ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Программная инженерия

ПрИн 12 2014

Учредитель: Издательство "НОВЫЕ ТЕХНОЛОГИИ"

Издается с сентября 2010 г.

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана, ОАО "Концерн "Сириус".

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индексы: по каталогу агентства "Роспечать" — **22765**, по Объединенному каталогу "Пресса России" — **39795**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/pi.html E-mail: prin@novtex.ru

Журнал включен в систему Российского индекса научного цитирования.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

№ 12
December
2014

Published since September 2010

CONTENTS

Information about the jornal is available online at: http://novtex.ru/pi.html, e-mail:prin@novtex.ru

УДК 004.43

А. М. Бородин, канд. техн. наук, доц., e-mail: amborodin@acm.org,
С. Г. Мирвода, ассистент, e-mail: sergey@mirvoda.com,
С. В. Поршнев, д-р техн. наук, доц., e-mail: sergey_porshnev@mail.ru,
Уральский федеральный университет имени первого президента России Б. Н. Ельцина, г. Екатеринбург

Анализ современных средств прототипирования языков программирования

Важный этап создания языка программирования — определение его синтаксиса. На этом этапе закладываются выразительные возможности и ясность языка, которые позволят завоевать интерес программистов.

В статье обсуждаются результаты сравнительного анализа основных перспективных подходов реализаций генераторов синтаксических анализаторов. Сравнительный анализ проведен с точки зрения прототипирования языка программирования.

Ключевые слова: *синтаксический анализ, контекстно-свободная грамматика, PB-грамматика, прототипирование языков программирования*

Введение

В настоящее время для обеспечения взаимодействия между пользователем и техническими средствами обработки и передачи информации активно применяют устройства, в которых использование текстовой информации сведено к минимуму или полностью исключено. Например, мыши, сенсорные панели, акселерометры, дигитайзеры, устройства распознавания жестов и голоса, в том числе Kinect, и др. С их помощью графические интерфейсы операционных систем персональных компьютеров обеспечивают пользователю возможность, используя только мышь, редактировать рисунки и фотографии. Сенсорные экраны позволяют пользователю прикосновением пальцев руки просматривать содержимое документов и писем, а также набирать тексты в WYSIWYG-редакторах. Кроме того, сегодня успешными с коммерческой точки зрения оказались бесконтактные управляющие электронные устройства, основанные на технологиях анализа аудио- и видеоинформации (голосовое управление и визуальное управление).

В то же время программисты, создающие программное обеспечение для "безтекстовых" технологий, используют при написании программ по-прежнему те или иные текстовые редакторы, в которых они создают инструкции для компьютеров, нажимая для этого соответствующие последовательности и определенные сочетания клавиш. Данные инструкции, имея различную функциональную направленность, обладают большим числом различных свойств. В то же время все они обладают общим универсальным свойством — инструкции представляют собой некоторые последовательные наборы символов, которые могут включать в себя буквы, цифры, знаки пунктуации и специальные знаки. Отметим, что существующие в способах представления программ исключения либо архаичны (например, перфокарты), либо пока крайне редки (например, проекционная среда разработки Meta Programming System, разработанная компанией Jet Brains [1]).

Преимущества подхода, основанного на использовании текстового языка, перечислены ниже.

- **Возможность использовать средства управления версиями.** Сегодня существует большое число различных программных средств, обеспечивающих совместную работу над одним и тем же текстом нескольких программистов. Такие средства позволяют разрешать возникающие конфликты, проводить анализ имеющихся версий, а также компоновать конечный результат с учетом намерения одновременно всех программистов (конечно, при этом потенциально возможно возникновение ситуаций, в которых это не реализуемо).
- **Портативность**. Текстовый файл можно, например, напечатать в журнале, продиктовать по телефону или вставить в презентацию. В то же время, если результат программирования представляется некоторой диаграммой с неявными элементами, то для его полноценной передачи и восприятия другими участниками процесса создания программного продукта необходимо привлекать соответствующие специализированные средства.
- **Автономность.** Текстовый файл в большинстве случаев является самодостаточной смысловой единицей. Это отличает его, например, от того случая, когда результат работы программиста представляет собой некоторое множество строк в СУБД, описывающих некоторое преобразование данных. В этом случае разделить текст команд, результат их выполнения и стороннюю информацию, содержащуюся в СУБД, оказывается весьма трудоемкой задачей.

- **Фрагментируемость**. Структурированные языки описания информации позволяют при необходимости легко извлекать отдельные фрагменты программного кода, не содержащие лишней информации. Например, если речь идет о решении некоторой относительно небольшой задачи, то оказывается достаточным скопировать соответствующие строки кода, но не требуется передавать всю программу.
- **Комбинируемость**. Как известно, текстовые программы могут создавать другие текстовые программы, которые, в свою очередь, также создают текстовые программы и так далее, до момента получения на определенном уровне кода, который, наконец, решает исходную задачу. Как следствие, в описанной ситуации требуется достаточный уровень абстрагирования и разделения функций кода. Здесь возможно возникновение проблемы, называемой проблемой "лазанья-кода". Например, она возникает в тех случаях, когда программа, написанная на языке PHP, содержит в строковых литералах программу, написанную на языке SQL. При этом последняя программа содержит программу на языке HTML, включающую, в свою очередь, программу на JavaScript, имеющую разметку Markdown, которая обеспечивает вызов в соответствии с HTTP-протоколом другой программы первого уровня, написанной на языке PHP.
- **Специфицируемость**. Текстовый язык программирования при наличии его спецификации позволяет большому числу различных сообществ создавать сторонние средства (компиляторы, системы статического анализа, расширения языка).

Отметим, что вопреки современному уровню развития информационных технологий, некоторые текстовые языки программирования, созданные в самом начале второй половины XX века, остаются актуальными и сегодня (например, Fortran, BASIC, Pascal, Cobol, C, SQL). Вместе с тем регулярно появляются новые языки программирования, создаваемые как крупными корпорациями (Go — 2009 г., Swift — 2014 г.), так и независимыми исследователями (Julia — 2009 г., Zonnon — 2003 г.). Таким образом, можно с высокой вероятностью предположить, что задача создания текстовых языков программирования и в ближайшем будущем останется актуальной.

Анализ современного состояния языков программирования позволяет сделать вывод о том, что некоторые языки уже не являются эффективным средством взаимодействия между программистом и ЭВМ. Однако они сохраняют свою значимость как промежуточный слой абстрагирования на пути преобразования программы из человекочитаемого представления в аппаратные инструкции. К данным языкам следует отнести:

— язык MS Common Intermediate Language (MS IL) [2], созданный для представления результата компиляции языков платформы .NET (C#, F#, VB.NET и др.)[1];

— код на языке JavaScript в проекте ASM.JS [3], по сути, представляющий собой результат компиляции ассемблера в JavaScript;

— язык HQL проекта Hibernate [4], код которого компилируется в реализации языков СУБД, например, в T-SQL, в виде, не предназначенном для чтения;

— язык Linq, позволяющий компилировать выражения на языке C# с помощью различных ORM-технологий в различные диалекты SQL.

Таким образом, принимая во внимание описанные выше достоинства текстового подхода, современный язык программирования должен одновременно выполнять функции создания команд, структурирования информации, а также разметки или структурирования данных. В связи с этим обоснование выбора средств, позволяющих создать язык программирования с заявленными функциональными возможностями, оказывается актуальным.

В статье изложены результаты сравнительного анализа средств разработки языков программирования с точки зрения прототипирования их грамматики. При этом сознательно за рамками статьи оставлен анализ дискуссии вокруг стратегии синтаксического анализа программ (парсеров): нисходящих (LL) и восходящих парсеров (LR), их модификаций (LL(*) и Packrat), а также оценок их производительности, начало которых было положено А. Ахо в "книге с красным драконом" [5]. Отметим при этом, что в современных работах (см., например, работу [6]) отличной считается производительность синтактического разбора "почти как у YACC[2]".

Синтаксический анализ текста

Важным процессом, реализующимся транслятором языка программирования, является синтаксический анализ (парсинг). В ходе его выполнения потоку символов программного кода ставится в соответствие синтаксическое дерево в соответствии с грамматикой языка [5]. В большинстве случаев синтаксическому анализу предшествует лексический анализ — преобразование потока символов в поток токенов (символов, слов, лексем). Однако с точки зрения рассмотрения прототипирования языка, лексический анализ можно считать этапом синтаксического анализа.

На первом этапе разработки языка программирования возникает необходимость выбора из следующих двух возможных технических решений:

- использование генератора парсеров;
- создание собственного парсера средствами применяемого языка общего назначения (хост-язык).

Отметим, что на настоящее время в продвинутых учебных курсах по программированию задача создания программной реализации парсера средствами языка общего назначения предлагается слушателям в качестве упражнения. Например, методика изучения языка Haskell предусматривает создание слушателями монадического и аппликативного парсеров [7]. Дан-

[1] Отметим, что удобочитаемость получаемого программного кода не является приоритетом языка MS IL, однако предусмотрено его структурированное текстовое представление.

[2] YACC — *Yet Another Compiler Compiler* (буквально — еще один компилятор компилятора).

ный подход также распространен и в сфере коммерческого программного обеспечения. Например, парсеры в поставляемой компанией Microsoft платформе компиляторов Roslyn [8] созданы без применения генераторов парсеров. При этом очевидно, что написанный без лишних слоев абстрагирования парсер будет работать быстрее и его работа будет прозрачна и понятна его авторам.

Вместе с тем подход, основанный на использовании генераторов парсеров, в значительной степени облегчает развитие языка. Один из наиболее известных генераторов YACC был создан С. Джонсоном для того чтобы добавить в язык B операцию исключающего ИЛИ [9]. После этого YACC стал наиболее часто используемым инструментом при разработке языков программирования для Unix.

На этапе прототипирования языка генератор парсеров позволяет существенно сократить сроки разработки за счет раннего обнаружения концептуальных противоречий. Однако использование генератора парсеров имеет ряд ограничений, рассматриваемых далее в статье.

Сравнение генераторов парсеров

Число известных на данный момент генераторов парсеров превышает 150 [10], однако только менее 10 % из них имеют интегрированную среду разработки. Сегодня наиболее популярным является генератор парсеров ANTLR (*ANother Tool for Language Recognition*). В данном генераторе парсеров использован метод парсинга LL(*) [6]. Он применяется в крупных коммерческих проектах (Twitter, Apple XCode IDE, Oracle SQL Developer). В качестве хост-языков в четвертой версии ANTLR применяются Java, Python или C#.

Отметим, что предыдущие версии генератора парсеров ANTLR поддерживали большее число хост-языков, поэтому можно ожидать, что в будущем создатели ANTLR реализуют поддержку и других современных языков программирования. С точки зрения прототипирования языка наибольший интерес представляет четвертая версия генератора ANTLR, так как в данной версии программист имеет возможность использовать интегрированную среду разработки — систему ANTLR Work, обеспечивающую тестирование парсера в интерактивном режиме, или создавать в среде разработки Jet Brains IntelliJ IDEA собственный плагин, который далее можно интегрировать в разработку парсера и применять в Java-проектах. Сегодня ANTLR является широко используемым программным инструментом, протестированным множеством признанных экспертов.

Относительно недавно был разработан генератор парсеров PEG.js, в котором в качестве хост-языка используется JavaScript; а также для эквивалентных PB-грамматик (PB-грамматики, parsing expression grammar — PEG) существуют генераторы в другие хост-языки [11]. В PEG.js реализован подход, основанный на грамматическом разборе выражений. При этом необходимо отметить, что динамически типизированный хост-язык с точки зрения прототипирования языка зачастую оказывается наиболее удобным. Среда разработки для PEG.js, реализованная на JavaScript, размещена на сайте его разработчика [12], что дает возможность заинтересованным пользователям экспериментировать с PB-грамматиками в режиме ONLINE.

На примере синтаксического разбора SQL-запроса, представленного на рис. 1, сравним тестовые грамматики, построенные генераторами парсеров ANTLR и PEG.js. Выбор данного выражения, представляющего собой запрос к некоторой СУБД, обусловлен тем, что в нем использованы наиболее типичные для SQL особенности: подзапрос, сортировка, реляционное соединение. При этом приведенные тестовые грамматики никак не реализуют ни DDL, ни DML.

Результаты синтаксического анализа, полученные с помощью генераторов парсеров ANTLR и PEG.js, представлены в таблице.

Отметим, что грамматики, представленные в таблице, не являются полноценными SQL-грамматиками, но приведены исключительно для демонстрации особенностей сравниваемых парсер-генераторов. Для получения полноценных SQL-грамматик следует использовать соответствующие программные инструменты, находящиеся в свободном доступе [13, 14].

Из таблицы видно, что формат записи обеих грамматик подобен структуре расширенной формы Бекуса-Наура [5], однако между ними существует ряд отличий. В грамматике ANTRL двоеточие разделяет название правила разбора, в грамматике PEG.js для этого используется знак "=". Для обозначения альтернативных вариантов в ANTLR используется символ «|», в PEG.js — символ "/". Синтаксис грамматик заимствован из регулярных выражений Клини [10]: знак "*" обозначает повторение предшествующей последовательности от 0 до *N* раз, знак "+" — повторение последовательности от 1 до *N* раз, знак "?" означает необязательность выражения. Постоянно повторяющееся правило sp в грамматике PEG.js соответствует наличию в разбираемом выражении разделителей (пробелов, знаков перевода строк и табуляции). Обозначения окончания правила разбора точкой с запятой в генераторе парсеров PEG.js является необязательным, однако в тестовой грамматике они оставлены для удобства чтения.

Важно отметить, что в ANTLR оператор "|" имеет семантику "альтернатива", тогда как в генераторе пар-

```
select "name" 'AuthorName', (select topic from post p where p.author_id=a.id
order by post_date limit 1) as 'Last topic' from author a inner join profile px on
px.id = a.id
```

Рис. 1. Тестовый SQL-запрос

Результаты синтаксического анализа тестового выражения

ANTLR 4-грамматика	PEG.js-грамматика
grammar AntlrSelect; select : 'select' column (',' column)* ('from' from_clause ('where' expr)? ('order' 'by' IDENTIFIER (','IDENTIFIER)*)? ('limit' NUMERIC_LITERAL)?)?; table_or_subquery : table_name ('as'? name)? | '(' select ')' ('as'? name)?; from_clause : table_or_subquery ('inner'? 'join' table_or_subquery 'on' expr)* ; expr : constv | (table_name '.')? IDENTIFIER | expr '||' expr | expr ('*' | '/' | '%') expr | expr ('+' | '−') expr | expr ('<<' | '>>' | '&' | '|') expr | expr ('<' | '<=' | '>' | '>=') expr | expr ('=' | '==' | '!=' | '<>') expr | expr 'and' expr | expr 'or' expr | '(' expr ')' ; column: '*' | name (('.' '*')|('as'? name))? | '('select')' ('as'? name)? | expr ('as'? name)?; table_name : name; name : IDENTIFIER | STRING_LITERAL | '(' name ')' ; constv : NUMERIC_LITERAL | STRING_LITERAL; IDENTIFIER: '"' (~'"' | '""')* '"' //| '`' (~'`' | '``')* '`' | '[' ~']'* ']' | [a-zA-Z_] [a-zA-Z_0-9]* ; NUMERIC_LITERAL: DIGIT+ ('.' DIGIT*)? (E [−+]? DIGIT+)? | '.' DIGIT+ (E [−+]? DIGIT+)?; STRING_LITERAL: '\'' (~'\'' | '\'\'')* '\'' ; SPACES:[\u000B\t\r\n] −> channel(HIDDEN); fragment DIGIT : [0-9]; fragment E : [eE];	select = sp 'select' sp column sp (',' sp column sp)* ('from' sp from_clause sp ('where' sp expr sp)? ('order' sp 'by' sp IDENTIFIER sp (',' sp IDENTIFIER sp)*)? ('limit' sp NUMERIC_LITERAL sp)?)?; table_or_subquery = table_name sp ('as'? sp name sp)? / '(' select ')' ('as'? sp name sp)? ; from_clause = table_or_subquery sp ('inner'? sp 'join' sp table_or_subquery 'on' sp expr sp)*; value = sp constv sp / sp name ('.' name)? sp / sp '(' sp expr sp ')' sp; expr = sp add (('=' / '==' / '!=' / '<>' /'<' / '<=' / '>' / '>=') add sp)+ / sp add sp; add = sp mlt (('+' / '−') mlt sp)+ / sp mlt; mlt = sp value (('*' / '/' / '%') value sp)+ / sp value sp; column = '*' / name (('.' '*')/('as'? sp name sp))? / '('select')' sp ('as'? sp name sp)? / expr sp ('as'? sp name sp)? ; table_name = name ; name = IDENTIFIER / STRING_LITERAL / '(' name ')' ; constv = NUMERIC_LITERAL / STRING_LITERAL; IDENTIFIER = '"' [^\"]* '"' / '[' [^\]]* ']' / !(keyword sp) [a-zA-Z_] [a-zA-Z_0-9]* ; keyword = 'select'/'as'/'from'/'inner'/'join'/'order'/'by'/'where' NUMERIC_LITERAL = DIGIT+ ('.' DIGIT*)? (E [−+]? DIGIT+)? / '.' DIGIT+ (E [−+]? DIGIT+)? ; STRING_LITERAL = '\'' [^\']* '\'' ; DIGIT = [0-9]; E = [eE]; sp = [\t\r\n]*

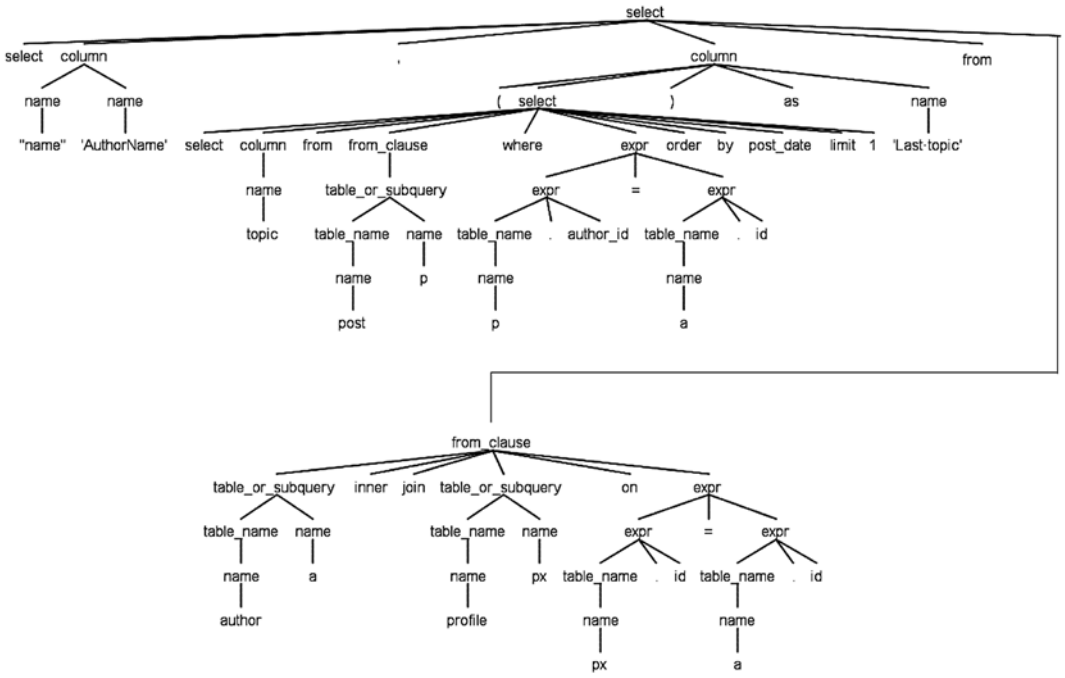

Рис. 2. Синтаксическое дерево, сгенерированное ANTLR

```
table_or_subquery
 = t:table_name sp m:( 'as'? sp n:name sp {return n;})? { t.alias=m; return t;}
 / '(' q:select ')' m:( 'as'? sp n:name sp {return n;})?
{return {type:"subquery",query:q,alias:m};};
```

Рис. 3. Код JavaScript, добавленный в текст PEG.js-грамматики анализируемого SQL-выражения

серов PEG.js оператор "/" имеет семантику "упорядоченный выбор". Таким образом, в PEG.js при изменении порядка альтернатив в грамматике для одной и той же входной строки могут быть получены существенно различные синтаксические деревья, в то время как ANTLR не допускает неоднозначных грамматик.

