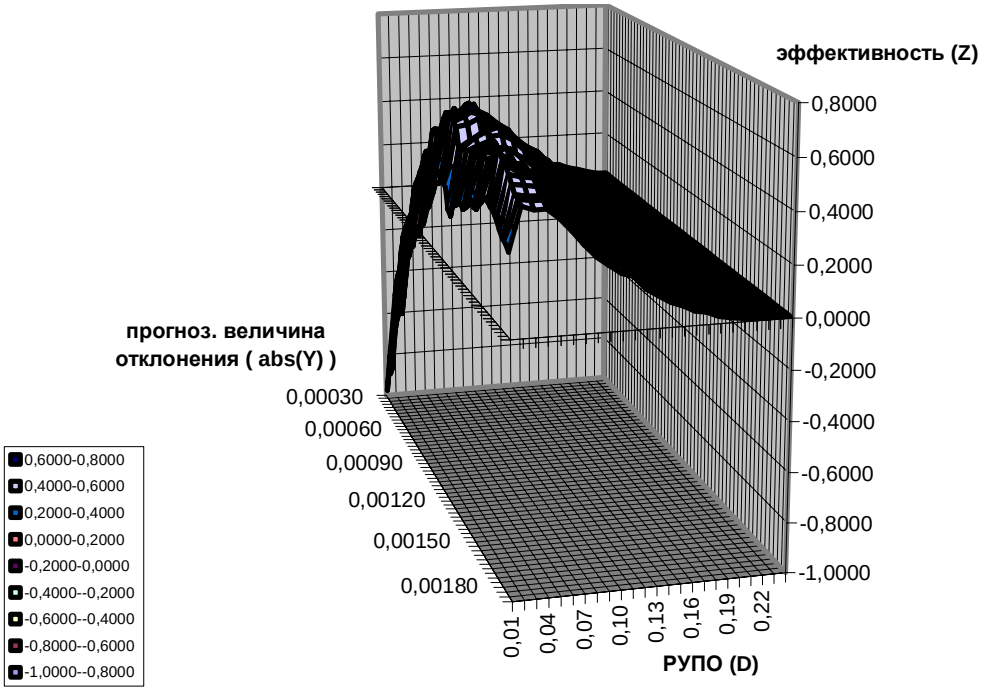


Рис. 2. Пример графика зависимости $Z(D_k^w, Y_l^w)$ ***(проекция 2)***

Как видно из графика (проекция 1), подтверждается гипотеза о том, что при низких значениях Y^w эффективность падает, особенно это заметно при низких значениях D^w. При $D^w = 0 \wedge Y^w = 0$ эффективность соответствует той, которая была бы при отсутствии качественных оценок, здесь она принимает отрицательное значение, т.е. сумма проигрышей перекрывает сумму выигрышей.

Проекция 2 того же графика подтверждает другую гипотезу, а именно, о балансе между надежностью и доходностью. По мере роста значений D_k^w Z_{kl} при любых Y_l^w демонстрирует одну и ту же тенденцию, – сперва резкий скачок, затем плавное снижение до 0, вызванное уменьшением общего кол-ва стратегических точек ($T_i \neq 0$). В данном случае, ТС име-

ла бы эффективность близкую к нулевой уже при $D^w = 0,21$, что существенно меньше максимально возможного значения, 1, тогда как максимальные значения Z_{kl} приходятся на $D_k^w \in [0,02;0,05]$, т.е. близких к нулю. Все это говорит о достаточно высокой волатильности прогнозируемости ряда, слабой корреляции выходной и управляющих переменных. Также стоит отметить здесь высокую чувствительность Z к малейшим изменениям D^w в диапазоне $[0;D_{\max}^w]$ и, наоборот, слабую для D^w в диапазоне $[D_{\max}^w;1]$.

Процедура определения подходящих значений D^w и Y^w может отличаться от (0.6). Более сложный вариант может включать анализ смежных значений Z, для минимизации риска, т.к. зачастую $Z_{\max}$, полученные по (0.6), расположен недалеко от места, где график резко «проваливается».

Двумя внешними параметрами этой процедуры являются размер выборки, на которой подбираются оптимальные значяения D^w и Y^w и периодичность их корректировки. И если объективность полученных значений очевидно будет тем выше чем чаще они будут пересчитываться по мере поступления новых фактических данных, то вопрос о размере выборки для расчета остается открытым.

8. Результаты исследования

Исходные данные:

- Пара валют: USD/CHF
- Тайм фрейм: 4 часа
- Общая выборка: 6 полных лет (2001−2006 гг.)
- Рабочая выборка: 5 лет (данные за 1-ый год использовались для начальной настройки параметров D^w и Y^w

Параметризация:

- МППО использовался в связке с НМГУА для моделирования Y в пределах скользящего окна.
- Параметры НМГУА:
  - кол-во вх. переменных: $N = 6$ содержащих значение курса на пред. N точках.
  - размер обучающей выборки: 5 точек (см. тайм фрейм)
  - размер проверочной выборки: 6 точек (см. тайм фрейм)
- В качестве модели T использовалась (0.5).
- D^w и Y^w рассчитывались по (0.6) на выборке размером 1 год и пересчитывались в начале каждого года по результатам предыдущего.
- При построении табличной ф-ии $Z(D_k^w, Y_l^w)$ шаг приращения для D^w: $D_{k+1}^w - D_k^w = 0,01$, шаг приращения для Y^w: $Y_{l+1}^w - Y_l^w = 0,00003$.

Результаты:

Табл. 1. Результаты экспериментального исследования ТС

Год	N	Ошибок прогноза тренда (%)	$\overline{Z^{neg}}$ $\times 10^{-4}$	$\overline{Z^{pos}}$ $\times 10^{-4}$	$\frac{Z}{N}$	Z $\times 10^{-4}$	$Z_{\max}$ $\times 10^{-4}$	$\frac{Z}{Z_{\max}}$ (%)	Z^D $\times 10^{-4}$	Z^Y $\times 10^{-4}$	Z_{GMDH} $\times 10^{-4}$
2001							*1698*				*-2374*
2002	326	42	-16	21	2.82	919	1168	79	345	-712	-2534
2003	272	36	-14	21	5.74	1562	1562	100	1003	-1751	-5472
2004	226	37	-12	20	5.08	1148	1474	78	-780	-1660	-6312
2005	212	42	-14	20	2.76	585	866	68	-3040	-267	-5852
2006	162	52	-15	28	2.69	436	660	66	-4418	-229	-6503
2007	53	42	-12	21	4.36	231	287	80	-4244	-389	-5267
AVG	**209**	**43**	**-14**	**22**	**3.91**	**814**	**1003**	**78**	**-1856**	**-835**	**-5323**

где N − кол-во стратегических точек в году, т.е. таких для которых $T_i \neq 0$;

Z − эффективность ТС за год (рабочие значения D^w и Y^w определялись на основании $Z^{\max}$, рассчитанного на фактических данных предыдущего года);

$\overline{Z^{neg}} = \frac{1}{M}\sum_{i=1}^{M} Z_i$, *при* $Z_i < 0$ − ср. проигрыш по убыточным сделкам в году;

$\overline{Z^{pos}} = \frac{1}{M}\sum_{i=1}^{M} Z_i$, *при* $Z_i > 0$ – ср. выигрыш по прибыльным сделкам в году;

$\frac{Z}{N}$ – средний выигрыш по сделке в году;

$\frac{Z}{Z_{\max}}$ – отношение, показывающее эффективность априорных значений D^w, Y^w полученных на основании данных за предыдущий год по сравнению с их апостериорными значениями, полученными по результатам года;

Z^D – годовая эффективность для модели T, учитывающей только D^w (Y^w игнорируется);

Z^Y – годовая эффективность для модели T, учитывающей только Y^w (D^w игнорируется);

Z_{GMDH} – годовая эффективность для модели T, игнорирующей критерии D^w, Y^w (соответствует применению базового метода НМГУА в каждой точке, т.е. кол-во стратегических точек равно общему кол-ву точек). GMDH – Group Method of Data Handly.

Примечание: результаты Z и $Z_{\max}$ учитывают расход на комиссию за торговую сделку в размере 3 курсовых ед. для каждой стратегич. точки.

Интерпретация полученных результатов:

Полученные результаты подтверждают ценность предоставляемого МППО показателя определенности сделанного прогноза. Значение $\overline{Z^{pos}}$ по всем годам стабильно больше $\overline{Z^{neg}}$. Что касается процентного отношения ошибок прогноза тренда, то только в 2006 году этот показатель превышает 50%, тем не менее разница между $\overline{Z^{pos}}$ и $\overline{Z^{neg}}$ позволяет достичь за этот год положительной эффективности в 436 курсовых единиц. Z^D, Z^Y, Z_{GMDH} приведены для того, чтобы показать как добавление качественных оценок улучшает результаты.

Значения показателя $Z / Z_{\max}$ дают основания полагать, что есть запас увеличения эффективности за счет сокращения периода корректировки параметров D^w, Y^w, и, возможно, изменения размера выборки для поиска наилучших значений.

Среди прочего стоит отметить некоторую корреляцию между N и Z. Практически весь диапазон значений, принимаемых $\frac{Z}{N}$ (за исключением 2003г.) можно описать треугольным нечетким числом (3.9, 1.2). Поэтому несмотря на тенденцию уменьшения N год от года, что говорит о возрастающей неопределенности временного ряда, средняя прибыль по сделке остается более или менее постоянной.

9. Заключение

На примере задачи построения формальной модели ТС, было продемонстрировано применение показателя определенности, предоставляемого МППО, а также приведены расчеты эффективности полученной модели. Показано значительное преимущество МППО по сравнению с «чистым» НМГУА, который не дает качественных оценок сделанного прогноза.

МППО имеет значительное кол-во параметров, что присуще методам индуктивного моделирования. Процедура настройки и адаптации некоторых из параметров частично поддается формализации. Был приведен пример формальной процедуры определения РУПО. В дальнейшем имеет смысл рассмотреть методы автонастройки значений и других параметров, значения которых сейчас определяются неформальными методами (являются эспертными оценками).

Список литературы

1. Ткаченко С.В. Метод прогнозирования с показателем определенности //Системные исследования и прогрессивные технологии. – 2007. – № 2. – С. 131-141.
2. Зайченко Ю.П., Кебкал О.Г., Крачковский В.Ф. Нечеткий метод группового учета аргументов и его применение в задачах прогнозирования макроэкономических показателей //Научные вести НТУУ КПИ, №2, 2000г., с. 18-26
3. Зайченко Ю.П. Основы проектирования интеллектуальных систем. - Киев: Слово, 2004. –352 с.
4. Лукашин Ю.П. Адаптивные методы краткосрочного прогнозирования временных рядов. – Москва: Финансы и статистика, 2003. – 416 с.

УДК 681.518.3

РОЛИК А.И.,
ТИМОФЕЕВА Ю.С.,
ТУРСКИЙ Н.И.

УПРАВЛЕНИЕ УСТРАНЕНИЕМ НЕИСПРАВНОСТЕЙ В ИТ-СИСТЕМАХ

Статья посвящена проблемам повышения эффективности поиска и устранения неисправностей в ИТ-системах. Предложены и проанализированы способы определения значений пороговых величин для трехпороговой схемы принятия решений. Предложен метод локализации неисправностей в ИТ-системах, объединяющий использование пассивного сбора симптомов и активных проверок. Разработана структура подсистемы управления устранением неисправностей, реализующая предложенные методы.

This article is dedicated to the problems of increasing of efficiency of fault search and elimination in IT-systems. Methods of threshold values determination for the three-threshold scheme are proposed and analyzed. Method of fault localization in IT-systems that incorporates passive symptom gathering and active probing is proposed. Fault management system which uses these methods was developed.

Введение

В настоящее время практически все организации и предприятия для автоматизации выполнения бизнес-процессов или процессов деятельности используют различные информационные технологии (ИТ), которые разворачиваются на основе ИТ-системы, включающей в себя разнообразные информационные и телекоммуникационные ресурсы. Для эффективного управления ИТ-системой и рационального использования ИТ-ресурсов разрабатываются и внедряются различные системы управления ИТ-инфраструктурой (СУИ) [1].

На СУИ, наряду с решением задач автоматизированного управления отдельными технологиями, оптимального распределения ограниченных ИТ-ресурсов [2—4], управления эксплуатацией и взаимодействием администраторов функциональных или технологических подсистем, а также решения множества других, чрезвычайно сложных и важных задач, возлагается ответственность за эффективное управление устранением неисправностей в ИТ-системе. Актуальность решения задачи восстановления работоспособности и качества функционирования ИТ-системы в кратчайшие сроки обусловлена высокой стоимостью информационных технологий, а также существенными потерями бизнеса от простоя или неэффективного использования ИТ-ресурсов, вызванных различного рода неисправностями или ненадлежащим функционированием компонентов ИТ-системы. Поэтому быстрый поиск неисправностей в ИТ-системе и проведение рациональных восстановительных мероприятий представляет собой важнейшую задачу с практической и экономической точек зрения. Кроме того, задача эффективного управления устранением неисправностей представляет большой научный интерес, который обусловлен необходимостью создания множества моделей функционирования компонентов ИТ-системы, идентификации классов и объектов управления, разработки алгоритмов обнаружения и локализации неисправностей, создания структуры подсистемы управления устранением неисправностей, а также множества других сложных и трудоемких задач. Поэтому данная работа посвящена решению вопросов, связанных с построением подсистемы управления устранением неисправностей (ПУН).

Анализ проблем управления устранением неисправностей

Все задачи управления, решаемые СУИ при устранении неисправностей в ИТ-системе, разделяются на три иерархических уровня: обнаружение, локализация и устранение неисправностей.

Для обнаружения неисправностей осуществляется сбор, обработка и анализ сведений о функционировании элементов и подсистем ИТ-системы. Для сбора информации о состоянии элементов ИТ-системы используются различные методы мониторинга и определения состояния элементов и подсистем ИТ-системы. В крупных корпорациях и организациях в ИТ-системе используется множество информационных и телекоммуникационных

технологий, а также большое количество средств вычислительной техники и коммуникационного оборудования различных производителей, которые не всегда поддерживают стандартные протоколы мониторинга. Поэтому определение состояния и качества функционирования множества элементов ИТ-системы, особенно программных, представляет существенные сложности для средств мониторинга. Для преодоления этих проблем могут использоваться программные агенты [5], устанавливаемые на элементы ИТ-системы в случаях, когда они не поддерживают стандартные протоколы мониторинга и управления, например, SNMP, CMIP и пр., когда использование стандартных протоколов запрещено из соображений безопасности или когда стандартными средствами нельзя получить всю необходимую информацию. В больших и сложных ИТ-системах выбор методов и средств мониторинга вызывает существенные затруднения даже для высококвалифицированного ИТ-подразделения.

Локализация неисправностей — один из самых сложных этапов жизненного цикла устранения проблем в ИТ-системах. Сложность поиска места возникновения неисправностей в ИТ-инфраструктуре обусловлена огромным количеством аппаратно-программных компонентов ИТ-системы и влиянием состояния одних элементов на работу других. Для решения этой проблемы используются многочисленные методы и алгоритмы, применимые только для решения отдельных задач в определенных условиях, поскольку сложная природа ИТ-инфраструктуры делает невозможным создание единого унифицированного метода или алгоритма локализации неисправностей.

Непосредственное устранение неисправностей после их точной локализации представляет собой самую простую задачу из всего спектра проблем управления устранением неисправностей. В то же время необходимость быстрого восстановления работоспособности ИТ-системы требует автоматизации процессов принятия решений администраторами и реализации возможности автоматического управления оборудованием и программным обеспечением для быстрой ликвидации неисправностей, предполагающего, например, автоматическую перезагрузку оборудования при зависании, переключение на резерв и пр. Существует ряд многофункциональных, в том числе и фирменных ПУН, которые часто входят в состав СУИ и позволяют автоматизировать ряд процессов восстановления работоспособности ИТ-систем. Однако, большое количество используемых информационных и телекоммуникационных технологий, сложность и многообразие структур и топологий корпоративных ИТ-систем, огромный спектр решаемых в системах задач, существенно отличающиеся требования к ИТ-системам со стороны бизнес-процессов и пр. делают невозможным принципиальное создание универсальной ПУН, способной решать все задачи автоматизации устранения неисправностей. Поэтому в настоящее время разрабатываются различные ПУН, предназначенные для автоматизации деятельности ИТ-подразделений. Для создания по сути специализированных ПУН необходимо разработать структуру и алгоритмы работы, определить механизмы реализации управляющих решений и решить множество других задач.

Таким образом, отсутствие хорошо проработанных принципов построения ПУН делают необходимым создание множества разнообразных моделей ИТ-систем для их эффективного мониторинга, разработки большого класса алгоритмов локализации неисправностей, методов и средств автоматизированного восстановления работоспособности ИТ-систем.

Анализ методов решения задач автоматизированного управления устранением неисправностей в ИТ-системах

Практически все бизнес-процессы крупных организаций поддерживаются информационными технологиями, поэтому незначительные нарушения в работе ИТ-системы могут вызвать простои в работе организации или ухудшить качество услуг, предоставляемых клиентам, что, как правило, влечет за собой существенные финансовые потери. Поэтому своевременная диагностика и локализация неисправностей, а также максимально быстрое восстановление работоспособности компонентов ИТ-системы является одной из важнейших задач СУИ.

Для обнаружения неисправностей определяется состояние и качество функционирования элементов ИТ-системы, после чего значения параметров работы элементов, подсистем

и пр. сравниваются с пороговыми значениями. Работа [6] посвящена анализу качества функционирования элементов информационно-телекоммуникационных систем, но в ней отсутствуют рекомендации о дифференциации состояний элементов ИТ-системы.

В [7] рассматриваются проблемы функционального мониторинга с использованием пороговых схем, однако при принятии решений участвуют только простые пороговые схемы без гистерезиса. Схемы принятия решений, предложенные в [8] и апробированные в [9], используют локальные механизмы гистерезиса, позволяющие существенно сократить количество срабатываний решающих схем, но при этом не рассматриваются вопросы, связанные с определением значений устанавливаемых порогов.

Для обнаружения неисправностей в компьютерных сетях предлагается использовать либо статистический, либо реактивный мониторинги [10]. При этом не рассматриваются вопросы использования этих видов мониторинга в тесной взаимосвязи.

В телекоммуникационных системах используется два основных метода — опрос и передача сообщений о событиях [11], а другие методы, как правило, не рассматриваются. В то же время использование только этих методов не позволяет обеспечить эффективный мониторинг ИТ-систем.

Для поиска и локализации неисправностей в телекоммуникационных системах обычно используют два подхода: пассивная диагностика [12, 13] и активные проверки [14, 15]. В первом случае происходит пассивный сбор информации о состоянии системы (например, с помощью программных агентов [5], [16]), которая затем обрабатывается для нахождения источников неисправностей. При пассивном подходе вмешательство в работу системы минимально, однако, поиск первоисточника неисправностей может занять длительное время при наличии большого процента потерянных симптомов. Активный подход подразумевает использование тестовых проверок для определения неисправностей. К недостаткам данного метода следует отнести значительное вмешательство в работу системы, а также невозможность выявления некоторых неисправностей с помощью проверок.

В [12] рассматривается пассивный метод локализации проблем в сетях с использованием матрицы вероятностей, связывающей симптомы с возможными неисправностями. Метод позволяет ранжировать по вероятности набор потенциальных неисправностей, отличается низкой вероятностью получения ложных симптомов, однако не исключает ошибочные выводы и требует большого времени анализа для крупномасштабных сетей.

В [13] описан механизм корреляции для многоуровневой диагностики неисправностей, основанный на использовании иерархической модели и позволяющий осуществлять корреляцию сообщений о неисправностях в сети и работе распределенных приложений. При этом не учитывается возможная потеря симптомов, что существенно снижает точность результатов.

Предложенный в [14] метод поиска неисправностей в сетях с помощью активных проверок, выбираемых с учетом результатов предыдущих проверок, испытывает затруднения с выявлением сбоев и аномалий при изменении параметров функционирования.

Метод генерирования набора активных транзакций для выявления источника нарушения функционирования в больших распределенных системах, предложенный в [15], сосредотачивается на поиске редких видов неисправностей.

Целью статьи является повышение эффективности устранения неисправностей в ИТ-системах, включая совершенствование методов и средств обнаружения и локализации неисправностей.

Обнаружение неисправностей в ИТ-системах

Для эффективного управления устранением неисправностей необходимо располагать достоверной информацией о функционировании элементов, подсистем и ИТ-системы в целом. Для получения этой информации осуществляется сбор и обработка данных о работе ИТ-системы с использованием методов мониторинга и анализа. Лучшим способом получения такой информации является непрерывный (или с малым периодом) контроль параметров работы компонентов ИТ-системы, при помощи которого обнаруживаются неисправности и принимаются управляющие решения о восстановлении работоспособности. Сложность современных ИТ-систем и большое количество предоставляемых ими услуг требуют кон-

троля большого числа различных параметров. Причем для передачи значений этих параметров используется та же телекоммуникационная сеть, что и для передачи информации пользователей, которая также является объектом контроля. В этом случае передача большого количества значений всевозможных параметров, необходимых для обнаружения и устранения неисправностей, создает большую нагрузку на ИТ-систему. Поэтому уменьшение объема передаваемой контрольной информации о функционировании ИТ-системы является важной задачей.

В телекоммуникационных сетях и ИТ-системах для обнаружения неисправностей используются различные методы мониторинга, из которых наибольшей популярностью пользуются статистический, реактивный и проактивный мониторинги.

При статистическом мониторинге СУИ на основе анализа поступающих к ней многочисленных исходных данных находит некоторые статистические закономерности, позволяющие предсказать тенденции в поведении компонентов ИТ-системы. В этом случае все необработанные данные должны поступить в ПУН, при этом возможности сокращения объема трафика мониторинга отсутствуют. Системы статистического мониторинга позволяют анализировать характеристики трафика и общую функциональность сети. Они хорошо подходят для анализа текущего состояния, перспектив и тенденций поведения телекоммуникационных сетей, но менее пригодны для обнаружения и локализации неисправностей в ИТ-системах.

При реактивном мониторинге СУИ получает информацию о состоянии компонентов ИТ-системы в реальном или псевдореальном времени. Это позволяет реагировать на множество аварийных ситуаций, которые могут прогрессировать в ИТ-системе. Аварийные ситуации обычно говорят о неисправности в сети или означают аномальное поведение компонентов ИТ-системы, которое может привести к неисправности. Такой вид мониторинга дает множество возможностей для поиска механизмов и алгоритмов минимизации объема контрольной информации, передаваемой по сети. Системы реактивного мониторинга позволяют определять только часть проблем в сложных ИТ-системах, испытывая затруднения при анализе функционирования сложных распределенных приложений. При этом диагностика и локализация ошибок в ИТ-системах производится после обнаружения неполадок, так же, как и определяются только проблемы, уже существующие в аппаратном или программном обеспечении.

Более совершенными являются средства проактивного мониторинга, которые не только обеспечивают дистанционный мониторинг в режиме реального времени, регулярные проверки исправности компонентов ИТ-системы, но и позволяют прогнозировать критические состояния системы и на ранней стадии генерировать предупреждения об ошибках, для того, чтобы предотвратить возникновение отказов в работе ИТ-системы. Такой мониторинг позволяет анализировать работоспособность распределенных многоуровневых приложений и пр. Главным отличием этих систем от реактивных является понимание логики распределенных приложений, а также способность предсказывать на основе анализа накопленных данных возможные сценарии развития текущей ситуации. За счет этого системы проактивного мониторинга могут выявлять и предсказать гораздо больше проблем в ИТ-системе, что позволяет устранять неполадки еще на этапе их зарождения и развития. Системы такого типа позволяют не только выявить конкретный некорректно работающий в данный момент аппаратный или программный элемент ИТ-системы, но и предсказать возможность отказа этого элемента в будущем, за счет чего обеспечивается более стабильная работа ИТ-системы и минимизируются издержки, вызванные с ее простоем. Недостатками такого рода систем являются относительная сложность в установке и настройке, высокая стоимость. Кроме того, такие системы, как правило, являются фирменными решениями, например Microsoft, которые хорошо работают в среде Windows и с продуктами Microsoft.

При мониторинге сетевых элементов используется два основных метода получения информации [11]: опрос и передача сообщений о событиях. В СУИ могут дополнительно использоваться методы мониторинга, основанные на сборе, обработке и передаче контрольной информации элементами функциональных и технологических подсистем, методы с использованием агентов СУИ [5, 16], а также методы получения информации о функ-

ционировании компонентов ИТ-системы по косвенным признакам. При этом могут использоваться агенты, созданные разработчиками СУИ или стандартные агенты, например, агенты SNMP, CMIP и пр.

Опрос предполагает отправку управляющей станцией запроса к элементу ИТ-системы на предоставление информации о параметрах, характеризующих его состояние. Как правило, опрос производится с фиксированным, заранее определенным периодом. Сообщения о событиях самостоятельно и асинхронно генерируются элементом ИТ-системы и передаются в ПУН. При этом используются стандартные протоколы, например, SNMP.

Сообщения о событиях вырабатываются сетевыми элементами или элементами распределенных функциональных подсистем в случае превышения заранее установленных пороговых значений.

Все типы систем мониторинга при анализе работоспособности элементов ИТ-системы производят сравнение нормированных значений параметров $S_i(t)$, где $i = 1,\ldots, I$ — количество параметров, определяющих состояние элементов, с пороговыми значениями L_i (рис. 1, а). При этом рассматривается множество параметров, характеризующих состояние элемента, либо вводится единый интегральный показатель качества функционирования простого или составного элемента, как это сделано в [6], по значению которого и оценивается состояние элемента ИТ-системы. В любом случае при пересечении величиной $S_i(t)$ порогового значения L_i изменяет значение дискретный сигнал $D_i(t)$ (рис. 1, б), получаемый на выходе компаратора, осуществляющего сравнение значений $S_i(t)$ с порогом L_i. При каждом изменении значения $D_i(t)$ в ПУН передается соответствующее сообщение.

При использовании порогов без гистерезиса (рис. 1, а) изменение $D_i(t)$ состояния $S_i(t)$ фиксируется при каждом пересечении порога L_i (рис. 1, б.). В этом случае колебания состояния в зоне порога вызывают большое количество срабатываний компаратора и значения $D_i(t)$ могут меняться очень часто. Каждое изменение состояния $S_i(t)$ вызывает необходимость передавать сведения о новом состоянии и значении параметров, приведших к изменению состояния в ПУН, с последующей активацией процедур реакции на новое состояние, а также выработкой управляющих решений, принимаемых в СУИ, и направленных на поддержание стабильной работы ИТ-системы. При этом происходит излишняя загруженность каналов связи. При использовании порогов без гистерезиса и большом количестве элементов ИТ-системы, что практически всегда имеет место в корпоративных системах, может произойти чрезмерная загруженность сети передачей служебного трафика СУИ и перегрузка программного обеспечения ПУН.

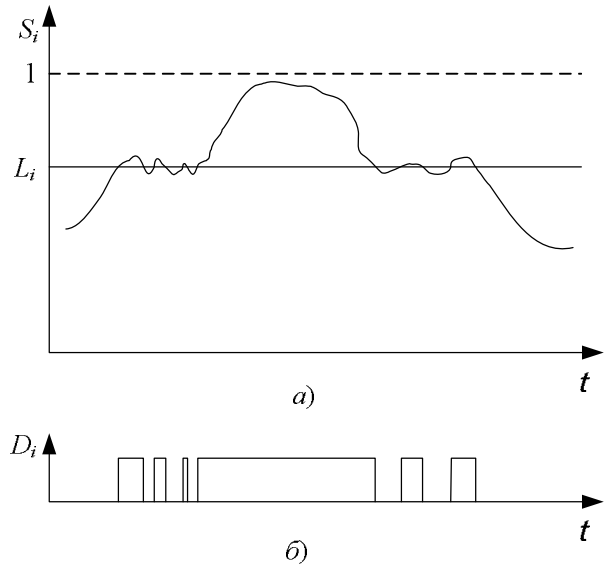


Рис. 1.

Поэтому в СУИ при вынесении решения о состоянии элемента ИТ-системы целесообразно использовать пороги со свойством гистерезиса (рис. 2, а), что позволяет существенно уменьшить количество срабатываний решающей схемы (рис. 2, б).

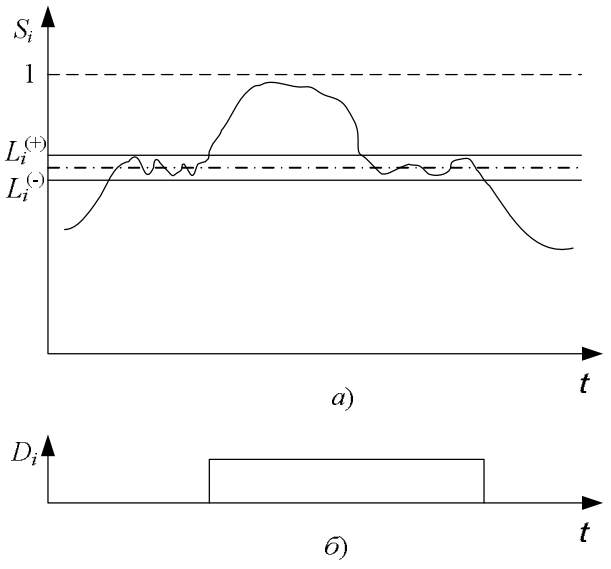


Рис. 2.

В этом случае значение каждого порога $L_i, i = \overline{1, I}$, преобразовывается в два пороговых

значения — $L_i^{(+)}$ и $L_i^{(-)}$, а решающая схема срабатывает таким образом, что сигнал $D_i(t)$ принимает значение логической единицы при превышении значения $S_i(t)$ порога $L_i^{(+)}$ и становится равным логическому нулю, когда значение $S_i(t)$ меньше порога $L_i^{(-)}$. Если выполняется условие $L_i^{(-)} \le S_i(t) \le L_i^{(+)}$, то сигнал $D_i(t)$ сохраняет прежнее значение.

Как видно из рис. 2, б) в этом случае существенно уменьшается количество срабатываний решающей схемы определения состояния элемента ИТ-системы, поскольку решающая схема вырабатывает сигнал изменения состоянии элемента только после того, как изменение состояния надежно зафиксировано.

При использовании порогов со свойством гистерезиса (трехпороговой схемы) возникает задача определения значений $L_i^{(-)}$ и $L_i^{(+)}$.

Учитывая тот факт, что ПУН решает большое количество разнообразных задач по обнаружению и устранению множества неисправностей, происходящих в многочисленных аппаратных и программных элементах ИТ-системы и имеющих сильно различающиеся симптомы, нельзя создать универсальную методику определения значений порогов L_i, $L_i^{(-)}$ и $L_i^{(+)}$. Поэтому метод выбора значений этих порогов определяется конкретной задачей, решаемой ПУН.

Так, для оценки работоспособности распределенного приложения или функционирования подсистемы необходимо выработать критерии оценки качества работы приложения либо определить регламент функционирования подсистемы.

Для функционирования любого распределенного приложения или подсистемы необходимо обеспечить работу аппаратного и программного обеспечения рассредоточенных серверов и рабочих станций, а также сети, через которую производится взаимодействие. При этом на качество функционирования будет влиять множество различных факторов, и их комплексное влияние измерить и оценить будет чрезвычайно сложно. Поэтому для оценки качества функционирования приложения можно поступить следующим образом. На рабочих станциях периодически в автоматическом режиме запускаются контрольные задания и оценивается время их выполнения. При этом можно использовать два подхода.

В первом случае оценивается значение эталонной производительности сервера. Для этого со стороны рабочей станции, после установки операционной системы, инсталляции программного обеспечения и оптимизационной настройки вычислительной системы, с небольшим интервалом времени запускается ряд тестовых заданий. Усредненное значение результатов выполнения тестовых заданий считается эталонным и последующие выполнения контрольных заданий сравниваются с ним.

Второй подход предполагает периодическое выполнение тестовых заданий на сервере, запускаемых автоматически с удаленной рабочей станции в течение длительного времени, которое может измеряться днями или неделями. После накопления статистики с помощью методов прикладного анализа данных производится определение показателей нормальной работы приложения, определяются значения порогов L_i, $L_i^{(-)}$ и $L_i^{(+)}$, и в дальнейшем происходит сравнение результатов проверок, автоматически запускаемых с заданным интервалом, с показателями нормальной работы. Все отклонения от нормальных показателей фиксируются ПУН, после чего статистика отклонений предъявляется администратору, который принимает управляющие решения и при этом производит обучение ПУН. При ухудшении работы распределенного приложения или подсистемы администратор анализирует дополнительные признаки. Если при этом, например, произошло увеличение количества пользователей приложения, то снижение производительности — естественное явление. В этом случае администратор сбрасывает накопленную статистику о показателях нормальной работы и производится автоматическое получение новых значений показателей и порогов, либо осуществляется адаптация порогов к новым условиям работы ИТ-системы. ПУН должна игнорировать кратковременное превышение порогов, не носящее систематический характер, а также обусловленное факторами, не связанными с работой анализируемой подсистемы, а реагировать только на сбои и перегрузки компонентов подсистемы, которые проявляются в результате возникших в ней неисправностей.

Второй подход определения значений нормального режима работы представляет наибольший практический интерес для контроля работы распределенных приложений, функциональных и технологических подсистем,

поэтому ниже предлагаются способы определения значений порогов для этого метода.

Предлагается несколько вариантов определения значений порогов для трехпороговых схем принятия решений.

Все варианты используют методы прикладного анализа данных [18], накопленных в течение продолжительного интервала времени. Множество предлагаемых вариантов определения значений порогов с гистерезисом вызвано необходимостью минимизировать срабатывание решающих схем в предположении, что при дальнейшей эксплуатации подсистемы характер изменения значений контролируемых параметров не будет сильно изменяться по сравнению с характером изменения значений этих же параметров на интервале времени, когда производился сбор статистики. Срабатывание решающих схем должно происходить тогда, когда значение контролируемых параметров начинает систематически отличаться от значений, характерных для нормального режима работы, а кратковременное ухудшение показателей функционирования должно игнорироваться.

Такие требования справедливы для контроля работы ряда функциональных и технологических подсистем, но могут быть неприемлемы, например, для анализа работоспособности сетевых элементов. В последнем случае должны применяться другие способы получения значений порогов или использоваться однопороговые схемы, что и имеет место в большинстве систем сетевого мониторинга.

В качестве примера для демонстрации использован график на рис. 3 изменения значений параметра S_i, полученных в результате выполнения контрольных задач на сервере, запущенных с удаленной рабочей станции.

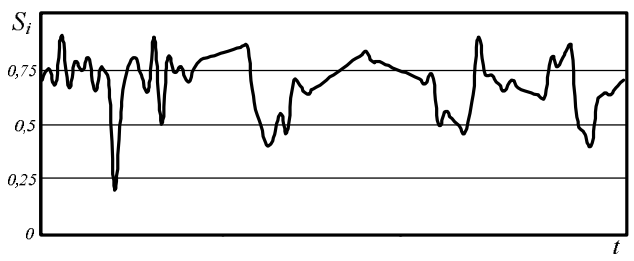


Рис. 3.

Рассмотрим три варианта построения трехпороговых схем.

В первом случае в течение продолжительного интервала времени T с периодом Δt выполняется $M = T / \Delta t$ тестовых заданий, а значение порога L_i i-го параметра определяется следующим образом:

$$L_i = \frac{\sum_{m=1}^{M} S_{i,m}}{M},$$

где M — количество тестовых заданий за время накопления статистики i-го параметра,

$S_{i,m}, m = \overline{1, M}$, — нормированный результат выполнения тестового задания для контроля i-го параметра в момент времени $t_m = m\Delta t$, например, нормированное время выполнения контрольного задания, запущенного на сервере с удаленной рабочей станции.

При этом в моменты времени $t_m = m\Delta t$, $m = \overline{1, M}$, кроме значения $S_{i,m}$, производится одновременная фиксация множества дополнительных признаков, например, доступность сервера, количество пользователей, обслуживаемых сервером, загруженность канала связи и пр. Эти признаки используются ПУН при локализации неисправностей в подсистеме.

Значение верхнего порога $L_i^{(+)}$ для каждого i-го параметра определяется следующим образом:

$$L_i^{(+)} = \frac{\sum_{m=1}^{M} B_{i,m}^{(+)} S_{i,m}}{\sum_{m=1}^{M} B_{i,m}^{(+)}},$$

причем

$$B_{i,m}^{(+)} = \begin{cases} 1, \text{ при } S_{i,m} > L_i; \\ 0, \text{ в противном случае.} \end{cases}$$

Аналогично определяется значение нижнего порога $L_i^{(-)}$ для мониторинга i-го параметра:

$$L_i^{(-)} = \frac{\sum_{m=1}^{M} B_{i,m}^{(-)} S_{i,m}}{\sum_{m=1}^{M} B_{i,m}^{(-)}},$$

здесь

$$B_{i,m}^{(-)} = \begin{cases} 1, \text{ при } S_{i,m} \le L_i; \\ 0, \text{ в противном случае.} \end{cases}$$

Для графика на рис. 3 нормированные значения порогов составляют: $L_i = 0,7$, $L_i^{(+)} = 0,79$, $L_i^{(-)} = 0,49$ (рис. 4). В этом случае сигнал на выходе решающей схемы будет иметь вид, представленный на рис. 5.

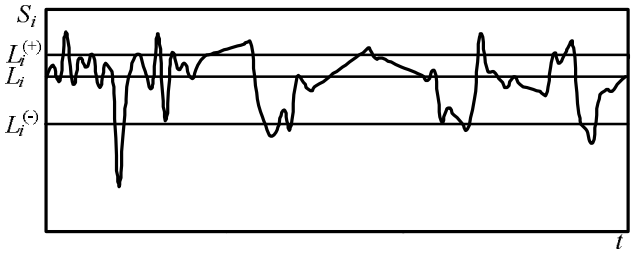


Рис. 4.

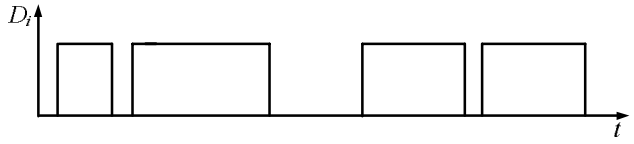


Рис. 5.

Второй способ определения значений порогов основан на аппроксимации исходных данных линейной функцией. Для этого отрезок времени, в течение которого производились замеры параметра S_i, разбивается на интервалы $\Delta T = R\Delta t$. Причем R выбирается таким образом, чтобы интервал ΔT соответствовал периоду, измеряемому десятками минут или часами и мог использоваться для отображения и учета суточной загруженности компонентов ИТ-системы. На каждом интервале ΔT определяется линейная функция, например, по методу наименьших квадратов [18], когда сначала вычисляются коэффициенты b_0 и b_1:

$$b_0 = \frac{2(2R+1)\sum_{r=1}^{R} S_{i,r} - 6\sum_{r=1}^{R} rS_{i,r}}{R(R-1)},$$

$$b_1 = \frac{12\sum_{r=1}^{R} rS_{i,r} - 6(R+1)\sum_{r=1}^{R} S_{i,r}}{\Delta tR(R-1)(R+1)},$$

где b_0 – свободный член регрессии, b_1 – угол наклона, а затем на рассматриваемом интервале строится график вида $S(t) = b_0 + b_1 t$.

Пример преобразования исходных данных (рис. 3) при разбитии отрезка времени на девять интервалов приведен на рис. 6.

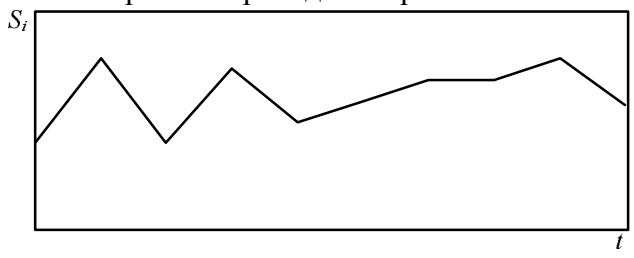


Рис. 6.

Далее производится определение порогов L_i, $L_i^{(+)}$ и $L_i^{(-)}$. Причем для вычисления значений порогов используется та же методика, что и в предыдущем способе, с той разницей, что минимальные и максимальные значения берутся из графика на рис. 6, а не из массива накопленных статистических данных о значении параметра S_i, как это имеет место в первом случае. Для графика на рис. 6 нормированные значения порогов будут $L_i = 0,58$, $L_i^{(+)} = 0,72$, $L_i^{(-)} = 0,45$ (см. рис. 7). Сигнал на выходе решающей схемы для этого случая представлен на рис. 8.

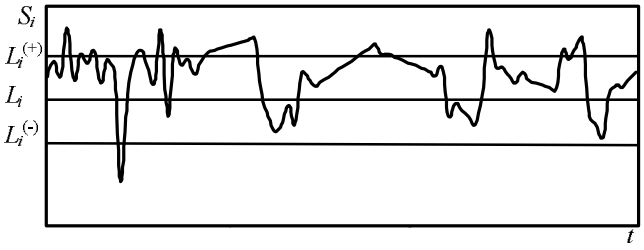


Рис. 7.

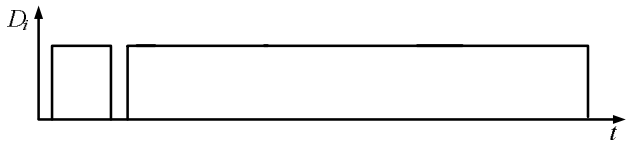


Рис. 8.

В данном случае по сравнению с предыдущим вариантом количество срабатываний решающей схемы значительно меньше, причем срабатывание произошло как реакция на явный провал кривой на рис. 7. Можно сказать, что этот метод отличается наименьшей чувствительностью.

Третий способ определения значений порогов основан на обработке накопленных данных (рис. 3) с использованием методов спектрального анализа. Для этого с помощью формул Бесселя определяются значения коэффициентов a_0, a_k и b_k:

$$a_0 = \frac{2}{M}\sum_{m=1}^{M-1} S_{i,m},$$

$$a_k = \frac{2}{M}\sum_{m=1}^{M-1} S_{i,m}\cos(km\Delta t),$$

$$b_k = \frac{2}{M}\sum_{m=1}^{M-1} S_i \sin(km\Delta t).$$

Для коэффициентов a_k и b_k, не близких к нулю, и исходных данных на рис. 3 уравнение разложения функции в ряд Фурье

$$S(t) = \frac{a_0}{2} + \sum_{k=1}^{K}(a_k\cos(km\Delta t) + b_k\sin(km\Delta t))$$

будет выглядеть следующим образом:

$$S(t) = 0,61 - 0,022\cos(6m\Delta t) + 0,0227\cos(7m\Delta t) + \\ + 0,079\sin(6m\Delta t) + 0,02\sin(8m\Delta t) + \\ + 0,036\sin(9m\Delta t),$$

а график соответствующей функции изображен на рис. 9.

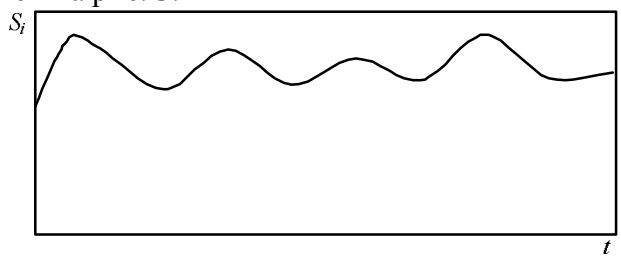


Рис. 9.

Для определения значения порогов L_i, $L_i^{(+)}$ и $L_i^{(-)}$ используется методика, описанная в первом варианте, с той разницей, что минимальные и максимальные значения берутся из графика на рис. 8. Для примера на рис. 3 нормированные значения порогов будут $L_i = 0,73$, $L_i^{(+)} = 0,81$, $L_i^{(-)} = 0,58$ (см. рис. 10), а сигнал на выходе решающей схемы для этого случая представлен на рис. 11.

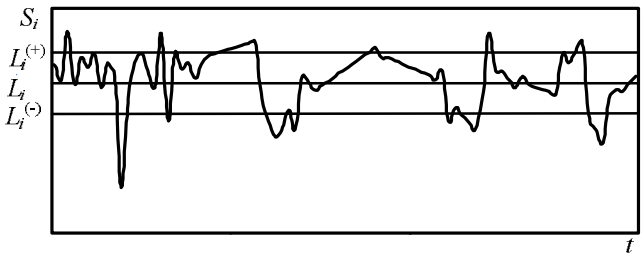


Рис. 10.

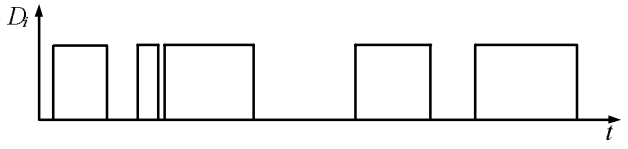


Рис. 11.

Анализируя график срабатываний решающей схемы (рис. 11), можно сказать, что данный вариант обладает наибольшей чувствительностью и позволяет выявлять практически все кратковременные отклонения от нормального режима работы.

Таким образом, для выявления большего количества отклонений от нормальной работы можно использовать первый и третий способы определения значений трехпороговых схем, для выделения только существенных и длительных отклонений от нормального режима работы — первый и второй.

Локализация неисправностей в ИТ-системах

СУИ должна обеспечивать надлежащее функционирование ИТ-системы, мониторинг и анализ работы ее составляющих, эффективное перераспределение ресурсов в случае возникновения неисправностей, диагностирование, локализацию и быстрое устранение неисправностей, а также и решение множества других задач управления ИТ-инфраструктурой. Одной из важнейших подсистем СУИ является подсистема управления неисправностями, в которой основную роль играет модуль локализации неисправностей, ответственный за выявление причины нарушения качества работы ИТ-системы, подсистем и компонентов, или полного прекращения их работы.

В [17] предложен метод локализации неисправностей в интернет-сетях и объединенных сетях, с использованием получаемых пассивно симптомов и активных проверок. На основании пассивно собранных симптомов с помощью матрицы, отображающей взаимосвязи между симптомами и неисправностями, выводятся гипотезы о наличии в сети определенных неисправностей. Проводится оценка выбранных гипотез и, если они объясняют все симптомы и не порождают симптомов, которые не наблюдаются в сети, то выдается сообщение об источниках неисправностей. В противном случае осуществляется поиск новых симптомов или обнаружение недостающих для подтверждения гипотез с помощью активных проверок. Однако в сложной ИТ-системе с большим количеством элементов сбор и обработка информации о взаимосвязях между неисправностями и их симптомами является сложной и ресурсоемкой задачей. Кроме того, в масштабах всей системы множество разных неисправностей могут вызвать одинаковые или похожие симптомы, что делает результаты локализации неисправностей менее точными и увеличивает количество ложноположительных результатов. Поэтому использование матрицы неисправность-симптом для локализации неисправностей является рациональным только для отдельных классов объектов, которые могут соответствовать как отдельным элементам системы, так и ее подсистемам, например, подсистеме документооборота, электронной почты и т. д.

В работе классификация объектов производится в зависимости от функциональности объекта, типизации его элементов и взаимосвязей внутри функциональной или технологической подсистемы. При этом учитывается степень влияния элементов и связей на качество функционирования подсистемы в целом.

Целесообразно выделение следующих классов.

Класс «Сервер — сеть — пользователи». Такому классу соответствует, например, технологическая подсистема — электронная почта. В этом случае на сервере работает приложение, которое предоставляет услуги пользователям. С точки зрения функционирования подсистемы, связь с отдельным пользователем не является критической, т. е. отсутствие этой связи еще не является неисправностью (пользователь может самостоятельно отключиться от сервера и т. д.). ПУН прежде всего должна следить за работоспособностью сервера и его доступностью по сети. Для данного класса симптомами неисправностей будут недоступность сервера или ухудшение параметров его функционирования.

Класс «Сервер – сеть – АРМ». Этому классу соответствует, например, функциональная подсистема. При этом на сервере работает приложение, которое выполняет вычисления и/или анализ на основе данных и решений, поступающих с АРМ пользователя. Связь с отдельными АРМ является критической для функционирования подсистемы. При отсутствии этой связи отсылается сообщение о неисправности. Также важной является работа сервера и отдельных АРМ. Для данного класса симптомами неисправностей будут недоступность одного из элементов подсистемы, ухудшение параметров их работы и т. д.

Класс «Сервер – сеть – сервер». Этому классу соответствует, например, взаимодействие сервера приложений с сервером баз данных, когда на сервере работает приложение, которое посылает запросы базе данных. Связь между сервером и базой данных является критической для функционирования подсистемы, так же, как и работоспособность сервера и базы данных.

Можно ввести классы, содержащие резервные сервера или другие элементы. В этом случае при недоступности или неисправности основного элемента, осуществляется автоматическое переключение на резервный. Сообщения о неисправности основного элемента не являются первоочередными, что и учитывается на этапе классификации сообщений о неисправностях по степени важности.

Предположим, что в ИТ-системе выделяется N классов. Для каждого n-го класса, $n=\overline{1,N}$, на основании анализа элементов класса и взаимосвязей между ними определяются:

– упорядоченное множество симптомов C_n т. е. различных признаков, свидетельствующих о наличии неисправностей в n-м классе, $C_n=\{c_{n,j}\}$, $j=\overline{1,J_n}$, J_n – количество симптомов для n-го класса;

– упорядоченное множество неисправностей F_n, порождающих симптомы C_n, $F_n=\{f_{n,i}\}$, $i=\overline{1,I_n}$, I_n – количество возможных неисправностей для n-го класса;

– упорядоченное множество активных проверок A_n, которые могут опровергнуть или подтвердить наличие симптомов множества C_n, $A_n=\{a_{n,k}\}$, $k=\overline{1,K_n}$, K_n – количество возможных проверок для объектов n-го класса;

– матрица размерностью $I_n\times J_n$ неисправность-симптом $Q_n=\left\|q_{n,ij}\left(c_{n,j}\mid f_{n,i}\right)\right\|$, элемент $q_{n,ij}\left(c_{n,j}\mid f_{n,i}\right)$ которой принимает значения

$$q_{n,ij}\left(c_{n,j}\mid f_{n,i}\right)=\begin{cases}1, \text{ если } j\text{-тый симптом возникает}\\ \text{при наличии } i\text{-ой неисправности;}\\ 0, \text{ в противном случае}\end{cases}$$

– матрица размерностью $J_n\times I_n$ симптом-неисправность $P_n=\left\|p_{n,ji}\left(c_{n,j}\mid f_{n,i}\right)\right\|$, элементы которой $p_{n,ji}\left(c_{n,j}\mid f_{n,i}\right)$ соответствуют вероятности того, что j-й симптом вызван i-ой неисправностью в n-м классе и вычисляется следующим образом:

$$p_{n,ji}\left(c_{n,j}\mid f_{n,i}\right)=\frac{q_{n,ij}\left(c_{n,j}\mid f_{n,i}\right)}{\sum\limits_{f_{n,i}\in F_n} q_{n,ij}\left(c_{n,j}\mid f_{n,i}\right)};$$

– матрица размерностью $K_n\times J_n$ симптом-проверка $V_n=\left\|v_{n,kj}\left(c_{n,j},a_{n,k}\right)\right\|$, элемент которой равен нулю $v_{n,kj}\left(c_{nj},a_{nk}\right)=0$, в случае, если k-я проверка не способна обнаружить j-й симптом. В противном случае $v_{n,kj}\left(c_{n,j},a_{n,k}\right)$

определяет затраты (временные или ресурсные) на проведение *k*-ой проверки для обнаружения *j*-го симптома в *n*-м классе.

В пределах класса связи между неисправностями и симптомами определяются на основе анализа изменений значений параметров элементов класса во времени, например, с использованием методов прикладного анализа данных.

Для каждого наблюдаемого в данный момент времени симптома $c_{H,m}$ из множества $C_H = \{c_{H,m}\}$, $m = \overline{1, M_H}$, где M_H – количество наблюдаемых в настоящее время симптомов в ИТ-системе, определяется его принадлежность к одному или нескольким из *N* классам. Например, симптом «отсутствие связи с сервером X» может быть отнесен к нескольким классам, если на этом сервере работают приложения разных подсистем.

На основании матрицы неисправность-симптом P_n для каждого из этих классов выбираются те неисправности, которые объясняют наибольшее количество наблюдаемых симптомов, т. е. для каждого *n*-го класса формируется множество гипотез $H_n = \{h_{n,l}\}$, $l = \overline{1, L_n}$, элементами которого являются отдельные неисправности, если они могут объяснить все наблюдаемые в пределах *n*-го класса симптомы или комбинации из нескольких неисправностей. В состав гипотез включаются неисправности, имеющие максимальное значение коэффициента вклада K_B, т. е. объясняющие наибольшее количество наблюдаемых симптомов,

$$K_B\left(f_{n,i}\right) = \frac{\sum\limits_{c_{n,j} \in C_H} p_{n,ji}\left(c_{n,j} \mid f_{n,i}\right)}{\sum\limits_{c_{n,j} \in C_{f_{n,j}}} p_{n,ji}\left(c_{n,j} \mid f_{n,i}\right)},$$

где $K_B\left(f_{n,i}\right)$ – коэффициент вклада для неисправности $f_{n,i}$,

$C_{f_{n,i}}$ – множество симптомов, вызываемых неисправностью $f_{n,i}$.

Формирование множества H_n осуществляется следующим образом.

Отбор неисправностей для формирования гипотез происходит циклически. В каждом цикле неисправности с максимальным вкладом включаются во множество гипотез H_n. Симптомы, которые объясняются этими неисправностями, удаляются из множества наблюдаемых симптомов, модифицируя C_H в C'_H, а выбранные неисправности удаляются из множества F_n рассматриваемых неисправностей, модифицируя F_n в F'_n для каждого *n*-го класса. Затем поиск неисправностей с максимальным коэффициентом вклада проводится заново, при этом в качестве входных данных используются модифицированные множества C'_H и F'_n. Процедура итеративно повторяется до тех пор, пока не будут объяснены все симптомы.

Выбранные на предыдущем этапе гипотезы для каждого *n*-го класса, объединенные во множество всех гипотез $H=\{H_1, \ldots, H_n, \ldots, H_N\}$, кроме объяснения симптомов C_H, могут также порождать симптомы, о которых не поступали сообщения от подсистемы мониторинга. Существует несколько возможных причин отсутствия сообщений об этих симптомах. Данные, поступавшие от соответствующего агента мониторинга, могли потеряться, симптом не был выявлен на этапах мониторинга и анализа или предположение о наличии в системе содержащихся в гипотезах неисправностей ошибочно.

Для подтверждения или опровержения выбранных гипотез на основе матриц V_n выбираются проверки из множеств A_n, для всех $n = \overline{1, N}$. Из всех проверок, связанных с определенной гипотезой, выбираются и выполняются те, для которых значение $v_{n,kj}\left(c_{n,j}, a_{n,k}\right)$ минимально.

Если часть наблюдаемых симптомов невозможно отнести к какому-либо отдельному классу, так как они отображают взаимное влияние подсистем, принадлежащих к различным классам, то для определения источников неисправностей, вызвавших эти симптомы, необходимо провести дополнительные активные проверки.

После подтверждения гипотез входящие в них неисправности классифицируются в зависимости от степени их влияния на функционирование элементов подсистем, подсистем и ИТ-системы в целом.

При этом классификация неисправностей может производиться следующим образом:

– аварийная неисправность – неисправность, которая приводит к приостановке

функционирования элементов подсистем, подсистем или всей системы;

– функциональная неисправность – неисправность, которая обуславливает невозможность выполнения одной или нескольких функций элементов подсистем, подсистем или всей системы;

– сбой – неисправность, обусловленная однократным отказом элемента подсистемы;

– потенциальная неисправность – означает, что некоторые параметры, которые характеризуют функционирование элементов подсистем, приближаются к граничному значению, при превышении которого функционирование элемента подсистемы не будет соответствовать заданному регламенту.

Затем данные об источниках неисправностей и, при наличии, рекомендации по их устранению выдаются администратору в соответствии с установленным на этапе классификации приоритетом.

Результаты работы модуля локализации, т.е. данные про наблюдаемые симптомы и вызвавшие их источники неисправностей, заносятся в базу известных неисправностей, что облегчает дальнейшую работу ПУН.

После получения сообщений об источниках возникших в системе неисправностей и степени влияния этих неисправностей на функционирование системы, администратор, основываясь на рекомендациях ПУН, выполняет действия по устранению самостоятельно или уведомляет соответствующую службу.

Подсистема управления устранением неисправностей

Структурная схема разработанной подсистемы управления устранением неисправностей представлена на рис. 12.

ПУН состоит из модулей анализа, корреляции, локализации и связи с администратором. ПУН взаимодействует с базой данных мониторинга, базой известных неисправностей, базой данных конфигурации.

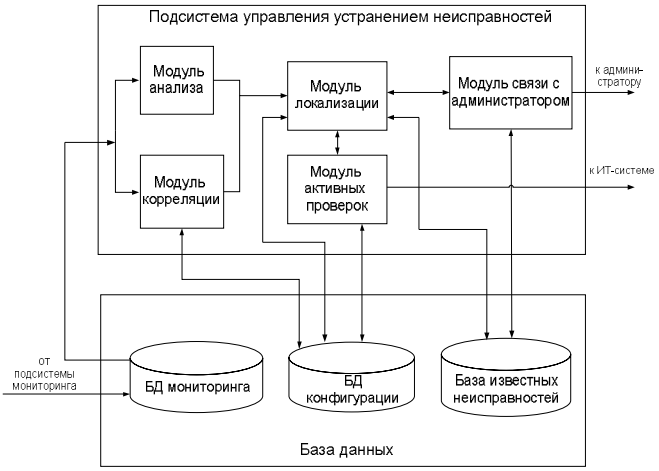


Рис. 12. Структура подсистемы управления устранением неисправностей

Модуль анализа производит предварительную обработку накопленных данных о трафике в телекоммуникационной сети и различных параметров функционирования элементов функциональных и технологических подсистем ИТ-системы. Эти данные запрашиваются модулем анализа из базы данных мониторинга, в которой содержатся результаты работы подсистемы мониторинга. В результате анализа определяются аномалии в телекоммуникационной сети и в функционировании элементов подсистем, подсистем и ИТ-системы в целом, которые не были выявлены программными агентами подсистемы мониторинга. Результаты анализа в виде сообщений о нарушении функционирования объектов (элементов подсистем, подсистем и системы в целом) передаются в модуль локализации.

Модуль корреляции уменьшает количество сообщений о неисправностях, поступивших от подсистемы мониторинга, объединяя сообщения от одного источника или имеющие одну причину и, при возможности, определяет эту причину. Выявление источника неисправности на данном этапе возможно при условии владения полной информацией о взаимосвязях элементов, от которых исходят сообщения о неисправностях. Модуль корреляции использует данные о конфигурации управляемой ИТ-системы и физических взаимосвязях ее элементов, получаемые из базы данных конфигурации. Результаты работы модуля в виде сообщений о неисправностях, т. е. симптомы неисправностей передаются модулю локализации для дальнейшего поиска источника нарушений функциональности управляемой системы.

Модуль локализации производит поиск источников неисправностей, признаками которых являются наблюдаемые симптомы. На первом этапе модуль запрашивает данные в базе известных неисправностей и проверяет, наблюдались ли подобные симптомы в системе ранее. В случае отсутствия в ней искомых данных, т. е. записей о взаимосвязях между наблюдаемыми симптомами и известными неисправностями, модуль производит поиск источников неисправностей на основании наблюдаемых симптомов. При этом задействуются данные о классификации подсистем и взаимосвязях их элементов, содержащиеся в базе данных конфигурации. Модуль находит наиболее вероятные источники неисправностей и классифицирует их в зависимости от степени влияния на функционирование элементов подсистем, подсистем и ИТ-системы в целом. Если во время поиска возникла необходимость подтвердить или опровергнуть какой-либо симптом для проверки справедливости гипотез, модуль локализации обращается с запросом к модулю активных проверок.

Модуль активных проверок подтверждает или опровергает наличие в системе симптомов, запрашиваемых модулем локализации. Происходит это с помощью выполнения активных проверок. Матрицы связи симптомов и проверок для каждого из классов хранятся в базе данных конфигурации, на основании этих матриц и происходит выбор проверок. Результаты проверок, т. е. обнаруженные симптомы или же сообщения об их отсутствии, передаются модулю локализации.

Модуль связи с администратором обеспечивает вывод информации о возникновении неисправностей, а также их причины, обнаруженные модулем локализации, что позволяет администратору изменять параметры ПУН, запрашивать и модифицировать данные в базе известных неисправностей, производить обучение ПУН и пр.

База данных мониторинга содержит информацию о значениях отслеживаемых параметров элементов и сообщения о неисправностях, т. е. о наличии симптомов. Эти данные являются результатом работы подсистемы мониторинга и собираются программными агентами с контролируемых объектов.

В базе известных неисправностей содержится информация о когда-либо обнаруженных в системе неисправностях, их симптомах и рекомендованных методах устранения этих неисправностей. После обнаружения новых неисправностей модуль локализации заносит в базу известных неисправностей данные о них: симптомы, причины и методы их устранения и пр.

Данные о конфигурации системы, о взаимосвязях элементов подсистем содержатся в базе данных конфигурации. Также в ней хранятся данные о классификации элементов и подсистем.

Выводы

В работе рассмотрены вопросы повышения эффективности устранения неисправностей в ИТ-системах. Проанализированы методы мониторинга ИТ-систем. Для выявления неисправностей в ИТ-системах с помощью порогов предложены варианты определения значений пороговых величин для трехпороговых схем.

Предложен метод локализации неисправностей в информационно-телекоммуникационной системе, объединяющий преимущества использования пассивного сбора симптомов и активных проверок, а именно: минимальное вмешательство в работу ИТ-системы и быстрое обнаружение источников неисправностей. Предложена структура подсистемы управления устранением неисправностей, которая производит обнаружение неисправностей с помощью анализа данных мониторинга, локализацию источников неисправностей, оповещает о них администратора и выдает рекомендации по устранению неисправностей. ПУН производит быстрый и достаточно точный поиск источников неисправностей в ИТ-системах. В дальнейшем необходимо ввести учет оценки степени влияния неисправностей на функционирование ИТ-системы.