ANTLR-грамматику анализируемого SQL-выражения, представленную в таблице, можно изобразить в виде синтаксического дерева (рис. 2). PEG.js-грамматику визуализировать средствами IDE аналогичным образом не удается. Отметим, что здесь непосредственно в текст грамматики дописывается JavaScript обработчиков приведенный на рис. 3 код (вставки кода JavaScript выделены подчеркиванием). В ANTLR также существует поддержка аннотирования грамматики обработчиками на хост-языке.

PEG.js позволяет представить результат синтаксического разбора SQL-выражения в виде следующего JSON-кода:

{"type":"select","columns":[{"type":"column","modifier":{"type":"alias","name":"AuthorName"},"name":"name"},{"type":"subquery","modifier":{"type":"alias","name":"Last topic"},"query":{"type":"select","columns":[{"type":"column","modifier":null,"name":"topic"}],"from":{"tables":{"table":{"type":"table","name":"p"},"joins":[]},"order":"post_date","limit":"1"}}}],"from":{"tables":{"table":{"type":"table","name":"a"},"joins":[{"type":"join","inner":true,"table":{"type":"table","name":"px"},"condition":{"type":"binary","left":{"type":"reference","name":"px","modifier":"id"},"right":{"type":"reference","name":"a","modifier":"id"},"sign":"="}}]},"order":null, "limit":null}}

и синтаксического дерева, фрагменты которого представлены на рис. 4.

Приведенный выше анализ текстов ANTL-грамматики и PEG.js-грамматики тестового запроса к СУБД, представленных в таблице, позволяет сделать вывод о том, что они оказываются достаточно похожими друг на друга. Однако более тщательный анализ обнаруживает их существенные различия.

Генератор парсеров PEG.js имеет минимальный порог вхождения. Для начала работы с ним достаточно перейти по ссылке [13]. После этого программист сразу получает готовую учебную грамматику разбора математических выражений и доступ к нескольким грамматикам, созданным сообществом пользователей PEG.js (CSS, JavaScript, JSON и др.).

В значительной степени работу с PEG.js-грамматиками упрощает применение в PEG.js динамического языка программирования, что позволяет писать код, выполняемый при разборе, на хост-языке непос-

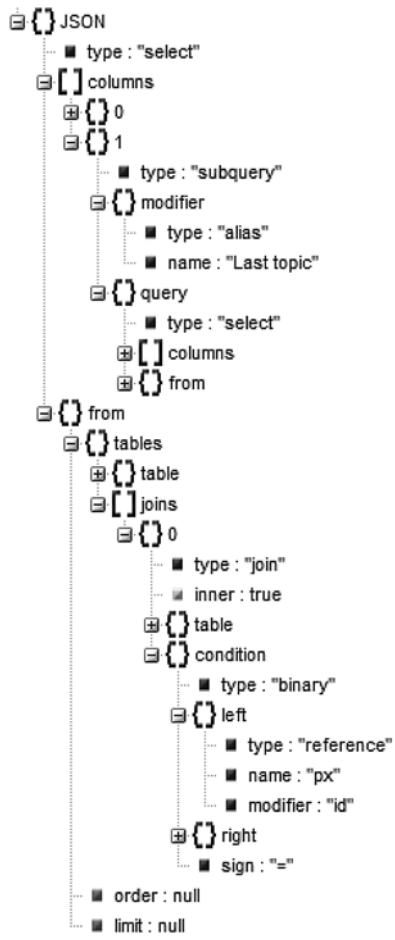

Рис. 4. Фрагменты синтаксического дерева, сгенерированного PEG.js

редственно в правилах грамматики. Каждый последовательный блок правила по умолчанию имеет обработчик, который возвращает массив всех вложенных элементов. Для представления синтаксического дерева в наиболее удобном виде при разборе определенной конструкции достаточно указать интересующие программиста признаки. Например, узел со свойством "type: "join"" отображает синтаксическое поддерево, содержащее информацию о SQL-операции соединения без текстовых фрагментов. При этом смысл скрытых фрагментов текста PEG.js-грамматики остается понятным программисту, так как указываются тип и другие свойства узла синтаксического дерева.

В генераторе парсеров PEG.js для каждого правила разбора арифметических выражений можно одновременно с процессом синтаксического анализа запускать обработчик, выполняющий действие, соответствующее разобранному правилу. Это позволяет программисту одновременно видеть и как выражение, написанное на языке программирования, представляется в виде синтаксического дерева, и как это дерево далее обрабатывается. Отметим, что для создания абстрактного синтаксического дерева на языке Java для рассматриваемой тестовой грамматики потребуется реализовать как минимум 11 классов, обеспечив при этом их взаимодействие.

Разработка в генераторе парсеров PEG.js происходит в интерактивном режиме в браузере, что оказывается достаточно удобным для разработчика. Вместе с тем во время работы с генератором парсеров PEG.js могут возникать определенные трудности. Например, если причины возникновения бесконечного цикла или переполнения стека, приводящие к закрытию страницы браузер и потере данных, обусловлены правилами грамматики, то среда разработки корректно обрабатывает данные ситуации. Однако можно создать бесконечный цикл в коде обработчика, например, случайно поставив точку с запятой "for(;;);{/*некоторое условие выхода*/}", что приведет к немедленной потере части выполненной работы.

Результатом работы генератора парсеров PEG.js является текст на языке JavaScript. Для промышленно эксплуатируемой системы это означает, что потребуется либо развернуть JavaScript на сервере (авторы PEG.js рекомендуют использовать Node.js), либо реализовать обработку запросов пользователя в браузере клиента. Отметим, что второй подход создает потенциальную угрозу безопасности информационной системы, которая может быть нарушена, например, SQL-инъекцией.

Приведем простой пример информационной системы, в которой возможно распространение SQL-инъекции. Предположим, что имеется информационная система, в которой пользователь вводит запросы на выбранном языке программирования. Далее проводится их синтаксический анализ парсером PEG.js, результаты которого обработчики парсера PEG.js преобразуют в SQL-запрос, который затем выполняется на сервере. При этом каждый пользователь считается доверенным лицом, имеющим право открывать отладчик браузера и выполнять любые запросы вплоть до "drop database ...". Отметим, что в подобных случаях один из возможных подходов, позволяющий снизить угрозы безопасности информационной системы, состоит в передаче на сервер только синтаксического дерева и его компиляции уже на сервере с соответствующей валидацией безопасности выполнения.

Отметим, что в генераторе парсеров ANTLR версии 4 появилась поддержка прямой левой рекурсии в правилах разбора [16]. Напомним, что прямая левая рекурсия грамматики означает наличие правил, которые предполагают наличие самих себя слева внутри своего описания. Например, в представленной тестовой грамматике правило *expr* может разбирать выражения, состоящие слева из выражения *expr*, затем из знака арифметической операции и после этого снова из выражения *expr*.

В генераторе парсеров PEG.js поддержка левой рекурсии грамматики (как и в большинстве других генераторов парсеров) отсутствует. Как следствие, в PEG.js-грамматике реализованы различные правила для результатов сложения и вычитания, умножения и деления, логических операций, применения скобок. Более того, в PEG.js-грамматике в силу левой ассоциативности операций вычитания и деления не представляется возможным описать эти операции как операции с двумя элементами. Например, результатом синтаксического

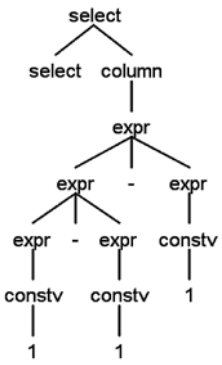

Рис. 5. Разбор левоассоциативных операций леворекурсивной грамматикой

анализа выражения "select 1-1-1" генератором парсеров ANTLR версии 4 будет синтаксическое дерево, представленное на рис. 5, в то время как генератор парсеров PEG.js вернет вместо истинного значения данного выражения, равного минус 1, значение 1.

В связи с этим в PEG.js-грамматике операцию сложения/вычитания определяют как серию из множества аргументов, но не только с двумя (левым и правым) аргументами: "add = sp mlt (('+'/'−') mlt sp)+/sp mlt ;". Отметим, что данный подход позволяет избавиться от унарного оператора инверсии арифметических выражений.

Автоматическое восстановление сгенерированного ANTLR-парсера после сбоев является еще одной его важной особенностью. Например, в случае разбора запроса "select 1++1 from test_tabe" генератор парсеров ANTLR сможет распознать оставшуюся часть запроса (рис. 6), в то время как генератор парсеров PEG.js укажет только место ошибки, поэтому для получения сходного результата разработчику грамматики, использующему генератор парсеров PEG.js, необходимо предусматривать возможные ошибки пользователя и соответствующие механизмы восстановления парсера в случаях их возникновения. С точки зрения авторов, такой подход при решении задачи прототипирования языка является вполне приемлемым, однако требующим от программиста дополнительных усилий.

Таким образом, язык описания грамматик является выразительным средством описания языка. При создании языка программирования применение генератора парсеров неизбежно приводит разработчика к стремлению сделать грамматику компактной. Мышление разработчика в терминах грамматики языка заранее дает ему возможность реализовать этот язык.

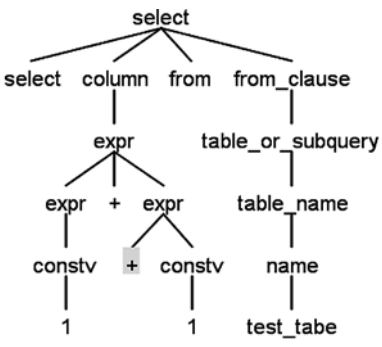

Рис. 6. Восстановление парсера ANTLR после сбоя

Можно ожидать, что для языка, созданного в соответствии с идеями грамматики языка, при разработке его синтаксического анализатора не должно возникать каких-либо трудностей. В то же время при его реализации приходится вносить в программирование новое, вообще говоря, нетехническое понятие — компактность грамматики. Вместе с тем наличие у языка компактной грамматики отнюдь не означает, что данный язык с точки зрения его изучения и использования окажется более понятным, чем язык программирования, имеющий "некомпактную грамматику".

Также демонстрацией ограничения генераторов парсеров являются трудности в реализации идеи адаптивной грамматики [17] средствами генератора парсеров. Этот факт означает, что если программа сама является описанием языка программирования, на котором она написана, сгенерированный синтаксический анализатор может разбирать только очень высокоуровневые фрагменты программы. Однако идеи адаптивных грамматик не распространены в разработке программного обеспечения промышленного применения.

Заключение

Проведенный анализ особенностей тестовых грамматик, построенных генераторами парсеров ANTLR и PEG.js, позволяет сделать следующие выводы.

1. Генератор парсеров PEG.js целесообразно использовать для решения следующих задач:

- обработка идей и общих концепций проектируемого языка;
- создание макета языка программирования с частичной реализацией функциональных возможностей либо с прямой компиляцией в существующий язык сходной парадигмы;
- создание парсера языка программирования, проверяющего дополнительные ограничения на языке программирования[3].

2. Генератор ANTLR применим для разработки промышленно используемых языков программирования, призванных обеспечить компромисс между идеями программиста и вычислительными возможностями компьютера.

Исследование проведено в рамках создания языка запросов данных для модуля подготовки данных системы моделирования выпуска металлургической продукции [18].

Работа выполнена в рамках договора № 02.G25.31.0055 (проект 2012-218-03-167) при финансовой поддержке работ Министерством образования и науки Российской Федерации.

[3] Например, в системе анализа данных достаточно реализовать экспертный и простой режим работы. В экспертном режиме пользователь будет создавать запросы при помощи SQL или MDX, которые затем будут сохраняться в соответствующей БД информационной системе в виде SQL\MDX. При загрузке запроса из БД запросов информационная система, применяя к нему парсер, будет определять, применим ли для его отображения простой пользовательский интерфейс.

Список литературы

1. **DSL** Development Environment [Электронный ресурс]. URL: http://www.jetbrains.com/mps (дата доступа 31.07.2014).
2. **Standard** ECMA-335 Common Language Infrastructure (CLI) [Электронный ресурс]. URL: http://www.ecma-international.org/publications/standards/Ecma-335.htm (дата доступа 31.07.2014).
3. **Asm.js** Working Draft. [Электронный ресурс]. URL: http://asmjs.org/spec/latest/ (дата доступа 31.07.2014).
4. **Elliott J**. Hibernate: A Developer's Notebook. 1st ed. O'Reilly Media, 2004. 190 p.
5. **Ахо А., Сети Р., Ульман Дж.** Компиляторы. Принципы, технологии, инструменты М.: Вильямс, 2003. 1184 с.
6. **Parr T., Fisher K. S.** LL(*): The Foundation of the ANTLR Parser Generator // Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation. 2011. P. 425—436.
7. **Hutton G., Meijer E.** Monadic parsing in Haskell // Journal of Functional Programming. 1998. Vol. 8, N. 4. P. 437—444.
8. **.NET** Compiler Platform ("Roslyn") [Электронный ресурс]. URL: https://roslyn.codeplex.com/ (дата доступа 31.07.2014).
9. **The A-Z** of Programming Languages: YACC [Электронный ресурс]. URL: http://www.techworld.com.au/article/252319/a-z_programming_languages_yacc/? (дата доступа 31.07.2014).
10. **Comparison** of parser generators [Электронный ресурс]. URL: http://en.wikipedia.org/wiki/Comparison_of_parser_generators (дата доступа 31.07.2014).
11. **Ford B.** Packrat Parsing: a Practical Linear-Time Algorithm with Backtracking. PhD thesis, Massachusetts Institute of Technology. 2002. 112 p.
12. **Online** version PEG.js [Электронный ресурс]. URL: http://pegjs.majda.cz/online (дата доступа 31.07.2014).
13. **Grammars** written for ANTLR v4 [Электронный ресурс]. URL: https://github.com/antlr/grammars-v4 (дата доступа 31.07.2014).
14. **SQL** parser in JavaScript [Электронный ресурс]. URL: https://github.com/steveyen/sqld3 (дата доступа 31.07.2014).
15. **Kozen D.** On Kleene algebras and closed semirings // Proc. Math. Found. Comput. Sci. Lecture Notes in Computer Science. 1990. Vol. 452. P. 26—47.
16. **Moore R. C.** Removing Left Recursion from Context-Free Grammars // Microsoft Research, Association for Computational Linguistics. 2000. P. 249—255.
17. **Henning C.** A Survey of Adaptable Grammars // ACM SIGPLAN Notices. 1990. Vol. 25, N. 11. P. 35—44.
18. **Aksyonov K. A., Bykov E. A., Aksyonova O. P., Antonova A. S.** Development of real-time simulation models: integration with enterprise information systems // Proceedings of ICCGI 2014: The Ninth International Multi-Conference on Computing in the Global Information Technology. Sevilla. 22—26 June 2014. P. 45—50.

A. M. Borodin, Associate Professor, e-mail: amborodin@acm.org, **S. G. Mirvoda,** Assistant, e-mail: sergey@mirvoda.com,
S. V. Porshnev, Head of Department, Associate Professor, e-mail: sergey_porshnev@mail.ru,
Ural Federal University named after the first President of Russia B. N. Yeltsin, Yekaterinburg

Analysis of Language Prototyping Toolset

Syntax definition is important stage of programming language design. This stage forms expressive capabilities and language clearance. These capabilities can help programming language to conquer minds of software developers.

This paper discuss state of text programming languages and result of comparative analysis of modern grammar development toolset. Two main technologies of parser generators are compared in view of programming language prototyping. Analysis is based on comparison of two simple grammars. These grammars are designed to implement some structured query language (SQL) features. Though simple this grammars are capable of parsing most of common select SQL statements. Advantages and disadvantages of parser generators technologies are discussed on example of ANTLR and PEG.js.

In addition, this paper describe arguments towards text programming languages. These arguments are given in the context of vertical integration of many different programming languages into technical stack.

Keywords: *syntax analisys, context-free grammar, parsing expression grammar, programming language prototyping*

References

1. **DSL** Development Environment. URL: http://www.jetbrains.com/mps
2. **Standard** ECMA-335 Common Language Infrastructure (CLI). URL: http://www.ecma-international.org/publications/standards/Ecma-335.htm
3. **Asm.js** Working Draft. URL: http://asmjs.org/spec/latest/
4. **Elliott J.** *Hibernate: A Developer's Notebook*. 1st ed. O'Reilly Media, — 2004. 190 p.
5. **Aho A., Seti R., Ul'man J.** *Kompiljatory. Principy, tehnologii, instrument*. M.: Vil'jams, 2003. 1184 p.
6. **Parr T., Fisher K. S.** LL(*): The Foundation of the ANTLR Parser Generator. *Proceedings of the 32nd ACM SIGPLAN conference on Programming language design and implementation*. 2011. P. 425—436.
7. **Hutton G., Meijer E.** Monadic parsing in Haskell. *Journal of Functional Programming*. 1998. Vol. 8, N. 4. P. 437—444.
8. **.NET** Compiler Platform ("Roslyn"). URL: https://roslyn.codeplex.com/
9. **The A-Z** of Programming Languages: YACC. URL: http://www.techworld.com.au/article/252319/a-z_programming_languages_yacc/?
10. **Comparison** of parser generators. URL: http://en.wikipedia.org/wiki/Comparison_of_parser_generators
11. **Ford B.** Packrat Parsing: a Practical Linear-Time Algorithm with Backtracking. PhD thesis. Massachusetts Institute of Technology. 2002. 112 p.
12. **Online** version PEG.js. URL:http://pegjs.majda.cz/online
13. **Grammars** written for ANTLR v4. URL:https://github.com/antlr/grammars-v4
14. **SQL** parser in JavaScript. URL: https://github.com/steveyen/sqld3
15. **Kozen D.** On Kleene algebras and closed semirings. *Proc. Math. Found. Comput. Sci., Lecture Notes in Computer Science*. 1990. Vol. 452. P. 26—47.
16. **Moore R. C.** Removing Left Recursion from Context-Free Grammars. *Microsoft Research, Association for Computational Linguistics*. 2000. P. 249—255.
17. **Henning C.** A Survey of Adaptable Grammars. *ACM SIGPLAN Notices*. 1990. Vol. 25, No. 11. P. 35—44.
18. **Aksyonov K. A., Bykov E. A., Aksyonova O. P., Antonova A. S.** Development of real-time simulation models: integration with enterprise information systems. *Proceedings of ICCGI 2014: The Ninth International Multi-Conference on Computing in the Global Information Technology*. Sevilla. 22—26 June 2014. P. 45—50.