Список литературы

1. Теленик С.Ф., Ролік О.І., Букасов М.М., Соколовський Р.Л. Система управління інформаційно-телекомунікаційною системою корпоративної АСУ// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. —2006. — № 45. — С. 112—126.

2. Ролик А.И. Модель управления перераспределением ресурсов информационно-телекоммуникАционной системы при изменении значимости бизнес-процессов// Автоматика. Автоматизация. Электротехнические комплексы и системы. ХГТУ. — 2007. — № 2 (20). — С. 73—82.
3. Теленик С.Ф., Ролік О.І., Букасов М.М. Моделі управління розподілом обмежених ресурсів в інформаційно-телекомунікаційній мережі АСУ// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — 2006. — № 44. — С. 234—239.
4. Теленик С.Ф., Ролік О.І., Букасов М.М., Терещенко П.І. Управління доступом до обмежених ресурсів інформаційно-телекомунікаційної мережі АСУ військового призначення// Сб. наук. праць ЦНДІ Збройних Сил України. — 2006. — №3 (37). — С. 33—43.
5. Ролик А.И., Соколовский Р.Л. Распределение мобильных компонентов системы управления информационно-телекоммуникационной системой// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — 2007. — № 47. — С. 113—124.
6. Ролик А.И., Глушко Е.В. Анализ качества функционирования элементов информационно-телекоммуникационных систем// Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. — 2008. — № 48. — С. 113—120.
7. Cormode G., Muthukrishnan S., Yi K. Algorithms for Distributed Functional Monitoring// Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms. — San Francisco, California. — 2008. — P. 1076—1085.
8. Wuhib F., Dam M., Stadler R., Clemm A. Decentralized computation of threshold crossing alerts// Proc. 16th IEEE/IFIP International Workshop on Distributed Systems. — Barcelona, Spain. — 2005. — Vol. 3775. — P. 220—232.
9. Wuhib F., Stadler R., Clemm C. Decentralized service-level monitoring using network threshold crossing alerts// IEEE Communications Magazine. — 2006. — Vol. 44. — № 10. — P. 70—76.
10. Dilman M., Raz D. Efficient reactive monitoring// IEEE JSAC. — 2002.— Vol. 20, № 4. — P. 668—676.
11. Stallings W. SNMP, SNMPv2, SNMPv3, RMON1 and 2. — 3rd edition. AdisonWesley. — 1998. — 640 p.
12. Steinder M., Sethi A. S. Probabilistic Fault Diagnosis in Communication Systems Through Incremental Hypothesis Updating// Computer Networks.— July 2004.— vol. 45.— no. 4.— pp. 537—562.
13. Appleby K., Goldszmidt G., Steinder M. Yemanja – A Layered Event Correlation Engine for Multi-domain Server Farms// Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on. — 2001.— pp. 329—344.
14. Rish I., Brodie M., Odintsova N., Ma S., Grabarnik G. Real-time Problem Determination in Distributed Systems using Active Probing// Network Operations and Management Symposium. NOMS 2004. IEEE/IFIP.— April 2004.— Vol. 1.— pp. 133—146.
15. Guo J., Kar G., Kermani P. Approaches to Building Self Healing System using Dependency Analysis// Network Operations and Management Symposium. NOMS 2004. IEEE/IFIP.— April 2004.— Vol. 1.— pp. 119—132.
16. Теленик С.Ф., Ролік О.І., Терещенко П.І., Літвінцов О.В. Модель управління розподілом ресурсів інформаційно-телекомунікаційної системи збройних сил України// Зб. наук. праць ННДЦ оборонних технологій і воєнної безпеки України. — 2006.— №5 (34).— С. 117—124.
17. Tang Y., Al-Shaer E. S. Boutaba R. Active Integrated Fault Localization in Communication Networks// Integrated Network Management Proceedings. IM’2005. IEEE/IFIP International Symposium on.— May 2005.— pp. 543—556.
18. Бендат Дж., Пирсол А. Прикладной анализ случайных данных. — М.: «Мир».— 1989. — 526 с.

УДК 004.3

ЖАБИН В.И.,
МАКАРОВ В.В.

ИСПОЛЬЗОВАНИЕ ТАБЛИЦ ФУНКЦИЙ ДЛЯ БЫСТРОГО ВЫЧИСЛЕНИЯ МНОГОЧЛЕНОВ

Предложен таблично-алгоритмический метод вычисления многочленов, основанный на предварительной обработке коэффициентов. Показана возможность ускорения вычисления многочленов по сравнению с реализацией известных таблично-алгоритмических методов.

The table and algorithmic method of calculation of polynomials based on preliminary coefficient processing is offered. Possibility of acceleration of calculation of polynomials in comparison with realization of the well-known table and algorithmic methods is shown.

Введение

Непрерывное расширение области применения цифровой вычислительной техники приводит к необходимости решения все более сложных задач обработки информации, что, в свою очередь, требует постоянного повышения эффективности аппаратных средств и программного обеспечения вычислительных систем. Одним из основных направлений ускорения процессов обработки информации является аппаратная реализация функций программного обеспечения. Такой подход является особенно актуальной для систем реального времени.

Особенность режима реального времени заключается в том, что на реакцию системы накладываются ограничения со стороны внешних факторов. Система должна отреагировать на события, происходящие на объекте управления, за определенный промежуток времени, величина которого определяется особенностями объекта управления. Выход за допустимые временные рамки приводит к снижению качества управления или потере управляемости объектом, что может повлечь нежелательные последствия, в том числе катастрофические.

Из широкого круга систем реального времени можно в первую очередь выделить весьма широкий класс системы управления объектами и различными технологическими процессами (системы числового программного управления, навигации и т.д.). Для таких систем характерным является необходимость решения траекторных задач, требующих вычисления различных функциональных зависимостей.

В связи с этим важной задачей является разработка аппаратных средств для вычисления различных функций, позволяющих обеспечить минимальное время вычисления функций.

Анализ методов, использующих таблицы функций

Высокую скорость воспроизведения функции обеспечивает табличный метод, основанный на хранении в памяти полной таблицы функции. В этом случае для получения значения функции достаточно одного обращения к памяти. Реализация такого похода требует наличия $K \cdot 2^n$ бит памяти, где K – число воспроизводимых функций, а n – разрядность операндов. При больших значениях K и n обеспечить такую емкость не всегда возможно.

С целью уменьшения аппаратурных затрат используют таблично-алгоритмические методы [1-3], которые сочетают обращение к таблицам и обработку полученной информации, например, с использованием полиномиальной аппроксимации. В связи с этим возникает необходимость разработки быстродействующих операционных устройств для вычисления многочленов.

Современные достижения в области интегральной технологии предоставляют средства, которые позволяют создавать сложные системы, в том числе, на одном кристалле (технология CSoC – Configurable System on Chip). Лидерами данного направления являются фирмы Triscend, Xilinx, Altera. В общем случае микросхемы содержат вычислительные ядра, память и программируемую логику FPGA (Field Programmable Gate Arrays), что

дает возможность эффективной реализации таблично-алгоритмических методов вычислений.

В статье [4] авторов данной работы проведен анализ эффективности различных таблично-алгоритмических методов вычисления многочленов и предложен метод, базирующийся на предварительной обработке коэффициентов. Показано, что метод обеспечивает вычисление многочлена

$$P(x) = \beta_0 + \beta_1 X + \beta_2 X^2 + ... + \beta_m X^m \qquad (1)$$

за время $T = t_C + 2t_П + t_Д$, где t_C – время сложения двух чисел, $t_П$ – время обращения к памяти, $t_Д$ – время сложения $(m+2)$ чисел с помощью дерева сумматоров (m – степень полинома). Показано, что предложенный метод позволяет уменьшить аппаратурные затраты по сравнению с известными таблично-алгоритмическими методами, причем, без снижения быстродействия.

В данной работе исследуется возможность ускорения вычислений с использованием предварительной обработки коэффициентов многочлена.

Обоснование метода

Представим многочлен (1) в виде

$$P_1(X) = \alpha_0 + (\alpha_1 + X)^2 + (\alpha_2 + X)^3 + \\ + ... + (\alpha_m + X)^{m-1} + F_1(X). \qquad (2)$$

Определим, при каких значениях $\alpha_0,...,\alpha_m, F_1(X)$ выполняется равенство

$$P_1(X) = P(X).$$

Разложив в (2) составляющие $(\alpha_j + X)^{j+1}$ для $j = \overline{1,m}$ в соответствии с формулой бинома Ньютона, получим

$$P_1(X) = \alpha_0 + \sum_{i=0}^{2} C_2^i \alpha_1^{2-i} X^i + \sum_{i=0}^{3} C_3^i \alpha_2^{3-i} X^i + ... + \\ + \sum_{i=0}^{j+1} C_{j+1}^i \alpha_j^{j+1-i} X^i + ... + \sum_{i=0}^{m+1} C_{m+1}^i \alpha_m^{m+1-i} X^i + F_1(X).$$

Выделим в полученном выражении составляющую $(X^2 + X^3 + ... + X^{m+1})$ и сгруппируем оставшиеся члены по степеням X:

$$P_1(X) = (\alpha_0 + \sum_{i=1}^{m} C_{i+1}^0 \alpha_i^{i+1}) + X \sum_{i=1}^{m} C_{i+1}^1 \alpha_i^i + \\ + X^2 \sum_{i=2}^{m} C_{i+1}^2 \alpha_i^{i-1} + ... + X^j \sum_{i=j}^{m} C_{i+1}^j \alpha_i^{i-j+1} + ... + \\ + C_{m+1}^m \alpha_m X^m + F_1(X) + (X^2 + X^3 + ... + X^{m+1}).$$

Приравняв полученное выражение исходному многочлену с учетом $C_k^0 = 1$, $C_{k+1}^k = k+1$, получим:

$$F_1(X) = -(X^2 + X^3 + ... + X^{m+1}),$$
$$\alpha_m = \beta_m / (m+1),$$
$$\alpha_j = (\beta_j - \sum_{i=j+1}^{m} C_{i+1}^j \alpha_j^{i-j+1}) / (j+1),$$
$$\alpha_1 = (\beta_1 - \sum_{i=2}^{m} C_{i+1}^1 \alpha_i^i) / 2,$$
$$\alpha_0 = \beta_0 - \sum_{i=1}^{m} \alpha_i^{i+1}.$$

Устройство, работающее в соответствии с выражением (2), может быть построено по схеме на рис. 1. Блоки $П_i$ $(i = \overline{1,m})$ предназначены для возведения чисел в степень $(i + 1)$, а блок $П_{m+1}$ обеспечивает хранение таблицы функции $F_1(X)$. Сумматоры См осуществляют суммирование двух чисел, а суммирующий блок СБ – $(m+2)$ чисел.

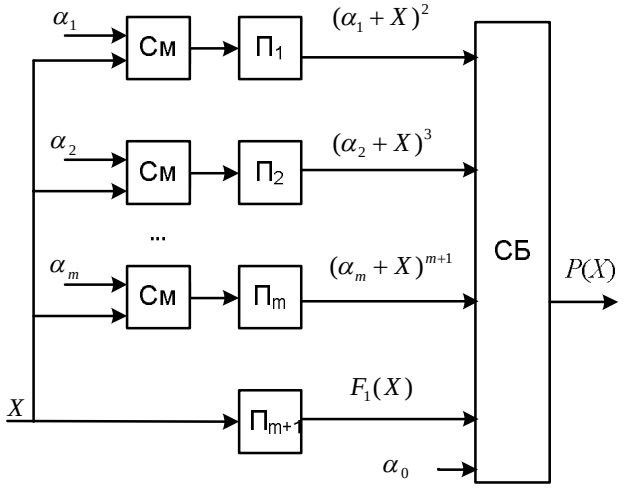


Рис. 1. Структура устройства

Общая емкость необходимой памяти определяется как

$$N = \sum_{i=1}^{m+1} 2^{n_{2i}} n_{1i}, \qquad (3)$$

где n_{1i} и n_{2i} – разрядность соответственно выходных и входных слов блоков Π_i.

Разрядность слов зависит от формата данных и требуемой точности вычислений. Рассмотрим эту зависимость.

Будем считать, что $n_{1i} = n'_{1i} + n''_{1i}$, $n_{2i} = n'_{2i} + n''_{2i}$, где n'_{1i} и n'_{2i} – разрядности целой части, n''_{1i} и n''_{2i} – разрядности дробной части соответственно выходных и входных слов.

Значения n'_{1i} и n'_{2i} для блоков Π_i $(i = \overline{1,m})$ определяются так:

$$\begin{aligned} n'_{1i} &=]\log_2(\alpha_i + X)^{i+1}_{\max}[, \\ n'_{2i} &=]\log_2(\alpha_i + X)_{\max}[. \end{aligned} \tag{4}$$

Здесь $]\cdot[$ – функция округления до ближайшего большего целого числа, если данное число не целое.

Для блока Π_{m+1} получим:

$$\begin{aligned} n'_{1i} &=]\log_2 \sum_{j=2}^{m+1} X^j[, \\ n'_{2i} &=]\log_2 X[. \end{aligned} \tag{5}$$

Очевидно, что величины n'_{1i} и n'_{2i} зависят от пределов изменения аргумента и коэффициентов α_i. Эти величины необходимо определить для всех многочленов, которые могут вычисляться на рассматриваемом устройстве. При расчете следует принять максимальные значения.

Разрядность дробной части выходных и входных слов блоков Π_i можно определить следующим образом:

$$\begin{aligned} n''_{1i} &=]\log_2(1/\Delta_{1i})[, \\ n''_{2i} &=]\log_2(1/\Delta_{2i})[, \end{aligned}$$

где Δ_{1i} и Δ_{2i} – погрешность представления соответственно выходных и входных слов.

Как следует из (2), значение $P_1(X)$ является результатом суммирования $(m+2)$ слагаемых. Так как погрешность Δ результата равна сумме погрешностей слагаемых, то приняв погрешности всех слагаемых равными Δ_1, найдем

$$\Delta_1 = \Delta/(m+2) = 2^{-n}/(m+2). \tag{6}$$

Тогда для блоков Π_i $(i = \overline{1,m+1})$ получим

$$n''_{1i} = n +]\log_2(m+2)[. \tag{7}$$

Исходя из значения Δ_1, определим погрешность Δ_{2i} представления входных слов блоков $\Pi_1 - \Pi_m$.

Погрешность Δ_1 возведения в степень $(i+1)$ числа $Z_i = (\alpha_i + X)$, представленного с погрешностью Δ_{2i}, определяется как

$$\begin{aligned} \Delta_1 &= (Z_i + \Delta_{2i})^{i+1} - Z_i^{i+1} = C_{i+1}^1 Z_i^i \Delta_{2i} + \\ &+ C_{i+1}^2 Z_i^{i-1} \Delta_{2i}^2 + \ldots + C_{i+1}^i Z_i \Delta_{2i}^i + C_{i+1}^{i+1} \Delta_{2i}^{i+1}. \end{aligned}$$

Пренебрегая членами, содержащими второй и более высокие порядки Δ_{2i}, получим

$$\Delta_1 = C_{i+1}^1 Z_i^i \Delta_{2i} = (i+1) Z_i^i \Delta_{2i}.$$

Отсюда видно, что погрешность результата максимальна при максимальном значении $Z_i = (\alpha_i + X)\max$. С учетом (6) определим

$$\Delta_{2i} = 2^{-n}/((i+1)(m+2)(\alpha_i + X)^i_{\max}).$$

Исходя из этого, разрядность представления дробной части входного слова блока Π_i будет составлять

$$n''_{2i} = n +]\log_2((i+1)(m+2)(\alpha_i + X)^i_{\max})[. \tag{8}$$

Суммарную длину выходных и входных слов блоков Π_i $(i = \overline{1,m})$ определим с учетом (4), (7) и (8) так:

$$\begin{aligned} n_{1i} &= n +]\log_2(\alpha_i + X)^{i+1}_{\max})[+ \\ &+]\log_2(m+2))[, \\ n_{2i} &= n +]\log_2(\alpha_i + X)_{\max})[+ \\ &+]\log_2((i+1)(m+2)(\alpha_i + X)^i_{\max})[. \end{aligned} \tag{9}$$

Разрядность выходных и входных слов блока Π_{m+1} получим из (5) и (7), с учетом того, что входное слово X блока Π_{m+1} представлено n разрядами после запятой. Тогда:

$$\begin{aligned} n_{1(m+1)} &= n +]\log_2 \sum_{j+2}^{m+1} X^j[+]\log_2(m+2)[, \\ n_{2(m+1)} &= n +]\log_2 X[. \end{aligned} \tag{10}$$

Таким образом, суммарная емкость требуемой памяти для хранения таблиц определяется выражением (3) с учетом (9) и (10). Следует заметить, что аппаратурные затраты на реализацию данного метода и метода, рассмотренного в [4], имеют примерно одинаковый порядок.

Время вычислений в устройстве складывается из времени t_C сложения двух чисел, обращения к памяти $t_П$ и суммировании $(m+2)$-х чисел в СБ, то есть

$$T = t_C + t_П + t_Д .$$

Одним из наиболее быстродействующих блоков для сложения нескольких чисел является дерево сумматоров [5]. Если СБ построен в виде дерева сумматоров, то время вычисления многочлена составит

$$T = (\lceil \log_2 (m+2) \rceil + 1) t_C + t_П .$$

Таким образом, по сравнению с методом, рассмотренным в работе [4], предлагаемый метод позволяет сократить время вычислений на длительность одного цикла обращения к памяти.

Выводы

Предложенный метод вычисления многочленов позволяет по сравнению с наиболее эффективным из рассматриваемых известных таблично-алгоритмических методов ускорить воспроизведение функций. Уменьшение времени вычислений при прочих равных условиях позволяет расширить область применения систем управления в реальном времени за счет сокращения цикла обработки информации.

Для смены воспроизводимой функции достаточно один раз пересчитать коэффициенты полинома. Схема устройства при этом не претерпевает изменений.

Метод может быть эффективно использован в случае необходимости многократного вычисления значений функций при разных значениях аргумента. Такой режим вычислений является естественным, например, для систем управления, когда алгоритмы обработки информации в реальном времени, включающие вычисление полиномов, на протяжении кадра управления не изменяются.

Список литературы

1. Информационные системы: Табличная обработка информации / Под ред. Е.П.Балашова и В.Б.Смолова. – Л.: Энергоатомиздат, 1985. – 184 с.
2. Попов Б.А., Теслер Г.С. Вычисление функций на ЭВМ. Справочник. – К.: Наукова думка, 1984. – 600 с.
3. Кнут Д. Искусство программирования для ЭВМ: В 3-х т., т. 2. – М.: Мир, 1977. – 723 с.
4. Жабин В.И., Макаров В.В. Таблично-алгоритмический метод вычисления многочленов // Вісник НТУУ “КПІ”, Інформатика, управління та обчислювальна техніка. – № 46. – 2007. – С. 206-210.
5. Карцев М.А., Брик В.А. Вычислительные системы и синхронная арифметика. – М.: Сов. радио, 1981. – 360 с.

УДК 004.3

ЖАБИНА В.В.

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ НА УРОВНЕ ОПЕРАЦИЙ В ПОТОКОВЫХ СИСТЕМАХ

В статье предложен метод ускорения вычислений на уровне параллельной обработки машинных слов в системах, управляемых потоком данных. Рассмотрена структура системы, позволяющая одновременное формирование и выполнение нескольких команд. Показана возможность автоматической идентификации слов акторов и данных на основе графа задачи.

In the article the method of acceleration of calculations at the level of the simultaneous processing of computer words in the systems, guided the flow of data is offered. The structure of the system, allowing the simultaneous forming and execution of a few instructions, is considered. Possibility of automatic identification of words of actors and dates on the basis of graph of task is shown.

Введение

Время решения параллельных задач во многом определяется эффективностью распараллеливания процессов с целью максимальной загрузки ресурсов вычислительной системы.

Большинство из современных технологий параллельного программирования относятся к средствам статического распараллеливания процессов. Задачи распараллеливания в этом случае решаются на этапе разработки программ, причем, их эффективность во многом определяется квалификацией программиста [1].

К оновным недостаткам средств статического распараллеливания следует отнести следующее.

При анализе алгоритмов не всегда удается выявить параллельные ветви, то есть не всегда можно выявить скрытый параллелизм. Это обусловлено целым рядом причин, к основным из которых можно отнести недостаток информации о динамике процессов.

При разработке программ, как правило, учитывается конфигурация аппаратных средств системы, изменение которой может потребовать повторной разработки программы.

Одним из перспективных подходов, позволяющих устранить ряд недостатков статического планирования, является разработка средства динамического распараллеливания вычислений. В этом случае назначение заданий на вычислительные узлы осуществляется системой автоматически в процессе решения задач. Такой подход дает возможность достичь большей степени параллелизма, так как позволяет выявить параллельные ветви, которые возникают непосредственно в процессе вычислений.

Поскольку динамическое распределение заданий осуществляется средствами самой системы, то важной задачей является уменьшение на это расходов времени в процессе обработки информации.

Модель вычислений, управляемых потоком данных

Для решения проблемы динамического распределения заданий между вычислительными ресурсами системы предложены модели вычислений, управляемых потоком данных. К наиболее ранним можно отнести работы [2-6].

В потоковых системах отсутствует необходимость решения задачи динамического распределения заданий между вычислительными узлами на программном уровне. Распределение работ между вычислителями осуществляется автоматически на аппаратном или микропрограммном уровне.

В системах, управляемых потоком данных, команды выполняются не в заданной программой последовательности, а при наличии всех операндов, то есть определяющим в данном случае является не порядок выполнения команд, а доступность данных для команды.

Будем рассматривать алгоритмы с мелкозернистой структурой, при реализации которых распараллеливание осуществляется на уровне выполнения отдельных команд, то есть на уровне обработки машинных слов. Для определенности будем рассматривать двуместные операции, что соответствует системам команд большинства современных процессо-

ров. Одноместные операции приводятся к двуместным добавления фиктивного операнда, который не участвует в вычислениях.

Потоковая система должна содержать вычислительные модули (ВМ), среду формирования команд (СФК), устройства ввода (УВв) и вывода (Увыв) данных. Компоненты системы связаны между собой через соответствующие коммуникационные средства.

Подготовка вычислений осуществляется на основе графа, каждой i-й вершине которого соответствует операция, а каждой дуге – операнд.

Операция для i-й вершины графа описывается информационным словом, которое называют актором (actor). Актор описывается кортежем

$$A_i = \langle I_i, F_i, N_i, T_i \rangle, \qquad (1)$$

где I_i – идентификатор (уникальное имя) данного актора; F_i – функция преобразования данных (код операции); N_i – имя актора, для которого i-й актор подготавливает операнд, а T_i – совокупность признаков этого операнда.

Акторы связаны между собой только по данным. Каждой дуге графа соответствует слово данных

$$D_i = \langle I_i, Q_i, N_i, T_i \rangle, \qquad (2)$$

где Q_i – значение операнда.

Из соответствующих элементов A_i и D_i в СФК формируется команда, которая выполняется в свободном ВМ или помещается в очередь. Известны различные алгоритмы формирования команд [2-5]. Для организации СФК используется ассоциативная память или ее эмуляция с применением других технических средств.

Компиляторы позволяют автоматизировать подготовку акторов и данных на базе графа, причем, без учета конкретного числа ВМ в системе.

Если время формирования команд при наличии готовых ее компонентов существенно меньше времени выполнения команд в ВМ, то в разных ВМ одновременно выполняются разные команды, за счет чего и достигается распараллеливание операций. Современные технологии реализации ВМ позволяют использовать аппаратные методы ускорения операций, в результате чего зачастую интенсивность формирования команд в СФК становится недостаточной для загрузки нескольких ВМ одновременно. Покажем это на примере реализации графа, изображенного на рис. 1. Считаем, что время формирование команды в СФК равно времени выполнения команды в ВМ.

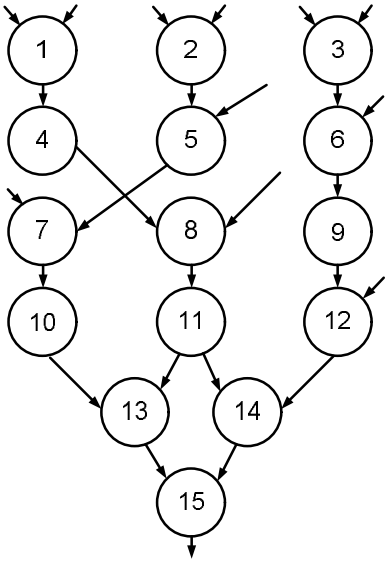


Рис. 1. Граф задачи

Возможные временные диаграммы занятости СФК и ВМ без учета времени пересылки данных показаны на рис. 2. Номера вершин графа соответствуют номерам интервалов на диаграмме. Команда может начать выполняться в ВМ только после ее формирования в СФК. Команда назначается на свободный ВМ.

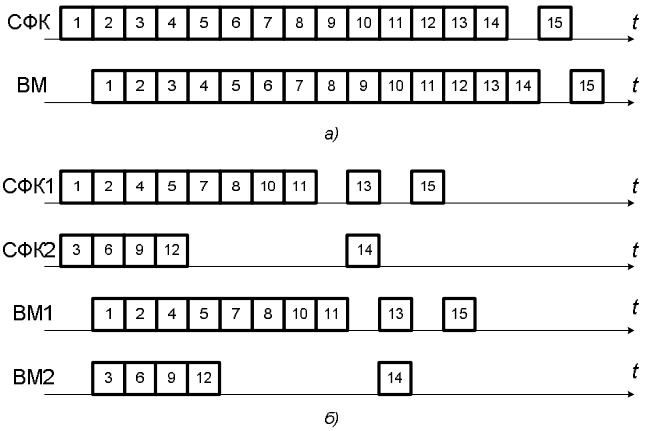


Рис.2. Временная диаграмма занятости структурных компонентов системы

Как видно из рис. 1 *а*, при наличии одной СФК достаточно иметь в системе только один ВМ, который успевает обрабатывать поток команд, формируемых в СФК. Интенсивности потока команд не хватает для загрузки второго ВМ.

Две СФК позволяют загрузить второй ВМ (рис.1 *б*). Очевидно, что время реализации вычислений при этом уменьшается.

В связи с этим важной проблемой является увеличение интенсивности потока готовых команд с целью ускорения параллельных вычислений.

Тривиальным подходом является дублирование потоковых процессоров, но при этом возникают недостатки, присущие статическим методам подготовки параллельных программ. Программист должен предварительно разрезать граф задачи на подграфы, предопределить идентификаторы акторов и данных в разных подграфах с учетом связи по данным между ними.

Более эффективным является подход, который позволяет автоматически формировать акторы и данные при наличии нескольких СФК [6, 7]. Система содержит модули, в состав которых входит СФК и ВМ. Модули организуются в кольцевую структуру. Данные циркулируют в такой структуре в поисках своего актора. Команды формируются в разных СФК и выполняются в соответствующих ВМ.

Недостатком такого подхода являются затраты времени на пересылку данных между модулями системы. Кроме того, выход из строя любого модуля приводит к неработоспособности всей системы, поскольку информация передается последовательно от одного модуля к другому.

Ниже предлагается метод параллельного формирования команд в разных СФК, который не требует ручного вмешательства в процесс назначения идентификаторов, а также реализации последовательного обмена данными непосредственно между СФК.

Концепция формирования параллельных потоков команд

Подготовку задачи для случая параллельного формирования различных потоков команд рассмотрим на примере структуры системы, показанной на рис. 3, где КС – коммуникационная среда.

Система может иметь любое число Увв, Увыв и ВМ. Количество СФК должно быть равно $k = 2^j$, где $j = 1,2,3,...$

Модифицируем объекты (1) и (2) следующим образом:

$$A_i = \langle M_i, I_i, F_i, N_i, T_i \rangle, \quad (3)$$

$$D_i = \langle M_i, I_i, Q_i, N_i, T_i \rangle, \quad (4)$$

где M_i – идентификатор СФК, который назначается компилятором автоматически с учетом связей акторов по данным.

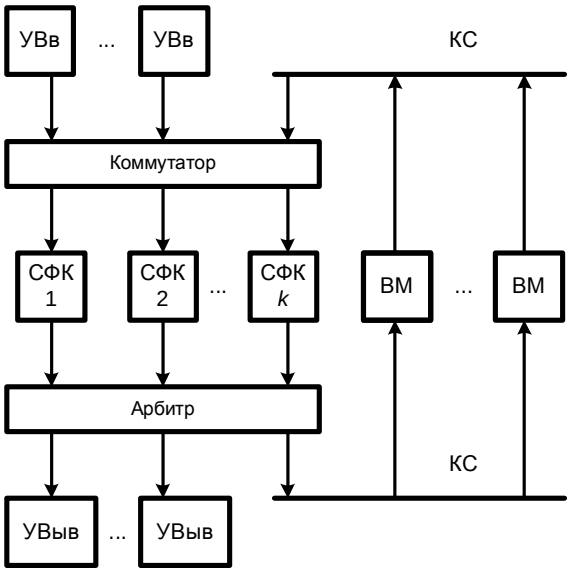


Рис.3. Структура потоковой системы

Алгоритм нахождения M_i представлен ниже.

1. В графе задачи вершинам, которые соответствуют одноместным операциям, добавить входную дугу с фиктивным операндом γ.

2. Используя известные в теории компиляторов алгоритмы, получить префиксную (постфиксную) бесскобочную форму записи вычисляемого результата в виде строки.

3. Просматривая полученную строку выделить все пары имен операндов вместе с именем соответствующего актора (выполняемой операции). В результате прохода выписать имена всех акторов, которые могут выполняться параллельно во времени. Далее рассматривать эти тройки объектов в качестве операндов для следующего прохода. Повторять аналогичные проходы до полного сворачивания строки, выписывая после каждого прохода имена акторов, которые могут выполняться одновременно.

4. Составить цепочки имен акторов следующим образом. В первую цепочку входят все акторы, полученные последними при каждом проходе (см. п.3), во вторую цепочку – предпоследними и т.д.

5. Первой цепочке назначить 0-й номер СФК, второй – 1-й номер и т.д. Полученные номера СФК присваивают акторам (3) и данным (4) в качестве имен M_i.

Команды, соответствующие разным цепочкам, формируются в разных СФК, за счет чего

создаются параллельные потоки команд в системе.

Некоторые вопросы, связанные, например, с реализацией циклических алгоритмов, а также разветвлением данных выходят за рамки данной статьи.

Заметим, что в теории компиляторов бесскобочные формы применяют для формирования последовательности операций с использованием стека. В рассматриваемом случае на основе такой записи определяются операции, которые могут выполняться параллельно во времени.

Пример автоматического назначения идентификаторов

Рассмотрим граф (рис. 4), в котором номера вершин соответствуют именам акторов, то есть определяют выполняемую операцию. Исходные данные заданы буквами.

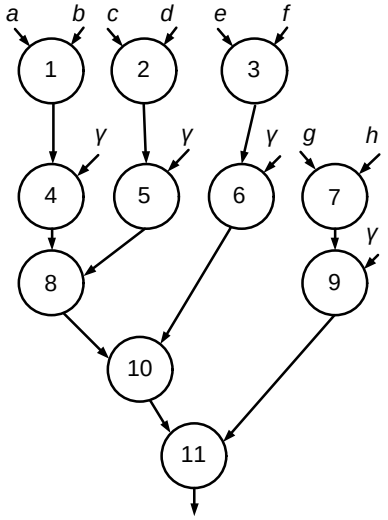


Рис. 4. Граф задачи

В соответствии с п.1 приведенного алгоритма вершинам 4, 5, 6 и 9 с одноместными операциями приписаны фиктивные операнды γ.

Методом обхода графа, начиная с вершины 11, формируем в соответствии с п.2 строку

$$11\,9\,\gamma\;7\;h\;g\;10\;6\;\gamma 3\;f\;e\;8\;5\gamma\;2\;d\;c\;4\gamma 1\;b\;a$$

Согласно п.3 выполняем пять проходов.

1-й проход.

Операции: $7hg$; $3fe$; $2dc$; $1ba$.

Акторы: 7, 3, 2, 1.

2-й проход.

Операции: $9\gamma-$; $6\gamma-$; $5\gamma-$; $4\gamma-$.

Акторы: 9, 6, 5, 4.

3-й проход.

Операция $8--$.

Актор 8.

4-й проход.

Операция $10--$.

Актор 10.

5-й проход.

Операция $11--$.

Актор 11.

Прочерками обозначены операнды, которые соответствуют результатам операций предыдущих проходов.

В соответствии с п.4 и п.5 составляем цепочки операций и присваиваем им номера M_i, начиная с нулевого.

Номер M_i=0 назначается для команд 1, 4, 8, 10, 11, M_i=1 – для команд 2 и 5, M_i=2 – для команд 3 и 6, M_i=3 – для команд 7 и 9.

При наличие в системе четырех ВМ параллельно могут выполняться четыре потока команд. Если в системе два ВМ, то команды с четными номерами будут выполняться в одном ВМ, а с нечетными – в другом. Функцию распределения данных между разными СФК выполняет коммутатор. Таким образом, при подготовке задачи нет необходимости учитывать число ВМ в системе.

Выводы

Предложенный метод формирования параллельных потоков команд в системах, управляемых потоком данных, имеет ряд преимуществ как по сравнению с классическими параллельными системами, так и по сравнению с кольцевыми потоковыми структурами.

По сравнению с традиционной моделью вычислений существенно упрощается процесс подготовки задач. Граф задачи может быть составлен весьма просто с использованием графического редактора. Нет необходимости учитывать число вычислителей в системе и длительность выполнения команд. Управляющие слова и данные могут быть сформированы автоматически компилятором на основе графа. Динамическое распределение операций позволяет выявить непосредственно в процессе вычислений скрытый параллелизм, связанный с различными длительностями об-

работки данных в различных ветвях алгоритмов.

Вычислительные модули могут выполнять операции, относящиеся к различным задачам, в любой последовательности. В связи с этим в системе могут одновременно решаться независимые задачи, причем, нет необходимости синхронизации задач, что позволяет начинать решение новой задачи в любой момент времени.

По сравнению с кольцевой организацией потоковых систем предложенный метод позволяет ускорить вычисления за счет сокращения непроизводительных затрат времени на последовательную пересылку данных между модулями системы. Кроме того, это способствует повышению надежности систем.

Все это повышается эффективность обработки данных в потоковых системах.

Список литературы

1. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2004. – 608 с.
2. Dennis J. B., Missunas D. P. A preliminary architecture for basic data flow processor// Proc. 2nd Annual Symp. Comput. Stockholm, May 1975. N. Y. IEEE. – 1975. – P. 126 – 132.
3. Silva J.G.D., Wood J.V. Design of processing subsystems for Manchester data flow computer // IEEE Proc. N.Y. – 1981. – Vol. 128, N 5. – P. 218 – 224.
4. Watson R., Guard J. A practical data flow computer // Computer. – 1982. – Vol. 15, N 2.– P. 51 – 57.
5. Hogenauer E.B., Newbold R.F. Inn Y.T. DDSP – a data flow computer for signal processing/ Proc. Int. Conf. Parall. Process. Ohio, August 1982. N.Y. // IEEE. – 1982. – P. 126 – 133.
6. Johnson D. Data flow machines threaten the program counter// Electronic Design. – 1980. – N 22. – P. 255 – 258.
7. Функционально ориентированные процессоры / Водяхо А.Н., Смолов В.Б., Плюснин В.У., Пузанков Д.В. / Под ред. В.Б.Смолова. – Л.: Машиностроение. Ленингр. отд-ние, 1988. – 224 с.

УДК 681.3.06

КУЛАКОВ А.Ю.

СПОСОБ ПОВЫШЕНИЯ ЭФФЕКТИВНОСТИ GRID СИСТЕМ НА БАЗЕ СЕТЕЙ MPLS

Предложен способ повышения эффективность функционирования GRID систем. Проведен анализ изменений пропускной способности каналов передачи данных на базе объединенных деревьев доставки информации туннелей LSP сетей MPLS. Показано, что распределение нагрузки по маршрутизаторам DR, а также балансировка пропускной способности при многоабонентской доставке информации существенно зависит не только от физической, но и от логической организации распределенной системы

A way to improve the effectiveness of the GRID systems is proposed. The analysis of changes in channel bandwidth capacity based on the combined trees information delivery of LSP tunnels MPLS networks is adopted. It has been shown that load distribution of the DR as well as balancing bandwidth capacity of multicast information delivery depends not only on the physical but also on the logical organization of a distributed system

Введение

В связи с расширением области применения GRID систем, увеличения доли мультимедийного и высокоскоростного трафика в компьютерных сетях и системах, актуальной становится задача организации быстрой и эффективной доставки информации.

GRID – это аппаратно-программная инфраструктура на основе компьютерной сети, которая обеспечивает эффективный доступ к высокопроизводительным компьютерным ресурсам [1]. В настоящее время для построения GRID систем все чаще используется многопротокольная коммутация на основе меток (MPLS – Multiprotocol Label Switching). Технология MPLS сочетает в себе возможности управления трафиком, присущие технологиям канального уровня [2] (уровень 2 модели OSI), масштабируемость и гибкость протоколов, характерные для сетевого уровня [3] (уровень 3 модели OSI). В соответствии с протоколом MPLS маршрутизаторы и коммутаторы присваивают на каждой точке входа в таблицу маршрутизации особую метку и сообщают эту метку соседним устройствам.

Сеть MPLS делится на две функционально различные области – ядро и граничную область. Ядро образуют устройства, минимальным требованием к которым является поддержка MPLS и участие в процессе маршрутизации трафика для того протокола, который коммутируется с помощью MPLS. Маршрутизаторы ядра занимаются только коммутацией. Все функции классификации пакетов по различным классам FEC (Forwarding Equivalence Class), а также реализацию дополнительных сервисов берут на себя граничные маршрутизаторы LER (Label Edge Router) и маршрутизаторы переключений LSR (Label Switching Router). В результате интенсивные вычисления приходятся на граничную область, а коммутация выполняется на уровне ядра. В связи с этим актуальной становиться задача повышения эффективности вычислений в граничной области MPLS.

Механизмы MPLS могут быть реализованы двумя способами: на основе модели перекрытий (overlay) или одноранговой (peer) модели. В модели перекрытий, называемой также UNI, маршрутизатор является клиентом оптического домена и взаимодействует только с непосредственно примыкающим к нему оптическим узлом. В одноранговой модели уровень IP/MPLS обладает теми же полномочиями, что и уровень оптической передачи. Основная задача MPLS в обоих случаях заключается в расширении сферы действия технологии коммутации по меткам и ее переносе от маршрутизаторов на оптический уровень, где решения о дальнейшей пересылке пакетов принимаются на основании временных интервалов, длин волн и физических портов «неявных меток», а не содержимого пакета. Технология MPLS устанавливает равноправные отношения между оптическими доменами за счет поддержки новых классов LSR.

В настоящее время для построения GRID систем все чаще используется многопротокольная коммутация на основе меток (MPLS – Multiprotocol Label Switching). Технология MPLS сочетает в себе возможности управления трафиком, присущие технологиям каналь-

ного уровня (уровень 2 модели OSI), масштабируемость и гибкость протоколов, характерные для сетевого уровня (уровень 3 модели OSI). В соответствии с протоколом MPLS маршрутизаторы и коммутаторы присваивают на каждой точке входа в таблицу маршрутизации особую метку и сообщают эту метку соседним устройствам.

Сеть MPLS делится на две функционально различные области – ядро и граничную область. Ядро образуют устройства, минимальным требованием к которым является поддержка MPLS и участие в процессе маршрутизации трафика для того протокола, который коммутируется с помощью MPLS. Маршрутизаторы ядра занимаются только коммутацией. Все функции классификации пакетов по различным классам FEC (Forwarding Equivalence Class), а также реализацию дополнительных сервисов берут на себя граничные маршрутизаторы LER (Label Edge Router) и маршрутизаторы ядра LSR (Label Switching Router). В результате интенсивные вычисления приходятся на граничную область, а высокопроизводительная коммутация выполняется в ядре, что позволяет оптимизировать конфигурацию устройств MPLS в зависимости от их местоположения в сети.

Главная особенность MPLS – отделение процесса коммутации пакета от анализа IP-адресов в его заголовке. Очевидным следствием описанного подхода является тот факт, что очередной сегмент LSP может не совпадать с очередным сегментом маршрута, который был бы выбран при использовании традиционных алгоритмов маршрутизации. Использование явно задаваемого маршрута в сети MPLS свободно от недостатков стандартной IP-маршрутизации от источника, поскольку вся информация о маршруте содержится в метке и пакету не требуется нести адреса промежуточных узлов, что улучшает управление распределением нагрузки в сети. По значению метки пакета определяется его принадлежность к определенному классу FEC на каждом из участков коммутируемого маршрута. Метка должна быть уникальной лишь в пределах соединения между каждой парой логически соседних LSR. Поэтому одно и то же ее значение может использоваться LSR для связи с различными соседними маршрутизаторами, если только имеется возможность определить, от какого из них пришел пакет с данной меткой. В рамках архитектуры MPLS вместе с пакетом разрешено передавать не одну метку, а целый стек. Результат коммутации задает лишь верхняя метка стека, нижние же передаются прозрачно до операции изъятия верхней. Такой подход позволяет создавать иерархию потоков в сети MPLS и организовывать туннельные передачи. Коммутируемый путь LSP одного уровня состоит из последовательного набора участков, коммутация на которых происходит с помощью метки данного уровня.

Для динамической маршрутизации в GRID системах наиболее эффективно использовать многоабонентскую передачу информации, на базе протокола «точка – многоточие» с использованием меток. Существуют два протокола рассматриваемые IETF (Internet Engineering Task Force) для построения LSP в сетях MPLS [4]: протокол резервирования ресурсов RSVP-TE (Resource Reservation Protocol – Traffic Engineering) и протокол распределения меток LDP (Label Distribution Protocol). Оба протокола могут быть расширены для передачи информации в LSP режиме «точка – многоточие», однако, RSVP-TE строит деревья «точка – многоточие» от корня к конечным вершинам, а LDP наоборот. В случае одновременной передачи одного IP-пакета по нескольким адресам, распределение меток LDP формируется после групповой передачи. В свою очередь, протокол RSVP-TE специально разработан для масштабируемого группового обслуживания с использованием сетей VPN и мульти-VPN на основе группового дерева. Поэтому, использование протокола RSVP-TE для совмещения резервирования ресурсов и организации LSP для различных потоков данных является более эффективным для поставленной задачи.

Применение RSVP-TE позволяет оптимизировать GRID систему под конкретные задачи путем создания туннелей LSP на основе VPN. Сообщения этих протоколов передаются от одного узла сети к другому в соответствии с данными об IP-адресах маршрута.

Необходимо отметить, что каждый граничный маршрутизатор должен располагаться как можно ближе к корню дерева (Root), обычно, такой граничный маршрутизатор является абонентом многоабонентской доставки или широковещательного потока который был послан для распределения. В этом случае, все граничные маршрутизаторы (PE) посылают

единому Root свои пакеты через LSP туннель ядра для получения внешней метки, осуществляя проверку предварительно назначенной внутренней метки соответствующему маршрутизатору распределения (DR) для данной VPN группы, и заменяют метку VPN группы на метку всего дерева. В таблице маршрутизатора DR существует вторая метка (вых. метка) для передачи информации вниз по дереву распределения другим VPN, как показано на рис. 1.

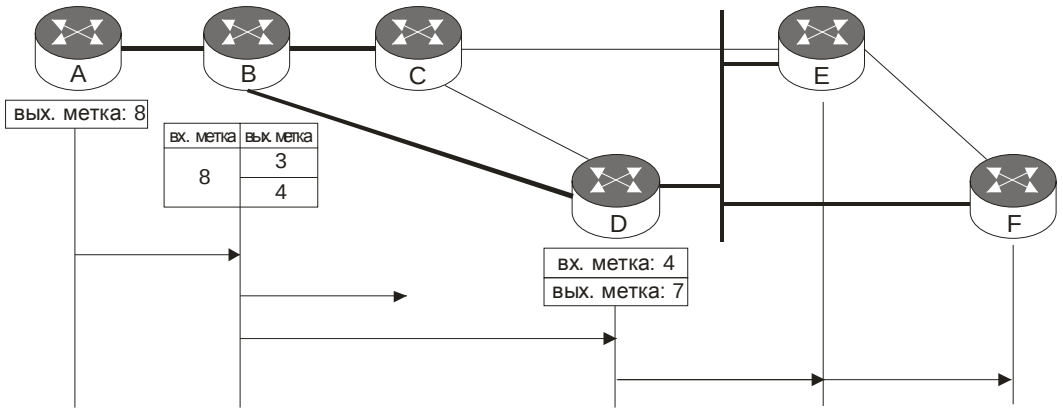


Рис. 1. Процесс передачи информации вниз по дереву распределения другим VPN

Выбор между единичным деревом и объединенным деревом распределения зависит от доступных ресурсов. Объединенное дерево обеспечивает надежность и минимальную задержку, за счет своей сложности и состояния передачи, поэтому защищенное одиночное групповое дерево в нашем случае является наиболее приемлемым и перестраиваемым. Перестройка дерева осуществляется при помощи сигнального протокола «точка-многоточие» LSP поддерживаемого RSVP-TE.

В связи с тем, что возрастает интерес к использованию GRID систем существенным требованием является повышение пропускной способности канала без повышения стоимости ресурсов системы. Для оценки состояния пропускной способности будем рассматривать простую модель, полученную в результате объединения маршрутизаторов PE с абонентами в сети VPN по не которым признакам. Предлагается не изменять физическую топологию системы, а применять VPN сети для поддержки необходимой структуры. Информация, предназначенная всем узлам GRID системы, которые принадлежат множеству сетей VPN объединенных в дерево распределения, передается при помощи протокола «точка-многоточие». Для этого необходимо, чтобы все маршрутизаторы PE передавали информацию корню дерева (Root). Если маршрутизатор не владеет информацией о составе группы, то передача информации осуществляется вниз по дереву граничным маршрутизаторам.

Для этого необходимо отметить, что VPN сети A и B объединены между собой маршрутизаторами DR, которые в свою очередь связаны между собой корневыми маршрутизаторами CR. Для передачи информации абонент AMb VPN сети A передает информацию в маршрутизатор CR Root через маршрутизаторы PE и DR.

Корневой маршрутизатор передает информацию маршрутизаторам DR, которые входят в состав VPN B, и далее осуществляется передача информации вниз по дереву через маршрутизаторы PE до абонентов.

Общую пропускную способность (BW_{LSP}) туннеля LSP сети VPN можно вычислить аналитически относительно соединения «точна-точка». Если свести многоадресную и широковещательную передачу всех VPN к единице, то пропускную способность можно определить по формуле:

$$BW_{LSP} = K * \left(N * M * 2 + M \sum_{i=1}^{\log_2(N)} 2^{i-1} * (2i-1) \right), \quad (1)$$

где: K – количество корневых маршрутизаторов CR;

N – количество маршрутизаторов DR,

M – количество абонентов.

В (1) определена стоимость репликации одного пакета в одной из вершин графа и пересылка его по туннелю LSP VPN соединения «точка-точка» другим вершинам. Первое слагаемое, N*M*2 определяет стоимость связи от маршрутизаторов DR до маршрутизаторов PE. Второе слагаемое

$$M \sum_{i=1}^{\log_2(N)} 2^{i-1} * (2i-1)$$

определяет стоимость от корня дерева до маршрутизаторов DR.

С другой стороны, если дерево передает информацию по соединению «точка-многоточие» для каждого из туннелей LSP сетей VPN, общая пропускная способность (BW_{mLSP}) может быть представлена в виде:

$$BW_{mLSP} = K * \left(1 + \log_2(N) + \sum_{i=1}^{\log_2(N)} 2^i + M * N\right), \quad (2)$$

где: K – количество корневых маршрутизаторов CR;

N – количество маршрутизаторов DR;

M – количество абонентов.

Слагаемое $1+log_2(N)$ определяет стоимость отправки пакета от маршрутизатора PE к корню дерева «точка-моготочие». Сумма $\sum_{i=1}^{\log_2(N)} 2^i$ определяет стоимости передачи информации от корня дерева всем маршрутизаторам PE, которые входят в состав туннеля LSP сети VPN.

После объединения сетей VPN в дерево распределения, изменяется только последнее слагаемое в (2), которое принимает вид $Z * N$, где Z ($Z \geq M$) обозначает среднее количество вершин, которые зависят от маршрутизатора DR в «точка-многоточие» туннеля LSP, после равномерного распределения K деревьев на W групповых деревьев ($W \leq K$).

Увеличение пропускной способности (BW_{abLSP}) возникает при условии, когда объединенное дерево VPN A не содержит в себе маршрутозаторов PE из группы VPN B. При этом передача информации через такой маршрутизатор PE считается бесполезной. В этой ситуации, маршрутизаторы DR поддерживают количество предающих связей равное $W * Z$. Необходимо отметить, что $M \leq Z \leq L$.

$$BW_{abLSP} = K * \left(1 + \log_2(N) + \sum_{i=1}^{\log_2(N)} 2^i + Z * W\right), \quad (3)$$

где: K – количество корневых маршрутизаторов CR;

N – количество маршрутизаторов DR;

Z – среднее количество вершин, которые зависят от маршрутизатора DR;

W – количество групповых деревьев.

В результате значение пропускной способности ($BW_{(ab)}$), которое не эффективно используется во время передачи от абонента AMb VPN сети A всем абонентам VPN сети B можно определить как разность (2) от (3), представленное в виде:

$$BW_{(ab)} = K * (M * N - Z * W) \quad (4)$$

Необходимо отметить, что распределение сетей VPN случайным образом является не эффективным для построения больших распределенных систем. Невзирая на то, что пропускная способность зависит от вида топологии сети, необходимо проводить анализ распределения VPN сетей для повышения скорости процесса передачи данных. Таким образом, можно сделать вывод, что анализ общей пропускной способности туннелей LSP сетей VPN во многом зависит от правильной организации распределенной системы.

Моделирование

Для оценки преимущества объединенных деревьев, проведем моделирование распределенной системы при условии, что количество VPN равно 1000. Возьмем по 20 граничных маршрутизаторов для каждого из P маршрутизаторов DR. Расположим сети VPN по случайному закону распределения таким образом, чтобы они были равномерно разделены между всем маршрутизаторами DR. Для большей эффективности будем рассматривать дополнительные параметр - плотность распределения LSP туннелей VPN по маршрутизаторами DR (VPN density). Предположим, что все VPN сети имеют одинаковую плотность. Определим, что количество объединенных деревьев W изменяется в приделах от 1 до 1000. На каждой итерации, значение Z рассчитывается путем группировки случайным образом разных VPN сетей в объединенное дерево, которое в свою очередь относится к другой группе VPN сети в маршрутизаторе DR.

Результаты моделирование представлены на рис. 2. Рисунок 2.а показывает отношения использование пропускной способности от количества объединенных деревьев системы.

Отметим, что параллельные линии отображают состояние пропускной способности при использовании метода передачи информации «точка-точка», а кривые – «точка-многоточие» с использованием групповых деревьев. На рисунке 2.b показано количество исходящих связей маршрутизаторов DR при многоабонентской доставке информации.

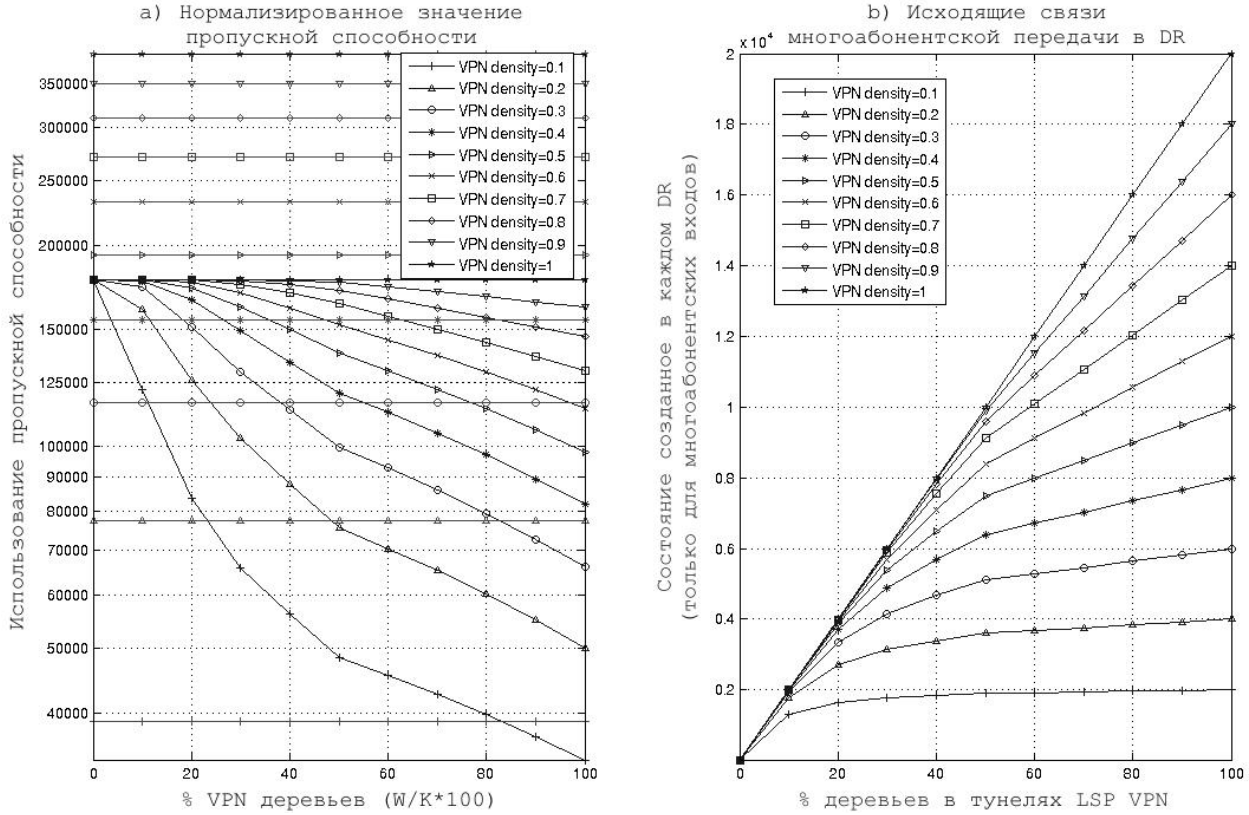


Рис. 2. Оценка пропускной способности и количества исходящих связей в GRID системе с использованием объединенных деревьев на базе туннелей LSP сетей MPLS.

Выводы

Предложенный метод многоабонентской доставки информации позволяет повысить эффективность функционирования GRID систем, за счет более эффективного распределения пропускной способности каналов передачи данных на базе объединенных деревьев доставки информации туннелей LSP сетей MPLS. Проведя анализ результатов моделирования можно сказать, что распределение нагрузки по маршрутизаторам DR, а также балансировка пропускной способности при многоабонентской доставке информации существенно зависит не только от физической, но и от логической организации распределенной системы.

Список литертуры

1. Ian Foster. What is the Grid? A Three Point Checklist, Argonne National Laboratory & University of Chicago, 2002, p.4
2. Столлингс В., Современные компьютерные сети. 2-е изд. – Спб.: Питер, 2003. - 783с.
3. Шиллер Й., Мобильные коммуникации.: Пер. с.англ. - М.: «Вильямс», 2002.-384 с.
4. A.Chaak, “Quality of service and Traffic Engineering in Consolidated Core and Metro Networks”, University of Toronto, September 2004

УДК 681.327.12.001.362

ДОРОГОЙ Я.Ю.,
ЯШИН В.Е.

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ СИМУЛЯЦИИ МНОГОПОТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Данная статья посвящена проблеме создания эффективного программного инструментария для симуляции нейронных сетей. Дан краткий обзор существующих программных решений, выявлен перечень недостатков рассмотренных средств. Описана архитектура разработанного многопоточного симулятора нейросетей, продемонстрирована его работоспособность.

This article deals with the problem of creating an effective software tool for simulating neural networks. A brief overview of existing software solutions was given, revealed a list of the deficiencies of discussed means. An Description of proposed multithread neural networks simulator architecture was given and results of its work.

Общая постановка проблемы

На сегодняшний день нейронные сети находят все большее применение в различных научных областях и промышленных отраслях. Для реализации готовых нейросетевых моделей уже выработаны определенные техники, такие как реализация с помощью элементов ПЛИС, DSP-процессоров, а также спеиализированных высокоскоростных нейропроцессоров. В большинстве этих аппаратных решений обученная выполнять определенную задачу нейросетевая система работает в параллельном режиме, т.е. затраты времени на выполнение задачи очень малы.

В это же время основной проблемой в моделировании и реализации нейросетей остается дизайн сети, выбор оптимальной архитектуры, а также обучение. Очень часто для решения определенной задачи приходится определять оптимальную конфигурацию и параметры нейросетевой системы методом проб и ошибок, проводя большое количество тестов и циклов обучения. Именно обучение нейронной сети на сегодняшний день является самым ресурсоемким и продолжительным по времени процессом.

Для проведения дальнейших исследований в области построения нейронных сетей нам был необходим хороший инструмент — набор программных средств, которые позволили бы с минимальными временными затратами создавать модели нейронных сетей, обучать нейронные сети и проводить их сравнительный анализ.

Обзор существующих решений

В ходе работы были рассмотрены некоторые из существующих ныне популярных программных решений поставленной проблемы. Среди них: Neural Network Toolbox от Matlab, пакет SNNS и библиотека Joone. Рассмотрим каждое решение подробней.

Пакет Matlab [1]. В состав программного комплекса Matlab входит популярный пакет neural network toolbox. Этот пакет позволяет достаточно быстро смоделировать нейронную сеть, провести ее обучение несколькими алгоритмами. Кроме того, в состав пакета Matlab входят и другие пакеты, которые позволяют проводить анализ, пре- и постобработку данных, подаваемых на вход нейронной сети и получаемых на ее выходе (например анализ главных компонентов(PCA), усреднение входных данных и др.). Но к сожалению, Neural Network Toolbox не дает требуемой гибкости. Он оперирует нейронной сетью, как набором слоев, что не позволяет смоделировать сложные типы сетей, не предусмотренные авторами пакета, такие как сверточные сети и неокогнитрон. Модификация пакета представляет слишком сложную задачу, т.к. требует значительного изменения большинства входящих в пакет модулей. Поэтому от использования этого пакета как основного средства моделирования и анализа нам пришлось отказаться.

Пакет SNNS. Пакет SNNS представляет собой мощную библиотеку, изначально разработанную в Штуттгардском университете. Ее основное назначение — анализ и моделирование нейросетей практически любой топологии. К сожалению, исходный код библиотеки SNNS является довольно сложным для моди-

фикации и не позволяет в приемлимые сроки выполнить его оптимизацию для современных аппаратных платформ, предоставляющих возможность параллельного выполнения нескольких процессов.

Библиотека Joone [2]. Библиотека Joone может использоваться для создания наглядных, простых моделей нейронных сетей, но библиотека спроектирована с допущением, что все нейронные сети состоят из слоев. Также предусмотрен только 1 алгоритм обучения.

Ограничения рассмотренных пакетов, а также их низкая производительность сыграли решающую роль в решении создать собственную реализацию библиотеки для симуляции нейронных сетей произвольной топологии.

Постановка задачи

Цель данной работы – разработка программного обеспечения для симуляции нейронных сетей с устраненными ограничениями, найденными в рассмотренных выше пакетах, а также повышение общей производительности обучения и тестирования нейронных сетей за счет использования многопоточной архитектуры.

Архитектура библиотеки

Библиотека нейросетевой симуляции PANN предполагает моделирование сети произвольной топологии. Пользователь сам задает количество входных нейронов, сам соединяет нейроны между собой. В получившейся структуре библиотека автоматически производит сегментацию на слои с использованием модифицированного волнового алгоритма, для дальнейшего разделения нейронов сети между рабочими потоками. Следует отметить, что архитектура сети не обязательно должна быть слоистой.

После подачи в сеть входного сигнала происходит его прямое распространение до выходных нейронов. Библиотека подразумевает, что в зависимости от алгоритма обучения, над каждым отдельно взятым нейроном проводятся различные действия. Объект, который выполняет над каждым нейроном необходимые действия передается сети как параметр. Таким образом, каждый алгоритм обучения или алгоритм прямого распространиния может сохранять в нейроне свой набор параметров, посвоему обрабатывать входы нейрона, вычислять значение функции активации и формировать выход. За счет этого достигается огромная гибкость комплекса, выигрыш в скорости.

Также в состав библиотеки входят модули для создания различных топологий нейронных сетей, алгоритмы обучения, модули формирования входных данных (могут подавать данные на вход сети как результат вычисления некоторой функции, либо после чтения набора данных или графического изображения из файлов на внешнем носителе).

Существует возможность сохранения существующей обученной нейронной сети на диск для последующей загрузки и продолжения обучения. Данные сохраняются в формате XML, в бинарном виде или как текстовый файл, что позволяет с легкостью переносить существующие модели нейронных сетей в другие программы.

Отдельно стоит рассмотреть вспомогательную модуль библиотеки – визуализатор топологии нейронной сети – **pann_viewer.** Довольно часто возникает необходимость визуализации созданной нейронной сети, особенно если ее части была сгенерирована автоматически. Для этого был разработан вспомогательный программный модуль, задачей которой является визуализация модели нейронной сети. Модуль автоматически визуализирует нейронную сеть из ранее сохраненного файла, позволяет вращать модель сети, масштабировать изображение и пр.

Для проверки работоспособности разработанной библиотеки было проведено множество различных экспериментов. Опишем два из них.