УДК 004.415.2

Ю. Л. Гик, зам. нач. отдела, e-mail: gikjuri@yandex.ru, АКБ "Росевробанк" (ОАО), г. Москва

Использование языка программирования Groovy в интеграционной шине Sonic ESB

Интеграционные шины получили распространение в крупных организациях и используются для решения задач интеграции информационных систем. Сложные задачи интеграции требуют особых подходов и для их решения удобно использовать специализированные языки программирования. Настоящая работа посвящена решению задач на интеграционной шине Sonic ESB с помощью языка программирования Groovy. Ключевое внимание уделено вопросам удобства применения языка и вопросам вычислительной производительности. Приведены результаты сравнительного тестирования производительности сервисов, написанных на стандартных компонентах Sonic ESB, Java и Groovy. Даны рекомендации по применению различных технологий в интеграционной шине Sonic ESB.

Ключевые слова: *Groovy, ESB, Sonic ESB, интеграционная шина, скриптовые языки программирования, динамические языки программирования*

Введение

Интеграционные шины (*Enterprise Service Bus*, ESB) являются важной составляющей ландшафта современных корпоративных информационных систем, применяемых на крупных предприятиях. Задачи интеграции, для решения которых используют такие шины, как правило, сложны и требуют значительных интеллектуальных затрат на их реализацию. Современные ESB обычно включают мощные наборы стандартных компонентов для решения типовых задач интеграции информационных систем. Они предоставляют возможность разрабатывать нестандартные компоненты (сервисы) с помощью универсальных языков программирования, таких как Java или C, не всегда удобных для решения этих задач.

Специализированный язык программирования Groovy хорошо подходит для решения задач интеграции информационных систем в силу сокращенного синтаксиса и удобных средств работы с XML, файлами и базами данных.

Целью исследования, результаты которого представлены в статье, является анализ возможностей применения Groovy в интеграционной шине Sonic ESB, а также анализ вычислительной производительности ее базовых программных компонентов. Анализ проводился по результатам тестовых испытаний, на основе сравнения производительности приложений, аналогичных основным сервисам Sonic, однако написанных на языках Groovy и Java. На основе анализа производительности решений и удобства разработки даны рекомендации по применению различных технологий в рамках Sonic ESB.

Язык программирования Groovy

Язык программирования Groovy появился в 2007 г. как один из динамических языков на платформе *Java Virtual Machine* (JVM). Он имеет синтаксис, похожий на синтаксис языка Java, но многие синтаксические конструкции в Groovy упрощены по сравнению с языком Java. Конструкции Groovy можно представить как "синтаксический сахар" аналогичных конструкций Java — поэтому разработчики на Java обычно достаточно быстро осваивают Groovy, к тому же, в Groovy-проектах можно смешивать код Groovy и Java. Производительность труда разработчика приложений на Groovy обычно выше производительности труда разработчика на Java, однако исполнение программного кода на Groovy происходит существенно медленнее аналогичного кода на Java.

Groovy является полноценным объектно-ориентированным языком с возможностью исполнения скриптов, т. е. для исполнения кода в Groovy необязательно оформлять классы. В Groovy можно использовать замыкания — функции, в теле которых могут присутствовать ссылки на переменные, объявленные вне параметров, и тела этих функций. Эта возможность позволяет сократить объем кода.

Язык программирования Groovy используется в популярных инструментальных средствах разработки веб-приложений Grails, приложений Griffon, сборки проектов Gradle. В основных для мира Java интеграционных средах разработки приложений Eclipse и Netbeans существуют модули поддержки Groovy, позволяющие легко компилировать и исполнять его код.

Создатели языка Groovy первоначально имели желание с помощью него вытеснить язык программирования Java. Такого не произошло, однако свое место в инструментарии разработчика Groovy нашел. Этот

язык программирования, как правило, используется при выполнении проектов для быстрого создания прототипов приложений, а также для кодирования их функциональных возможностей, не требующих высокой производительности.

Язык программирования Groovy является живым и динамично развивающимся. На нем частично реализованы такие большие проекты, как интернет-портал linkedin и инструмент нагрузочного тестирования LoadUI. Поддержка Groovy встроена в среду веб-разработки JBoss Seam. Он является основным скриптовым языком в популярной среде тестирования веб-сервисов SOAPUI.

Про язык программирования Groovy издан ряд книг [1, 2]. Знакомству с этим языком посвящены выступления и мастер-классы на последних отраслевых конференциях: JPoint 2014[1] (Москва, 18 апреля 2014 г., выступление Баруха Садогурского "Мета-программирование на уровне компилятора в Groovy" и его же мастер-класс по Groovy) и JEEConf 2014 (Киев, 21—24 мая 2014 г., мастер-класс Евгения Борисова "Groovy и Grails для Java-разработчиков", выступление Питера Ледбрука "Groovy для Java-разработчиков"). Отмеченные факты свидетельствуют о растущем интересе к Groovy у разработчиков программного обеспечения.

Слабым местом Groovy является скорость исполнения кода. Код на Groovy приблизительно в 10 раз медленнее аналогичного кода на Java. Проблемы с производительностью являются платой за удобство разработки. Язык Groovy с динамической типизацией, транслятор на этапе компиляции не знает о реальных типах данных в коде. Это обстоятельство — основная причина малой производительности.

Производительности Groovy и смежным вопросам перевода кода из Groovy в Java посвящен ряд работ. В работе [3], например, представлено решение задачи портирования кода Groovy в Java с анализом соответствий основных семантических конструкций Groovy и Java.

В статье [4] проведено сравнение последних на середину 2013 г. версий Java и Groovy. Проделав тесты на базовых алгоритмах (быстрая сортировка, Fork/Join и др.), автор пришел к выводу, что такие приложения на Groovy в 2—8 раз медленнее, чем на Java.

Интеграционная шина Sonic ESB

Дэвид А. Шаппелл, один из архитекторов Sonic ESB и признанный в отрасли эксперт по интеграции информационных систем, дает следующее определение интеграционной шины: "Сервисная шина предприятия — это основанная на стандартах интеграционная платформа, объединяющая обмен сообщениями, веб-сервисы, преобразование данных и интеллектуальную маршрутизацию. Все эти средства необходимы для обеспечения надежного соединения и координации взаимодействия значительного числа разнообразных приложений в пределах расширенного предприятия с транзакционной целостностью" [5].

На сайте производителя Sonic ESB[2] содержится следующее определение интеграционной шины: "Предприятия сегодня сталкиваются с проблемами несовместимости и неэффективности, что препятствует дальнейшим инновациям и росту. Следствием слияний и приобретений, а также разрозненных бизнес-инициатив является зоопарк несовместимых технологий, плохо приспособленных для изменяющихся нужд бизнеса. Эти проблемы усугубляются организационными, культурными и географическими ограничениями.

Интеграционная шина Sonic ESB отвечает на эти вызовы, предоставляя доступ организациям к старым и новым бизнес-системам, независимо от технологии и месторасположения, а также давая возможность создавать новые бизнес-сервисы, способные удовлетворить новые потребности" (перевод автора).

Отметим, что определение интеграционной шины не является четким и унифицированным. Многие компании-производители интеграционных шин и специалисты по интеграции дают свои определения и соответствующий различный набор функциональных возможностей. Так, Томас Эрль, эксперт по вопросам сервис-ориентированной архитектуры (СОА) и интеграции информационных систем, определяет интеграционную шину как один из составных шаблонов СОА.

Интеграционная шина предназначена для связи между собой информационных систем и облегчения решения сопутствующих задач, прежде всего, преобразования данных, протоколов, а также для маршрутизации сообщений. В Sonic ESB для решения этих задач существуют встроенные компоненты (сервисы), реализующие:

- преобразование XML-сообщений (*XSL Transform* или XSLT);
- маршрутизацию по контенту (*content based routing* или CBR);
- доступ к СУБД (*DBservice*);
- доступ к файлам (*File Service*);
- доступ к веб-сервисам (*Web Service*).

Этого набора компонентов хватает для решения большинства типовых задач, которые возникают при интеграции информационных систем.

Программный комплекс Sonic ESB является исторически первой интеграционной шиной. В России Sonic ESB успешно конкурирует с интеграционными шинами других производителей, прежде всего IBM. С помощью Sonic ESB решают задачи интеграции такие крупные организации, как банк ВТБ24, Россельхозбанк, РосЕвроБанк, сеть магазинов "Спортмастер".

Производительность интеграционных шин — тема достаточно сложная вследствие закрытого характера коммерческих интеграционных шин: компании-производители публиковать исследования по производительности не любят, а для внешнего тестирования не-

[1] http://javapoint.ru

[2] http://www.aurea.com/technology-solutions/enterprise-service-bus/

обходимо покупать дорогую лицензию. В частности, вопросы производительности Sonic ESB не исследованы. Существуют исследования производительности открытых (*open source*) интеграционных шин, реже проводится сравнение производительности открытых и закрытых ESB. Кратко отметим наиболее интересные работы, обратив внимание на представленную в них методику тестирования.

В работе [6] проведено сравнение производительности интеграционных шин ServiceMix, Mule и JBoss ESB на примере сервиса кредитного бюро, обращающегося к внешнему веб-сервису. Измерялись время отклика и производительность шины (число транзакций в секунду). Каждый тест исполнялся не менее 10 раз с разным числом одновременно работающих пользователей. Тесты исполнялись как из клиентских приложений, вызывающих сервисы, так и из оркестровщика сервисов BPEL (*Apache ODE*).

В работе [7] описано попарное тестирование Mule, WSO2 и ServiceMix. Методика тестирования идентична методике, описанной в работе [6], но было добавлено тестирование с разными размерами сообщений для таких компонентов, как сквозные, маршрутизация по контенту и преобразование сообщений. По результатам тестирования сделан вывод об одинаковой в среднем производительности различных шин при одинаковой нагрузке.

В статье [8] проведено сравнение производительности интеграционных шин Mule и BEA AquaLogic на примере сервиса кредитного бюро, аналогичного представленному в работе [6]. Работа интересна как сравнение производительности открытой (Mule) и закрытой (BEA) интеграционных шин.

На сайте http://esbperformance.org можно найти детальный анализ производительности открытых интеграционных шин.

Groovy-сервис в Sonic ESB

Интеграционная шина Sonic ESB была разработана на Java и имеет развитый набор стандартных сервисов для задач интеграции информационных систем. Сервисы используются в качестве минимальных строительных элементов в ESB-процессах, указывающих маршрут движения сообщений по интеграционной шине. На рис. 1 представлен процесс с единственным сервисом — сервисом Groovy. Если возможностей стандартных сервисов не хватает, то по методологии разработки Sonic необходимо написать на языке программирования Java дополнительный сервис.

Sonic облегчает работу создания новых сервисов, предоставляя шаблон класса сервиса с автоматически сгенерированными основными методами.

Для написания Groovy-скриптов непосредственно в среде Sonic Workbench, построенной на Eclipse-технологии, можно использовать дополнение (plug-in) для Eclipse[3]. Это дополнение позволяет редактировать, отлаживать и исполнять Groovy-код непосредственно из среды разработки Sonic Workbench.

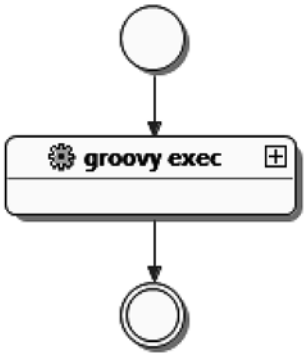

Рис. 1. ESB-процесс с Groovy-сервисом

```
try {
      binding.setVariable("env", env);
      shell.evaluate(groovy_content);
} catch (Exception e) {
      throw new XQServiceException(
      "Exception performing GROOVY script: "
      + e.getMessage(), e);
}
```

Рис. 2. Сервис исполнения Groovy-скриптов на Java (скрипт 1)

Для исследования функциональных возможностей Groovy автором был разработан на языке Java сервис Groovy (основная часть сервиса, помимо тривиальных считывания файла скрипта и записи в строковую переменную groovy_content, рис. 2, скрипт 1), позволяющий загружать и исполнять файл со скриптом Groovy. Сервис принимает путь с именем файла скрипта в качестве параметра. При ошибке исполнения соответствующее сообщение записывается в журнал (лог) контейнера исполнения сервиса.

Команда shell.evaluate осуществляет исполнение строки Groovy_content как скрипта, содержащего код на Groovy. Сервис требует подключения следующих библиотек для Groovy 2.2: Antlr-2.7.7.jar, Asm-4.1.jar, Groovy-2.2.0.jar, Groovy-xml-2.2.0.jar (для обработки XML).

С помощью Groovy-сервиса можно провести тестирование производительности базовых возможностей интеграционной шины, а именно маршрутизацию по контенту и преобразование XML-сообщений. Целью тестирования является выяснение спектра интеграционных задач, которые можно решать на Groovy. Проведем сравнительное тестирование производительности Groovy-сервиса с аналогичными стандартными сервисами Sonic и аналогичными сервисами, написанными на Java. Также обратим внимание на удобство разработки сервисов на Groovy в сравнении с Java.

Тестирование проводилось по методике, аналогичной описанной в предыдущем разделе. Тестировались локальные ESB-процессы (чтобы исключить влияние сети). На точку входа через JMS-соединение подавалось XML-сообщение. Измерялось время прохождения этого сообщения по ESB-процессу между точка-

[3] http://groovy.codehaus.org/Eclipse+Plugin

ми входа и выхода. Тестирование велось с помощью инструментария JMeter сериями с разным числом последовательно отправляемых сообщений, имитирующих нагрузку (от 100 до 1000 сообщений). Каждая серия повторялась 10 раз, результатом являлось среднее значение времени прохождения сообщения в серии. Таким образом, получили зависимость времени прохождения сообщения по процессу от нагрузки и технологии разработки.

Характеристики тестовой среды: процессор Intel Core 2 Duo 2,93 ГГц; ОЗУ 3,18 ГБайт; операционная система Windows XP Professional SP3; жесткий диск 465 Гбайт; версия Sonic 8.5.1; версия Java 1.6.0_25; версия Groovy 2.2; версия JMeter 2.9.

Тестирование маршрутизации по контенту

Маршрутизация по контенту (*Content Based Routing*, CBR) — одна из основных функциональных возможностей интеграционных шин. Сообщение внутри интеграционной шины идет по определенному маршруту, внутри которого возможны точки ветвления по некоторым условиям, зависящим от содержимого сообщения. Сообщение обычно представляет собой XML-текст, для его разбора используется стандарт XML XPath. Для маршрутизации по контенту в Sonic ESB используется стандартный CBR-сервис с возможностью настройки XPath-условий на соответствующие маршруты (рис. 3). Проводя сравнение с традиционными языками программирования можно сказать, что CBR похож на конструкцию CASE (в вырожденном случае двух выходных точек маршрута — конструкцию IF), где условие выражено в форме XPath.

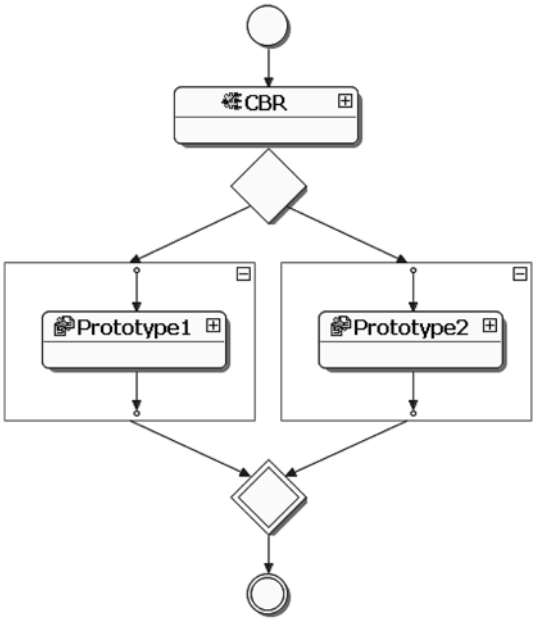

Рис. 3. Стандартный Sonic-процесс CBR с двумя точками ветвления

```
XQPart prt = msg.getPart(0);
String scontent = (String) prt.getContent(); //Берем содержимое сообщения с помощью Sonic API
Document doc = null;
DocumentBuilderFactory fact = DocumentBuilderFactory.newInstance();
fact.setIgnoringElementContentWhitespace(true);
DocumentBuilder builder;
try {
    fact.setValidating(false);
    builder = fact.newDocumentBuilder();
    doc = builder.parse(new InputSource(new StringReader(scontent))); //Определяем содержимое
сообщения как XML
} catch (SAXException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
} catch (ParserConfigurationException e) {
    e.printStackTrace();
}
String select_by_xpath = "//Service/@Name"; //XPath-условие
try {
    ServiceName = XPathAPI.selectSingleNode(doc.getDocumentElement(),
select_by_xpath).getNodeValue(); //Получаем информацию в переменную по XPath-условию
} catch (DOMException e2) {
    e2.printStackTrace();
} catch (TransformerException e2) {
    e2.printStackTrace();
}
NServiceName = Integer.parseInt(ServiceName);
} catch (XQMessageException me) {
    throw new XQServiceException("Exception accessing XQMessage: "
+ me.getMessage(), me);
}
if (NServiceName == 1) {
    XQPart part1 = null;
try {
    part1 = (XQPart) msg.getPart(0).clone();
} catch (XQMessageException e) {
    e.printStackTrace();
}
part1.setContentId("NewPart1"); //Устанавливаем заголовок сообщения по условию
part1.setContent("<choice1/>", "text/xml");
try {
    msg.addPart(part1);
} catch (XQMessageException e) {
    e.printStackTrace();
}
} else {
    XQPart part2 = null;
try {
    part2 = (XQPart) msg.getPart(0).clone();
} catch (XQMessageException e) {
    e.printStackTrace();
}
part2.setContentId("NewPart2"); //Устанавливаем заголовок сообщения по условию
part2.setContent("<choice2/>", "text/xml");
try {
    msg.addPart(part2);
} catch (XQMessageException e) {
e.printStackTrace();
}
```

Рис. 4. Маршрутизация по контенту на Java (скрипт 2)

Для анализа производительности CBR необходимо написать отдельный сервис на Java, имитирующий CBR, который затем будет встроен в соответствующий ESB-процесс.

Основная часть написанного автором кода Java-сервиса представлена на рис. 4 (скрипт 2).

Отметим большое количество шаблонного кода, связанного с обработкой XML, обработкой исключений и блоками транзакций try-catch. Фактически, к прикладной логике относятся лишь строки, выделенные полужирным шрифтом. Прикладной код теряется на фоне шаблонного кода, что может привести к неверным выводам программиста, читающего подобный текст. Вместе с тем необходимость писать в основном шаблонный код может привести к ошибкам — программист может написать его некорректно или просто часть его не написать. Стандартным решением вопроса избыточности шаблонного кода в Java является использование средств IDE, генерирующих этот

```
import com.sonicsw.xq.*
def env = binding.getVariable('env')
def xqmsg = env.getMessage()
def xqPart = xqmsg.getPart(0)
def cnt = xqPart.getContent().toString()
def Service = new XmlSlurper().parseText(cnt)
def ServiceName = Service.@'Name'
if (ServiceName == 1) {
        def part1=xqmsg.getPart(0).clone()
        part1.setContentId('NewPart1')
        part1.setContent('<choice1/>', "text/xml")
        xqmsg.addPart(part1)
}
else {
        def part2=xqmsg.getPart(0).clone()
        part2.setContentId('NewPart2')
        part2.setContent('<choice2/>', "text/xml")
        xqmsg.addPart(part2)
}
```

Рис. 5. Маршрутизации по контенту на Groovy (скрипт 3)

шаблонный код, как, например, шаблон сервиса в Sonic Workbench. Такой подход помогает при написании кода, но слабо помогает (скорее, даже усугубляет ситуацию) при чтении кода. Это обстоятельство служит одной из причин появления Groovy и других более лаконичных языков программирования.