Компьютерный эксперимент 1. Для демонстрации возможностей созданного программного комплекса, был проведен ряд симуляций. Использованная в данном опыте нейронная сеть – многослойный персептрон с конфигурацией 1 – 9 – 4 – 1. Для скрытых нейронов использовалась модифицированная функция гиперболичес-кого тангенса: y = 1.7179*tanh(2/3 * x). Были сгенерированы 2 набора данных — для функций y = x^2, на интервале [-1;+1] и y = sin(x), на интервале [-3;+3]. Каждый набор данных включал 20 точек. Алгоритм обучения – метод обратного распространения ошибки. На рисунке 1 показана 3D модель нейронной сети эксперимента 1 (построена при помощи модуль визуализации pann_viewer):

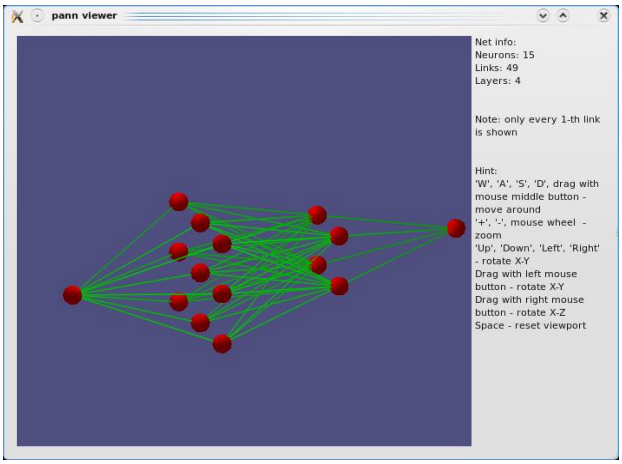


Рис. 1. 3D модель нейронной сети эксперимента 1

На рисунках 2 и 3 показаны графики распределения ошибки для двух функций (по оси абсцис указан номер эпохи обучения, по оси ординат – абсолютное значение усредненной среднеквадратической ошибки нейронной сети за данную эпоху):

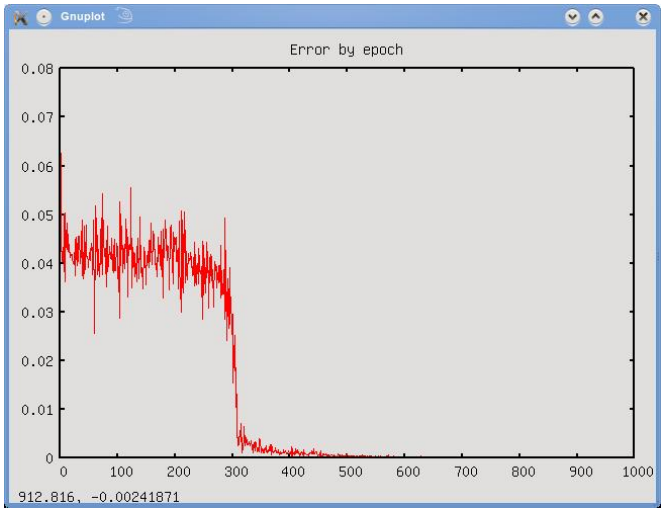


Рис. 2. Распределение ошибки обучения сети для функции y=x2 на интервале [-1;+1]

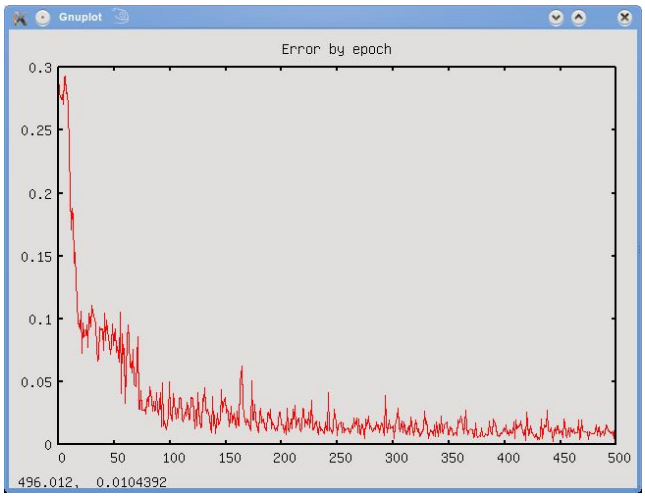


Рис. 3. Распределение ошибки обучения сети для функции y=sin(x) на интервале [-3;+3]

На рисунках 4 и 5 представлены результаты тестирования обученной нейронной сети:

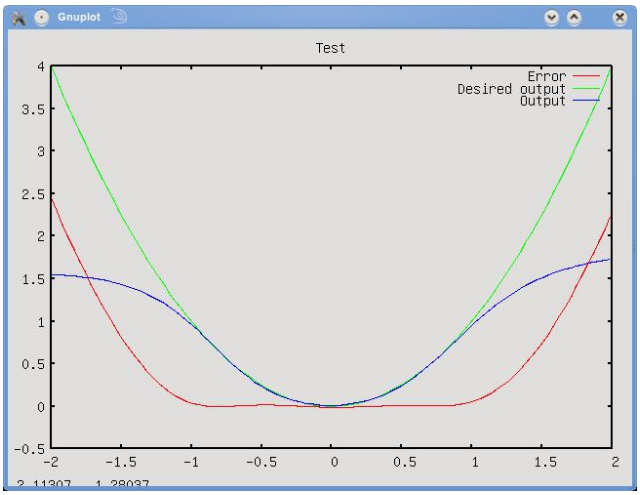


Рис. 4. Тестирование обученной нейронной сети для функции $y=x^2$ на интервале [-1;+1]

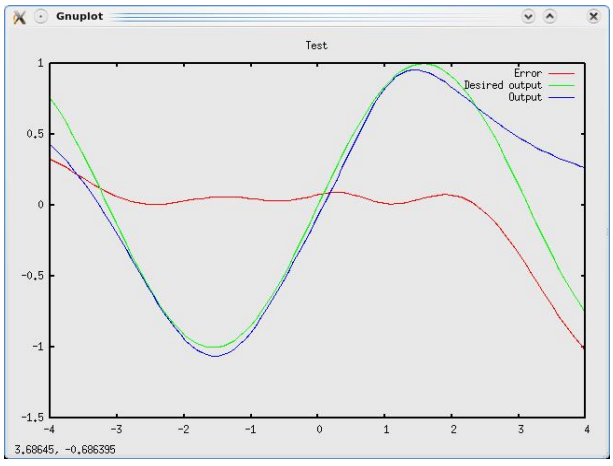


Рис. 5. Тестирование обученной нейронной сети для функции y=sin(x) на интервале [-3;+3]

Таким образом, обученная на примерах нейронная сеть демонстрирует правильную работу используемых алгоритмов обучения.

Компьютерный эксперимент 2. Далее приведены результаты нескольких симуляций, выполненных на многопроцессорной системе с различным количеством используемых рабочих потоков. Эти тесты позволяют количественно оценить преимущество, полученное в результате оптимизации библиотеки для работы в режиме нескольких потоков.

Используемая для симуляции нейроная сеть – многослойный персептрон с конфигурацией 1 – 1000 – 1000 – 1000 – 1. Проводились замеры времени обучения такой нейронной сети для задачи предсказания значения функции на заданном интервале с использованием от 1 до 8 параллельно выполняемых потоков команд.

Для тестов использовалась машины с 8 процессорами с тактовой частотой 2.0ГГц (машина 1) и 4 процессорами с тактовой частотой 3.2 ГГц (машина 2). В ходе эксперимента было проведено 10 независимых тестов. Усредненные данные проведенных тестов представлены в таблице 1 и на рисунке 6.

Табл. 1. Результаты эксперимента (время в с)

Кол-во потоков	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Машина 1	89.8	67.9	55.7	50.8	46.7	45.9	47.4	49.2
Машина 2	68.6	40.9	36.5	33.8	35.2	36.0	34.2	34.4

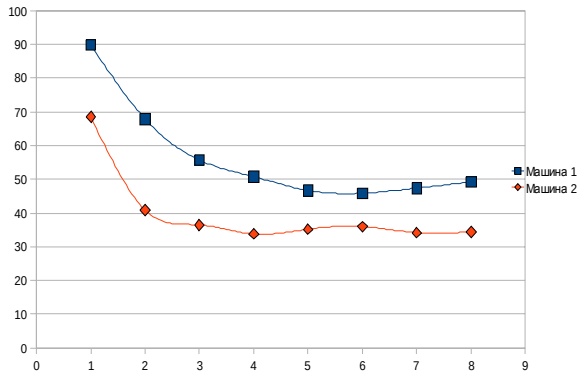


Рис. 6. Время обучения нейронной сети в зависимости от количества потоков

По полученным данным можно сделать следующие выводы: увеличение количества рабочих потоков не дает огромного прироста производительности. Реальное значение прироста сильно зависит от сложности алгоритма обучения. На получаемое преимущество в скорости влияют задержки, связанные с синхронизацией потоков друг с другом, что является неизбежным явлением при многопоточном программировании. Также заметно, что при количестве потоков превышающем реальное количество процессоров, задержки вносимые операционной системой при переключении между несколькими потоками одного процессора становятся достаточно велики, в то время как из-за затрат на синхронизацию потоков отсутствует какой-либо выигрыш.

Резюмируя все вышесказанное, можно сказать, что использование нескольких параллельных потоков действительно дает преимущество по производительности для больших нейронных сетей и довольно сложных алгоритмов обучения. Полученные результаты позволяют рекомендовать разработанное ПО для симуляции нейронных сетей для использования в задачах связанных с распознаванием образов и компьютерным зрением.

Выводы

В процессе работы над библиотекой нейросетевой симуляции был создан программный комплекс для моделирования нейронных сетей произвольной топологии. Спроектированный комплекс обладает следующими отличительными особенностями:

– высокий уровень абстракции (за счет использования объектно-ориентированного языка программирования), что позволяет добавлять или менять функционал библиотеки за короткое время, с минимальной переработкой существующего кода;

– высокий уровень модульности – все алгоритмы обучения, функции активации, модули нейросетевого прототипирования – являются модулями. Таким образом можно легко добавлять новые нейросетевые модели, алгоритмы обучения и т.д., используя существующий простой, но эффективный интерфейс взаимодействия с другими модулями;

– ориентация на современные процессорные архитектуры(SMP). Вся обработка данных выполняется параллельно в несколько потоков, в зависимости от количества присутствующих в системе процессоров. Взаимодействие между отдельными потоками сведено к минимуму для устрания эффекта «гонок» за общие ресурсы, и как следствие – повышение производительности. Данное решение позволяет получить значительный прирост в

скорости при тестировании и обучении нейронных сетей;

– независимость от топологии – возможность моделирвания сетей, связи в которых распределены любым образом, т.е. архитектура сети не обязательно предполагает наличие слоев. Также поддерживаются рекуррентные связи.

Разработанное ПО позволяет выполнить быстрое моделирование обучение и анализ полученных результатов, с задействованием современных многопроцессорных архитектур. Этот инструментарий в дальнейшем будет пополняться новыми нейросетевыми моделями, алгоритмами обучения и др., а его гибкость и расширяемость позволит провести ряд важных исследований архитектур нейронных сетей.

Дальнейшие перспективы развития данного ПО

В дальнейшем предполагается реализация моделей сверточной нейронной сети и компактной ячеистой нейронной сети [3,4] для исследований в области распознавания человека.

Список литературы

1. Hyun-Yeop Lee, Kyung-Il Moon. Fuzzy Logic and Neural Networks Using MATLAB. 2008.
2. Jeff Heaton. Using Joone for Artificial Intelligence Programming. Журнал Developer.com. 2008. – Vol. 2. – P.40-46.
3. Дорогой Я.Ю. Компактные ячеистые сверточные нейронные сети для локализации лица человека.//Збірник тезисів міжнародної науково-практичної конференції «Обробка сигналів і негауссівських процесів», Черкаси, 2007.
4. Дорогой Я.Ю. Применение компактных ячеистых сверточных нейронных сетей для биометрической идентификации человека по лицу.//Вісник НТУУ «КПІ», «Інформатика, управління та обчислювальна техніка», №46. – 2007. – С. 135-149.

УДК 004.3

СТІРЕНКО С.Г.,
ШАУРІН Д.О.

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОБОТИ ІТ ІНФРАСТРУКТУРИ НА ОСНОВІ ТЕХНОЛОГІЇ ВІРТУАЛІЗАЦІЇ

Віртуалізація ІТ інфраструктури – ключова технологія, яка допомагає об'єднати додатки на різних платформах і апаратних засобах попередніх поколінь з використанням меншого числа сучасних, більш потужних серверів з низьким енергоспоживанням. Можливості, що запропоновані, можуть бути істотно розширені за рахунок її використання для задоволення потреб багаторівневого зберігання даних (віртуалізація зберігання), а також для так званої віртуалізації клієнтських місць, яка забезпечує користувачеві доступ до робочих матеріалів з будь-якого терміналу, включаючи територіально видалені. У роботі розглядається віртуалізація рівнів ІТ інфраструктури ВУЗу: сервери, системи зберігання, робочі місця клієнта, інфраструктура центру обробки даних.

Virtualization is a key technology which helps to unite applications on various platforms and hardware of the previous generations with use of smaller number of modern, more powerful servers with low energy consumption. Now, the opportunities offered by this technology can be essentially expanded by its using for satisfaction of requirements of a multilevel data storage (storage virtualization), and also for so-called client place virtualization which provides to the user access to working materials from any terminal, including territorially remote. In the article it is considered virtualization of all levels of IT infrastructure: servers, systems of storage, workplaces of a client, an infrastructure of a data processing centre.

Вступ

Сучасна ІТ-інфраструктура дозволяє гнучко розподіляти ресурси і ефективніше їх використовувати. Підхід, що пропонується, має тенденцію розвитку убік сервіс-орієнтованої ІТ-інфраструктури. Діяльність ВУЗу вимагає від ІТ гнучкості, високої якості обслуговування і ефективності, а ці запити можна задовольнити за допомогою сервіс-орієнтованої архітектури (SOA) і сервіс-орієнтованої ІТ-інфраструктури (SOI). Перша дозволяє об'єднати сервіси, що надаються додатками, для підтримки учбово-виробничих процесів, а друга – динамічно виділяти додаткам ресурси.

Основними характеристиками SOI є: автоматичне призначення ресурсів додаткам; інтегроване управління і моніторинг ресурсів; стандартизація, що забезпечує сумісність, і автоматизація ІТ-операцій.

Стратегія SOI втілена в концепції динамічного центру даних (DDC). Її реалізація передбачає віртуалізацію ресурсів, їх динамічний розподіл і інтеграцію стандартних рішень і ІТ- сервісів.

Для динамічних ІТ-сервісів пропонується створення віртуального серверного пулу, в якому ресурси організовані за допомогою спеціальної архітектури. Вона наділяє це рішення всіма атрибутами SOA для використання в центрах обробки даних (ЦОД) нового покоління. Максимальна утилізація продуктивності, що надається новими процесорами, краще всього досягається за допомогою віртуалізації. Віртуалізація є ефективною, якщо її підтримують всі системи. Такий підхід гарантує, що у віртуальному середовищі додатки працюватимуть з високою швидкодією.

Віртуалізація в будь-якій своєї реалізації належить до найбільш перспективних і вигідних рішень в області ІТ індустрії. Технологія віртуалізації, що дозволяє абстрагувати один від одного різні компоненти ІТ-систем, давно використовується для консолідації серверних ресурсів, в системах зберігання даних, а також для створення клієнтських робочих місць, що дозволяють забезпечити користувачеві доступ до робочих матеріалів з будь-якого терміналу, включаючи територіально видалені. Окрім консолідації, серверна віртуалізація може застосовується для здійснення роботи застарілих застосувань, здатних функціонувати тільки як «гостьові» на сучасному устаткуванні.

Іншою сферою застосування віртуалізації є розробка технології тонких терміналів, які віртуалізують робочі місця користувачів настільних систем, дозволяючи здійснювати доступ до робочого середовища з будь-якої географічної точки, причому робоче середовище зберігає стан, в якому вона перебувала в той момент, коли користувач працював з нею останній раз.

Така технологія фактично «відчеплює» користувача від конкретного фізичного персонального комп'ютера, зберігаючи всі його файли і додатки на одному центральному сервері.

Але основним завданням віртуалізації повинна стати консолідація всіх комп'ютерних ресурсів організації в єдиний резервуар ресурсів ЦОД. Тоді вдасться істотно збільшити коефіцієнт корисного використання ЦОД і підвищити ефективність інвестицій вкладених в його створення.

Застосування технології на рівні ВУЗу

У НТУУ «КПІ» навчається близько 41 тис. студентів [10]. При такій кількості потенційних користувачів необхідний центр, де будуть сконцентровані основні обчислювальні ресурси, які поки що розподілені по всій території університету. Тут важливо заощадити як робочий час адміністраторів, так і місце в серверній кімнаті. Тримати для розміщення сайту кожної кафедри по окремому потужному серверу нераціонально.

Крім того, існує проблема балансування навантаження між всіма фізичними серверами, що існують в організації. Як правило, здійснити таке балансування раціональним чином украй важко, і, як наслідок, частина серверів виявляється переобтяженою, тоді як існує велика кількість малозавантажених серверів, потужності яких використовуються на 5-10%. (Згідно статистиці, середній рівень завантаження процесорних потужностей у серверів на Windows не перевищує 10%, а у Unix-систем даний показник не більше 30%.)

Зростання парку серверного устаткування приводить до істотного збільшення витрат, пов'язаних з оплатою споживаної ними енергії, а також оплатою енергії, що витрачається на охолоджування серверів, яка, у міру збільшення потужності комп'ютерного устаткування, безперервно зростає. Збільшуються витрати на знаходження і зміст нових серверних приміщень, покупку ліцензій на серверну ОС.

Все вищевикладене приводить до необхідності мінімізації кількості фізичних серверів в організації і підвищення ефективності їх використання. Вирішити ці завдання дозволяє використання технології віртуалізації.

Один з підходів – шлях контейнерної віртуалізації. Вона дозволяє запускати на одному фізичному сервері безліч віртуальних, що дуже актуально для учбового процесу. Контейнерна віртуалізація забезпечує роботу в одній ОС (Windows, Linux, Solaris) декількох віртуальних сесій, які один одного «не бачать». Їх ізоляція йде не на рівні перехоплення звернення до апаратури, як в технології гіпервізора, а на рівні операційної системи.

Віртуальні машини (ВМ) є повністю ізольованими програмними контейнерами, здатними працювати з власною операційною системою і додатками, як фізичний комп'ютер. Віртуальна машина працює абсолютно так само, як фізичний комп'ютер, і містить власні віртуальні (тобто програмні) процесор, оперативну пам'ять, жорсткий диск і мережеву інтерфейсну карту. Операційна система, додатки і інші комп'ютери в мережі не здатні відрізнити віртуальну машину від фізичного комп'ютера. Навіть сама віртуальна машина вважає себе матеріально існуючим комп'ютером. Проте, він складається виключно з програмного забезпечення і абсолютно не містить апаратних компонентів. Тому ВМ володіють рядом істотних переваг в порівнянні з фізичними серверами.

При такій організації виділяються декілька переваг:

1. Зростає щільність розміщення віртуальних серверів (контейнерів), оскільки в кожному з них знаходиться не повноцінна ОС, а лише процеси, індивідуальні для даного користувача.

2. Збільшується продуктивність, оскільки при контейнерній віртуалізації не перехоплюється звернення до апаратури.

3. Така віртуалізація обходиться дешевше, як з погляду ліцензійної підтримки, так і при емуляції апаратних ресурсів (її забезпечує гіпервізор).

4. Віртуальні машини, використовуючи загальні фізичні ресурси одного комп'ютера, залишаються повністю ізольованими один від одного, неначебто вони були окремими фізичними машинами. Наприклад, якщо на одному фізичному сервері запущено чотири ВМ, і одна з них дає збій, це не впливає на доступність трьох машин, що залишилися. Ізольованість – важлива причина набагато вищої доступності і безпеки додатків, що виконуються у віртуальному середовищі, в порівнянні з додатками, що виконуються в стандартній, невіртуалізованій системі.

5. Віртуальні машини є дуже мобільними і зручними в управлінні за рахунок того, що вони об'єднують в єдиному програмному пакеті повний комплект віртуальних апаратних ресурсів,

а також ОС і всі її застосування. Це дозволяє легко переміщати або копіювати ВМ з одного місцеположення в інше так само, як будь-який інший програмний файл. Крім того, віртуальну машину можна зберегти на будь-якому стандартному носієві даних: від компактної карти Flash-памяти USB до кампусових мереж зберігання даних (SAN).

6. Віртуальні машини є повністю незалежними від базового фізичного устаткування, на якому вони працюють. Наприклад, для віртуальної машини з віртуальними компонентами (процесором, мережевою картою, контролером SCSI) можна задати настройки, абсолютно не співпадаючі з фізичними характеристиками базового апаратного забезпечення. Віртуальні машини можуть навіть виконувати різні операційні системи (Windows, Linux і ін.) на одному і тому ж фізичному сервері.

Разом з наявністю безперечних переваг технологія віртуалізації, має ряд недоліків, зв'язаних, наприклад, з неможливістю забезпечити максимальну надійність при консолідації серверів, оскільки між віртуальними машинами немає електричної ізоляції. Так збій в роботі операційної системи хоста приводить до необхідності перезавантажувати все ВМ і їх застосування. Крім того, ПО віртуалізації є достатньо «важким», таким, що вимагає для свого функціонування наявності могутніх серверних конфігурацій. Недоліком контейнерної віртуалізації є те, що всі ВМ вимушені працювати під управлінням однієї і тієї ж версії ОС. Це не дозволяє, наприклад, запустити Linux і Windows на одній машині.

Вибираючи контейнерні технології, одночасно вибирається операційна система, під управлінням якої вони працюватимуть. Оскільки для різних проектів потрібні різні ОС, необхідно користуватися універсальними рішеннями. Як правило, на серверах частіше встановлено ПО для Linux, на клієнтських машинах – для Windows. Можливе використання учбових курсів на платформі Solaris.

Досвід роботи показує, що розгортання одного сервера, в кращому разі, триває близько трьох годин, а на створення контейнера йде від хвилини до декількох десятків хвилин залежно від того, що саме потрібно встановити в нім.

Можливі проекти і результати

Технологія контейнерної віртуалізації дозволить, з одного боку, упорядкувати і раціональніше використовувати обчислювальні потужності ВУЗу, а з іншої – підвищити ефективність їх застосування для потреб конкретних учбових і наукових лабораторій. Ось, наприклад, типовий випадок: студентський проект, для виконання якого необхідний сервер. Купувати або виділяти для цих цілей окрему потужну машину абсолютно нераціонально. Оптимальне рішення – створити віртуальний сервер під цей проект.

Один з проектів – реплікація реляційних баз. Співробітники університету спільно із студентами створюють ПО, яке дозволить працювати в єдиному інформаційному середовищі з базами даних, маючи у себе на місцях репліку цієї бази.

Другий проект – застосування даної технології в роботі олімпіади по програмуванню. Вирішення завдань, які підготували учасники олімпіади, викладаються на сервер за допомогою Web-интерфейса. До цього ж сервера підключаються тестуючі комп'ютери, які дістають рішення з черги, компілюють їх і запускають на наборах тестів. Під час олімпіади тестування проводиться на традиційних персональних комп'ютерах, під час очного туру для тестування декілька машин, а то і цілий термінальний клас. Проте під час тренувань в ролі тестуючих комп'ютерів використовуються контейнери під Windows.

Сам сайт олімпіади, база даних, обслуговування черги рішень, визначення рейтингу учасників – все це може працювати на віртуальній машині. Щоб забезпечити обчислювальні потужності для олімпіади, може бути задіяним додатковий сервер з університетського обчислювального кластера. Віртуальний сервер олімпіадної системи переноситься на нього за допомогою засобів міграції протягом декількох хвилин.

Факультети зазвичай мають декілька серверів під управлінням Linux і Windows (все це без віртуалізації). Можлива консолідація всього цього господарства на двох або трьох серверах.

Технології віртуалізації може бути використана в учбовому процесі, студенти інженерних факультетів працюють в розподіленому паралельному середовищі з пакетом MATLAB, а студенти факультетів ІТ займаються в ній програмуванням. Лабораторії ж можуть бути оснащені «тонкими клієнтами».

Перспективи

У найближчих планах може бути використання напрацювань по віртуалізації в учбових класах в режимі «тонкого клієнта». Наступним кроком могла б стати консолідація ресурсів на базі технологій контейнерної віртуалізації в масштабах учбових корпусів університету, а згодом – студгородка. І у факультетів і у інститутів є власні обчислювальні мінікластери, але поки вони працюють автономно і обчислювальні ресурси між ними не перерозподіляються. Факультетські кластери використовуються в учбовому процесі, на них працюють студенти. Зрозуміло, що давати студентам доступ в загальний кластер ВУЗу з їх учбовими програмами недоцільно, необхідно розмежувати і перерозподілити доступ до кластерів.

Роботи по впровадженню технологій віртуалізації повинні йти в загальному руслі розвитку інформаційних технологій університету. Повинен бути створений єдиний віртуальний учбовий простір університету, основу якого складуть, в інтеграції з Українським інститутом інформаційних технологій в освіті [1], інструментальні портали підготовки і доставки електронних засобів навчання, сервери тестування знань, спеціалізовані системи відеоконференцій для дистанційного проведення лекцій, бібліотеки і сховища учбових матеріалів. Все це об'єднується в єдине сервіс-орієнтоване середовище.

Приклад побудови віртуального обчислювального GRID-середовища

Кластер – група комп'ютерів, об'єднаних високошвидкісними каналами зв'язку, що з погляду користувача є єдиним апаратним ресурсом [2]. У сучасному світу існує велика безліч обчислювальних центрів. Продуктивність сучасних суперкомп'ютерів досягає 1.2 PetaFlops [3]. Для забезпечення працездатності таких комп'ютерів потрібні потужні системи охолоджування, висококваліфікований персонал і значні фінансові витрати. Але і до цього дня процесори персональних комп'ютерів простоюють більше 90% часу.

Такі проекти як Folding@home, Seti@home, mFluids@home засновані на ядрі BOINC, яке дозволяє організувати розподілені обчислення використовуючи ресурси персональних комп'ютерів. Але ці проекти вузькоспеціалізовані і не дають можливості звичайному користувачеві запускати свої завдання.

Запропонована нижче система дозволяє використовувати ресурси існуючого устаткування без додаткових фінансових витрат. Обчислювальними вузлами є віртуальні машини, що встановлені на персональних комп'ютерах. Можливості, бібліотеки і компілятори віртуального кластера ідентичні реальним кластерам, різниця полягає лише в продуктивності, відмовостійкості і ціні.

Структурна організація системи

Для функціонування системи на кожному персональному комп'ютері потрібно встановити віртуальний сервер, що буде забезпечувати роботу віртуальних машин, і доступ до основного сервера кластера по мережі, який повинен мати статичну адресу IP. Для підвищення відмовостійкої бажано, щоб головний вузол був встановлений не на віртуальній машині. Основний сервер і обчислювальні вузли можуть бути розташовані в різних локальних мережах, підключатися до головного вузла через HTTP proxy, NAT або WAN.

Вузли кластера підключаються до провідного вузла за допомогою віртуального каналу OPENVPN [4] і організовують віртуальну мережу з повнозв'язною топологією. Таке з'єднання має можливості стискування і шифрування трафіку, а також є відмовостійким. При нетривалих розривах зв'язку між головним і обчислювальним вузлом кластера OPENVPN не припиняє з'єднання і, при нагоді, відновлює всі зупинені з'єднання прозоро для користувача. Але тільки за допомогою OPENVPN організувати взаємодію вузлів з використанням MPI не можливо із-за помилки, що виникає в модулі btl (MPI point-to-point byte transfer layer). Для усунення цієї помилки на основі тунелів OPENVPN налаштовуються IP-in-IP тунелі (RFC2003), через які і відбувається обмін даними між вузлами кластера. За допомогою IP-in-IP тунелів можна налаштовувати будь-які топології. Повнозв'язна топологія вимагає $\frac{N(N-1)}{2}$ тунелів, гіперкубічна – $N \cdot 2^{N-1}$ IP-in-IP тунелів, замкнуті грати (MESH) – $2N$ і так далі. Кількість тунелів визначається кількістю зв'язків (ребер) в системі. Приклади настройки IP-in-IP тунелів представлені в [5]. Доцільно написати shell-сценарій, який за допомогою SSH підключення виконуватиме на вузлах кластера необхідні команди.

Варіанти реалізації

Систему можна умовно розділити на 5 рівнів: хост операційна система, віртуальна машина, гостьова операційна система, VPN, засоби розпаралелювання.

У якості хост системи може виступати будь-яка ОС, в якій є або можуть бути встановлені засоби віртуалізації. Для Windows є такі віртуальні машини як HYPER-V, VMware, Virtual Server, MS VIRTUALPC. Зокрема, VMware і HYPER-V підтримують технологію Intel VT, яка істотно збільшує продуктивність віртуальних машин. Для Unix систем є Xen, OPENVZ, KVM, VIRTUALBOX та інші.

Гостьовою ОС може бути FREEBSD [6], RHL 5, Debian [7] і будь-яка інша Unix-сумісна ОС, в якій є можливість встановити всі необхідні користувачам кластера бібліотеки. Рекомендується все ж таки використовувати один дистрибутив для всіх вузлів. Це спрощує процес налаштування і зменшує ризик виявлення підводних каменів.

Завданням VPN є забезпечення працездатності, відмовостійкості і високої продуктивності віртуальної мережі, що організована між вузлами кластера. З цим завданням повністю справляється OPENVPN. До того ж в нім є можливість підключення через HTTP proxy. Істотно меншу відмовостійкість забезпечує mpd і в нім немає підтримки підключення через проксі. У разі проблем із зв'язком mpd розриває з'єднання, а разом з ним завершуються і всі запущені процеси MPI.

Засобами розпаралелювання є такі відомі бібліотеки як MPI, OPENMP, PVM. Непогано зарекомендували себе MPICH і OPENMPI реалізації бібліотеки MPI.

Оцінка продуктивності

Для оцінки продуктивності системи був створений віртуальний кластер на базі лабораторій кафедри обчислювальної техніки НТУУ «КПІ». Хост ОС – Windows, гостьова ОС – FREEBSD, 100Mbit мережеві адаптери, середовище розпаралелювання – OPENMPI. Елементи кластера розташовані в різних лабораторіях, що відповідає WAN. Тестування проводилося за допомогою системи тестів mpi-bench-suite, розроблених в НДОЦ МДУ [7].

На рис. 1 і 2 та у таблиці представлені результати тестування пропускної спроможності мережі і ефективності основних операцій MPI. Детальніше з тестом mpi-bench-suite можна ознайомитися в [8].

Табл. 1.

Топологія	Максимальна швидкість обміну Мб/с	Операція MPI	Максимальна швидкість обміну Мб/с
Star	2,19	Reduce	1,697
Star2	2,19	Allreduce	4,744
Chaos	2,06	Broadcast	1,394
Chaos2	2,14	Gather	1,69
Ring	1,95	Allgather	6,23
Ring2	2,13	All-to-all	6,39
		Isend & Wait	0,628
		Send	0,627
		Sendrecv	0,994
		Send & Receive	1,25

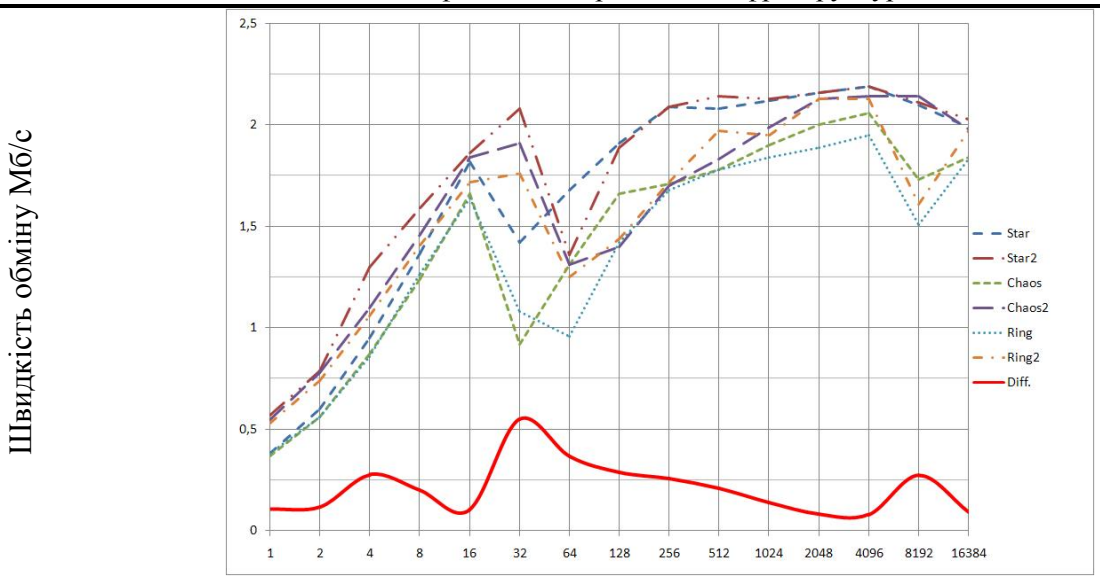


Рис. 1. Тест пропускної спроможності мережі різних топологій.

Star – топологія «Зірка». Використовуються блокуючі пересилки MPI_SEND(RECIEVE).

Star2 – топологія «Зірка». Використовуються неблокуючі пересилки MPI_ISEND(IRECIEVE).

Chaos – повнозв'язна топологія. Блокуючі пересилки.

Chaos2 – повнозв'язна топологія. Неблокуючі пересилки.

Ring – топологія «Кільце». Блокуючі пересилки.

Ring2 – топологія «Кільце». Неблокуючі пересилки.

Difference – різниця між швидкостями пересилок різних топологій.

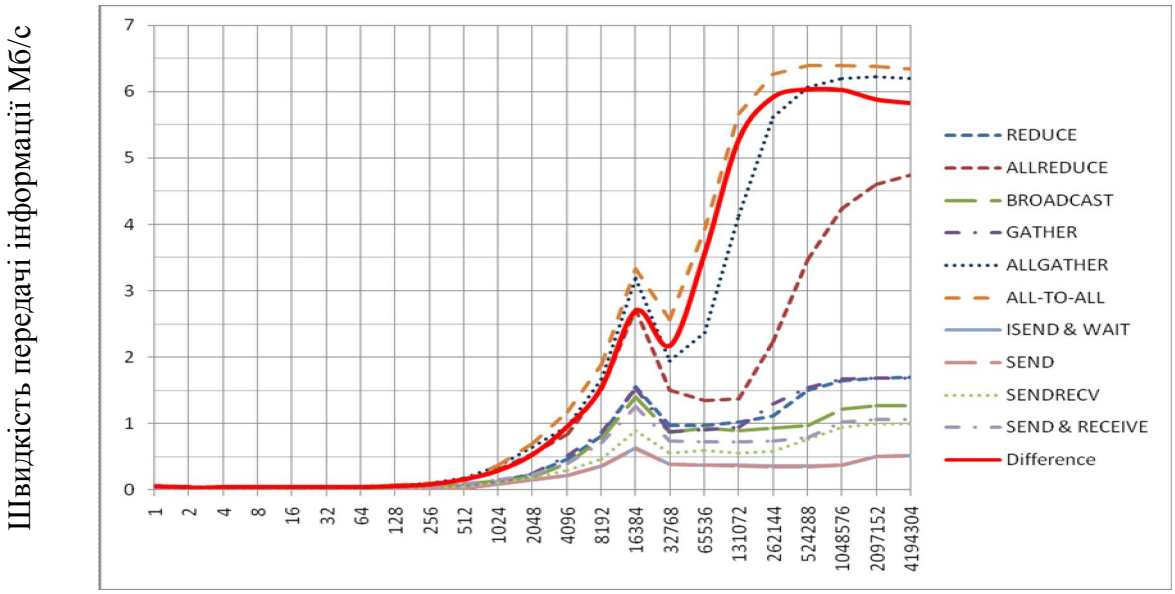


Рис. 2. Тест основних операцій MPI.

ALLREDUCE – підсумовування N цілочисельних змінних по всіх вузлах з розсилкою результату по всіх вузлах.

REDUCE – підсумовування N цілочисельних змінних по всіх вузлах, результат на одному вузлі

BROADCAST – розсилка N цілочисельних значень з одного вузла по всіх вузлах.

GATHER – збір N цілочисельних значень зі всіх процесів на один головний процес; в результаті, головний процес приймає $N*(np–1)$ цілочисельних значень.

ALLGATHER – Кожен процес посилає всім рештою N цілочисельних значень; в результаті, кожен процес посилає N, а приймає $N*(np–1)$ цілочисельних значень; цю операцію можна реалізувати як GATHER (збір). а потім BCAST (розсилка).

ALL-TO-ALL – Кожен процес посилає кожному по N цілочисельних значень; в результаті, ко-

жен процес посилає і приймає по $N*(P-1)$ цілочисельних значень, де P – кількість процесів.

ISEND & WAIT – Не блокуюча посилка N цілочисельних значень від одного процесу іншому з очікуванням завершення.

BLOCKING SEND – блокуюча пересилка N цілочисельних значень від одного процесу іншому цілочисельних значень.

SENDRECV – операція посилки і отримання N цілочисельних значень.

SEND & RECV – посилка, а потім отримання N цілочисельних значень.

Difference – різниця між швидкостями пересилок при різних операціях MPI.

Операція синхронізації всіх процесів (MPI_Barrier) виконується за 2,775 секунд, при цьому латентність рівна 1,05 секунд.

Докладніше програмування з використанням бібліотеки MPI описано в [9].

Розглянутий підхід дозволяє створювати віртуальні кластера як в межах локальної або WAN мережі, так і через інтернет, завдяки можливостям OPENVPN. Перевагами таких класстерів є низька вартість і відсутність проблем з габаритами системи. Як недоліки слід відмітити відносно невисоку надійність вузлів, велику латентність, обмежену пропускну спроможність мережі. Слідуючи перерахованим способам, забезпечується ізоляція хост-ситеми від гостьових ОС і надаються зручні засоби адміністрування кластера через веб-сервер-доступ до вузлів системи, користувачі можуть підключатися по протоколу SSH і запускати завдання за допомогою команди mpirun або за допомогою менеджера ресурсів slurm, PBS і та інш.

Висновки

Таким чином, використання технології віртуалізації дозволяє вирішити багато завдань по вдосконаленню IT-інфраструктури ВУЗу. Рішення, що існують на даний момент, для віртуалізації серверних ресурсів, систем зберігання даних, робочих місць клієнта дають можливість істотно поліпшити ефективність використання устаткування і понизити витрати на його обслуговування.

Проте слід пам'ятати, що ефект від впровадження віртуалізації може гарантувати тільки грамотна оцінка ситуації, що склалася з інформационно-обчислювальними ресурсами в конкретному підрозділі, а також вибір тих варіантів віртуалізації, які сприятимуть підвищенню ефективності використання ресурсів.

Перелік посилань

1. Інформаційний портал Українського інституту інформаційних технологій в освіті http://www.udec.ntu-kpi.kiev.ua
2. Корнеев В. Параллельные вычислительные системы. – М.: Издательство «Knowledge», 1999. – 320 с.
3. http://top500.org
4. http://openvpn.net
5. Сайт OpenNET. — http://www.opennet.ru/docs
6. FreeBSD manual. — http://www.freebsd.org/docs.html
7. The Linux documentation project. — http://tldp.org
8. Інформаційних портал лабораторії паралельних інформаціних технологій НДОЦ МДУ. – http://parallel.ru
9. Жуков И., Корочкин А. Параллельные и распределённые вычисления. – К.: «Корнейчук», 2005. – 226 с.
10. Інформаційний портал Національного технічного університету України "КПІ" http://kpi.ua/ru/education

УДК 004.057.6

ИЛАРИОНОВ Р.

ПРОГРАММНЫЙ ДРАЙВЕР СВЯЗИ 3D СКАНИРУЮЩЕГО УСТРОЙСТВА С СИСТЕМАМИ КОМПЬЮТЕРНОГО ПРОЕКТИРОВАНИЯ

В работе предложены подходы к оптимизации обработки результатов трехмерного сканирования. Эти подходы основаны на анализе процедур преобразования набора точек в различные формы представления объектов. Предложенные подходы реализованы в системах компьютерного проектирования для машиностроения.

In article the approaches for optimization of results of three-dimension scanning processing has been proposed. Those approaches are based on procedure of processing of sets of three-dimension points to different forms of object representations. Proposed approaches has been realizing on system for computer designing in mechanical engineering.

Введение

Важной сферой практического применения компьютерных технологий являются системы автоматизированного проектирования в машиностроении. Их использование позволяет многократно сократить сроки проектных разработок, повысить качество проекта и оперативно выполнять его модификацию.

Современные системы компьютерного проектирования (CAD -Computer Aided Design) в машиностроении представляют собой сложный комплекс алгоритмических, программных и аппаратных средств. К числу последних относятся устройства двумерного (2D) и трехмерного (3D) сканирования реальных объектов, параметры которых вводятся в CAD. Необходимость ввода описания объектов путем их многомерного сканирования возникает при решении задач создания математических моделей объектов, измерения деталей и заготовок, контроля параметров готовых изделий. При этом сам процесс сканирования и описания 3D-объектов представляет собой сложную задачу.

Проблема совершенствования технологии ввода в системы компьютерного проектирования информации о внешних объектах реального мира является актуальной и практически значимой.

Анализ технологий ввода

Трехмерное (3D) сканирование представляет собой процесс ввода изображений трехмерных объектов и их дискретизации с целью эффективной обработки компьютерных систем. При 3D-сканировании осуществляется снятие рельефа исследуемого объекта. Оно состоит в обхождении поверхности объекта, с использованием специального сенсора, который выдает при этом последовательность координат точек объекта в рамках виртуальной трехмерной системы координат. Пространственные координаты точек получаются путем объединения относительного положения сенсора и его показаний по трем взаимно-перпендикулярным осям.

Построение эффективной модели представления 3D-объектов в компьютерных технологиях представляет собой сложную задачу вычислительной геометрии, которая в значительной степени зависит от целевого назначения и требуемых параметров модели.

К настоящему времени предложено ряд способов аппроксимирующего представления 3D–объектов в компьютерных системах. Одним из наиболее распространенных на практике является представление 3D–объекта в виде множества пространственных треугольников. Для получения такого представления используются такие базовые процедуры: диаграммы Вороной, триангуляция Делоне и процедура построения огибающей фигуры (выпуклого многогранника). Ниже эти процедуры рассмотрены более подробно.

Для построения огибающей фигуры (выпуклого многогранника) используется последовательный алгоритм. Вначале, находятся такие точки, которые подчиняются следующему правилу (рис. 1) [3]. Точки 1 и 2 задают прямую, точка 3 не лежит на ней. Эти три точки определяют поверхность в трехмерном пространстве, точка 4 не должна лежать на ней. В результате, получается тетраэдр, к которому добавляются другие точки, положение которого относительно тетраэдра определяет видимость или невидимость точки. Если добавленная точка расположена внутри тетраэдра, то она не лежит на выпуклой огибающей, то есть является невиди-

мой. В противном случае, точка видимая. Строится пирамида с ребрами от видимой точки до всех видимых вершин уже образовавшейся фигуры (рис. 1), после чего стираются те вершины, которые находятся внутри пирамиды.

Процесс продолжается с объединением двух фигур. В результате получается выпуклый многогранник, от которого удаляются те грани, которые расположены внутри фигуры, а именно те грани, которые являются общими между двумя фигурами. Потом берется следующая точка для добавления и так далее до исчерпывания всех точек. В конечном результате получается выпуклый многогранник на основе множества точек в трехмерном пространстве.

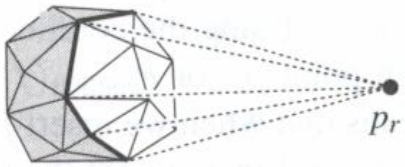


Рис.1. Пример построения выпуклого многогранника по видимым точкам и вершинам.

Диаграмма Вороной представляет собой специальный вид декомпозиции метрического пространства, задаваемая множеством дискретных точек в этом пространстве. Фактически метрическое пространство разделяется на области, каждая из которых включает только одну точку и границы областей находятся на одинаковом расстоянии от двух точек. Пример диаграммы Вороной представлен на рис.2.

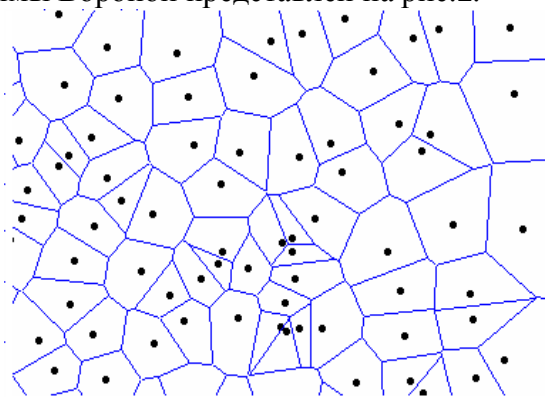


Рис.2. Пример диаграммы Вороной

В самом простом и самом общем случае в плоскости, для заданного множества точек S с координатами X и Y, диаграмма Вороной для S представляет деление на плоскости, которые ассоциируют область V_P с каждой точкой p из S таким образом, что все точки V_P находились ближе к p, чем к любой другой точки из S.

Все области Вороной являются выпуклыми многоугольниками, некоторые из которых бесконечны. Диаграмма Вороной делит плоскость по правилу самого ближнего соседа. Осюда можно заключить, что биссектриса двух сегментов l_i и l_j равна $B_{ij} = \{ x \in R_2 \mid d(x,l_i) = d(x, l_j)\}$, где l_i и l_j точки главного множества, определяющие прямую, а любая клетка $v_i = \{ x \in R_2 \mid d(x, l_i) \le d(x, l_j)$, для любого $j\}$, где d функция расстояния.

Если S конечное множество точек, то граница между двумя соседними областями является симметраллой отрезка между соответствующими двумя точками. Отсюда можно заключить, что три области Вороной определяют точку Вороной, которая является центром окружности, описанной около соответствующих точек, в которую не попадает никакая другая точка из S.

Существуют два вида диаграмм Вороной - статическая и динамическая. Статическая диаграмма исчисляется для наперед заданного множества точек, а динамическая позволяет осуществлять дополнительные операции добавление или стирание точки. над уже существующей диаграммой. В процессе добавления точки, сначала в диаграмму добавляется один конец сегмента, а потом он расширяется путем перемещения точки до другого его конца. При стирании используется операция уменьшения сегмента до точки, после чего эта точка удаляется из диаграммы.

Каждая ячейка Вороной образуется в результате пересечения плоскостей, образованных двумя соседними точками. Любая найденная плоскость пересекается с остальными до тех пор, пока не найдутся такие плоскости, которые огибают все точки.

Триангуляция Делоне DT(p) для множества лежащих в одной плоскости точек p с координатами X_p и Y_p, определяется как такая триангуляция, при которой ни одна точка p не находится в окружности, описанной около любого из треугольников. Триангуляция Делоне позволяет сделать максимально равносторонними все треугольники, которые представляют объемную аппроксимацию объекта [5]. Пример триангуляции Делоне показан на рис.3.

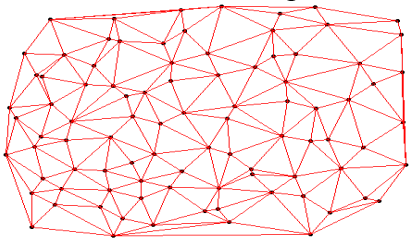


Рис.3. Пример триангуляции Делоне

Триангуляцию Делоне для множества точек можно представить и как набор ребер, удовлет-

воряющую условию „пустых кругов“. Это означает, что для любого ребра можно найти окружность, содержащую конечные точки ребра, но не содержащую других точек (рис.4).

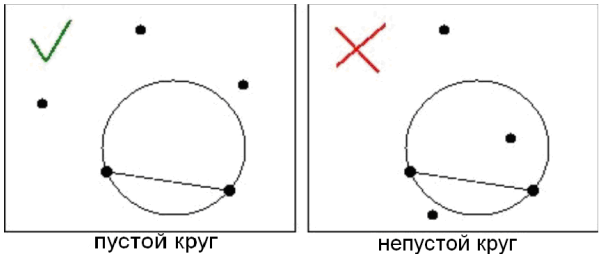


Рис.4. Примеры ребер, удовлетворяющих (слева) и не удовлетворяющих (справа) условию Делоне

Это правило следует из определения Делоне, что описанная окружность формируется из трех точек треугольника. Описанная окружность называется пустой, если она не содержит других точек, кроме трех ее образующих. Все остальные точки множества должны оставаться вне описанной окружности.

Триангуляция Делоне для заданной сети треугольников единственна, если описанные окружности пустые. Это правило верно только для двумерного пространства. Триангуляция Делоне в своем трехмерном варианте представляет совокупность тетраэдров, описанная сфера около которых не содержит ни одной из точек главного множества точек, кроме точек тетраэдра, задающих сферу (фиг. 5) [5].

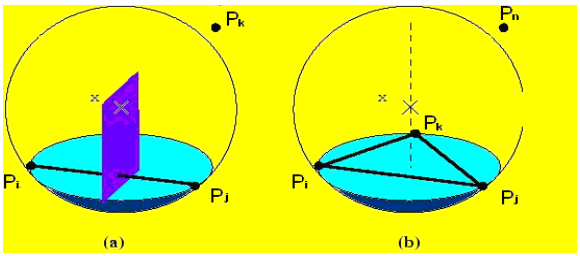


Рис.5. Ребро Делоне (а) и треугольник Делоне (б) в пространстве 3D

Триангуляция Делоне для заданного конечного множества *S* представляет собой планарный граф с вершинами в точках *S* и прямыми ребрами, который максимален в смысле, что нельзя добавить ни одного ребра без пересечения с другим. Каждая триангуляция *S* содержит ребра выпуклой оболочки *S*. Ее ограниченные стенки являются треугольниками. Это подмножество ребер данной триангуляции можно представить и как мозаику (дробление) S, если оно содержит ребра выпуклой оболочки и у любой точки S присутствуют хотя бы два прилежащих ребра. Пример триангуляции Делоне в трехмерном пространстве показан на рис.6.

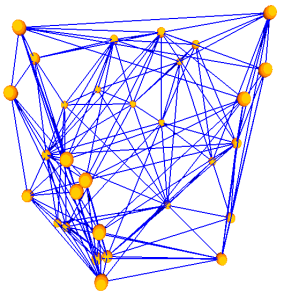


Рис.6. Пример 3D триангуляции Делоне

Все алгоритмы вычисления построения триангуляции Делоне базируются на решении задачи о том, располагается ли данная точка на описанной около треугольника окружности.

Следует отметить дуализм представления трехмерного объекта диаграммами Вороной и триангуляциями Делоне. Это означает, что существуют процедуры, позволяющие переходить от диаграмм Вороной к триангуляции Делоне и обратно [5]. Построение одной структуры, исходя из другой, реализуется очень легко, так как алгоритмы преобразования линейны [6].

Целью работы является повышение эффективности преобразований результатов сканирования трехмерных объектов в пригодные для дальнейшей обработки формальные представления, создание, на этой основе, драйвера 3D лазерного сканера для систем компьютерного проектирования и контроля.

Структура модульной сканирующей системы

На фиг. 7 представлена структура модульной сканирующей системы и ее связь с внешней CAD системой. Связь между двумя системами осуществляется через файл унифицированного формата для описания объектов в пространстве. Потребитель задает команду сканирования фигуры. Система активирует сканирующий модуль и инициирует процесс лазерного 3D сканирования. В результате сканирования получается множество точек, которые сохраняются в памяти.

После этапа сканирования начинает работать модуль создания 3D фигуры по множеству точек. При этом полученные в результате сканирования точки поступают на входной интерфейс модуля.

Рис.7. Структура системы сканирования

На выходе модуль генерирует множество полигонов, сохраняющиеся в памяти. По запросу пользователя система экспортирует сгенерированную полигональную фигуру в унифицированный файловый формат на жесткий диск потребителя. Потребитель имет возможность запустить CAD систему с целью оценки результата сканирования. CAD система реализует не только 3D визуализацию объекта, но и позволяет модифицировать объект. Кроме того возможен экспорт представления объекта другие CAD форматы (STL, DXF).

Программный модуль построения фигуры по точкам реализует алгоритм, представленный на рис.8.

Алгоритм состоит из пяти основных этапов работы, каждый из которых решает определенную комплексную проблему. На вход поступает множество точек с координатами (X, Y, Z), описывающих поверхность объекта. Точки записываются в отдельный массив S.

Первый этап состоит в исчислении диаграммы Вороной по входному множеству точек S. Диаграмма дает в результате список ячеек диаграммы Вороной и точек, к которым эти ячейки соотносятся.

На втором этапе находится выпуклый многогранник по множеству точек S.

Третий этап связан с нахождением полюсов ячейки Вороной. Выполняется итерация для каждой точки из главного множества S. Если точка лежит на выпуклом многограннике, то по построенной диаграмме Вороной находятся ячейки, являющиеся соседними текущей и образуются треугольники из точек, которые закрыты ячейками (триангуляция Делоне). На образовавшихся треугольниках строятся нормальные вектора и находится вектор, средний по отношению их ориентации. Если обрабатываемая точка не лежит на выпуклом многограннике, то находится самая отдаленная вершина ячейки Вороной и образуется вектор от текущей точки до найденной точки. Эта вершина ячейки Вороной отмечается как *p+*. На следующем шаге находится самая отдаленная вершина ячейки, такую, что скалярное произведение по отношению найденного уже вектора отрицательно. Эта вершина отмечается как *p-*. Процесс продолжается до окончания обхода всех точек S.

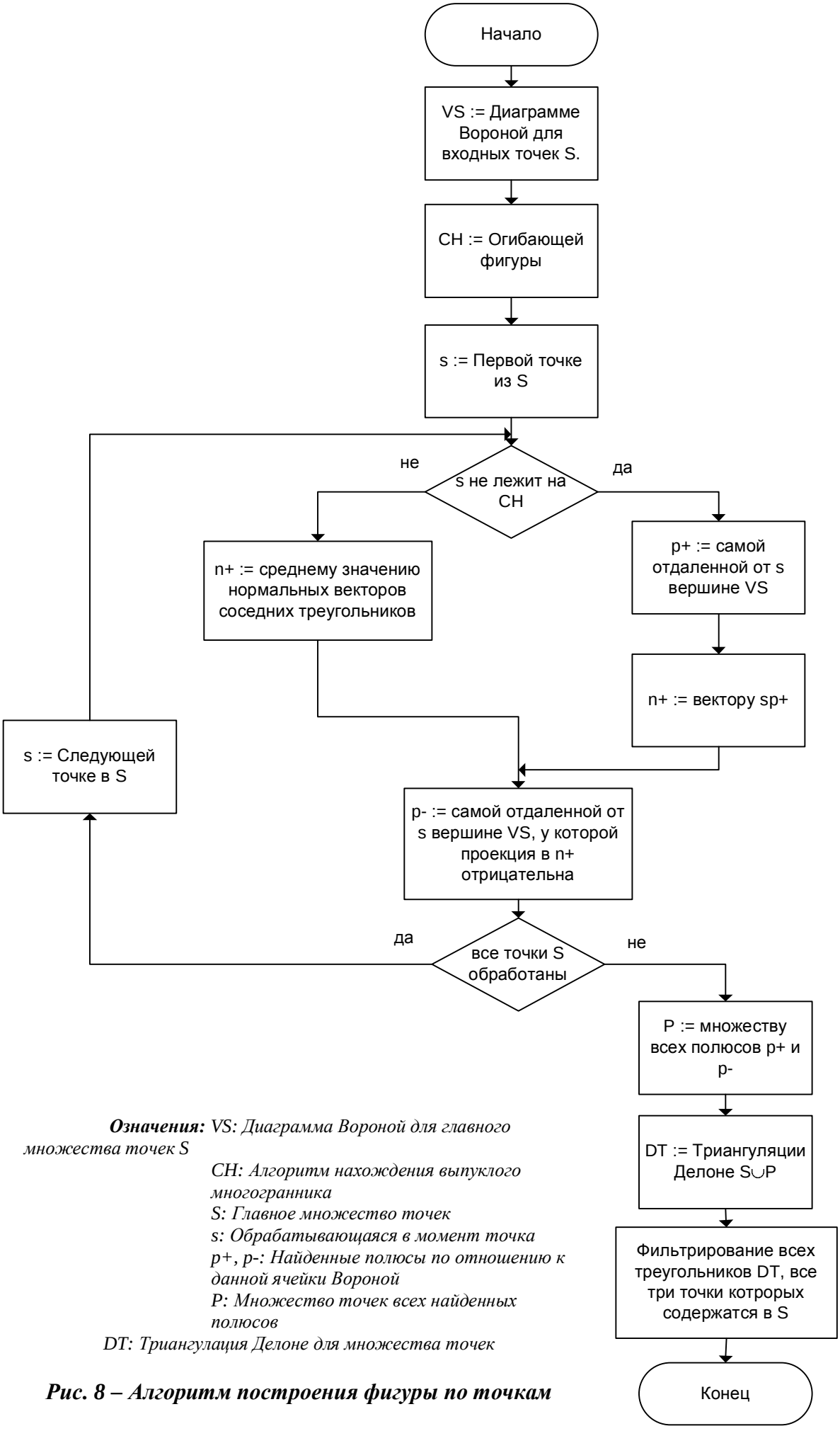
На четвертом этапе вычисляется триангуляция Делоне для главного множества S и найденных полюсов (*p+* и *p-*). Процесс заканчивается после исключения треугольников, все три точки которых являются точками главного множества S.

Таким образом, предложенный алгоритм построения фигуры имеет на входе множество точек, а на выходе генерирует множество треугольников. Рассматриваются два подхода для сохранения дискретных данных о фигуре на выходе.

Подход 1 : Сохраняется структура треугольника, содержащая информацию о трех точках. Недостаток этого подхода состоит в дублировании данных.

Подход 2 : Сохраняется таблица с точками, идентифицируемыми индексом, причем никакая точка в таблице не повторяется. Сохраняется структура треугольника как три числа, которые соотносятся с индексами таблицы и описывают треугольник. Этот подход позволяет значительно уменьшить требуемый объем памяти, но более сложен в реализации.

Информация, полученная на выходе алгоритма построения 3D фигуры, представляется в форме полигонов. Объект 3D составлен из множества полигонов, склеенных один к другому. Хорошо было бы представить формат данных в коде ASCII во избежании необходимости исполнять дополнительные проверки конвенции "endian" для представления чисел - от младшего байта к старшему или от старшего к младшему (little endian, big endian).



Означения: *VS: Диаграмма Вороной для главного множества точек S*

CH: Алгоритм нахождения выпуклого многогранника
S: Главное множество точек
s: Обрабатывающаяся в момент точка
p+, p-: Найденные полюсы по отношению к данной ячейки Вороной
P: Множество точек всех найденных полюсов
DT: Триангулация Делоне для множества точек

Рис. 8 – Алгоритм построения фигуры по точкам

Важным аспектом проекта является создание CAD системы. Создание настоящей системы имеет два отличия: 1) концептуальное – необходимо было отделить процессы сканирования в одну систему, а процесс визуализации – в другую; 2) направленность на облегчение работы потребителя. На рис. 9 показана общая схема CAD системы (среды).

Рис.9. Структура CAD системы

Концептуально в системе заложен множественный документный интерфейс (MDI – Multiple Document Interface). Таким образом, CAD система предлагает возможность работы с множеством открытых документов и множеством видов на документы. Понятие документа используется только для абстракции. Документ, который загружается в CAD систему, является созданный сканирующей системой исходный файл.

После загрузки файла в CAD среду создается документ, который добавляется к списку открытых документов в системе. Во время загрузки создаются две таблицы - таблица точек и таблица полигонов (таблицы представляются в памяти массивами данных). Таблица полигонов содержит индексы точек для соответствующей таблицы. Наличие этих двух таблиц гарантирует отсутствие повторных копий сохраненных данных. Документ содержит как список, содержащий нормальные векторы для каждого полигона, так и исчисление границ фигуры. Нормальные векторы полигонов используются для черчения, потому что эта информация нужна системе визуализации. Все операции, которые исполняются на фигуре, заносятся в конкретный документ, так как он содержит всю информацию о фигуре.

Одна из задач системы CAD состоит в визуализации фигуры. Создание системы, преобразующей трехмерные координаты в двумерные является сложной задачей. Поэтому, в качестве языка управления черчением выбирается OpenGL. Причина такого выбора состоит в том, что этот стандарт является открытым, что упрощает его использование. С точки зрения потребителя над видом объекта возможны операции ротации, масштабирования и трансляции фигуры, что облегчает работу потребителя, так как любая из них осуществляется мышью. Система визуализации использует разные виды – вид только поверхности фигуры, вид с очертанием полигонов, вид с очертанием полигонов и поверхностей, вид одних только точек фигуры. Вид снабжен дополнительными возможностями черчения, как черчение границ фигуры черчение координатной системы. Добавлена возможность простого выбора вида фигуры – вид свыше, вид снизу, вид слева, вид справа, вид спереди, вид сзади.

Выводы

В результате выполненных исследований предложены подходы к оптимизации процессов обработки результатов сканирования трехмерных объектов. На основе этих подходов создана реальная система компьютерного проектирования с использованием 3D-сканера, разработанном совместно Техническим Университетом - Габрово и АМК ЕООД – Габрово. Практическое использование системы доказало эффективность разработанных подходов к оптимизации обработки результатов 3D сканирования.

Список литературы

1. Иларионов, Р. Компютърна периферия, Университетско издателство „В. Априлов”, 2008, Габрово, България.
2. Иларионов, Р. и др. Въвеждане на 3D обекти в изчислителна среда. Сборник с доклади на International Scientific Conference “Uniteh 07”, Volume I, 23-24 ноември 2007, Габрово, България.
3. http://www.cse.ohio-state.edu/~tamaldey/ paper/tcocone/tcocone.pdf
4. http://www.cs.utexas.edu/users/amenta/ pubs/sm.pdf
5. http://interviso.openfmi.net/index.php
6. http://web.mit.edu/manoli/www/ publications/Amenta _Siggraph_98.pdf

УДК 519.85

СЕРАЯ О.В.,
КАТКОВА Т.И.,
БАЧКИР Л.В.

ОЦЕНИВАНИЕ СОСТОЯНИЯ С ИСПОЛЬЗОВАНИЕМ НЕЧЕТКОЙ РЕГРЕССИИ

Для построения нечеткой экспертной системы предложена методика расчета функций принадлежности нечеткой регрессии, вычисляемой по результатам измерения нечетко заданных контролируемых параметров.