Для тестирования производительности маршрутизации по контенту используем сервис Groovy (см. рис. 2, скрипт 1), который будет исполнять скрипт 3, представленный на рис. 5, выполняющий функцию маршрутизации по контенту.

Этот код фактически идентичен приведенному на рис. 4 коду на Java, выделенному полужирным цветом, т. е. в коде на Groovy отсутствует шаблонный код. Этот шаблонный код "появляется" при компиляции скрипта. При необходимости у разработчика есть возможность ввести необходимые исключения и try-catch блоки. Однако это целесообразно только для нестандартной обработки, стандартные исключения формируются на этапе компиляции автоматически. Для обработки XML-сообщений в скрипте используется объект XMLSlurper, способный удобно для разработчика читать XML-формат и понимающий XPath. Обработка XML является одним из основных преимуществ Groovy по сравнению с Java.

Результат тестирования производительности стандартного сервиса маршрутизации по контенту (рис. 2, скрипт 1), сервиса на Groovy (рис. 5, скрипт 3) и сервиса, написанного на Java (рис. 4, скрипт 2) приведен на рис. 6.

В первую очередь отметим, что сервис на Groovy в 5—7 раз медленнее сервиса на чистой Java и в 2—4 раза медленнее стандартного CBR-сервиса. Интересно также, что стандартный сервис Sonic в 2—3 раза медленнее сервиса, написанного на Java, что свидетельствует о его неэффективной реализации. Отметим также эффект деградации производительности при увеличении числа сообщений независимо от технологии разработки. Это можно объяснить особенностями конфигурации тестирования и нехваткой ресурсов при нагрузке.

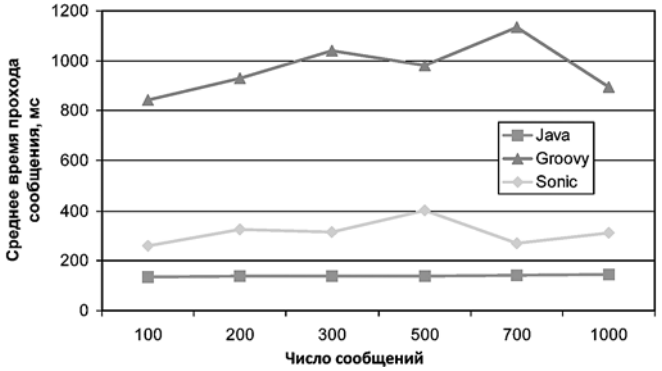

Рис. 6. Зависимость производительности CBR-процессов от числа сообщений

Тестирование преобразования XML

Также как и маршрутизация по контенту, преобразование XML является одной из основных функциональных возможностей интеграционных шин. Обычно преобразование делается в рамках интеграционного шаблона "Каноническое преобразование". Согласно этому шаблону, при поступлении сообщения в интеграционную шину его нужно преобразовывать в канонический формат. Движение по маршруту в шине сообщение проходит в этом каноническом формате. При обращении по маршруту к другим информационным системам или при возврате ответа в исходную информационную систему происходит преобразование формата сообщения из канонического формата в формат информационной системы.

Для преобразования XML обычно используется стандарт XSLT. В Sonic ESB существует специальный компонент для проведения XSLT-преобразований. В Groovy существует возможность работы с XSLT, но в скрипте будут использоваться конструкции для обычной работы с XML.

В ходе тестирования выполняется простое преобразование XML по замене названий тэгов, входное и выходное сообщения приведены на рис. 7, 8, скрипты 4, 5 соответственно.

```
<btask>
        <field1>1</field1>
        <field2>2</field2>
        <field3>3</field3>
</btask>
```

Рис. 7. Входное сообщение XML (скрипт 4)

```
<btask>
        <field_new1>1</field_new1>
        <field_new2>2</field_new2>
        <field_new3>3</field_new3>
</btask>
```

Рис. 8. Преобразованное сообщение XML (скрипт 5)

```
<xsl:stylesheet version="2.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
         <btask>
            <field_new1><xsl:value-of select="//field1"/></field_new1>
            <field_new2><xsl:value-of select="//field2"/></field_new2>
            <field_new3><xsl:value-of select="//field3"/></field_new3>
         </btask>
   </xsl:template>
</xsl:stylesheet>
```

Рис. 9. Трансформация XSLT (скрипт 6)

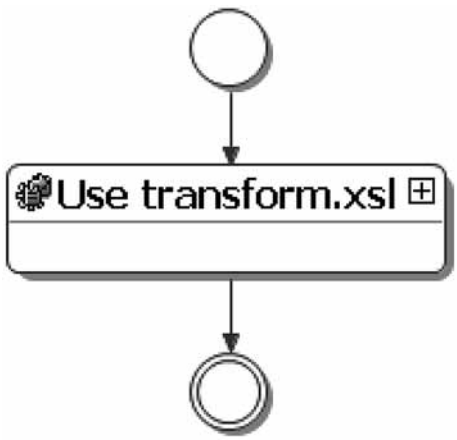

Рис. 10. ESB-процесс с сервисом XSL-трансформации

```
XQPart prt = msg.getPart(i);
Object content = prt.getContent();
                m_xqLog.logInformation("Content at Part [" + i + "]:\n" + content);
                String xslFile = "c:\\tmp\\transform.xsl";
                InputStream xslInputStream = null;
try {
     xslInputStream = new FileInputStream(xslFile);
} catch (FileNotFoundException e) {
     e.printStackTrace();
}
Source xslSource = new StreamSource(xslInputStream);
TransformerFactory factory = TransformerFactory.newInstance();
Transformer transformer = factory.newTransformer(xslSource);
Reader reader = new StringReader((String) content);
StreamSource in = new StreamSource(reader);
Writer writer = new StringWriter();
 StreamResult out = new StreamResult(writer);
try {
     transformer.transform(in, out);
} catch (TransformerException e) {
     e.printStackTrace();
}
prt.setContent(writer.toString(), "test/xml");
```

Рис. 11. Сервис преобразования на Java (скрипт 7)

Для выполнения преобразования служит простая трансформация, рис. 9, скрипт 6.

Процесс со стандартным сервисом трансформации Sonic показан на рис. 10. Файл с XSLT-преобразованием (см. рис. 9, скрипт 6) указывается в качестве настройки этого сервиса.

Для анализа производительности XSLT необходим сервис, написанный на Java, код которого представлен на рис. 11, скрипт 7.

```
import com.sonicsw.xq.*
def env = binding.getVariable('env')
def xqmsg = env.getMessage()
def xqPart = xqmsg.getPart(0)
def cnt = xqPart.getContent().toString()
def btask = new XmlSlurper().parseText(cnt) //Разбираем входящий XML
def field1 = btask.field1
def field2 = btask.field2
def field3 = btask.field3
writer = new StringWriter()
builder = new groovy.xml.MarkupBuilder(writer) //Создаем новый
преобразованный XML
builder.btask{
  field1_new(field1)
  field2_new(field2)
  field3_new(field3)
}
xqPart.setContent(writer.toString(), "text/xml")
```

Рис. 12. Скрипт Groovy по преобразованию XML (скрипт 8)

Код, касающийся непосредственно преобразования, выделен на рис. 11 полужирным шрифтом. Этот код более лаконичен по сравнению с аналогичным кодом Java по CBR, но в нем также присутствует шаблонный код, не имеющий прямого отношения к прикладной логике программы. Лаконичность объясняется использованием библиотеки javax.xml.transform (JAXP).

Для анализа производительности XSLT нужен также сервис Groovy (см. рис. 2, скрипт 1), исполняющий скрипт 8, представленный на рис. 12.

Скрипт 8 не использует XSLT, для такого простого преобразования в Groovy оно избыточно. Для разбора XML используется объект Groovy XmlSlurper (см. рис. 4, скрипт 2). Для формирования нового XML-сообщения используется объект Groovy xml.MarkupBuilder.

Результаты тестирования сервисов XSLT на Java, Groovy и стандартного Sonic-сервиса приведены на рис. 13.

Можно заметить фактически одинаковую производительность чистого Java-сервиса и стандартного сервиса Sonic (расхождение не более 20 % при небольших нагрузках). Расхождение более 50 % на 1000 сообщений можно объяснить недостатком ресурсов рабочей станции, где проводилось тестирование. Сервис Groovy в 6—7 раз медленнее сервиса на Java и в 4—6 раз медленнее стандартного сервиса Sonic.

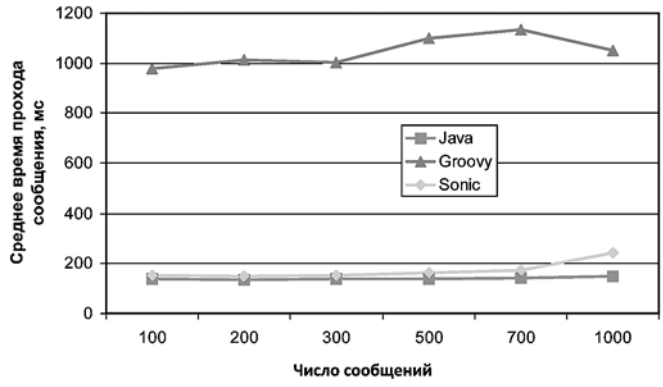

Рис. 13. Зависимость производительности процессов трансформации XML от числа передаваемых сообщений

Рассмотрим более сложный пример. Стандарт XSLT достаточно громоздок. Некоторые задачи преобразования XML-сообщения он решает довольно сложным способом, другие не решает вообще. Например, задача добавления константы ко всем содержимым тэгов в XSLT решается громоздко. Нужно учитывать структуру входящего XML-сообщения, изящное решение нетривиально.

Предположим, существует входной файл (рис. 14, скрипт 9).

Необходимо добавить константу "Luxury" к каждому элементу и получить XML-файл (рис. 15, скрипт 10).

```
<shopping>
        <category type="groceries">
            <item>Chocolate</item>
            <item>Coffee</item>
        </category>
</shopping>
```

Рис. 14. Входящий XML-файл для сложного преобразования (скрипт 9)

```
<shopping>
  <category type="groceries">
    <item>
     Luxury Chocolate
    </item>
    <item>
     Luxury Coffee
    </item>
  </category>
</shopping>
```

Рис. 15. Результирующий XML-файл для сложного преобразования (скрипт 10)

```
class gj_class1 {
    public static void main(String[] args) {
        def input = '''
        <shopping>
            <category type="groceries">
                <item>Chocolate</item>
                <item>Coffee</item>
            </category>
        </shopping>
        '''
        def root = new XmlParser().parseText(input)
        // modify groceries
        def groceries = root.category.findAll{ it.@type == 'groceries'
}.item[0]
        groceries.each { g ->
            g.value = 'Luxury ' + g.text()
        }

        def writer = new StringWriter()
        new XmlNodePrinter(new PrintWriter(writer)).print(root)
        def result = writer.toString()
        println result
    }
}
```

Рис. 16. Скрипт на Groovy для сложного XML-преобразования (скрипт 11)

Этой цели позволяет добиться скрипт 11, написанный на Groovy, представленный на рис. 16.

Для разбора XML-сообщений в скрипте используется Groovy-объект XmlParser, который позволяет в цикле найти необходимые элементы и обновить их.

Таким образом, Groovy может помочь не только при написании специализированных сервисов за счет снижения объема шаблонного кода по сравнению с Java, но и в решении нестандартных задач XML-преобразований. Отметим, что на задачи преобразования XML-сообщений уходит обычно основное время разработчиков интеграционных процессов.

Работа с файлами

Работа с файлами является важной функциональной составляющей интеграционной шины. В Sonic ESB есть встроенные сервисы по считыванию файла (*FilePickup*) и созданию файла (*FileDrop*).

В Groovy существуют удобные возможности по работе с файлами. Скрипт по созданию файла представлен на рис. 17 (скрипт 12).

```
class gj_class2 {
    public static void main(String[] args) {
        def writer=new File('c:/tmp/groovy/HelloWorld.txt').newWriter()
        writer.writeLine('Hello World!')
        writer.close()
    }
}
```

Рис. 17. Создание файла в Groovy (скрипт 12)

В скрипте 12 используется класс Groovy file.newWriter. Аналогичный код, написанный на Java, был бы гораздо более пространным.

Работа с базами данных

Другой важной составляющей интеграционной шины является доступ к базам данных с вызовом запросов по выборке данных и возможностью исполнения хранимых процедур. В Sonic ESB есть встроенный сервис DBService, позволяющий обращаться к базам данных по протоколу JDBC и вызывать в них хранимые процедуры.

Groovy также имеет возможность обращаться к базам данных по протоколу JDBC. На рис. 18 представлен скрипт 13, обращающийся к СУБД Oracle и выбирающий записи из таблицы по определенному условию.

Необходимо отметить спорный момент в обращении к базам данных из интеграционных шин. С точки зрения теории СОА [9], непосредственное обращение к базам данных из сервисов (например, интеграционных шин) является антишаблоном, нарушающим принцип слабой связи сервисов. Практически, вследствие возникающей сильной связи, появляется необходимость перезапуска сервисов в интеграционной шине при изменениях в СУБД (например, при перекомпиляции пакетов хранимых процедур). Тем не менее большинство интеграционных шин имеют возможность доступа к СУБД и эту возможность в определенных пределах нужно использовать.

```
import groovy.sql.Sql
def query = "select * from TBL_REF_STREAMS";
def tns = "jdbc:oracle:thin:server_name:port_number:service_name";
def sql = Sql.newInstance(tns, "login",
      "password", "oracle.jdbc.driver.OracleDriver")
sql.eachRow(query)
      { row -> if (row.ID>50) println row}
```

Рис. 18. Пример обращения к СУБД Oracle из Groovy (скрипт 13)

Заключение

Язык программирования Groovy в Sonic ESB применим для решения как типовых, так и нестандартных задач в области интеграции информационных систем. В типовых задачах интеграции (маршрутизация по контенту, преобразование XML, работа с файлами, работа с базами данных) Groovy конкурирует со стандартными сервисами Sonic ESB в скорости и удобстве разработки. В нестандартных задачах (сложные преобразования XML-сообщений и т. п.) Groovy выигрывает в скорости разработки у Java, но резко проигрывает в производительности решений.

Анализ производительности базовых функциональных компонентов шины показывает преимущество решений на Java. Решения на стандартных компонентах Sonic в 2—3 раза медленнее, а решения на Groovy в 5—7 раз медленнее решений на Java.

Критичные по быстродействию интеграционные решения рекомендуется разрабатывать на Java. Типовые интеграционные задачи без особых требований к производительности нужно решать с помощью стандартных компонентов Sonic ESB. Нестандартные интеграционные задачи, некритичные к производительности, рекомендуется делать на Groovy, особенно в условиях сокращенных сроков разработки.

Язык программирования Groovy хорошо подходит для создания прототипов интеграционных решений. Например, прототип решения быстро разрабатывают на Groovy. Затем, после уточнения функциональных и нефункциональных требований, решение полностью или частично переписывают на Java или с помощью стандартных компонентов Sonic.

Список литературы

1. **Смит Г., Ледбрук П.** Grails. Гибкость groovy и надежность Java. СПб.: Символ-Плюс, 2010.
2. **Абдул-Джавад Б**. Groovy и Grails. Практические советы. М.: ДМК-Пресс, 2010.
3. **Медведев М. Ю.** Трансляция кода из Groovy в Java в IntelliJ IDEA. Дипломная работа. Санкт-Петербургский государственный университет. 2011. URL: http://www.myshared.ru/slide/647415/
4. **Kubrynski J.** Java 7 vs Groovy 2.1 Performance Comparison. URL: http://java.dzone.com/articles/java-7-vs-Groovy-21
5. **Шаппелл Д. А.** ESB. Сервисная Шина Предприятия. СПб.: БХВ-Петербург, 2008.
6. **Shezi T., Jembere E., Adigun M.** Performance Evaluation of Enterprise Service Buses towards Support of Service Orchestration // Proc. of International Conference on Computer Engineering and Network Security (ICCENS'2012). Dubai. 26—27 December 2012. URL: http://psrcentre.org/images/extraimages/1412164.pdf
7. **Ahuja S. P., Patel A.** Enterprise Service Bus: A Performance Evaluation // Communications and Network. 2011. N. 3. P. 133—140. URL: http://www.scirp.org/journal/PaperInformation.aspx?PaperID=6768
8. **Liu Y., Gorton I., Zhu L.** Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus // Proc. of 31st Annual International Computer Software and Applications Conference (COMPSAC). Beijing. 24—27 Jule 2007. URL: http:// ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4291021&url=http%3A%2F%2Fieeexplore.ieeeorg%2Fxpls%2Fabs_all.jsp%3Farnumber %3D4291021
9. **Erl T.** SOA Principles of Service Design. Upper Saddle River: Prentice Hall, 2008. 184 p.

Yu. L. Gik, Deputy Head of the Department, e-mail: gikjuri@yandex.ru, The joint-stock Bank "Rosevrobank"

The Use of the Programming Language Groovy in the Sonic ESB

Enterprise Service Buses have proliferated in large enterprises and used to solve problems of integration of information systems. Complex tasks require the integration of specific approaches and the use of specialized programming languages. The work is devoted to the study of the Groovy programming language features in solving problems of integration. The key focus is on ease of use and performance issues. Services written on standard Sonic ESB components, Java and Groovy were tested and results analyzed. Article provides recommendations on the use of various technologies in the integration bus Sonic ESB.

Keywords: *integration bus, Groovy, ESB, Sonic ESB, script programming languages, dynamic programming languages*

References

1. **Smith G., Ledbrook P.** *Grails. Gibkost groovy i nadezhnost Java*. SPb.: Simvol-Plus, 2010.
2. **Abdul-Javad B.** *Groovy i Grails. Practicheskiie sovety*. M.: DMK-Press, 2010.
3. **Medvedev M. Ju.** Transliatsija koda iz Groovy v Java v IntelliJ IDEA. Diplomnaya rabota, Sankt-Peterburgskiy gosudarstvennyj universitet. 2011. URL: http://www.myshared.ru/slide/647415/
4. **Kubrynski J.** Java 7 vs Groovy 2.1 Performance Comparison.URL: http://java.dzone.com/articles/java-7-vs-Groovy-21
5. **Schappell D. A.** *ESB. Servisnaja shina predprijatija*. SPb.: BHV-Peterburg, 2008.
6. **Shezi T., Jembere E., Adigun M**. Performance Evaluation of Enterprise Service Buses towards Support of Service Orchestration. *Proc. of International Conference on Computer Engineering and Network Security (ICCENS'2012)*. Dubai. 26—27 December 2012. URL: http://psrcentre.org/images/extraimages/1412164.pdf
7. **Ahuja S. P., Patel A.** Enterprise Service Bus: A Performance Evaluation. *Communications and Network*. 2011. N. 3. P. 133—140. URL: http://www.scirp.org/journal/PaperInformation.aspx?PaperID = 6768
8. **Liu Y., Gorton I., Zhu L.** Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus. *Proc. of 31st Annual International Computer Software and Applications Conference (COMPSAC)*. Beijing. 24—27 July 2007. URL: http:// ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=4291021&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D4291021
9. **Erl T.** *SOA Principles of Service Design*. Upper Saddle River: Prentice Hall, 2008. 184 p.