For the construction of fuzzy expert system the calculation method of membership functions of fuzzy regression, calculated on results measuring of fuzzy controlled parameters is offered.

Постановка проблемы, анализ последних публикаций

Конструктивно необходимым элементом любой системы поддержки принятия решений является подсистема оценивания состояния среды и объекта, реализующего принимаемые решения. Одно из перспективных направлений совершенствования технологий оценивания состояния состоит в исследовании экспертных систем (ЭС) [1,2]. Такая система преобразует набор измеренных значений $x_1, x_2, \ldots, x_n$ контролируемых параметров объекта в значение y – параметра, оценивающего состояние этого объекта. Если при этом механизм логического вывода системы – продукционный, то процедура преобразования сводится к применению совокупности правил вида

$$\begin{aligned} &\textit{ЕСЛИ}\ \ x_1\ \textit{это}\ A_1, \\ &\qquad x_2\ \textit{это}\ A_2, \\ &\qquad \ldots\ldots\ldots\ldots, \\ &\qquad x_n\ \textit{это}\ A_n, \\ &\textit{ТО}\qquad y\ \textit{это}\ B. \end{aligned} \qquad (1)$$

Точность оценивания состояния в такой системе может быть сделана как угодно высокой и ограничивается только числом контролируемых параметров, точностью их измерения и правильностью заключений, образующих правила (1). В случаях, когда число контролируемых параметров велико, может быть использована декомпозиция продукционного механизма (1) с введением совокупности правил субпродукции. Неопределенность, неизбежно сопровождающая все этапы процедуры оценивания состояния объектов с использованием ЭС, приводит к появлению и все более широкому использованию нечетких ЭС [3-4]. Одной из наиболее известных таких систем является система нечеткого вывода Мамдани – Заде [5]. Применительно к задаче оценки состояния объекта эта система работает следующим образом. Для каждого из возможных состояний объекта $S_1, S_2, \ldots, S_p$ формируются функции принадлежности контролируемых параметров $\mu^{(k)}(x_j), k = 1,2,\ldots,p,\ j = 1,2,\ldots,n,$ диапазону возможных своих значений, определяемому состоянием. В соответствии с этим при получении конкретного набора измеренных значений параметров $X^{(0)} = \{x_1^{(0)}, \ldots, x_n^{(0)}\}$ осуществляется вычисление значений функций принадлежности $\mu^{(k)}(X_j^{(0)}), k = 1,2,\ldots,p,\ j = 1,2,\ldots,n$. Далее значения функций принадлежности, относящихся к каждому из состояний, агрегируются (чаще всего с использованием операции логического суммирования). При этом получают

$$\begin{aligned} &\mu^{(k)}(X^{(0)}) = max\{\mu^{(k)}(x_1^{(0)}), \\ &\mu^{(k)}(x_2^{(0)}), \ldots, \mu^{(k)}(x_n^{(0)})\}, \\ &k = 1,2,\ldots,p. \end{aligned} \qquad (2)$$

Завершающей является операция оценки состояния путем дефуззификации, выполняемая, например, следующим образом:

$$\hat{k} = \frac{\sum_{k=1}^{p} \mu^{(k)}(x^{(0)}) \cdot k}{\sum_{k=1}^{p} \mu^{(k)}(x^{(0)})}. \qquad (3)$$

Представленная в форме (3) процедура получения нечеткого логического вывода обладает принципиальными недостатками, присущими продукционным системам. Среди них наиболее существенными являются следующие.

1. Отсутствует возможность учета различий в важности контролируемых входных переменных $x_1, x_2, ... x_n$.

2. Для каждого из продукционных правил отсутствует возможность учета различий в важности субправил.

3. Жесткая схема логического вывода, реализуемая правилами агрегирования, может привести к неконтролируемым ошибкам в диагнозе (для этого достаточно, чтобы минимальное значение функции принадлежности только для одного из правил субпродукции оказалось больше остальных).

4. Множество правил базы знаний много меньше числа возможных вариантов значений входных переменных. Поэтому на практике могут возникать варианты, не предусмотренные в базе знаний.

5. В системах такого типа, в особенности, если число входных переменных велико, практически невозможно учесть синергетический эффект, который возможен при совместном появлении некоторых конкретных значений отдельных переменных.

Подход, не требующий дефуззификации, реализован в модели нечеткого вывода Такаги-Сугено [6]. В этой модели для каждого из возможных состояний объекта рассчитывается уравнение регрессии

$$y_k = a_{k_0} + a_{k_1} x_1 + a_{k_2} x_2 + ... + a_{k_n} x_n, \quad (4)$$
$$k = 1,2,..., p,$$

связывающее некоторый результирующий параметр y с результатами непосредственных измерений параметров $x_1, x_2, ..., x_n$. При этом коэффициенты (a_{k_j}) в (4) оцениваются статистически. С другой стороны, тот же что и в модели Мамдани – Заде набор функций принадлежности $\mu^{(k)}(x_j)$ используется для формирования весовых коэффициентов W_k, $k = 1,2,..., p$, по правилу

$$W_k = min\{\mu^{(k)}(x_j)\}, \quad k = 1,2,..., p. \quad (5)$$

Теперь с учетом этих коэффициентов рассчитывается оценка достоверности состояния объекта

$$\hat{k} = \sum_{k=1}^{p} \frac{W_k}{\sum_{k=1}^{p} W_k} \cdot y_k. \quad (6)$$

Слабые звенья этой процедуры: необоснованный выбор структуры уравнения регрессии (4) и использование операции логического умножения (5) при расчете весовых коэффициентов.

Другая идея реализуется в предложенной в [7] нечеткой байесовой ЭС. В этой системе нечеткость исходных данных отображается в описании с помощью функций принадлежности $\mu^{(k)}(x_j)$ нечетких значений априорных вероятностей наблюдения значений контролируемых параметров x_j при условии, что объект находится в состоянии S_k. Байесова система преобразует контролируемый набор параметров $X^{(0)}$ с учетом совокупности $\{\mu^{(k)}(x_j)\}$ в набор апостериорных вероятностей $P(S_k / X^{(0)})$, $k = 1,2,..., p$. Понятно, что нечеткость исходных данных навязывает нечеткость результата. К сожалению, в [7] содержится лишь паллиативное решение задачи: получены формулы для расчета носителей нечетких чисел, описывающих апостериорное распределение вероятностей состояний.

Более радикальный подход состоит в получении функций принадлежности нечетких апостериорных вероятностей состояний.

Естественно предположить, что по результатам предварительной обработки реальных данных или экспертного оценивания получены основные статистические характеристики (математическое ожидание и дисперсия) значений априорных вероятностей наблюдения каждого из контролируемых параметров x_j, $j = 1,2,..., n$, при условии, что объект находится в каждом из возможных состояний $S_1, S_2, ..., S_p$, то есть имеются наборы $(m_1^{(k)}, m_2^{(k)}, ..., m_n^{(k)})$, $(D_1^{(k)}, D_2^{(k)}, ..., D_n^{(k)})$, $k = 1,2,..., p$.

В соответствии с формулой Байеса набор апостериорных вероятностей состояний объекта в ситуации, когда в результате контроля получен вектор значений параметров $X^{(0)} = (x_1^{(0)}, ..., x_n^{(0)})$, отыскивается рекуррентно по формуле

$$P\left(S_k \middle/ x_1^{(0)}, x_2^{(0)}, ..., x_r^{(0)}\right) = \frac{P\left(x_r^{(0)} \middle/ S_k\right) \cdot \hat{P}_{r-1}(S_k)}{\sum_{k=1}^{p} P\left(x_r^{(0)} \middle/ S_k\right) \cdot \hat{P}_{r-1}(S_k)},$$

$$\hat{P}_{r-1}(S_k) = P\left(S_k \middle/ x_1^{(0)}, x_2^{(0)}, ..., x_{r-1}^{(0)}\right), \quad (7)$$

$$k = 1,2,..., p,\ r = 1,2,..., n.$$

Для получения функции принадлежности апостериорных вероятностей (7) рассчитаем математическое ожидание и дисперсию числителя и знаменателя в соотношении (7). Для $r = 1$ имеем

$$M\left[P\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)\right] = m_1^{(k)} \cdot P(S_k), \quad (8)$$

$$D\left[P\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)\right] = D_1^{(k)} \cdot P^2(S_k), \quad (9)$$

$$M\left[\sum_{k=1}^{p}\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)\right] = \sum_{k=1}^{p} m_1^{(k)} \cdot P(S_k), \quad (10)$$

$$D\left[\sum_{k=1}^{p}\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)\right] = \sum_{k=1}^{p} D_1^{(k)} \cdot P^2(S_k), \quad (11)$$

Тогда

$$M\left[P\left(S_k \middle/ x_1^{(0)}\right)\right] = M\left[\frac{P\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)}{\sum_{k=1}^{p} P\left(x_1^{(0)} \middle/ S_k\right) \cdot P(S_k)}\right] = \frac{m_1^{(k)} \cdot P(S_k)}{\sum_{k=1}^{p} m_1^{(k)} \cdot P(S_k)}. \quad (12)$$

Для оценки дисперсии величины $P\left(S_k \middle/ x_1^{(0)}\right)$ используем следующее известное соотношение: для независимых случайных величин X и Y

$$D[XY] = D[X]D[Y] + M^2[X] \cdot D[Y] + M^2[Y] \cdot D[X].$$

Тогда, после несложных выкладок, получим

$$D\left[P\left(S_k \middle/ x_1^{(0)}\right)\right] = \left(D_1^{(k)}\right) \cdot P^2(S_k) \times \frac{4\left(D_1^{(k)}\right)}{P^2(S_k)\left[\left(m_1^{(k)}\right)^2 - \frac{D_1^{(k)}}{1-\gamma}\right]^2} + \left(m_1^{(k)}\right)^2 \cdot \frac{4D_1^{(k)}}{\left[\left(m_1^{(k)}\right)^2 - \frac{D_1^{(k)}}{1-\gamma}\right]^2} + \frac{1}{\left(m_1^{(k)}\right)^2} \cdot D_1^{(k)}. \quad (13)$$

Используя соотношения (12) и (13) рекуррентно получим наборы значений математического ожидания $\left(m_A^{(1)}, m_A^{(2)}, ..., m_A^{(p)}\right)$ и дисперсии $\left(D_A^{(1)}, D_A^{(2)}, ..., D_A^{(p)}\right)$ для всех компонентов апостериорного распределения вероятностей состояний объекта. С использованием этих наборов естественно для описания функций принадлежности соответствующих нечетких чисел выбрать гауссовы модели

$$\mu_A\left(P(S_k)\right) = exp\left\{-\frac{\left(P(S_k) - m_A^{(k)}\right)^2}{2D_A^{(k)}}\right\},$$

$$k = 1,2,..., p.$$

Применение байесовой технологии снимает проблемы размерности задачи, однако, принципиальным ее недостатком является необходимость получения объемной статистической информации для предварительного расчета элементов матриц $\left(m_j^{(k)}\right)$, $\left(D_j^{(k)}\right)$, $j = 1,2,..., n$, $k = 1,2,..., p$. Кроме того, понятно, что рекуррентная процедура расчета апостериорных вероятностей состояний приводит к накоплению погрешностей в оценивании априорных параметров задачи, результатом чего могут быть непрогнозируемые ошибки оценки диагноза состояния. В этой ситуации привлекательной представляется идея усовершенствования технологии Такаги-Сугено за счет использования достаточно полного уравнения регрессии (4) и получения соотношений для непосредственного расчета функции принадлежности результата оценивания состояния объекта.

Цель работы состоит в расчете функции принадлежности нечеткого значения экзогенной переменной уравнения регрессии, оценивающего состояние объекта по нечетким значениям контролируемых параметров.

Постановка задачи

Пусть $X=\left(x_1,x_2,...,x_n\right)$ – набор нечетких значений контролируемых переменных, для которых введены функции их принадлежности $\mu^{(k)}\left(x_j\right)$, $j=1,2,...,n$, $k=1,2,...,p$, диапазону возможных своих значений, определяемому состоянием. Введем совокупность линейных по параметрам, но нелинейных по факторам уравнений регрессии

$$y_k=a_{k_0}+\sum_{j=1}^{n}a_{k_j}x_j+\sum_{j_1=1}^{n}\sum_{j_2\neq j_1}a_{kj_1j_2}x_{j_1}x_{j_2}+$$
$$+\sum_{j_1=1}^{n}\sum_{j_2\neq j_1}\cdots\sum_{j_d\neq j_{d-1}}a_{kj_1j_2\ldots j_d}x_{j_1}x_{j_2}\ldots x_{jd}, \quad (14)$$
$$k=1,2,...,p.$$

Если при этом предполагается, что парные взаимодействия в (14) в достаточной мере определяют возможное появление синергетического эффекта, то уравнение регрессии упростится к виду

$$y_k=a_{k_0}+\sum_{j=1}^{n}a_{k_j}x_j+\sum_{j_1=1}^{n}\sum_{j_2\neq j_1}a_{kj_1j_2}x_{j_1}x_{j_2}, \quad (15)$$
$$k=1,2,...,p.$$

При этом коэффициенты $\left(a_{kj}\right)$ оцениваются статистически. Поставим задачу расчета функции принадлежности нечетких чисел y_k, $k=1,2,...,p$, рассчитываемых в соответствии с (14).

Основные результаты

Понятно, что вид искомых функций принадлежности зависит от того, каким образом заданы функции принадлежности $\mu^{(k)}\left(x_j\right)$, $j=1,2,...,n$, $k=1,2,...,p$. Рациональный выбор функций принадлежности для нечетких значений контролируемых параметров очень важен, поскольку он определяет уровень простоты и удобства применения правил выполнения операций над соответствующими нечеткими числами. Пусть, например, каждая из этих функций является функцией (*L-R*)- типа [8-10], которая имеет вид

$$\mu(x)=\begin{cases}L\left(\dfrac{a-x}{\alpha}\right), & x\le a,\\ \\ R\left(\dfrac{x-a}{\beta}\right), & x>a,\end{cases}$$

где *L* и *R* являются произвольными невозрастающими на множестве неотрицательных действительных чисел функциями, $\alpha>0$, $\beta>0$. При этом параметр *a* задает моду нечеткого числа *x*, а параметры α и β являются соответственно левым и правым коэффициентами нечеткости. Из этого следует, что нечеткое число (*L-R*)-типа при фиксированных *L* и *R* функциях однозначно определяется тройкой параметров (a,α,β). Соответствующее нечеткое число обозначается следующим образом: $B_{LR}=\langle a,\alpha,\beta\rangle$.

Удобство использования моделей (L-R)-типа для описаний функций принадлежности нечетких чисел определяется простотой выполнения алгебраических операций над соответствующими нечеткими числами [4, 5], которые реализуются следующим образом.

Результатом сложения двух нечетких чисел $U_{LR}=\langle a_u,\alpha_u,\beta_u\rangle$ и $V_{LR}=\langle a_v,\alpha_v,\beta_v\rangle$ является число (L-R)-типа $W_{LR}=\langle a_w,\alpha_w,\beta_w\rangle$, причем $a_w=a_u+a_v$, $\alpha_w=\alpha_u+\alpha_v$, $\beta_w=\beta_u+\beta_v$.

Результатом умножения нечеткого числа $U_{LR}=\langle a_u,\alpha_u,\beta_u\rangle$ на положительную константу *c* является число (L-R)-типа $W_{LR}=\langle a_w,\alpha_w,\beta_w\rangle$, причем $a_w=a_uc$, $\alpha_w=\alpha_uc$, $\beta_w=\beta_uc$.

Результатом умножения нечеткого числа $U_{LR}=\langle a_u,\alpha_u,\beta_u\rangle$ на отрицательную константу *c* является нечеткое число (L-R)-типа $W_{LR}=\langle a_w,\alpha_w,\beta_w\rangle$, причем $a_w=a_uc$, $\alpha_w=-c\alpha_u$, $\beta_w=-c\beta_u$.

Результатом умножения двух нечетких чисел с положительными носителями $U_{LR}=\langle a_u,\alpha_u,\beta_u\rangle$ и $V_{LR}=\langle a_v,\alpha_v,\beta_v\rangle$ является число (L-R)-типа $W_{LR}=\langle a_w,\alpha_w,\beta_w\rangle$, причем $a_w=a_ua_v$, $\alpha_w=a_u\alpha_v+a_v\alpha_u$, $\beta_w=a_u\beta_v+a_v\beta_u$.

Приведенные правила могут быть использованы для получения функций принадлежности нечетких чисел $\hat{y}_k$, $k=1,2,...,p$, рассчитываемых в соответствии с (15). Применим описанные правила выполнения операций над нечеткими числами. Пусть функция принадлежности контролируемого параметра x_i нечеткому множеству значений, соответствующему *k*-му состоянию описывается функцией *(L-R)*-типа

$$\mu^{(k)}\left(x_j\right)=\begin{cases} L\left(\dfrac{\bar{x}_j^{(k)}-x_j}{\alpha_{kj}}\right), \\ R\left(\dfrac{x_j-\bar{x}_j^{(k)}}{\beta_{kj}}\right). \end{cases}$$

Реализуем приведенные выше правила выполнения операций над нечеткими числами. При этом функция принадлежности нечеткого числа $u_{kj}=a_{kj}x_j$ имеет вид

$$\mu^{(k)}\left(u_{kj}\right)=\begin{cases} L\left(\dfrac{a_{kj}\bar{x}_j^{(k)}-u_{kj}}{a_{kj}\alpha_{kj}}\right), \\ R\left(\dfrac{u_{kj}-a_{kj}\bar{x}_j^{(k)}}{a_{kj}\beta_{kj}}\right), \end{cases}$$

а функция принадлежности нечеткого числа $u_k=\sum\limits_{j=1}^{n}u_{kj}$ определяется по формуле

$$\mu^{(k)}\left(u_k\right)=\begin{cases} L\left(\dfrac{\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}-u_k}{\sum\limits_{j=1}^{n}a_{kj}\alpha_{kj}}\right), \\ R\left(\dfrac{u_k-\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}}{\sum\limits_{j=1}^{n}a_{kj}\beta_{kj}}\right). \end{cases}$$

Далее, функция принадлежности нечеткого числа $v_{j_1j_2}=x_{j_1}x_{j_2}$ имеет вид

$$\mu^{(k)}\left(v_{j_1j_2}\right)=\begin{cases} L\left(\dfrac{\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}-v_{j_1j_2}}{\bar{x}_{j_1}^{(k)}\alpha_{kj_2}+\bar{x}_{j_2}^{(k)}\alpha_{kj_1}}\right), \\ R\left(\dfrac{v_{j_1j_2}-\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}}{\bar{x}_{j_1}^{(k)}\beta_{kj_2}+\bar{x}_{j_2}^{(k)}\beta_{kj_1}}\right). \end{cases}$$

Функция принадлежности нечеткого числа $w_{j_1j_2k}=a_{j_1j_2}v_{j_1j_2}$ имеет вид

$$\mu^{(k)}\left(w_{j_1j_2k}\right)=\begin{cases} L\left(\dfrac{a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}-w_{j_1j_2k}}{a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\alpha_{kj_2}+\bar{x}_{j_2}^{(k)}\alpha_{kj_1}\right)}\right), \\ R\left(\dfrac{w_{j_1j_2k}-a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}}{a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\beta_{kj_2}+\bar{x}_{j_2}^{(k)}\beta_{kj_1}\right)}\right), \end{cases}$$

а функция принадлежности нечеткого числа $w_k=\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j_1}w_{j_1j_2k}$ определяется выражением

$$\mu^{(k)}\left(w_k\right)=\begin{cases} L\left(\dfrac{\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j_1}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}-w_k}{\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\alpha_{kj_2}+\bar{x}_{j_2}^{(k)}\alpha_{kj_1}\right)}\right), \\ R\left(\dfrac{w_k-\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}}{\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\beta_{kj_2}+\bar{x}_{j_2}^{(k)}\beta_{kj_1}\right)}\right). \end{cases}$$

Наконец, функция принадлежности нечеткого числа $y_k=u_k+w_k$ рассчитывается по формуле

$$\mu^{(k)}\left(y_k\right)=$$

$$=\begin{cases} L\left(\dfrac{\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}-y_k}{\sum\limits_{j=1}^{n}a_{kj}\alpha_{kj}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\alpha_{kj_2}+\bar{x}_{j_2}^{(k)}\alpha_{kj_1}\right)}\right), \\ R\left(\dfrac{y_k-\left(\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}\right)}{\sum\limits_{j=1}^{n}a_{kj}\beta_{kj}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\beta_{kj_2}+\bar{x}_{j_2}^{(k)}\beta_{kj_1}\right)}\right). \end{cases}$$

Пусть теперь в определенной ситуации принятия решения получен вектор контролируемых переменных $X^*=\left(x_1^*,x_2^*,\ldots,x_n^*\right)$. Тогда с использованием полученных соотношений можно рассчитать оценки достоверности для каждого из состояний. Соответствующее число для k – го состояния равно

$$\mu^{(k)}\left(X^*\right)=$$

$$=\begin{cases} L\left(\dfrac{\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}-\left(\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}x_{j_1}^*x_{j_2}^*\right)}{\sum\limits_{j=1}^{n}a_{kj}\alpha_{kj}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\alpha_{kj_2}+\bar{x}_{j_2}^{(k)}\alpha_{kj_1}\right)}\right), \\ R\left(\dfrac{\left(\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}x_{j_1}^*x_{j_2}^*\right)-\left(\sum\limits_{j=1}^{n}a_{kj}\bar{x}_j^{(k)}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\bar{x}_{j_1}^{(k)}\bar{x}_{j_2}^{(k)}\right)}{\sum\limits_{j=1}^{n}a_{kj}\beta_{kj}+\sum\limits_{j_1=1}^{n}\sum\limits_{j_2\neq j}a_{kj_1j_2}\left(\bar{x}_{j_1}^{(k)}\beta_{kj_2}+\bar{x}_{j_2}^{(k)}\beta_{kj_1}\right)}\right), \end{cases}$$

$$k=1,2,\ldots,p\,.$$

Сравнение этих чисел для разных состояний позволяет выбрать то из них, степень достоверности которого в ситуации, когда набор контролируемых параметров образует вектор X^*, является наибольшей.

Предложенная методика обладает рядом важных достоинств. Во-первых, она реализует процедуру, не требующую использования систем нечеткого логического вывода. Во-вторых, она позволяет рассчитать степени достоверности состояний объекта для любого набора контролируемых параметров. В-третьих, она обеспечивает возможность учета различий в важности контролируемых параметров. Наконец, методика дает возможность при расчете степени достоверности состояний учитывать не только значения влияющих факторов, но и их взаимодействий требуемого порядка.

Выводы

Таким образом, предложена методика построения нечеткой экспертной системы, в которой функция принадлежности для каждого возможного состояния объекта оценивается с использованием уравнения регрессии, а контролируемые параметры – нечеткие числа с соответствующими функциями принадлежности. Для описания введенных функций принадлежности используется представление в виде $L-R$ функций, обеспечивающее простоту выполнения операций расчета функции принадлежности результирующей переменной.

Список литературы

1. Уотермен Д. Руководство по экспертным системам: пер. с англ. / Д.Уотермен. – М.: МИР, 1989. – 338с.
2. Нейлор К. Как построить свою экспертную систему: пер. с англ. / К. Нейлор. – М.: Энергоатомиздат, 1991. – 285с.
3. Zadeh L.A. The concept of linguistic variable and its application to approximate reasoning / L.A. Zadeh // Information Sciences, 1975. – Vol.4. – pp 199-249.
4. Бочарников В.П. Fuzzy Technology: основы моделирования и решения экспертно-аналитических задач. – К.: Эльга, Ника – Центр, 2003. – 296с.
5. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы: пер. с польск. / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2004. – 452с.
6. Takagi T. Fuzzy identifications of systems and its application to modeling and control / T. Takagi, M. Cugeno // IEEE Trans. SMC, 1985. – pp 116-132.
7. Серая О.В. Модели и информационные технологии оценки и прогнозирования состояния многомерных динамических объектов в условиях нечетких входных данных: дис…канд. техн. наук: 05.13.06; защищена 17.01.02; утв. 13.03.02 / Серая Оксана Владимировна; НТУ «ХПИ». – Х., 2001. – 251 с.
8. Дюбуа Д. Теория возможностей. Приложение к представлению знаний в информатике: пер. с франц. / Д. Дюбуа, А. Прад. – М.: Радио и связь, 1990. – 286с.
9. Леоненков А.В. Нечеткое моделирование в среде MATLAB и fuzzy TECH. / А.В. Леоненков. – СПб.: БХВ – Петербург, 2003. – 736с.
10. Орловский С.А. Проблемы принятия решений при нечеткой исходной информации. / С.А. Орловский. – М.: Наука, 1984. – 206с.

УДК 004.942

ТОМАШЕВСЬКИЙ В.М.,
ДМИТРИК І.М.

АНАЛІЗ МОДЕЛЕЙ НАВЧАННЯ ТА КОНТРОЛЮ ЗНАНЬ

У статті розглянуто процеси навчання й научіння у формально-психологічному аспекті через можливість побудови математичної моделі. Проведено аналіз методів і моделей статистичної теорії навчання, теорії стохастичних процесів, які використовуються при обробці результатів контролю та прогнозуванні й плануванні навчання.

Educatory and learning processes were studied in psychological aspect in the article, which gives a possibility to make a mathematical model. Analysis of methods and models of statistics educational theory was also made, as well as theory of stochastic processes, which are used during processing of results of control and education planning, was studied.

Вступ

Проблеми моделювання і розробка моделей їх користувачів (студентів), що є ключовими в навчальному процесі з використанням локальних комп'ютерів і комп'ютерних мереж, особливо актуальні в умовах бурхливого розвитку дистанційної освіти. Більшість з цих систем не враховує індивідуальні особливості тих, кого навчають, проте саме когнітивні можливості вирішальним чином часто впливають на успішність навчання. Для ефективного розв'язання основних проблем навчання необхідні теорії, які пояснюють, як знання представлені у пам'яті, яким чином зі структур знань вилучається необхідна інформація, як до цих структур додається нова інформація, а також, яким чином ця система здійснює розширення структур знань шляхом процесів саморозвитку. Процеси навчання й научіння розглядаються в різних аспектах: біологічному, фізіологічному, психологічному, філософському й ін. Основна увага приділяється формально-психологічному аспекту, тому що саме він дає можливість побудувати математичні моделі, які необхідно використовувати при побудові адаптивних навчальних систем.

Аналіз останніх досліджень

Вагомий внесок у розвиток досліджень у цьому напрямі здійснили Андерсон і Бауер, Ньюелл і Саймон, Ліндсей і Норман, Румелхарт, Шенк тощо. Існує багато робіт і вітчизняних авторів – П.І. Зінченко, Л.С. Виготський, А.М. Леонтьєв, В.Я. Ляудіс, П.П. Булонський, А.Р. Лурія, А.А. Смірнов, О.П. Свиридов, Л.А. Растригін й ін.

Формулювання цілей статті

Завданням даної статті є розглянути методи теорії стохастичних процесів, які використовуються при обробці результатів контролю та прогнозуванні навчання, а також методи та моделі статистичної теорії навчання.

Основний матеріал дослідження

Уперше систематичними науковими дослідженнями проблем научіння та навчання сто років тому займалися психологи Еббінгауз і Торндайк. У психології научіння визначають як засвоєння студентами певної системи знань, умінь, навичок. Цей процес припускає зміни зовнішньої (фізичної) і внутрішньої (психічної) діяльності (або поведінки) студента відповідно до мети цієї діяльності (або поведінки). Діяльність людини, спрямовану на учіння, що має своєю прямою метою научіння, називають навчанням. Навчання в психології визначається як процес стимуляції й управління зовнішньою й внутрішньою активністю студента, у результаті якої у нього формуються певні знання, навички й уміння. Причому стимуляція – це вплив, який викликає певну відповідну активність студента, а управління – це вплив, який направляє активність студента таким чином, щоб у результаті досягалася певна, заздалегідь поставлена мета.

Винятково важливу роль у вивченні процесів научіння й навчання відіграє дослідження пам'яті. Пам'ять є одним з найважливіших психічних процесів, що реалізує засвоєння знань. Початок експериментальних досліджень пам'яті пов'язаний з дослідами Г. Еббінгауза. Він перший розробив кількісні методи дослідження запам'ятовування й забування. Еббінгаузом побудована крива зміни об'єму пам'яті в залежності від часу, що пройшов після запам'ятовуван-

ня, тобто крива часу забування (рис. 1). Цю криву називають кривою забування або зберігання. Вона виражає об'єм пам'яті через різні проміжки часу у «відсотках заощадження». При цьому під об'ємом пам'яті (короткочасної) розуміється найбільше число одиниць матеріалу, який запам'ятовується і може бути відразу відтворений при одному повторенні. Що стосується довгострокової пам'яті, то вимірюють число повторень, необхідних для запам'ятовування й безпомилкового відтворення наданого для запам'ятовування матеріалу. Об'єм пам'яті визначають як відношення числа символів, що запам'ятовуються, до числа повторень. При цьому крива Еббінгауза – це об'єм пам'яті як функція часу. Описується вона наступним виразом:

$$b = \frac{100k}{(\log t)^c + k},$$

де b – процент утримуваного в пам'яті матеріалу в момент експерименту (або контролю) або об'єм пам'яті у «відсотках зберігання»; t – час з моменту повного оволодіння матеріалом у годинах; c і k – константи, отримані методом найменших квадратів на основі експериментальних даних, описаних у роботі [1].

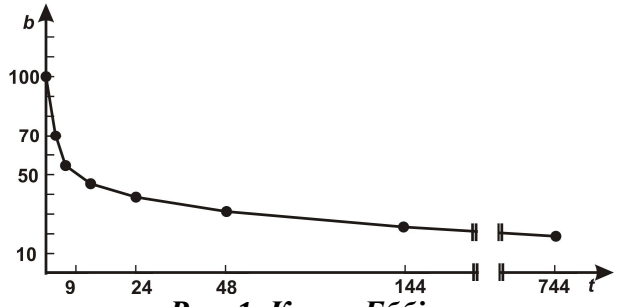


Рис. 1. Крива Еббінгауза

Аналогічні криві забування були отримані Радоссавлевичем, Фінкенбиндером, Пьєроном, Лу та Бореасом, які також проводили досліди в цьому напрямку. Проте існують деякі розбіжності відносно швидкості та величини першочергового спаду кривої, які пояснюються відмінностями в умовах експерименту, матеріалі, який запам'ятовується, й індивідуальних особливостях піддослідних.

О. Щукаревим було виведено рівняння, яке лише інтерпретує наявні дані та не спирається на конкретну теорію:

$$y = a - be^{cn},$$

де y – засвоювання, яке визначається як число правильних відтворень (успіхів) за одиницю часу; n – число випробувань; a – границя засвоювання при $n \to \infty$; b і c – константи.

Робертсоном Т. було запропоновано рівняння, отримане за аналогією з мономолекулярною автокаталітичною реакцією:

$$y = \frac{be^{\lambda n}}{c + e^{\lambda n}},$$

де y – засвоювання; n – число випробувань; $A = a\ b$; a і c – константи (параметри користувача); b – границя засвоювання при $n \to \infty$.