УДК 004.932.2

И. Р. Ахмадеева[1], программист 1 категории, e-mail: ah.irishka@gmail.com,
Ю. А. Загорулько[1, 2], канд. техн. наук, зав. лаб., e-mail: zagor@iis.nsk.su,
А. С. Серый[1, 2], мл. науч. сотр., e-mail: Alexey.Seryj@iis.nsk.su,
В. К. Шестаков[1, 2], мл. науч. сотр., e-mail: shestakov@iis.nsk.su,
[1] Новосибирский государственный университет,
[2] Институт систем информатики им. А.П. Ершова СО РАН, г. Новосибирск

Методы автоматического анализа цветовой гаммы изображения и их применение при создании веб-сайтов*

Предложены методы автоматического построения согласованных палитр изображений и поиска изображений, гармонирующих с заданной цветовой палитрой. Рассмотрены различные подходы к реализации данных методов. Представлены результаты экспериментального исследования возможности применения предлагаемых методов при разработке веб-сайтов.

Ключевые слова: *квантизация цветов, анализ цветовой гаммы изображений, поиск изображений, веб-дизайн*

Введение

На сегодняшний день трудно найти веб-сайты, на которых бы не размещались изображения. Изображения могут отражать тематику веб-сайта, иллюстрировать представленную на нем текстовую информацию, выступать в качестве составных частей меню и т. д. Кроме того, каждый веб-сайт имеет собственное цветовое оформление — палитру используемых цветов, где каждому цвету соответствует его назначение (цвета ссылок, основного текста, фона, элементов оформления и пр.). Если цвета палитры плохо сочетаются друг с другом, это вызывает отрицательную реакцию у посетителя, и дизайн сайта не может считаться хорошим. Эстетическая оценка сочетаемости цветов во многом определяется субъективным мнением, но существуют общие цветовые правила и законы. Эти законы веб-дизайнеры позаимствовали у художников.

Современная теория цвета была разработана художником Иоганнесом Иттеном [1]. В основе теории Иттена лежит цветовой круг, опирающийся на три основных цвета: красный, желтый и синий. По Иттену гармония — это равновесие, симметрия сил. Согласно этому принципу сочетание двух или более цветов является гармоничным, если их смесь дает нейтральный серый цвет. С помощью цветового круга можно подбирать гармоничные сочетания трех, четырех и большего числа цветов. Например, чтобы получить гармоничное сочетание трех цветов, в цветовой круг вписывают равнобедренный треугольник и выбирают цвета, на которые указывают его углы, чтобы получить сочетание из четырех цветов, в цветовой круг надо вписать прямоугольник и т. д.

Цветовые круги широко используются дизайнерами, в том числе и веб-дизайнерами, для подбора гармоничных цветовых сочетаний. Существуют различные инструментальные средства, такие как Adobe Kuler [2], ColoRotate [3], Copaso [4] и др., помогающие дизайнеру подобрать нужную ему палитру, или автоматически формирующие палитру по заданному изображению. В последнем случае предполагается, что цвета на изображении, будь то рисунок или фотография, гармонируют друг с другом, соответственно, составленная из них палитра также будет гармоничной.

Однако формирование палитры из цветов изображения исходя лишь из соображений сочетаемости цветов друг с другом не гарантирует того, что полученная палитра будет хорошо гармонировать с самим изображением. Некоторые присутствующие на изображении цвета могут занимать незначительную площадь, сливаться с соседними цветами и быть практически не видны невооруженным глазом. Чтобы построить палитру, гармонирующую с изображением, необходимо подбирать не произвольные цвета, а *ос-*

* Работа выполнена в Новосибирском государственном университете при финансовой поддержке Министерства образования и науки Российской Федерации (договор № 02.G25.31.0054).

новные цвета изображения — цвета, наиболее заметные для наблюдателя, либо их комбинации. Заметность цвета на изображении является субъективным понятием и не имеет четкого определения. В данной работе заметные цвета определяются авторами как занимающие на изображении достаточную площадь, чтобы быть различимыми невооруженным глазом.

Составлением цветовой палитры сайта и ее согласованием с размещаемыми изображениями занимается веб-дизайнер. Как правило, он знает, какие цвета сочетаются друг с другом, какое число цветов лучше включить в палитру и какая палитра хорошо сочетается с данным изображением. Другое дело, когда созданием веб-сайта занимается человек, не обладающий навыками дизайнера. Для грамотного оформления веб-сайта ему необходимы дополнительные инструментальные средства, помогающие как построить ассоциированную палитру для заданного изображения, так и найти среди множества изображений те, которые гармонируют с заданной цветовой палитрой. В настоящей работе предложен подход к созданию инструментария, позволяющего строить ассоциированные с изображением палитры и искать изображения, гармонирующие с заданной цветовой палитрой. Помимо рядовых пользователей такие средства могут оказаться полезными и профессиональным веб-дизайнерам, так как они сокращают объем ручной работы по сопоставлению изображений и палитр.

1. Построение ассоциированной палитры изображения

Практически для любого полноцветного изображения можно построить несколько ассоциированных палитр. Данный раздел включает два подраздела. В подразд. 1.1 приведен обзор алгоритмов свертки цветового пространства изображения, в подразд. 1.2 описаны методы построения ассоциированных палитр, разработанные авторами на основе данных алгоритмов.

1.1. Свертка цветового пространства изображения

Изображение может содержать десятки тысяч различных цветов и их оттенков, в то же время на практике редко требуется палитра, включающая более десяти цветов. Для построения палитры изображения предлагается осуществлять квантизацию его цветов. Квантизация цветов изображения — это сокращение цветового пространства изображения таким образом, чтобы новое изображение визуально как можно меньше отличалось от оригинала [5]. Сделать это можно несколькими путями.

Большинство стандартных алгоритмов трактуют квантизацию цветов как задачу кластеризации точек в трехмерном пространстве, где точки представляют цвета оригинального изображения, а три оси — три цветовых канала. После выделения кластеров точки в каждом из них усредняются для получения представительного цвета, к которому будут приведены все цвета кластера [6].

Самый простой алгоритм сокращения цветов — однородная квантизация (*uniform quantization*) [7]. В этом случае все цветовое пространство делится на области одинакового размера. Все цвета, попавшие в одну область, усредняются. Этот алгоритм работает быстро и прост в реализации, однако результаты его применения часто оказываются неудовлетворительными. Проблема заключается в том, что при таком способе квантизации изображения в большинстве случаев будут присутствовать области, в которые не попал ни один из цветов, т. е. почти всегда палитра будет содержать меньше цветов, чем требуется. Также усреднение цветов происходит неравномерно в силу того, что одни области содержат больше цветов, чем другие. Последнее особенно критично при малых размерах палитры (4...10 цветов). Совокупность данных факторов приводит к снижению качества палитры в целом.

Наиболее популярными алгоритмами считаются изобретенный в 1980 г. Полом Хекбертом (*Paul Heckbert*) алгоритм медианного рассечения (*median cut algorithm*) [8] и его многочисленные вариации. Суть его сводится к тому, чтобы разделить цветовой куб на требуемое количество частей-параллелепипедов, каждый из которых содержит примерно одинаковое число точек. Сначала строится прямоугольный параллелепипед, содержащий все точки. Далее, до тех пор пока число параллелепипедов меньше, чем требуемое число кластеров, повторяются следующие шаги: находится параллелепипед с самым длинным ребром, он разрезается на два по медианной точке таким образом, чтобы каждый из них содержал примерно половину точек из исходного параллелепипеда, затем каждый новый параллелепипед сокращается до такого минимального размера, чтобы все его точки находились внутри него. Представитель для каждого параллелепипеда выбирается путем усреднения в нем всех точек.

Более современный алгоритм, впервые описанный Михаэлем Герваутцем (*Michael Gervautz*) и Вернером Пургатхофером (*Werner Purgathofer*) в 1988 г. [9], а затем улучшенный исследователем компании Xerox PARC Дэном Блумбергом (*Dan Bloomberg*) [10], основан на кластеризации с использованием октодеревьев (*octree*). Он включает три основных этапа: построение октодерева, содержащего информацию о распределении цветов на изображении; свертку дерева путем объединения нижележащих узлов в вышележащие до тех пор пока число листьев не станет равным требуемому числу цветов; построение итоговой цветовой палитры на основе свернутого дерева.

Кроме них существуют такие менее распространенные алгоритмы, как алгоритм локальных K-средних (*local K-means*), придуманный Олегом Веревка (*Oleg Verevka*) в 1995 г. [11], или алгоритм NeuQuant [12]. Первый использует схему пост-кластеризации (*post-clustering*), согласно которой вначале делается предположение о палитре, а затем проводится ее итеративное улучшение. Второй алгоритм редуцирует число цветов до 256 путем обучения нейронной сети Кохонена (*Kohonen neural network*) [13].

При выборе алгоритма свертки цветового пространства авторами были рассмотрены два самых распространенных из них — алгоритм медианного рассечения и алгоритм октодерева. В работе [9] утверждается, что временная сложность алгоритма октодерева составляет $O(N)$, где N — общее число пикселей на изображении. Сложность алгоритма медианного рассечения зависит не только от входных данных, но и от того, как сортируются кластеры (чтобы найти медиану кластера, его необходимо отсортировать по одной из координат). В общем случае, если выбран быстрый алгоритм сортировки, медианное рассечение имеет немного меньшую временную сложность, чем алгоритм октодерева. Последний при этом показывает лучшие результаты при обработке изображений [14]. Авторами был выбран алгоритм октодерева, который при незначительной разнице во временной сложности дает более качественный результат.

При необходимости, после квантизации цветов изображение приводится к сокращенному цветовому пространству. Это тоже можно сделать не единственным способом. Простая замена оригинальных цветов на усредненные цвета из сокращенного набора ведет к потере деталей в полученном изображении. Для решения этой задачи разработано большое число алгоритмов дизеринга (*dithering*). Области изображения, которые должны быть окрашены отсутствующими цветами, составляются из имеющихся цветов, рассредоточенных по области таким образом, чтобы кажущийся итоговый цвет как можно меньше отличался от оригинального. Авторами была исследована возможность применения дизеринга при построении контрастной палитры изображения (см. подразд. 1.2). Для этого был использован один из самых распространенных алгоритмов дизеринга — алгоритм Флойда-Стейнберга [15]. Анализ результатов экспериментов показал, что применение дизеринга не дает заметного улучшения итоговой палитры, поэтому авторы не стали включать его в результирующий вариант алгоритма построения палитры.

1.2. Палитра изображения

В результате свертки цветового пространства получается палитра, состоящая из усредненных цветов изображения. В дальнейшем будем называть ее *основной,* или *базовой, палитрой.* Заметим, чем меньше размер базовой палитры, тем меньше вероятность того, что в ней будут представлены все заметные цвета изображения. Этот факт означает, что наряду с базовой существуют и другие подходящие палитры, состоящие из цветов изображения или близких к ним.

Как показала практика, существуют две основные причины, по которым могут возникнуть дополнительные палитры. Первая из них заключается в присутствии на изображении множества равноправных цветов (изображение "пестрое"), когда размера палитры просто не хватает для того, чтобы представить все цвета. Вторая причина — присутствие выделяющихся на общем фоне объектов сравнительно небольшого размера.

При свертке все цвета изображения приводятся к цветам базовой палитры, и чем меньше ее размер, тем сильнее усредняются цвета исходного изображения. После отбрасывания базовых цветов и близких к ним, на изображении остается набор цветов, из которого можно сформировать дополнительную палитру.

На рис. 1 (см. вторую сторону обложки) приведены примеры изображений, где имеются предпосылки для построения дополнительной палитры. Изображение на рис. 1, *а*, очевидно, является многоцветным. На рис. 1, *б* внимание привлекает контрастный объект (автомобиль), расположенный почти в центре изображения.

Отметим, что при выбранном авторами размере палитры, равном четырем, необходимость построения дополнительной палитры возникает для подавляющего большинства полноцветных изображений. Чтобы сделать это, необходимо было выработать пути обнаружения областей изображения, достаточно больших и однородных, для того чтобы быть заметными для человеческого глаза, и контрастирующих с базовыми тонами изображения. В итоге авторами был разработан алгоритм построения контрастной палитры, формируемой путем комбинации цветов базовой палитры и палитры дополнительных цветов, включающей те, которые не попали в базовую. Далее представлено его описание.

Шаг 1. На этом шаге, если это не было сделано ранее, из изображения выделяются основные цвета, т. е. строится его базовая палитра.

Шаг 2. Базовые цвета удаляются с изображения (чтобы они не участвовали в дальнейшем анализе изображения, все пиксели, окрашенные в базовые цвета, делаются прозрачными).

Шаг 3. На изображение накладывается медианный фильтр с заданной апертурой для очистки от несущественных деталей. В итоге остаются только однородные области заметного размера. Оптимальный размер апертуры фильтра определен экспериментально.

Шаг 4. Выполняется квантизация полученного изображения до заданного числа цветов. Для этого сначала с помощью алгоритма октодерева, аналогич-

ного тому, который используется для получения базовой палитры, выполняется свертка цветового пространства изображения, затем результирующие цвета накладываются на изображение.

Шаг 5. Строится гистограмма полученного на шаге 4 изображения и подсчитывается площадь, занимаемая каждым из оставшихся на нем цветов. Цвета, превысившие заданный порог по площади, попадают в дополнительную палитру.

Шаг 6. Контрастная палитра строится из цветов базовой палитры, полученной на шаге 1, и цветов дополнительной палитры, полученной на шаге 5. Выбор цветов осуществляется исходя из соображений, что цвета в контрастной палитре должны как можно сильнее отличаться друг от друга. В соответствии с этой установкой комбинация цветов подбирается таким образом, чтобы минимум расстояний, вычисляемых по формуле

$$\Delta E^*_{00} = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \frac{\Delta C'}{k_C S_C}\frac{\Delta H'}{k_H S_H}} \qquad (1)$$

между всеми цветами комбинации, был максимальным. Формально это можно записать следующим образом: если P_B и P_A — соответственно базовая и дополнительная палитры, то искомая комбинация цветов $p' = (c'_1, \ldots, c'_n)$ такова, что

$$\min_{\substack{i,j \\ i \neq j}} \Delta E^*_{00}(c'_i, c'_j) = \max_{p \in P}\left(\min_{\substack{i,j \\ i \neq j}} \Delta E^*_{00}(c_i, c_j)\right),\ p = (c_1, \ldots, c_n),$$

где $P = \{(c_1, \ldots, c_n) \in (P_B \cup P_A)^n | c_i \neq c_j\ \forall i, j = 1, \ldots, n\}$, n — число цветов в итоговой палитре.

Формула (1), называемая формулой цветового расстояния CIEDE2000 [16], была предложена международной комиссией по освещению (CIE — *Commission Internationale de l'Eclairage*) в 2000 г. в качестве замены предыдущей формуле CIE94. Для вычисления расстояний все цвета конвертировались в цветовое пространство CIELab. Пространство CIELab, разработанное в качестве аппаратно-независимой модели на основе пространства CIE XYZ, было призвано обеспечить линейность изменения цвета в различных областях спектра. Цвет в пространстве CIELab имеет координаты L, a и b, где $L \in [0; 100]$ задает светлоту цвета (*lightness*), a, $b \in [-128; 128]$ задают хроматическую составляющую (первая обозначает положение цвета в диапазоне от зелено-цианового до красного, вторая — положение цвета в диапазоне от синего до желтого).

Если (L^*_1, a^*_1, b^*_1), (L^*_2, a^*_2, b^*_2) — координаты цветов в пространстве CIELab, между которыми вычисляется расстояние, то подвыражения формулы (1) вычисляются следующим образом:

$$\Delta L' = L^*_2 - L^*_1,\quad \overline{L} = \frac{L^*_1 + L^*_2}{2};$$

$$C^*_1 = \sqrt{a^{*2}_1 + b^{*2}_1},\ C^*_2 = \sqrt{a^{*2}_2 + b^{*2}_2},\ \overline{C} = \frac{C^*_1 + C^*_2}{2};$$

$$a'_1 = a^*_1 + \frac{a^*_1}{2}\left(1 - \sqrt{\frac{\overline{C}^7}{\overline{C}^7 + 25^7}}\right),$$

$$a'_2 = a^*_2 + \frac{a^*_2}{2}\left(1 - \sqrt{\frac{\overline{C}^7}{\overline{C}^7 + 25^7}}\right);$$

$$C'_1 = \sqrt{a'^2_1 + b^{*2}_1},\quad C'_2 = \sqrt{a'^2_2 + b^{*2}_2},$$
$$\Delta C' = C'_2 - C'_1, \overline{C}' = \frac{C'_1 + C'_2}{2};$$

$$h'_1 = \operatorname{atan2}(b^*_1,\ a'_1) \bmod 360^\circ,$$
$$h'_2 = \operatorname{atan2}(b^*_2,\ a'_2) \bmod 360^\circ;$$

$$\Delta h' = \begin{cases} h'_2 - h'_1, & |h'_2 - h'_1| \leqslant 180^\circ; \\ h'_2 - h'_1 + 360^\circ, & |h'_2 - h'_1| > 180^\circ;\quad h'_2 \leqslant h'_1; \\ h'_2 - h'_1 - 360^\circ, & |h'_2 - h'_1| > 180^\circ;\quad h'_2 > h'_1; \end{cases}$$

$$\Delta H' = 2\sqrt{C'_1 C'_2}\sin(\Delta h'/2),$$
$$\overline{H}' = \begin{cases} (h'_1 + h'_2 + 360^\circ)/2, & |h'_1 - h'_2| > 180^\circ; \\ (h'_1 + h'_2)/2, & |h'_1 - h'_2| \leqslant 180^\circ; \end{cases}$$

$$T = 1 - 0{,}17\cos(\overline{H}' - 30^\circ) + 0{,}24\cos(2\overline{H}') + 0{,}32\cos(3\overline{H}' + 6^\circ) - 0{,}20\cos(4\overline{H}' - 63^\circ);$$

$$S_L = 1 + \frac{0{,}015(\overline{L} - 50)^2}{\sqrt{20 + (\overline{L} - 50)^2}},\ S_C = 1 + 0{,}045\,\overline{C}',$$
$$S_H = 1 + 0{,}015\,\overline{C}\,T;$$

$$R_T = -2\sqrt{\frac{\overline{C}'^7}{\overline{C}'^7 + 25^7}}\sin\left[60^\circ \mathbf{e}^{-[(\overline{H}' - 275^\circ)/25^\circ]^2}\right].$$

Коэффициенты k_L, k_C и k_H за редким исключением равны 1, так как используются для управления относительными поправками по светлоте, насыщенности и цветовому тону соответственно при вычислении

цветовых расстояний для различных условий просмотра, отличающихся от эталонных, утвержденных CIE в 1995 г.

При таком подходе базовая и контрастная палитры могут совпадать. Эксперименты на множестве изображений показали, что данный подход в большей степени эффективен именно для однородных изображений с контрастными объектами, подобных изображенному на рис. 1, *б*, см. вторую сторону обложки. Базовая палитра таких изображений включает близкие цвета, что повышает вероятность появления в контрастной палитре цветов из дополнительной.

На рис. 2 и 3 (см. вторую сторону обложки) показаны в сравнении, соответственно базовая и контрастная палитры размера 4, построенные для изображений рис. 1.

Путем модуляции (осветления и затемнения) из двух полученных палитр, базовой и контрастной, можно получить широкий набор палитр, предоставив пользователю возможность выбрать наиболее подходящую.

2. Поиск изображений по цветовой палитре

Если при создании веб-сайта разработчик пользуется библиотекой изображений, выбирая те, которые он хочет разместить на веб-сайте, то помимо задачи подбора палитры по изображению появляется и обратная задача: по заданной палитре найти изображения, которые сочетаются с ней.

Большие коллекции изображений, как правило, индексируются, чтобы при необходимости их можно было отфильтровать по какому-либо признаку — жанру, размеру и др. Когда требуется чтобы изображение вписывалось в цветовую гамму веб-сайта, коллекцию нужно фильтровать по палитре, состоящей из нескольких цветов.

Если наряду с коллекцией изображений задан и набор предопределенных цветовых палитр, например, сформированных предварительно веб-дизайнером, то фильтрация по цвету предоставляет ему возможность автоматически составить таблицу соответствия палитр и изображений. Это особенно важно для больших коллекций. В данном разделе приведен разработанный авторами подход к индексированию коллекций изображений и к их фильтрации по заданной цветовой палитре.

Многие из существующих алгоритмов поиска изображений по цвету (по крайней мере, открытых) основаны на оценке площади, занимаемой на изображении тем или иным цветом, т. е. на анализе гистограммы изображения. Глобальные поисковые системы, такие как Яндекс и Google, предоставляют возможность поиска изображения по цвету. Очевидно, что база поиска таких систем очень велика, а индекс изображений универсален, так как поиск по цвету — лишь одна из большого числа задач, которые решаются глобальной поисковой системой. За такой подход к решению задачи приходится расплачиваться тем, что поиск ведется только по одному из небольшого набора цветов. Цветовая гамма веб-сайта, по которой требуется отфильтровать коллекцию изображений, в свою очередь, обычно включает не менее 2...3 цветов (начинающим не рекомендуется использовать в оформлении сайта более 4...5 цветов). Кроме того, как показали проведенные авторами эксперименты, палитра, на 75 % состоящая из цветов, достаточно близких к цветам изображения, не всегда хорошо с ним сочетается. Для проверки такой сочетаемости были введены дополнительные критерии.

В связи с поиском изображений по цветовой палитре следует упомянуть веб-сервис Multicolor engine от TinEye [17]. Данный сервис на коммерческой основе предоставляет услуги индексирования коллекции изображений и последующего поиска, как по изображениям, так и по цветовым палитрам, включающим до пяти цветов. Поиск по цветам позволяет задать пропорции, в которых заданные цвета должны содержаться на необходимых пользователю изображениях. Все алгоритмы индексирования и поиска, которые используются в этом сервисе, составляют коммерческую тайну. Рассматриваемый сервис обеспечивает высокие релевантности поиска похожих изображений и поиска по цветовым палитрам. При этом следует заметить, что Multicolor engine является именно поисковым сервисом. С его помощью нельзя построить палитру изображения или его гистограмму.

Для фильтрации коллекции необходимо ее предварительное индексирование. В качестве индекса изображения авторами использовалась гистограмма его редуцированной копии. Такой подход означает, что сначала проводится свертка цветового пространства изображения до N ($15 \leqslant N \leqslant 256$) цветов, затем полученные цвета накладываются на изображение, далее все цвета вместе с занимаемой ими площадью на изображении фиксируются в гистограмме. Задача фильтрации коллекции изображений сводится к задаче сравнения цветовой палитры с такой гистограммой. Результатом сравнения должен быть ответ "да" или "нет" (является ли изображение подходящим для данной цветовой палитры или нет). Все арифметические операции над цветами проводились в цветовом пространстве CIELab, цветовое расстояние вычислялось по формуле (1). Далее перечислены этапы принятия решения о том, является ли изображение подходящим для заданной палитры.

Шаг 1. В индексе изображения отмечаются "заметные" цвета, т. е. те, у которых размер занимаемой площади выше определенного порога. Данный порог является параметром алгоритма и обозначается T_{SA}.

Шаг 2. Для всех элементов палитры C_P и для всех элементов гистограммы C_H по формуле (1) вычисляется цветовое расстояние $\Delta E^*_{00}(C_P, C_H)$. Если выполняется условие $\Delta E^*_{00} < T_{CD}$, где T_{CD} — параметрическое пороговое значение, то цвет C_H называется *парным цвету* C_P и тогда констатируется, что цвет C_H *покрывается палитрой.*

Шаг 3. Для каждого элемента палитры вычисляется суммарная площадь, занимаемая на изображении парными ему цветами. Если данная площадь меньше параметрической величины T_{LA}, то элемент считается не имеющим соответствий в гистограмме изображения.

Шаг 4. Если все элементы палитры имеют соответствия в гистограмме изображения и суммарная площадь, занимаемая на изображении всеми парными цветами, больше порогового значения T_{TA}, то возвращается ответ "да". Если более одного элемента палитры не имеют соответствий в гистограмме изображения или суммарная площадь не превосходит T_{TA}, то возвращается ответ "нет".

Шаг 5. Отдельно рассматривается случай, когда ровно один элемент C_0 палитры не имеет пары в гистограмме изображения. Если все заметные цвета изображения покрываются палитрой, то возвращается ответ "да". В противном случае вычисляется минимум цветовых расстояний между C_0 и всеми заметными цветами изображения, не покрываемыми палитрой. Если данный минимум не превосходит порогового значения, обозначаемого T_{SD}, то возвращается ответ "да". В противном случае возвращается ответ "нет".

В общем случае, чтобы отфильтровать коллекцию по произвольной палитре необходимо проанализировать индексы всех изображений и отобрать среди них те, для которых решение было положительным. Эффективность данного решения обеспечивается тем фактом, что число операций, необходимых для проверки соответствия изображения заданной палитре, как видно из описания, зависит только от размера индекса изображения, являющегося постоянной величиной, и не зависит от характеристик самого изображения.

В случае когда палитра содержится в некотором заранее известном наборе, достаточно проанализировать таблицу соответствия, построенную с помощью данного метода так, как это было указано выше. На рис. 4, см. третью сторону обложки, приведены примеры изображений, найденных по заданной палитре.

3. Эксперименты и результаты

Апробация предлагаемых методов и алгоритмов проводилась на коллекции из 512 изображений. Средний размер изображения в коллекции 2036×1364 пикселя, среднее число уникальных цветов на изображении — 142 531. Среднее время построения базовой и контрастной палитр составило 0,5 и 1,6 с соответственно. Тестовые реализации выполнены на языке PHP с применением библиотеки ImageMagick для работы с изображениями. Все эксперименты по апробации алгоритмов, а также решение задачи оптимизации параметров (см. подразд. 3.1) проводились с использованием следующих вычислительных средств: процессор Intel Core i5 2310 2,9 GHz, 8 GB DDR3 RAM, Windows 7×64 SP1, библиотека для работы с изображениями ImageMagick 6.7.7-4.

Эксперименты показали, что лучшие результаты поиска изображений достигаются, когда число цветов N, определяющих размер индекса коллекции, находится в промежутке от 20 до 30. Это связано с тем обстоятельством, что при малых N цвета изображения сильно усредняются, что искажает результат сравнения изображения с цветовой палитрой. При больших N в индексе остаются цвета, незаметные на изображении, влиянием которых на результат поиска можно пренебречь. С учетом отмеченного выше, при апробации алгоритмов авторами использовалось значение $N = 25$. При этом время индексирования коллекции изображений составило 985 с.

3.1. Оптимизация параметров алгоритма поиска изображений

Как следует из описания алгоритма, представленного в разд. 2, результат его работы зависит от перечисленных далее пяти параметров:

- пороговое значение площади заметных цветов T_{SA} (*significant area threshold*);
- пороговое значение цветового расстояния T_{CD} (*color distance threshold*);
- пороговое значение площади цветов, парных данному T_{LA} (*local area threshold*);
- порог суммарной площади всех парных цветов T_{TA} (*total area threshold*);
- порог расстояния между элементом палитры, не получившим соответствия в гистограмме, и заметным цветом T_{SD} (*significant distance threshold*).

Для подбора оптимальных значений перечисленных параметров авторами был реализован один из вариантов генетического алгоритма [18]. При этом решение задачи, представляющее вектор, состоящий из значений пяти перечисленных выше параметров, кодировалось хромосомой, включающей пять генов, каждый из которых соответствует одному из этих параметров. На начальном этапе задавалась начальная популяция — множество произвольных решений в виде хромосом указанного вида. На последующих этапах на основе текущей популяции с помощью таких операторов, как селекция и скрещивание, формировалась новая популяция, состоящая из более приспособленных хромосом.

Селекция хромосом для родительской популяции проводилась вращением рулетки. Этот метод селекции, при котором родительские хромосомы выбираются пропорционально значениям их функций приспособленности, считается для генетических алгоритмов основным методом отбора хромосом для родительской популяции в целях последующего их преобразования генетическими операторами.

Эталонные данные, помимо коллекции изображений, включали 50 четырехцветных палитр и таблицу соответствия изображений и палитр, составленную вручную веб-дизайнером.

По причине недостатка у авторов вычислительных ресурсов, необходимых для проведения вычислений на всей области значений параметров, предварительно было проведено большое число экспериментов для определения границ, в которых с наибольшей вероятностью находятся их оптимальные значения (в общем виде, области значений параметров, отвечающих за площади и расстояния, представляют собой отрезки от 0 до 1 и от 1 до 100 соответственно). Точность при таком подходе уменьшается, однако существенно снижается число необходимых вычислений. По результатам экспериментов выделены следующие дискретные области определения:

- $T_{SA} \in [0{,}005;\ 0{,}1]$ с шагом 0,005;
- $T_{CD} \in [11;\ 30]$ с шагом 1;
- $T_{LA} \in [0{,}005;\ 0{,}1]$ с шагом 0,005;
- $T_{TA} \in [0{,}31;\ 0{,}5]$ с шагом 0,01;
- $T_{SD} \in [31;\ 50]$ с шагом 1.

Нетрудно видеть, что каждая из областей определения состоит из 20 значений. Отсюда можно получить оптимальный размер для каждой популяции алгоритма.

Пусть G — размер популяции, $\mathbf{X}$ — случайная хромосома из популяции, $\mathbf{X} = (x_1, x_2, x_3, x_4, x_5)$. Вероятность появления на позиции x_1 любого из доступных значений равняется $p_1 = \frac{1}{20} = 0{,}05$, так как все значения равноправны. Соответственно, с вероятностью $p_2 = (1 - 0{,}05)^G = 0{,}95^G$ значение не появится ни в одной из хромосом популяции. Это верно для любого значения из области определения гена x_1. Чтобы получить вероятность того, что каждое значение встретится в популяции хотя бы один раз, необходимо из единицы отнять вероятности всех событий, которые неприемлемы, т. е. тех событий, которые предусматривают отсутствие какого-либо значения:

$$p = 1 - 0{,}95^G - 0{,}95^G - \ldots - 0{,}95^G = 1 - 20 \cdot 0{,}95^G. \quad (2)$$

Формула (2) верна для любого из генов, так как область определения каждого из них состоит из 20 значений. Возведя выражение из формулы (2) в пятую степень получим вероятность того, что в популяции будут представлены все возможные значения для всех генов:

$$P = (1 - 20 \cdot 0{,}95^G)^5. \quad (3)$$

Положив $P = 99{,}9\ \% = 0{,}999$, получим

$$G = \log_{0{,}95}\left(\frac{1 - \sqrt[5]{0{,}999}}{20}\right) \approx 224. \quad (4)$$

Таким образом, согласно формулам (2)—(4), чтобы получить популяцию, где с вероятностью 99,9 % представлены все возможные значения всех генов, достаточно 224 хромосомы (при общем числе возможных хромосом $20^5 = 3\ 200\ 000$). Для надежности значение G было увеличено до 250.

Понятие соответствия изображений заданной цветовой палитре не является полностью объективным. По этой причине в данном случае трудно формально говорить о таких характеристиках, как полнота и точность. Целевая функция генетического алгоритма выбиралась с таким расчетом, чтобы результат работы алгоритма поиска изображений как можно полнее покрывал эталон, и при этом минимизировалось общее расхождение. Другими словами, если множество изображений A получено автоматическим поиском по цветовой палитре, а множество B — эталонное множество изображений, соответствующих той же палитре, то необходимо минимизировать $|B \setminus A|$ таким образом, чтобы вместе с этим число элементов в симметрической разности $|A \Delta B|$ было как можно меньше.

Ниже приведена целевая функция генетического алгоритма.

$$F(\mathbf{X}) = \frac{1}{50}\sum_{i=1}^{50}\frac{|\ A_i(\mathbf{X}) \Delta B_i\ |}{|\ A_i(\mathbf{X}) \cup B_i\ |} \to \min. \quad (5)$$

В этой формуле 50 — число эталонных палитр и приняты следующие обозначения:

$\mathbf{X}$ — хромосома;

$A_i(\mathbf{X})$ — множество изображений, найденных по i-й эталонной палитре с набором параметров $\mathbf{X}$;

B_i — эталонное множество изображений, соответствующих i-й палитре;

$A_i(\mathbf{X}) \Delta B_i$ — симметрическая разность множеств $A_i(\mathbf{X})$ и B_i.

Лучший набор параметров $\mathbf{X}_{top}$ в среднем по i давал значения $\frac{\left|B_i \setminus A_i\left(\mathbf{X}_{top}\right)\right|}{|B_i|} \approx 0{,}74$, т. е. в среднем результат автоматического поиска изображений по цветовой палитре покрывал 74 % эталонного множества (по сравнению с 40 % на начальной популяции). Значение целевой функции (5) на лучшем наборе параметров $F(\mathbf{X}_{top}) \approx 0{,}688$ свидетельствует о том, что долю

симметрической разности множеств $A_i(\mathbf{X})$ и B_i удалось снизить всего до ~ 68,8 %. Этот факт означает, что кроме изображений из эталонного множества результат автоматического поиска содержит много других изображений. Тем не менее оценка результатов показала, что за редким исключением, все дополнительные изображения могут считаться подходящими для соответствующих палитр.

Заключение

Предложены подходы, реализация которых позволяет автоматизировать построение цветовой палитры изображения и ввести в дополнение к уже существующим способам фильтрации коллекции изображений фильтрацию по цветовой палитре. Эти подходы могут найти применение в веб-дизайне — области достаточно широкой и актуальной. С их помощью опытные дизайнеры и в большей степени обычные пользователи смогут эффективно подобрать цветовое оформление веб-сайта. Возможны и другие применения предложенных методов, например при обработке больших коллекций изображений.

Качество получаемых палитр и индексов изображений сильно зависит от того, какой алгоритм применяется при квантизации цветов. Используя более сложные алгоритмы квантизации, можно строить более качественные палитры и более информативные индексы для изображений. Увеличивая размер индекса, можно повысить точность фильтрации коллекций. Однако все это повлияет на производительность алгоритмов с точки зрения времени выполнения, что критично при работе в режиме реального времени. Поиск баланса между качеством результата и производительностью является возможным направлением будущих исследований в данной области.

Список литературы

1. **Иттен И.** Искусство цвета. М.: Аронов, 2001. 96 с.
2. **Adobe** Kuler [Электронный ресурс]. URL: http://kuler.adobe.com/ (дата доступа 18.07.2014).
3. **ColoRotate** [Электронный ресурс]. URL: http://web.colorotate.org/ (дата доступа 18.07.2014).
4. **Copaso** [Электронный ресурс. URL: http://www.colourlovers.com/copaso/ColorPaletteSoftware (дата доступа 18.07.2014).
5. **Orchard M. T., Bouman C. A**. Color quantization of images // Signal Processing, IEEE Transactions on. 1991. Vol. 39, N. 12. P. 2677—2690.
6. **Segenchuk S.** An Overview of Color Quantization Techniques [Электронный ресурс]. URL: http://web.cs.wpi.edu/~matt/courses/cs563/talks/color_quant/CQindex.html (дата доступа 23.06.2014).
7. **Hill F. S. Jr.** Computer Graphics. Macmillna Publishing Co., 1990.
8. **Heckbert P.** Color image quantization for frame buffer display // ACM. 1982. Vol. 16, N. 3. P. 297—307.
9. **Gervautz M., Purgathofer W.** A simple method for color quantization: Octree quantization // New trends in computer graphics. Berlin Heidelberg: Springer 1988. P. 219—231.
10. **Bloomberg D. S.** Color quantization using octrees. [Электронный ресурс]. URL: http://leptonica.net/papers/colorquant.pdf (дата доступа 23.06.2014).
11. **Verevka O.** Color Image Quantization in Windows Systems with Local K-means Algorithm // Proceedings of the Western Computer Graphics Symposium. 1995. Vol. 95. P. 20—22.
12. **Dekker A. H.** Kohonen Neural Networks for Optimal Colour Quantization // Network: Computation in Neural Systems. 1994. Vol. 5, N. 3. P. 351—367.
13. **Kohonen T.** The self-organizing map // Proceedings of the IEEE. 1990. Vol. 78, N. 9. P. 1464—1480.
14. **Bloomberg D.** Color quantization using modified median cut. [Электронный ресурс]. URL: http://leptonica.com/papers/mediancut.pdf (дата доступа 18.07.2014).
15. **Floyd R. W., Steinberg L.** An adaptive algorithm for spatial greyscale // SPIE Milestone Series. 1999. Vol. 154. P. 281—283.
16. **Sharma G., Wu W., Dalal E. N.** The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations // Color Research & Application. 2005. Vol. 30, N. 1. P. 21—30.
17. **TinEye** Multicolor Engine [Электронный ресурс]. URL: http://services.tineye.com/MulticolorEngine (дата доступа 18.07.2014).
18. **Whitley D.** A genetic algorithm tutorial // Statistics and computing. 1994. Vol. 4, N. 2. P. 65—85.

I. R. Akhmadeeva[1], Programmer, e-mail: ah.irishka@gmail.com,
Yu. A. Zagorulko[1, 2], Head of Laboratory, e-mail: zagor@iis.nsk.su,
A. S. Sery[1, 2], Junior Researcher, e-mail: Alexey.Seryj@iis.nsk.su,
V. K. Shestakov[1, 2], Junior Researcher, e-mail: shestakov@iis.nsk.su
[1] Novosibirsk State University,
[2] A. P. Ershov Institute of Informatics Systems Russian Academy of Sciences Siberian Branch, Novosibirsk

Experimental Techniques and Methods for the Automatic Analysis of Image Colors and their Applications to Web Design

When designing a website, choosing appropriate decoration colors may not be an easy task, especially if the designer is not professional but an ordinary internet user. This paper proposes methods for automating some tasks involving working with images and color schemes. Such methods may prove useful for both professional designers and

beginners. Some of these methods, based on various color quantization algorithms, help to automatically pick up colors that blend well with each other and with the images the designer wants to place on his/her website. The other method is intended for searching for appropriate images within a specified collection on a given color scheme. It is based on an analysis of the image histogram and its result depends on several parameters, which were optimized with genetic algorithm. The paper also presents a brief overview of various tools designed for constructing color palettes on a basis of images and searching for images on color palettes.