Л. Терстоун запропонував наступний так званий гіперболічний закон навчання:

$$y = \frac{a(n+c)}{(n+c)+b},$$

де y – засвоювання; n – число випробувань; a і c – константи; b – швидкість засвоювання.

К. Халл увів змінну «сила навички», що виражається формулою:

$$H_R{}^S = M(1 - e^{-bn}),$$

де $H_R{}^S$ – «сила навички», або асоціативна змінна, яка пов'язує стимул і реакцію; M – асимптотичне значення «сили навички»; b – параметр, який виражає швидкість научіння; n – число навчальних дослідів (або спроб з підкріпленням). Модель Халла дозволяє передбачати результати, які отримані при навчанні парним асоціаціям.

Р. Буш і Ф. Мостеллер розглядали навчання як стохастичний процес, вважаючи основною змінною ймовірність відповіді або реакції. Базуючись на тому, що процес научіння є марківським, вони побудували так звані стохастичні моделі навчання. Приблизно у той же час В. Естес, К. Берк, Дж. Міллер, У. Мак-Гілл й інші розроблювали подібні стохастичні моделі, які отримали назву «лінійні моделі навчання». При побудові даних моделей вводиться ймовірність p_n того, що студент у n-ому випробуванні дасть відповідь A_1. Альтернативною є відповідь A_2. Відповідно ймовірність того, що студент у n-ому випробуванні дасть відповідь A_2, дорівнює $1 - p_n$. У кожному випробуванні студент дає відповідь, отримуючи при цьому якесь підкріплення, наприклад, дізнається про правильну відповідь [2]. У залежності від підкріплюваної події E_j у n-му випробуванні змінюється ймовірність відповіді в $n+1$-ому випробуванні:

$$p_{n+1} = a_j p_n + b_j,$$

де параметри a_j і b_j збільшують або зменшують ймовірність відповіді. Ці параметри залежать від того, чи підкріплює подія E_j відповідь

A_1 або A_2. Так, у моделі Буша-Мостеллера для випадку двох альтернатив вводяться оператори:

$$p_{n+1} = \begin{cases} \alpha_1 p_n + (1-\alpha_1)\lambda_1 \ \textit{у випадку відповіді } A_1; \\ \alpha_2 p_n + (1-\alpha_2)\lambda_2 \ \textit{у випадку відповіді } A_2, \end{cases}$$

де λ_1, λ_2 $(0 \le \lambda_1, \lambda_2 \le 1)$ – непорушні точки, тобто якщо $p_n = \lambda_i (i = 1,2)$, то $p_{n+1} = p_n$.

В. Естесом була побудована стохастична модель для задачі навчання парним асоціаціям. Упродовж кожного досліду студенту пред'являвся деякий збуджуючий образ (стимул) і вказувалось його правильне найменування. Таке сполучення збудження та правильної відповіді у психології називають підкріпленням. Під час перевірочного досліду надавався тільки збуджуючий образ, на який студент повинен дати правильну відповідь. При цьому вводилась наступна формалізація. Припустимо $E_1, E_2, ..., E_N$ – елементи збудження, $A_1, A_2, ..., A_R$ – альтернативні відповіді, $p_{ij,n}$ – ймовірність того, що елемент збудження E_i під час n-го досліду викличе відповідь A_j. Тоді процес набування навички описується наступною функцією:

$$p_{ij,n+1} = p_{ij,n} + c(1 - p_{ij,n}),$$

де c – константа $(0 < c < 1)$.

Описані вище моделі Халла і Терстоуна у роботі Буша і Мостеллера інтерпретуються в термінах стохастичних моделей. Так, модель Халла приймає вигляд:

$$p_{n+1} = p_n + (1-\alpha)(1 - p_n),$$

де p_n – ймовірність набуття навички (або правильної відповіді) у n-му досліді; $\alpha (0 < \alpha < 1)$ – константа.

Порівнюючи дану модель з моделлю Естеса, можна помітити, що вона є окремим випадком моделі Естеса [3].

У моделі Терстоуна величина y інтерпретується як ймовірність набуття навички p_n, яка дорівнює 0 при $n = 1$ і прямує до 1 при $n \to \infty$. Рівняння Терстоуна набуває вигляд:

$$p_n = \frac{n-1}{n-1+b},$$

де b – швидкість научіння.

У моделі Рестла ймовірність змінюється за наступним правилом:

$$p_n = 1 - \frac{(1-\theta)^{n-1}}{\theta + (1-\theta)^n},$$

де θ – константа.

Кричевський висунув гіпотезу про те, що після початкового періоду навчання виникає «раптова» навченість, яка лежить в основі так званої теорії «стрибків». Вводиться випадкова величина x_n:

$$x_n = \begin{cases} 1, \textit{ якщо у } n-\textit{ому випробуванні має} \\ \textit{місце подія } A_1, \\ 0, \textit{ якщо у } n-\textit{ому випробуванні має} \\ \textit{місце подія } A_2. \end{cases}$$

Тоді $p_n = P\{x_n = 1\}$ – ймовірність події $A_1, q_n = P\{x_n = 0\} = 1 - p_n$. Припустимо, що студент на початку досліду знаходився у деякому стані S_1, а потім після деякого досліду n_i переходить у стан S_2 і залишається у ньому до кінця експерименту. У цьому випадку ймовірність p_n має вигляд

$$p_n = \begin{cases} p, \textit{ якщо студент в } n-\textit{ому випробуванні} \\ \textit{знаходиться у стані } S_1; \\ 1, \textit{ якщо студент в } n-\textit{ому випробуванні} \\ \textit{знаходиться у стані } S_2. \end{cases}$$

Крім того, вводиться ймовірність переходу зі стану S_1 у стан S_2 на n-му випробуванні:

$$P\{S_2 | S_1 \textit{ при } n-1\} = \beta,$$

де β – деякий параметр, $0 \le \beta \le 1$.

Підхід Кричевського можна застосувати при моделюванні динаміки вивчення навчального курсу, причому з використанням мереж Петрі. Тоді позиція відповідатиме деякому стану процесу навчання, мітка зіставлятиметься тому, кого навчають, перехід асоціюватиметься з вивченням деякої теми або окремого модулю (рис. 2). Використання моделі мережі Петрі дозволить створити індивідуальний сценарій навчання для студента.

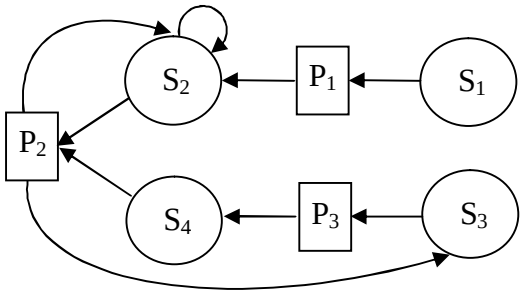


Рис. 2. Мережа Петрі

Подальший розвиток подібні моделі навчання набули у роботі Р. Аткінсона, Г. Бауера, Е. Кротерса.

Так, вищенаведені автори вважали, що співвідношення між стимулами та відповідями є результатом двох процесів: сенсорного процесу та процесу прийняття рішень. Їх модель може бути використана при аналізі експериментальних ситуацій, у яких застосовується проста структура відповідей, і в кожній спробі піддослідному надається обернений зв'язок, що містить інформацію про правильність його відповідей [4].

На основі проведених досліджень було розроблене наступне рівняння, яке може бути моделлю процесу навчання:

$$P(t,i) = A(i) - B(i)\exp[-tC(i)].$$

Вищенаведена формула прогнозує ймовірність виконання P стандартизованого тесту студентами i як функцію часу t, витраченого на взаємодію з системою навчання з ЕОМ впродовж навчального року. Параметри $A(i)$, $B(i)$ і $C(i)$ характеризують студента i та є різними для різних студентів [5] і можуть бути отримані шляхом накопичення статистики в процесі роботи системи навчання.

Серед досліджень в області статистичної теорії навчання та контролю знань найбільш відомі роботи О.П. Свиридова, у яких вивчається статистична динаміка знань, встановлюється зв'язок між потоком навчального матеріалу, його засвоюванням і забуванням.

Нехай у момент часу $t = 0$ навчальна інформація сприйнята студентом, а при $t > 0$ йому задається запитання за цим матеріалом. Якщо у момент $t = \tau$ студент дає неправильну відповідь на це запитання, то τ – відповідно час забування. Припускається, що час τ неперервна випадкова величина з функцією розподілу

$$P(t) = 1 - e^{-\lambda t},$$

де λ – інтенсивність забування [1/с]. Середній час забування дорівнює $1/\lambda$ [с].

Для опису процесу забування використовуються також розподіл Вейбулла, Ерланга і гамма-розподіл.

Розподіл Вейбулла визначається формулою:

$$Q(t) = 1 - \exp\left(-\frac{\lambda t^{a+1}}{a+1}\right).$$

Існують номограми для визначення значень розподілу Вейбулла.

Розподіл Ерланга полягає в наступному:

$$Q(t) = 1 - \sum_{r=0}^{a-1} \frac{(\lambda t)^r}{r!} e^{-\lambda t},$$

де a – додатне ціле число. Можна помітити, що експоненціальний розподіл є частинним випадком розподілу Ерланга (при $a = 1$).

Узагальнення розподілу Ерланга можна отримати, якщо вважати, що параметр a може приймати будь-яке додатне значення. При цьому щільність розподілу має вигляд:

$$q(t) = \frac{\lambda(\lambda t)^{a-1}}{\Gamma(a)} e^{-\lambda t}.$$

Необхідно зауважити, що інтенсивність забування для експоненціального розподілу постійна, для гамма-розподілу і розподілу Вейбулла при $a > 1$ інтенсивність $\lambda(t)$ збільшується [6].

Б.А. Смірнов досліджував довготривалість зберігання інформації. При цьому ймовірність правильної відповіді при відстроченому на час τ відтворенні визначається за формулою

$$P_\tau = P_0 e^{-\gamma\tau},$$

де P_0 – та ж ймовірність при негайному відтворенні, γ – швидкість забування інформації, яка визначається за методом найменших квадратів.

У роботі О.Ф. Шленського і Б.В. Боде досліджуються процеси накопичення та забування інформації та робиться спроба їх математичного виразу. Нехай $\Delta\tau$ –довготривалість окремого періоду часу надходження інформації, i_0 – середня швидкість сприйняття інформації студентом (в одиницях інформації за секунду). Через те, що в «кодуючому приладі мозку» внаслідок втрати частини інформації надходить деяка її частина $0 < \sigma < 1$, кількість інформації, сприйнятої студентом за час $\Delta\tau$, визначається як $\Delta J = \sigma i_0 \Delta\tau$, а сумарна кількість накопиченої інформації $J = \sum^{n} \Delta J$, де n – загальне число періодів $\Delta\tau$. Однак, через те, що відбувається втрата інформації в результаті її забування, вводиться деяка поправка, яка враховує розсіювання набутої у момент τ порції інформації до моменту часу t, як функція $K(t-\tau)$. Параметри даної функції і параметр σ залежать від індивідуальних особливостей студента. Із врахуванням процесу забування зміна кількості інформації в залежності від часу t виражається у вигляді [7]:

$$J = \sum^{n} K(t-\tau)\sigma i_0 \Delta\tau.$$

Далі розглянемо моделі відновлення знань.

При миттєвому відновленні знань, коли часом вивчення або повторення навчального матеріалу можна зневажити в порівнянні з часом забування, момент часу забування *n*-го запитання математично описується наступним чином:

$$t_n = \sum_{j=1}^{n} \tau_j ,$$

тобто після вивчення *i*-го питання в момент часу $t = 0$ студент дає на нього правильну відповідь. Але через час τ_1 він його забуває. У цей момент миттєво відбувається відновлення знань студента. Однак через якийсь час τ_2 студент знову забуває запитання. Цей процес може тривати багаторазово.

Якщо відновлення знань за забутим запитанням відбувається миттєво, то моменти забування або відновлення знань $t_1, t_2, ..., t_n$ утворюють потік $П_{1i}$ навчального матеріалу за *i*-м запитанням.

У загальному випадку інтервали часу τ_j, $j = 1,2,...$ є випадковими величинами, тому відповідний потік також є випадковим, який можна охарактеризувати функцією розподілу у вигляді:

$$F(z_1, z_2, ..., z_j) = P\{\tau_1 < z_1, \tau_2 < z_2, ..., \tau_j < z_j, \}.$$

Якщо інтервали часу *t* надання навчального матеріалу розподілені за експоненціальним законом з функцією розподілу $F(t) = 1 - e^{-\lambda t}$, то потік називається найпростішим або стаціонарним пуассонівським потоком.

Процес забування і відновлення з кінцевим часом відновлення знань за *n*-м запитанням (відновлення знань співвідносно з часом забування) можна представити у вигляді інтервалів, що чергуються, через забування або збереження (*стан* E_0) і відновлення знань (*стан* E_1). У момент часу $t = t_0^{1i}$ відповідно до навчальної програми починається вивчення навчального матеріалу деякого питання. Для цього потрібен час φ_1. Потім починається забування даного питання. Тривалість цього проміжку часу дорівнює τ_1. Для повторного відновлення знань з даного питання студенту потрібен час φ_2. Моменти часу

$$t_n^{1i} = \varphi_1 + \tau_1 + \varphi_2 + \tau_2 + ... + \tau_{n+1} + \varphi_n + \tau_n,$$

$$t_n^{2i} = \varphi_1 + \tau_1 + \varphi_2 + \tau_2 + ... + \tau_{n-1} + \varphi_n, n = 1,2,...$$

називаються відповідно моментами забування й відновлення знань. Час відновлення забутого навчального матеріалу менший за час початкового вивчення, але ця різниця невелика.

Якщо функція розподілу часу відновлення знань дорівнює

$$G(t) = 1 - e^{-\mu}, \mu > 0, t \geq 0,$$

відновлення знань називають кспоненціальним. При цьому математичне сподівання й дисперсія часу відновлення знань визначаються наступними формулами:

$$M\{\varphi\} = T_{вос.} = \frac{1}{\mu}, D\{\varphi\} = \frac{1}{\mu^2}$$

Особливість експоненціального відновлення знань полягає в тому, що якщо в момент часу *t* студент зайнятий відновленням знань, то розподіл часу відновлення, що залишився, буде експоненціальним з тим же параметром μ.

Як приклад використання експоненціального розподілу часу відновлення знань, можна розглянути варіант процесу навчання, коли студенту надається спочатку інформаційно-довідковий матеріал, а потім пропонується кілька вправ з відповідними поясненнями. При такій організації навчання ймовірність засвоєння знань або умінь підвищується зі зростанням числа вправ.

У статистичній теорії навчання розглядаються стандартизовані методи контролю знань, сутність яких полягає в тому, що студенту пропонується вибірка спеціальних завдань і за відповідями на неї робиться висновок про його знання, розумовий розвиток або здібності. Стандартизовані методи мають наступні позитивні властивості: скорочення часу контролю; стандартизованість проведення перевірки й аналізу результатів; можливість подання результатів перевірки в числовій формі; можливість математичної обробки результатів перевірки. До недоліків стандартизованого контролю знань можна віднести наступні: стандартизовані методи не завжди враховують індивідуальні особливості студентів; при використанні стандартизованих методів до уваги береться кінцевий результат розв'язання завдання й не враховується спосіб розв'язання.

Висновки та перспективи подальших досліджень

На сьогодні розроблена велика кількість різних підходів до моделювання процесу навчання. Однак відсутня уніфікована й інтегрована модель процесу навчання, яка б враховувала індивідуальні особливості студентів і адаптува-

ла форми та методи подання знань у залежності від їх когнітивних можливостей, створюючи при цьому індивідуальні стратегії навчання. Тому подальші дослідження спрямовані на розробку моделей компонентів навчальної адаптивної системи, у якій індивідуальні особливості користувачів враховуватимемо наступним чином:

1. Для дослідження процесів запам'ятовування та забування обрано експоненціальну залежність, у якій зміна ймовірностей незнання елементів вивченого матеріалу залежить від швидкостей їх забування, які, у свою чергу, визначаються індивідуальними особливостями пам'яті студента, і часу забування вивчених елементів після їх заучування. В подальшому здійснюється накопичення статистичного матеріалу для уточнення функції розподілу швидкості забування.

2. При вивченні окремих блоків (порцій) теоретичного матеріалу здійснюється перевірка знань студента і будуються прогнозні «криві забування».

3. На основі прогнозних «кривих забування» та моделей відновлення знань здійснюється адаптація системи подачі нового матеріалу та блоків повторення забутого матеріалу і створюється індивідуальний сценарій навчання студента.

Перелік посилань

1. Растригин Л.А., Эренштейн М.Х. Адаптивное обучение с моделью обучаемого. – Рига: Зинатне, 1988. – 160 С.
2. Буш Р., Мостеллер Ф. Стохастические модели обучаемости. – М.: Физматгиз, 1962. – 484 С.
3. Растригин Л.А., Эренштейн М.Х. Адаптивное обучение с моделью обучаемого. – Рига: Зинатне, 1988. – 160 С.
4. Аткинсон Р., Бауэр Г., Кротерс З. Введение в математическую теорию обучения: Пер. с англ. – М.: Мир, 1969. – 486 С.
5. Аткинсон Р. Человеческая память и процесс обучения. – М.: Прогресс, 1980. – 542 С.
6. Свиридов А.П. Введение в статистическую теорию обучения и контроля знаний. Ч. 2. Элементы статистической динамики знаний. – М., 1974. – 152 С.
7. Смирнов Б.А. Определение характеристик оперативной памяти // Психологические механизмы памяти и ее закономерности в процессе обучения: Материалы I Всесоюз. Сипмоз. По психологии памяти. – Харьков, 1970. – С.225-228.

УДК 004.942(045)

ТАЛАЄВ О.К,
ДУДНІК А.С.,
БЕРЕЗОВСЬКИЙ О.М.

МОДЕЛЮВАННЯ БЕЗДРОТОВИХ КОМП'ЮТЕРНИХ МЕРЕЖ В ЗАЛЕЖНОСТІ ВІД АЛГОРИТМУ УПРАВЛІННЯ ТА РОЗПОДІЛУ ТРАФІКУ

Побудовані моделі управління та розподілу трафіку в мережах стандарту IEEE 802.11. Параметри моделей повністю відповідають параметрам роботи реальних мережі. Змодельоване управління трафіком у мережевих стандартах WI-FI. Проведений порівняльний аналіз алгоритмів керування чергами з точки зору управління та розподілу трафіку.

Case and distributing of traffic frames are built in the networks of standard of IEEE 802.11. The parameters of models fully answer parameters works real of network. A management is modeled a traffic in the network standards of WI-FI. The comparative analysis of algorithms of management turns is conducted from the point of view a management and distributing of traffic.

Вступ

Алгоритми управління чергами потрібні для роботи в періоди тимчасових перевантажень, якщо мережевий пристрій не може впоратися з передачею бітів на вихідний інтерфейс в тому темпі, в якому вони поступають. Якщо причиною перевантаження є недостатня продуктивність процесорного блоку мережевого пристрою бездротової мережі, то необроблені біти тимчасово накопичуються у вхідній черзі відповідного вхідного інтерфейсу. Черг до вхідного інтерфейсу може бути декілька, якщо диференціюються запити на обслуговування по декількох класах. У тому ж випадку, коли причина перевантаження полягає в обмеженій пропускній спроможності вихідного інтерфейсу, біти тимчасово зберігаються у вихідній черзі (або чергах) цього інтерфейсу.

Аналіз останніх досліджень

Як видно зі вступної частини, вирішення цієї проблеми є дуже важливим питанням. Алгоритми управління трафіком прошиваються розробниками в інтегральні схеми бездротових мережевих пристроїв (точок доступу WI-FI). Ці алгоритми мають велике значення для точки доступу, адже в них прописана її поведінка під час перевантажень.

Нажаль існуючі спеціалізовані засоби моделювання комп'ютерних мереж не розглядають трафік з точки зору теорії черг. Існуючі засоби можуть лише відображати окремі параметри трафіку як черги, але відобразити всі параметри в комплексі не може жоден спеціалізований засіб моделювання.

Рішення, що пропонується в даній статті, побудоване за допомогою засобу імітаційного моделювання загального призначення. Даний засіб може розглянути будь-який процес, де є черга та обробляючий пристрій. Тобто задавши параметри роботи бездротової мережі, можна вирішити завдання, що описане вище.

Постановка задачі

Основним завданням даних досліджень являється моделювання бездротової передачі даних із застосуванням різних алгоритмів управління чергами. Визначення оптимального алгоритму управління трафіком, за допомогою проведення порівняльних характеристик. Алгоритми, що застосовуються для моделювання беруться довільно, не прив'язуючись до реальних пристроїв.

Види алгоритмів управління чергами

Серед видів алгоритмів управління чергами є три основних, а всі інші являють собою різноманітні їх комбінації. Далі наведемо коротке описання основних алгоритмів.

У традиційному *алгоритмі FIFO* у разі перевантаження всі біти поміщаються в *одну* загальну чергу і вибираються з неї в тому порядку, в якому поступили. У всіх пристроях з комутацією бітів алгоритм FIFO використовується за умовчанням. Достоїнствами його є простота реалізації і відсутність потреби в конфігурації. Проте йому властивий і корінний недолік – неможливість *диференційованої обробки бітів різних потоків. Всі* біти стоять в загальній черзі на рівних підставах. Разом виявляються і біти чутливого до затримок голосового

трафіку, і біти нечутливого до затримок, але дуже інтенсивного трафіку резервного копіювання, тривалі пульсації якого можуть надовго затримати голосовий пакет.

Алгоритми пріоритетного обслуговування дуже популярні в багатьох областях обчислювальної техніки, зокрема в операційних системах, коли одним застосуванням потрібно віддати перевагу перед іншими при обробці їх в мультипрограмній суміші. Застосовуються ці алгоритми і для переважної в порівнянні з іншими обробки одного класу трафіку.

Механізм пріоритетного обслуговування заснований на розділенні всього мережевого трафіку на невелику кількість класів і подальшого призначення кожному класу деякої числової ознаки – *пріоритету*.

Класифікація трафіку є окремим завданням. Біти можуть розбиватися на пріоритетні класи на підставі різних ознак: адреси призначення, адреси джерела, ідентифікатора додатку, що генерує цей трафік, будь-яких інших комбінацій ознак, які містяться в заголовках бітів. Правила класифікації бітів є частиною політики адміністрування мережі.

Алгоритм зважених черг розроблений для того, щоб можна було надати всім класам трафіку певний мінімум пропускної спроможності або гарантувати деякі вимоги до затримок. Під *вагою* даного класу розуміється відсоток такою, що надається класу трафіку пропускної спроможності від повної пропускної спроможності вихідного інтерфейсу.

При зваженому обслуговуванні так само, як при пріоритетному, трафік ділиться на декілька класів, і для кожного класу ведеться окрема черга бітів. Але з кожною чергою зв'язується *не пріоритет, а відсоток пропускної спроможності* ресурсу, що гарантується даному класу трафіку при перевантаженнях цього ресурсу. Для вхідного потоку таким ресурсом є процесор, а для вихідного потоку (після виконання комутації) – вихідний інтерфейс.

Розв'язання задачі

Концептуальні моделі мереж з застосуванням різних алгоритмів управління трафіком в мережах протоколу IEEE 802.11 представлені відкритою (незамкнутою) багатофазною системою масового обслуговування. З погляду класифікації Еталонної моделі описувалися два нижні рівні. Транзакт, що є неподільним об'єктом в системі імітаційного моделювання загального призначення, породжувався бітом, що переміщається в мережі WI-FI від джерела інформації до споживача. Кожна фаза моделювалася СМО (системи масового обслуговування) G\G\1 побудована згідно умов того чи іншого алгоритму управління чергами. Універсальна система імітаційного моделювання забезпечує збір і статистичну обробку даних про транзакти, затримані в кожній точці моделі, а також інтенсивності потоків відмов.

Імітація передачі даних в даній моделі буде проходити від станції-передавача до станції приймача через бездротову точку доступу, при моделюванні якої і будуть застосовані ті чи інші алгоритми управління чергами.

Хід експерименту

Побудуємо 3 моделі мереж з наступними параметрами:

1. Мережа WI-FI IEEE 802.11a
   - Частота роботи мережі 5 ГГц;
   - Швидкість передачі даних 54 Мб/с;
   - Кількість каналів 1;
   - Кількість пристроїв 2;
   - Алгоритм управління чергами "*FIFO*".
2. Мережа WI-FI IEEE 802.11b
   - Частота роботи мережі 2,4 ГГц;
   - Швидкість передачі даних 11 Мб/с;
   - Кількість каналів 4;
   - Кількість пристроїв 2;
   - Алгоритм управління чергами "*Алгоритм пріоритетного обслуговування*".
3. Мережа WI-FI IEEE 802.11g
   - Частота роботи мережі 2,4 ГГц;
   - Швидкість передачі даних 54 Мб/с;
   - Кількість каналів 4;
   - Кількість пристроїв 2.
   - Алгоритм управління чергами "*Зважені черги*".

Реалізуємо дану задачу за допомогою одного із засобів моделювання загального призначення, з використанням вище зазначених параметрів та алгоритмів управління трафіком бездротової мережі.

По тій причині, що в даній системі моделювання не можливо задати частоту буквально, вона виражена через період генерації за формулою: $\tau = 1/f$, де τ – період генерації який буде застосовано в моделі замість частоти, f – часто роботи мережі. Знайдемо період генерації для кожного окремого стандарту WI-FI:

$\tau_a = 1/f_a = 1/5000000000$ Гц = 0,0000000002

$\tau_b = 1/f_b = 1/2400000000$ Гц = 0,0000000004

$\tau_g = 1/f_g = 1/2400000000$ Гц = 0,0000000004

Таким же чином виразимо швидкість роботи в мережі WI-FI через середній час обслуговування за формулою: $b = 1/m$.

Де b – середній час обслуговування, m – швидкість передачі даних. Знайдемо швидкість передачі для кожного окремого стандарту WI-FI:

$b_a = 1/m_a = 1/54000000$ біт/с = 0,0000000185

$b_b = 1/m_m = 1/11000000$ біт/с = 0,00000009

$b_g = 1/m_g = 1/54000000$ біт/с = 0,0000000185

З умов задачі видно, що стандарти b, g, n мають по 4 канали, стандарти 802.11 a, g, n – однакову швидкість передачі даних. Тому 0,0000000185 і 0,0000000Ɛ розділимо на 4. Отримаємо середній час обслуговування для кожного каналу. В 802.11g буде дорівнювати 0,0000000046, а в 802.11b – 0,0000000227. Підставивши ці числа у модель.

Аналіз результатів експерименту

Запустивши програмну модель на виконання при 1000000 прогонів і зімітувавши максимальне навантаження на канали передачі, отримаємо наступні результати експерименту:

1. Час затримки в каналах передачі:
   - Стандарт 802.11a – t = 0,000321 с;
   - Стандарт 802.11b: t (канал №1) = 0,001652 с, t (Канал №2) = 0,001619 с, t (Канал №3) = 0,001619 с, t (Канал №4) 0,001589 с;
   - Стандарт 802.11g: t (канал №1) = 0,000328 с, t (Канал №2) = 0,000326 с, t (Канал №3) = 0,000329 с, t (Канал №4) = 0,000329 с.

   Обчислимо пропускну спроможність каналів для кожного з стандартів. Так, як транзакти в даній моделі замінюють роль бітів то пропускну спроможність можна за формулою: $C = \sum_{k=1}^{k} 1/t$ (C – пропускна спроможність, t – час затримки, k= кількість каналів). Маємо наступні результати:
   - Пропускна спроможність для стандарту 802.11a – C_a= 1 / 0,000321с = 3,115 Мб/с;
   - Пропускна спроможність для стандарту 802.11b – C_b = 2,5 Мб/с;
   - Пропускна спроможність для стандарту 802.11g – C_g = 12,18 Мб/с;

   Результати були підтверджені на спеціалізованому засобі моделювання.
2. Заповнимо отриманими значеннями порівняльну таблицю:

Табл. 1. Порівняльна характеристика результатів моделювання

ТЕХНОЛОГІЯ	СТАНДАРТ	СУМАРНА ПРОПУСКНА СПРОМОЖНІСТЬ	ДАЛЬНІСТЬ
Wi-Fi	802.11a	3,115 Мб/с	Приблизно 300 метрів
Wi-Fi	802.11b	2,5 Мб/с	Приблизно 300 метрів
Wi-Fi	802.11g	12,18 Мб/с	Приблизно 300 метрів

Оцінюючи дані результати можна сказати наступне:

- Стандарт 802.11a (Алгоритм управління чергами "*FIFO*")має наступні **переваги**: він вигідний при передачі файлів не великих розмірів, потребує порівняно низьких витрат енергії та має низьку ціну. **Недоліки:** за наявністю лише одного каналу при порівняно високій швидкості передачі даних має низьку пропускну спроможність. У разі перевантаження точки доступу буде застосовуватись алгоритм управління чергами "*FIFO*", який не пришвидшує передачу даних, а лише впорядковує її. Він доцільний там де потрібне впорядкування потоку даних;
- Стандарт 802.11b (*Алгоритм пріоритетного обслуговування*) має наступні **переваги**: він також вигідний при передачі файлів не великих розмірів, потребує порівняно низьких витрат енергії та має низьку ціну. За різницею між пропускною спроможністю і швидкістю він кращий ніж 802.11a (за рахунок 4 каналів). При сумарній пропускній спроможності 2,5 Мб/с його можна використовувати як Bluetooth з розширеними можливостями (він нагадує його за характеристиками). **Недоліки:** має низьку швидкість передачі даних. Алгоритм пріоритетного обслуговування хоча і підвищує швидкість за рахунок введення правил розподілу трафіку, але швидкість підвищується лише при передачі

трафіку високих класів, а швидкість нижчих класів може значно погіршитись. Його доцільно застосовувати там, де є важливою ієрархія класів трафіку за тими чи іншими ознаками.

– Стандарт 802.11g має наступні **переваги**: найоптимальніший за показниками швидкості, пропускної спроможності та дальності передачі. Алгоритм зважених черг є найбільш оптимальним тому, що він розподіляє пропускну спроможність кожного класу трафіку так, що біти кожного окремого класу надходять на обробку рівномірно, що на змушує адресата довго чекати. **Недоліки**: високі затрати енергії та ціна.

За даними результатами, отриманими за допомогою засобів моделювання загального призначення, вже можна судити про те, яке обладнання потрібне в тій чи іншій ситуації. Та який алгоритм краще застосовувати в тому чи іншому обладнанні.

Висновки

Після виконання даного дослідження були отримані наступні результати:

– Проведено порівняльну характеристику моделей стандартів WI-FI, та алгоритмів управління чергами;
– Побудовано моделі стандартів WI-FI за допомогою засобів загального призначення;
– Обґрунтоване використання засобів загального призначення для моделювання бездротових комп'ютерних мереж.

Перелік посилань

1. Олифер В. Г., Олифер Н. А. Компьютерные сети. – С-Пб., Питер Пресс : 2008. – 957 с.
2. Боев В. Д. Моделирование систем GPSS WORLD. – С-Пб., БХВ-Петербург: 2004. – 405 с.