Keywords: *color quantization, image color analysis, image search, web design*

References

1. **Itten J.** *Iskusstvo cveta*. M.: Aronov, 2001. 96 p.

2. **Adobe** Kuler [Electronic resource]. URL: http://kuler.adobe.com/ (last visited 18.07.2014).

3. **ColoRotate** [Electronic resource]. URL: http://web.colorotate.org/ (last visited 18.07.2014).

4. **Copaso** [Electronic resource]. URL: http://www.colourlovers.com/copaso/ColorPaletteSoftware (last visited 18.07.2014).

5. **Orchard M. T., Bouman C. A.** Color quantization of images. *Signal Processing, IEEE Transactions on*. 1991. Vol. 39, N. 12. P. 2677—2690.

6. **Segenchuk S.** An Overview of Color Quantization Techniques. [Electronic resource]. URL: http://web.cs.wpi.edu/~matt/courses/cs563/talks/color_quant/CQindex.html (last visited 23.06.2014).

7. **Hill F. S. Jr.** *Computer Graphics*. Macmillna Publishing Co., 1990.

8. **Heckbert P.** Color image quantization for frame buffer display. *ACM*. 1982. Vol. 16, N. 3. P. 297—307.

9. **Gervautz M., Purgathofer W.** A simple method for color quantization: Octree quantization. *New trends in computer graphics*. Berlin Heidelberg: Springer, 1988. P. 219—231.

10. **Bloomberg D. S.** Color quantization using octrees [Electronic resource]. URL: http://leptonica.net/papers/colorquant.pdf (last visited 23.06.2014).

11. **Verevka O.** Color Image Quantization in Windows Systems with Local K-means Algorithm. *Proceedings of the Western Computer Graphics Symposium*. 1995. Vol. 95. P. 20—22.

12. **Dekker A. H.** Kohonen Neural Networks for Optimal Colour Quantization. *Network: Computation in Neural Systems*. 1994. Vol. 5, N. 3. P. 351—367.

13. **Kohonen T.** The self-organizing map. *Proceedings of the IEEE*. 1990. Vol. 78, N. 9. P. 1464—1480.

14. **Bloomberg D.** Color quantization using modified median cut. [Electronic resource]. URL: http://leptonica.com/papers/mediancut.pdf (last visited 18.07.2014).

15. **Floyd R. W., Steinberg L.** An adaptive algorithm for spatial grayscale. *SPIE Milestone Series*. 1999. Vol. 154. P. 281—283.

16. **Sharma G., Wu W., Dalal E. N.** The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*. 2005. Vol. 30, N. 1. P. 21—30.

17. **TinEye** Multicolor Engine [Electronic resource]. URL: http://services.tineye.com/MulticolorEngine (last visited 18.07.2014).

18. **Whitley D.** A genetic algorithm tutorial. *Statistics and computing*. 1994. Vol. 4, N. 2. P. 65—85.

ИНФОРМАЦИЯ

УДК 519.854

Ю. А. Зак, д-р техн. наук, наук. эксперт, консультант, e-mail yuriy_zack@hotmail.com, Аахен, Германия

Алгоритмы линейного булевого программирования в условиях размытых исходных данных

Рассмотрены постановки, детерминированные эквиваленты и методы решения задачи булевого Fuzzy-линейного программирования в условиях, когда параметры функции цели и ограничений, а также допустимые граничные значения вектора ограничений задачи — нечеткие множества с заданными функциями принадлежности. Решение задачи ищется в виде детерминированного вектора булевых переменных. Предложены методы сравнения и ранжирования нечетких множеств, а также оригинальные алгоритмы решения задач булевого линейного программирования методом ветвей и границ.

Ключевые слова: *нечеткие множества, Fuzzy-булевое линейное программирование, детерминированный эквивалент, сравнение и ранжирование нечетких множеств, метод ветвей и границ*

Введение

Многие математические модели задач управления, анализа состояния и принятия решений в экономических и технических системах сводятся к задачам булевого линейного программирования в условиях неопределенности, когда коэффициенты функций цели, ограничений и граничные значения являются не действительными числами, а некоторыми нечеткими множествами.

В литературе рассматривались различные постановки и математические модели задач нечеткого линейного программирования, например, в работах [1—10], а также анализ различных приложений [10, 11] и алгоритмов решения этих задач [1, 2]. Наибольшее число публикаций касалось общей задачи нечеткого линейного программирования и ее практических приложений [1, 2, 4, 6—8, 11]. Задачам булевого нечеткого программирования посвящено сравнительно небольшое число работ, например, работы [3, 5, 6, 9, 12]. В отличие от подавляющего большинства публикаций по нечеткому линейному программированию, в рассматриваемой автором постановке нечеткими числами могут быть как коэффициенты левых частей ограничений, целевых функций, так и граничные значения функций ограничений.

Эти задачи имеют свои характерные особенности и для них могут быть предложены эффективные детерминированные эквиваленты и более эффективные алгоритмы решения на основе подходов, описанных в работе [12]. Полученные в последние годы автором [13—16] методы сравнения и ранжирования Fuzzy-множеств, устанавливающие правила безусловного и относительного предпочтения, дали возможность в данной работе рассмотреть условия выполнения ограничений и принятия решений с различной степенью риска и надежностью.

Задача булевого линейного программирования формулируется как задача максимизации (или минимизации) линейной целевой функции на заданном множестве допустимых решений, которое описывается системой равенств или неравенств. Например:

$$f(X^{\bullet}) = \max_{X \in G} \sum_{j=1}^{n} c_j x_j, \tag{1}$$

$$G = \left\{ \begin{matrix} X \mid \sum\limits_{j=1}^{n} a_{ij} x_j \leqslant b_i, i = 1, \ldots, m; \\ x_j = 0 \vee 1, j = 1, \ldots, n \end{matrix} \right\}, \tag{2}$$

где $X = (x_1, x_2, \ldots, x_j, \ldots, x_n)$ — вектор переменных задачи, каждая компонента которого может принимать только два значения: 0 или 1; $f(X)$ — целевая функция задачи; $X^{\bullet}$ — значение вектора переменных в оптимальном решении; c_j и a_{ij} — коэффициенты целевой функции и правых частей ограничений соответственно, b_i — граничные значения. Все эти величины принадлежат множеству действительных чисел.

В публикациях по этой теме преимущественно рассматривались постановки этих задач для функций принадлежности частного вида. При этом не исследовались различные критерии и условия безусловного и относительного предпочтения нечетких множеств и выполнения размытых ограничений, а также связанные с ними различные виды детерминированных эквивалентов сформулированных задач и их математические особенности.

В настоящей публикации рассматриваются постановка и методы решения этой задачи в условиях, когда параметры целевой функции и ограничений, а также допустимые граничные значения вектора переменных

задачи — нечеткие множества с заданными функциями принадлежности. Решение задачи ищется в виде вектора булевых переменных:

$$\bar{F} = \bar{C}_1 x_1 \oplus \bar{C}_2 x_2 \oplus ... \oplus \bar{C}_j x_j \oplus ... \oplus \bar{C}_n x_n \to \max, \quad (3)$$

$$\bar{\Psi}_i = \bar{A}_{i1} x_1 \oplus \bar{A}_{i2} x_2 \oplus ... \oplus \bar{A}_{ij} x_j \oplus ... \oplus \bar{A}_{in} x_n \lesseqgtr_{\mathrm{Re}} \bar{B}_i, \quad i = 1, ..., m. \quad (4)$$

Здесь $\bar{C}_j$, $j = 1, ..., n$; $\bar{B}_i$, $i = 1, ..., m$; $\bar{A}_{ij}$, $i = 1, ..., m$, $j = 1, ..., n$ — нечеткие множества с заданными функциями принадлежности; $\bar{\Psi}_i$ — нечеткое множество левой части i-го ограничения; $\bar{C}_j x_j$ и $\bar{A}_{ij} x_j$ — операции умножения нечетких множеств на действительное число, результатом которой является нечеткое множество; $\bar{C}_j x_j \oplus \oplus \bar{C}_p x_p$, а также $\bar{A}_{i1} x_1 \oplus \bar{A}_{i2} x_2 \oplus ... \oplus \bar{A}_{ij} x_j \oplus ... \oplus \bar{A}_{in} x_n$ — операции сложения двух или нескольких нечетких множеств, результатом которых является нечеткое множество.

Условия $\lesseqgtr_{\mathrm{Re}}$ или $\gtreqless_{\mathrm{Re}}$ определяют условия доминирования (предпочтения) двух нечетких множеств соответственно в сторону уменьшения или увеличения их значений. Знаком "⊕" обозначается алгебраическая сумма двух нечетких множеств.

В дальнейшем будет показано, что рассматривая различные сечения (дискретные α-уровни) нечетких множеств целевой функции и ограничений, исходная задача сводится к решению ряда детерминированных задач линейного программирования. В результате нечеткие ограничения принимают определенный интервальный вид. Однако при этом число ограничений увеличивается как минимум в 2 раза, но полученную задачу уже можно решить симплексным методом. Следовательно, за гибкость в целом ряде случаев приходится платить ценой увеличения размерности задачи. В отличие от известных публикаций по этой тематике, в настоящей работе рассматриваются различные условия предпочтений, безусловного (строгого) и условного (с большей или меньшей строгостью) выполнения размытых ограничений. На основе таких условий предлагаются различные, не рассматриваемые ранее в литературе детерминированные эквиваленты в виде задачи детерминированного булевого Fuzzy-программирования различной размерности. На основе правил Fuzzy-арифметики для представлений множеств приводятся формулы расчета коэффициентов правых частей ограничений, целевых функций и граничных значений. Для всех рассматриваемых в работе видов функций принадлежности нечетких множеств получены формулы для расчета всех коэффициентов и левых частей ограничений всех построенных (см. разд. 2) детерминированных моделей. Учитывая специфику рассматриваемого булевого программирования, включающего большое число ограничений, предложены эффективные алгоритмы решения данного класса задач.

С учетом отмеченной ранее прикладной стороны рассматриваемого класса задач, предложенные в работе алгоритмы могут быть использованы при построении систем подготовки принятия решений.

1. Правила сравнения и определения предпочтений нечетких множеств

Для решения задачи булевого линейного программирования (1), (2) в условиях нечетких исходных данных, представленных в самом общем виде (3), (4), должны быть сформулированы правила сравнения и определения предпочтений и ранжирования нечетких множеств. В данной работе ограничимся рассмотрением случаев, когда

$\bar{C}_1$, $\bar{C}_2$, ..., $\bar{C}_j$, ..., $\bar{C}_n$ — Fuzzy-числа (нечеткие множества), представленные LR-интервалами

$$\{a_1(c_j), m_1(c_j), m_2(c_j), a_2(c_j)\}_{LR}, j = 1, ..., n;$$

где $a_1(c_j)$, $a_2(c_j)$ и $m_1(c_j)$, $m_2(c_j)$ — соответственно крайние точки нижнего и верхнего оснований трапеции, определяющей нечеткое множество $\bar{C}_j$;

$\bar{A}_{ij}$ — нечеткие множества, представленные LR-интервалами вида

$$\{a_1(\bar{A}_{ij}), m_1(\bar{A}_{ij}), m_2(\bar{A}_{ij}), a_2(\bar{A}_{ij})\}_{LR}, \quad i = 1, ..., m, j = 1, ..., n;$$

$\bar{B}_i$ — нечеткие множества, представленные LR-интервалами вида

$$\{a_1(\bar{B}_i), m_1(\bar{B}_i), m_2(\bar{B}_i), a_2(\bar{B}_i)\}_{LR}, \quad i = 1, ..., m.$$

Функции принадлежности таких нечетких множеств $\bar{A}$, представленных трапецевидными LR-интервалами, вычисляются следующим образом:

$$\mu_{\bar{A}}(x) = \begin{cases} 0, \text{ if } m_1(\bar{A}) - a_1(\bar{A}) \geqslant x, m_2(\bar{A}) + a_2(\bar{A}) \leqslant x, \\ \dfrac{x - m_1(\bar{A}) - a_1(\bar{A})}{a_1(\bar{A})}, \text{ if } m_1(\bar{A}) - a_1(\bar{A}) < x < m_1(\bar{A}), \\ 1, \text{ if } m_1(\bar{A}) \leqslant x \leqslant m_2(\bar{A}), \\ \dfrac{m_2(\bar{A}) + a_2(\bar{A}) - x}{a_2(\bar{A})}, \text{ if } m_2(\bar{A}) < x < m_2(\bar{A}) + a_2(\bar{A}). \end{cases} \quad (5)$$

Если $m_1(\bar{A}) = m_2(\bar{A}) = m(\bar{A})$, то функция принадлежности (5) вырождается в центральный треугольник, описываемый следующей формулой:

$$\mu_{\bar{A}}(x) = \begin{cases} 0, \text{ if } m(\bar{A}) - a_1(\bar{A}) \geqslant x, m(\bar{A}) + a_2(\bar{A}) \leqslant x, \\ \dfrac{x - m(\bar{A}) - a_1(\bar{A})}{a_1(\bar{A})}, \text{ if } m(\bar{A}) - a_1(\bar{A}) < x < m(\bar{A}), \\ \dfrac{m(\bar{A}) + a_2(\bar{A}) - x}{a_2(\bar{A})}, \text{ if } m(\bar{A}) < x < m(\bar{A}) + a_2(\bar{A}). \end{cases} \quad (6)$$

Отметим, что если в формуле (5) *LR*-интервала $a_1(\bar{A}) = 0$, то функция принадлежности вырождается в правостороннюю трапецию, если в формуле (5) $a_2(\bar{A}) = 0$, то функция принадлежности вырождается в левостороннюю трапецию. Если $m_1(\bar{A}) = m_2(\bar{A}) = m(\bar{A})$ и $a_1(\bar{A}) = 0$, то формула (6) преобразуется в выражение, определяющее правосторонний треугольник, а в случае, если $m_1(\bar{A}) = m_2(\bar{A}) = m(\bar{A})$ и $a_2(\bar{A}) = 0$ — в выражение, определяющее левосторонний треугольник.

Результаты выполнения соответствующих операторов Fuzzy-арифметики, описанные, например, в работах [1—3], для выражений (5), (6) в данном случае могут быть записаны в виде

$$\left[\bar{T}_j x_j \oplus \bar{T}_p x_p\right]_{LR} = \left\{ \begin{matrix} \left[m_1(\bar{T}_j)x_j \pm m_1(\bar{T}_p)x_p\right]; \\ \left[m_2(\bar{T}_j)x_j \pm m_2(\bar{T}_p)x_p\right]; \\ \left[a_1(\bar{T}_j)x_j \pm a_1(\bar{T}_p)x_p\right]; \\ \left[a_2(\bar{T}_j)x_j \pm a_2(\bar{T}_p)x_p\right] \end{matrix} \right\}_{LR}, \quad (7)$$

где $\bar{T}_j = \bar{C}_j$ или $\bar{T}_j = \bar{A}_j$, а $\bar{T}_p = \bar{C}_p$ или $\bar{T}_p = \bar{A}_p$.

В выражении (7) справедлив знак "+", если $x_p = 1$ и коэффициент $\bar{T}_p$ — число положительное, а знак "–" — если $x_p = 1$ и $\bar{T}_p$ — число отрицательное.

Рассмотрим правила доминирования для нечетких множеств, представленных *LR*-Fuzzy-интервалами, основанные на сравнении их сечений. Эти методы ранее были предложены автором и для данного типа функций принадлежности подробно представлены в виде конкретных формул в книге [2]. Следующие ниже формульные выражения для функций принадлежности треугольного и трапецевидного типов конкретизированы далее и в ранее опубликованных работах автора [10, 13, 14, 16].

Рассмотрим два нечетких множества

$$\bar{\Psi}_i = \{a_1(\bar{\Psi}_i), m_1(\bar{\Psi}_i), m_2(\bar{\Psi}_i), a_2(\bar{\Psi}_i)\}_{LR}$$
$$\text{и } \bar{\Psi}_j = \{a_1(\bar{\Psi}_j), m_1(\bar{\Psi}_j), m_2(\bar{\Psi}_j), a_2(\bar{\Psi}_j)\}_{LR}$$
$$\text{или } \bar{\Psi}_i = \{a_1(\bar{\Psi}_i), m(\bar{\Psi}_i), a_2(\bar{\Psi}_i)\}_{LR}$$
$$\text{и } \bar{\Psi}_j = \{a_1(\bar{\Psi}_j), m(\bar{\Psi}_j), a_2(\bar{\Psi}_j)\}_{LR}.$$

Правила абсолютного предпочтения. Если для двух Fuzzy-множеств $\bar{\Psi}_i$ и $\bar{\Psi}_j$ справедливо неравенство $a_2(\bar{\Psi}_i) \leq a_1(\bar{\Psi}_j)$, то Fuzzy-множество $\bar{\Psi}_i$ имеет абсолютное предпочтение перед Fuzzy-множеством $\bar{\Psi}_i$ в смысле $\bar{\Psi}_i \overline{\leqslant}_{\text{Re}} \bar{\Psi}_j$.

При выполнении неравенство $a_1(\bar{\Psi}_i) \geqslant a_2(\bar{\Psi}_j)$ Fuzzy-множество $\bar{\Psi}_i$ имеет абсолютное предпочтение перед Fuzzy-множеством $\bar{\Psi}_j$ в смысле $\bar{\Psi}_i \overline{\geqslant}_{\text{Re}} \bar{\Psi}_j$.

Правила относительного предпочтения. Если для двух Fuzzy-множеств $\bar{\Psi}_i$ и $\bar{\Psi}_j$ справедлива система неравенств

$$a_1(\bar{\Psi}_i) \leqslant a_1(\bar{\Psi}_j),\ m_1(\bar{\Psi}_i) \leqslant m_1(\bar{\Psi}_j),\ m_2(\bar{\Psi}_i) \leqslant m_2(\bar{\Psi}_j)$$
$$(\text{или } m(\bar{\Psi}_i) \leqslant m(\bar{\Psi}_j)),\ a_2(\bar{\Psi}_i) \leqslant a_2(\bar{\Psi}_j),$$

то Fuzzy-множество $\bar{\Psi}_i$ имеет относительное предпочтение перед Fuzzy-множеством $\bar{\Psi}_j$ в смысле $\bar{\Psi}_i \overline{\leqslant}_{\text{Re}} \bar{\Psi}_j$.

Если для двух Fuzzy-множеств $\bar{\Psi}_i$ и $\bar{\Psi}_j$ справедлива система неравенств

$$a_1(\bar{\Psi}_i) \geqslant a_1(\bar{\Psi}_j),\ m_1(\bar{\Psi}_i) \geqslant m_1(\bar{\Psi}_j),\ m_2(\bar{\Psi}_i) \geqslant m_2(\bar{\Psi}_j)$$
$$(\text{или } m(\bar{\Psi}_i) \geqslant m(\bar{\Psi}_j)),\ a_2(\bar{\Psi}_i) \geqslant a_2(\bar{\Psi}_j),$$

то Fuzzy-множество $\bar{\Psi}_i$ имеет относительное предпочтение перед Fuzzy-множеством $\bar{\Psi}_j$ в смысле $\bar{\Psi}_i \overline{\geqslant}_{\text{Re}} \bar{\Psi}_j$.

Более сильное правило относительного предпочтения может быть построено на основании системы неравенств относительно крайних точек соответствующих α_p-сечений, т. е. системы неравенств относительно значений $L_1^{\alpha}(\bar{\Psi}_i)$, $L_2^{\alpha}(\bar{\Psi}_i)$ и $L_1^{\alpha}(\bar{\Psi}_j)$, $L_2^{\alpha}(\bar{\Psi}_j)$, $\alpha = \alpha_0 = 0$, $\alpha_1, \alpha_2, \ldots, \alpha_P, \alpha_P = 1$.

Иллюстрация выбора крайних точек для случая пяти сечений функций принадлежности трапецевидного и треугольного типов приведены на рис. 1 и 2.

На рис. 1, 2 приняты следующие обозначения: $L_1^s(\bar{A}) = L_1^{\alpha_s}(\bar{A})$, $L_2^s(\bar{A}) = L_2^{\alpha_s}(\bar{A})$, $s = 1, 2, 3, 4$.

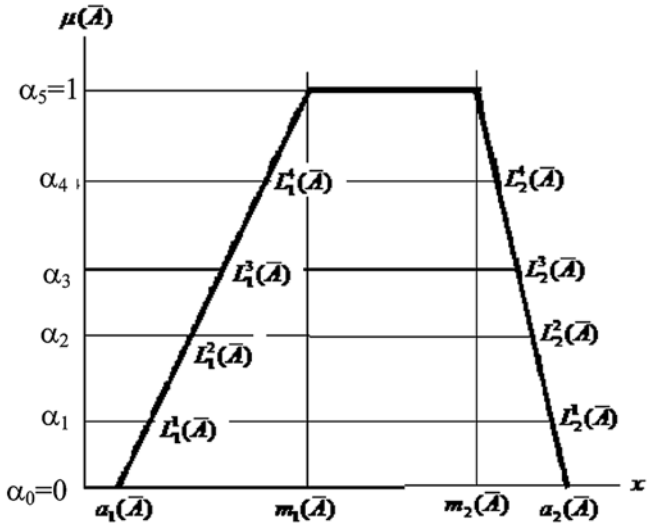

Рис. 1. Сечения функции принадлежности трепецевидного типа

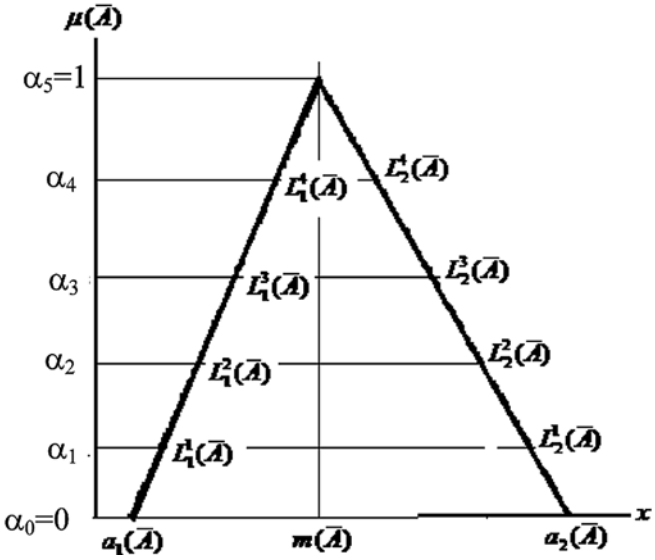

Рис. 2. Сечения функции принадлежности треугольного типа

Для функций принадлежности трапецевидного типа значения $L_1^{\alpha}(\bar{\Psi}_i)$, $L_2^{\alpha}(\bar{\Psi}_i)$ вычисляются по следующим формулам:

$$L_1^{\alpha}(\bar{\Psi}_i) = m_1(\bar{\Psi}_i) - [m_1(\bar{\Psi}_i) - a_1(\bar{\Psi}_i)](1 - \alpha),$$
$$L_2^{\alpha}(\bar{\Psi}_i) = m_2(\bar{\Psi}_i) + [a_2(\bar{\Psi}_i) - m_2(\bar{\Psi}_i)](1 - \alpha),$$

а для функций принадлежности треугольного типа по формулам

$$L_1^{\alpha}(\bar{\Psi}_i) = m(\bar{\Psi}_i) - [m(\bar{\Psi}_i) - a_1(\bar{\Psi}_i)](1 - \alpha),$$
$$L_2^{\alpha}(\bar{\Psi}_i) = m(\bar{\Psi}_i) + [a_2(\bar{\Psi}_i) - m(\bar{\Psi}_i)](1 - \alpha).$$

Если справедлива система неравенств

$$L_1^{\alpha}(\bar{\Psi}_i) \leqslant L_1^{\alpha}(\bar{\Psi}_j),\ L_2^{\alpha}(\bar{\Psi}_i) \leqslant L_2^{\alpha}(\bar{\Psi}_j),$$
$$\alpha = \alpha_0 = 0,\ \alpha_1,\ \alpha_2,\ ...,\ \alpha_P,\ \alpha_P = 1,$$

то Fuzzy-множество $\bar{\Psi}_i$ имеет относительное предпочтение перед Fuzzy-множеством $\bar{\Psi}_j$ в смысле $\bar{\Psi}_i \ \overline{\leqslant}_{\mathrm{Re}}\ \bar{\Psi}_j$.

Если справедлива система неравенств

$$L_1^{\alpha}(\bar{\Psi}_i) \geqslant L_1^{\alpha}(\bar{\Psi}_j),\ L_2^{\alpha}(\bar{\Psi}_i) \geqslant L_2^{\alpha}(\bar{\Psi}_j),$$
$$\alpha = \alpha_0 = 0,\ \alpha_1,\ \alpha_2,\ ...,\ \alpha_P,\ \alpha_P = 1,$$

то Fuzzy-множество $\bar{\Psi}_i$ имеет относительное предпочтение перед Fuzzy-множеством $\bar{\Psi}_j$ в смысле $\bar{\Psi}_i \ \overline{\geqslant}_{\mathrm{Re}}\ \bar{\Psi}_j$.

От выбора значений α_p, $p = 1, 2, ..., (P - 1)$ и числа сечений P во многом зависит результат решения задач сравнения и ранжирования. Выбор вида сечений может осуществляться по рекомендации экспертов или лица, принимающего решение. В качестве примера в работах [4, 10] рекомендуется выбирать следующие виды сечений: $\alpha_0 = 0$; $\alpha_1 = 0{,}25$; $\alpha_2 = 0{,}75$; $\alpha_3 = 0{,}75$; $\alpha_4 = 1{,}0$ или ($\alpha_0 = 0$; $\alpha_1 = 0{,}3$; $\alpha_2 = 0{,}6$; $\alpha_3 = 0{,}9$; $\alpha_4 = 1{,}0$).

Вычислим значение некоторой функции свертки крайних точек сечений нечетких множеств

$$f(\bar{\Psi}_i) = \sum_{p=1}^{P} \frac{1}{2} \beta_p [L_1^{\alpha_p}(\bar{\Psi}_i) + L_2^{\alpha_p}(\bar{\Psi}_i)],$$

где $0 < \beta_p < 1$, $p = 0, 1, 2, ..., P$ — некоторые весовые коэффициенты, удовлетворяющие условиям нормировки $\sum_{p=1}^{P} \beta_p = 1$. Например, могут быть выбраны следующие значения: $\beta_0 = 0{,}075$; $\beta_1 = 0{,}125$; $\beta_2 = 0{,}2$; $\beta_3 = 0{,}25$; $\beta_4 = 0{,}35$.

Другое более слабое правило относительного предпочтения может быть сформулировано следующим образом:

- в случае $f(\bar{\Psi}_i) < f(\bar{\Psi}_j)$ справедливо утверждение $\bar{\Psi}_i \ \overline{\leqslant}_{\mathrm{Re}}\ \bar{\Psi}_j$;
- в случае $f(\bar{\Psi}_i) > f(\bar{\Psi}_j)$ справедливо утверждение $\bar{\Psi}_i \ \overline{\geqslant}_{\mathrm{Re}}\ \bar{\Psi}_j$.

На основе приведенных выше правил и формульных выражений безусловного и относительного предпочтения нечетких множеств и в зависимости от предпочтений лица, принимающего решения (ЛПР), или эксперта могут быть сформулированы различные виды детерминированных эквивалентов задачи

нечеткого булевого программирования в зависимости от поставленных требований строгости и надежности выполнения всей системы ограничений и гарантий получения ожидаемого значения целевой функции.

2. Детерминированные эквиваленты задачи линейного Fuzzy-булевого программирования

Четкий детерминированный эквивалент проверки выполнения ограничений. Пусть задан некоторый детерминированный вектор булевых переменных $X = (x_1, x_2, ..., x_j, ..., x_n)$. Отметим, что специфика задач Fuzzy-булевого программирования состоит в том, что при любом способе сравнения нечетких множеств эта задача может быть сведена к детерминированной одно- или многокритериальной модели линейного булевого программирования. На основе принципа расширения и пользуясь операторами Fuzzy-арифметики может быть построено нечеткое множество $\bar{\Psi}_i$ левой части i-го неравенства системы ограничений (4). Его функция принадлежности трапециевидной формы также может быть представлена с помощью LR-Fuzzy-интервала $\{a_1(\bar{\Psi}_i), m_1(\bar{\Psi}_i), m_2(\bar{\Psi}_i), a_2(\bar{\Psi}_i)\}_{LR}$, а функция принадлежности треугольного типа с помощью интервала $\{a_1(\bar{\Psi}_i), m(\bar{\Psi}_i), a_2(\bar{\Psi}_i)\}_{LR}$.

Обозначим $L_1^{\alpha}(\bar{\Psi})$, $L_2^{\alpha}(\bar{\Psi})$ крайние точки α-сечений нечеткого множества $\bar{\Psi}$, где $0 < \alpha < 1$. В качестве множества $\bar{\Psi}$ могут рассматриваться нечеткие множества $\bar{F}$, определенные выражением (3), а также $\bar{\Psi}_i$, вычисляемые по формулам (4), и $\bar{B}_i$, $i = 1, ..., m$.

На основе сформулированных в разд. 1 правил предпочтения нечетких множеств приведем простейшее и не требующее доказательств утверждение, определяющее условия выполнения ограничений задачи с различной степенью строгости.

Утверждение 1. Если для двух нечетких множеств $\bar{\Psi}_i$ и $\bar{B}_i$ выполняется неравенство

$$a_2(\bar{\Psi}_i) \leqslant a_1(\bar{B}_i)$$

или системы неравенств

$$a_1(\bar{\Psi}_i) \leqslant a_1(\bar{B}_i),\ m_1(\bar{\Psi}_i) \leqslant m_1(\bar{B}_i),$$

$$m_2(\bar{\Psi}_i) \leqslant m_2(\bar{B}_i),\ a_2(\bar{\Psi}_i) \leqslant a_2(\bar{B}_i),$$

$$L_1^{\alpha}(\bar{\Psi}_i) \leqslant L_1^{\alpha}(\bar{B}_i),\ L_2^{\alpha}(\bar{\Psi}_i) \leqslant L_2^{\alpha}(\bar{B}_i),$$
$$\alpha = \alpha_1, \alpha_2, ..., \alpha_P,$$

$$\sum_{p=1}^{P} \beta_p \frac{1}{2} [L_1^{\alpha_p}(\bar{\Psi}_i) + L_2^{\alpha_p}(\bar{\Psi}_i)] \leqslant$$
$$\leqslant \sum_{p=1}^{P} \beta_p \frac{1}{2} [L_1^{\alpha_p}(\bar{B}_i) + L_2^{\alpha_p}(\bar{B}_i)],$$

то детерминированный вектор переменных $X \in H$ обеспечивает выполнение Fuzzy-отношений $\bar{\Psi}_i \overline{\leqslant}_{\mathrm{Re}} \bar{B}_i$, и следовательно выполнение i-го ограничения из системы ограничений (4).

Если эти условия утверждения 1 выполняются для каждого из ограничений ($i = 1, ..., m$) системы (4), то детерминированный вектор $X \in H$ удовлетворяет всей системе ограничений задачи. Здесь H — множество всех ограничений на булевые переменные сформулированной задачи (3), (4).

В качестве детерминированного эквивалента задачи нечеткого булевого линейного программирования может рассматриваться следующая многокритериальная задача детерминированного линейного булевого программирования, число ограничений которой равно $(P + 1)m$.

Задача 1.

$$F_p^{\alpha} = \sum_{j=1}^{n} L_p^{\alpha}(\bar{C}_j) x_j \to \max,$$
$$\alpha = \alpha_0 = 0, \alpha_1, \alpha_2, ..., \alpha_{P-1}, \alpha_P = 1, \qquad (8)$$

$$\varphi_i^{\alpha_p} = \sum_{j=1}^{n} L_s^{\alpha_p}(\bar{A}_{ij}) x_j \leqslant L_s^{\alpha_p}(\bar{B}_i),\ s = 1, 2,\ i = 1, ..., m,$$
$$\alpha = \alpha_0 = 0, \alpha_1, \alpha_2, ..., \alpha_{P-1}, \alpha_P = 1. \qquad (9)$$

Рассмотрим некоторые частные случаи формулировки ограничений и целевой функции задачи (1), (2) в условиях нечетких данных в виде (3), (4). Самое строгое выполнение ограничений задачи может быть представлено в виде

$$\sum_{j=1}^{n} a_2(\bar{A}_{ij}) x_j \leqslant a_1(\bar{B}_i),\ i = 1, ..., m. \qquad (10)$$

Другие менее строгие формы выполнения ограничений задачи (8), (9) могут быть представлены в следующем виде:

$$\sum_{j=1}^{n} a_1(\bar{A}_{ij}) x_j \leqslant a_1(\bar{B}_i),\ \sum_{j=1}^{n} m_1(\bar{A}_{ij}) x_j \leqslant m_1(\bar{B}_i),$$
$$\sum_{j=1}^{n} m_2(\bar{A}_{ij}) x_j \leqslant m_2(\bar{B}_i),$$

$$\sum_{j=1}^{n} a_2(\bar{A}_{ij}) x_j \leqslant a_2(\bar{B}_i),\ i = 1, ..., m; \qquad (11)$$

$$\sum_{j=1}^{n} m_2(\bar{A}_{ij})x_j \leqslant m_1(\bar{B}_i),\ i = 1, ..., m; \qquad (12)$$

$$\sum_{p=1}^{P} \beta_p \frac{1}{2} \left[L_1^{\alpha_p}\left(\sum_{j=1}^{n} \bar{A}_{ij}x_j \right) + L_2^{\alpha_p}\left(\sum_{j=1}^{n} \bar{A}_{ij}x_j \right) \right] \leqslant$$

$$\leqslant \sum_{p=1}^{P} \beta_p \frac{1}{2} [L_1^{\alpha_p}(\bar{B}_i) + L_2^{\alpha_p}(\bar{B}_i)],\ i = 1, ..., m. \qquad (13)$$

В качестве компромиссного критерия многокритериальной задачи 1 может быть использована некоторая линейная свертка локальных критериев с различными весовыми коэффициентами:

$$\Phi_0 = \sum_{p=0}^{P} \frac{1}{2} \sum_{j=1}^{n} [\lambda_1^{\alpha_p} L_1^{\alpha_p}(\bar{C}_j) + \lambda_2^{\alpha_p} L_2^{\alpha_p}(\bar{C}_j)]x_j \to \max. \qquad (14)$$

Здесь $0 \leqslant \lambda_1^{\alpha_p} \leqslant 1$ и $0 \leqslant \lambda_2^{\alpha_p} \leqslant 1$, $\alpha = \alpha_0 = 0$, α_1, α_2, ..., α_{P-1}, $\alpha_P = 1$ — весовые коэффициенты, значения которых удовлетворяют условиям нормировки $\sum_{p=0}^{P} (\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p}) = 1$.

В качестве частного случая критерия (14) может быть принят более простой критерий вида

$$\Phi_1 = \sum_{j=1}^{n} \frac{1}{2} [\lambda_1(a_1(\bar{C}_j) + a_2(\bar{C}_j)) + \lambda_2(m_1(\bar{C}_j) + m_2(\bar{C}_j))]x_j \to \max, \qquad (15)$$

где $0 \leqslant \lambda_1 \leqslant 1$, $0 \leqslant \lambda_2 \leqslant 1$; $\lambda_1 + \lambda_2 = 1$.

Следовательно, в качестве детерминированных эквивалентов задачи Fuzzy-булевого программирования в зависимости от требований жесткости выполнения ограничений (4) может быть сформулирована одна из задач выбора детерминированного вектора булевых переменных $\bar{X}^{\bullet}$, обеспечивающих максимизацию критерия (14) или (15) в условиях выполнения одной из систем ограничений (10)—(13).

Каждая из сформулированных выше задач является задачей булевого линейного программирования вида (1), (2) с рассчитанными на основании приведенных выше формульных выражений коэффициентами линейных ограничений g_{ij}, целевых функций q_j и правых частей ограничений G_i:

$$\sum_{j=1}^{n} q_j x_j \to \max, \qquad (16)$$

$$\sum_{j=1}^{n} g_{ij}x_j \leqslant G_i,\ i = 1, ..., M;\ x_j = 0 \vee 1,\ j = 1, ..., n. \qquad (17)$$

Здесь M — число функций линейных ограничений в детерминированном эквиваленте задачи.

Однако число ограничений N в этих задачах, как правило, существенно больше, чем в задаче (3), (4), т. е. $N \geqslant n$. Следовательно, последняя из сформулированных задач может быть решена известными алгоритмами решения задач линейного булевого программирования [17, 18]. Однако, учитывая особенности рассматриваемой задачи (сравнительно небольшое число переменных и большое число ограничений), для ее решения могут быть использованы алгоритмы решения, предложенные в работе автора [19]. Далее рассматриваются модификации этих алгоритмов, предложенные автором для случая большого числа ограничений.

3. Алгоритмы решения задач булевого линейного программирования

Решение сформулированной задачи выполним модифицированным методом ветвей и границ. Существенной особенностью рассматриваемого ниже алгоритма является применение для каждого из анализируемых вариантов операторов исключения областей, не содержащих допустимых и оптимальных планов. С учетом специфики сформулированной задачи линейного булевого программирования для различных практических приложений (большое число ограничений по сравнению с числом переменных) ниже приводится модификация разработанного и опубликованного автором в 1977 г. [19] алгоритма точного решения данной задачи для этих условий. Доказательства всех приведенных далее нетривиальных утверждений представлены в работе автора [17] и поэтому здесь не повторяются.

Свойства задач булевого линейного программирования. Обозначим $\tilde{J}_t^{k,1} = \{x_j = 1, j \in \tilde{J}\}$, $\tilde{J}_t^{k,0} = \{x_j = 0, j \in \tilde{J}\}$ — множества компонент вектора X, значения которых на t-м шаге итеративного процесса в k-м подмножестве вариантов (обозначим его $\tilde{\Omega}_t^k$) определены и равны 1 и 0 соответственно; $j \in \tilde{J}_t^{k,-}$; $\tilde{J}_t^{k,-} = \tilde{J}/(\tilde{J}_t^{k,1} \cup \tilde{J}_t^{k,0})$ — множество компонент вектора X, значения которых на t-м шаге итеративного процесса в k-м подмножестве еще не определены.

Определим следующие подмножества множества $\tilde{J}_t^{k,-}$:

$$\bar{Y}^{k,-} = \{j \in \tilde{J}_t^{k,-} | q_j < 0\},\ \bar{Y}^{k,0} = \{j \in \tilde{J}_t^{k,-} | q_j = 0\},$$
$$\bar{Y}^{k,'+} = \{j \in \tilde{J}_t^{k,-} | q_j > 0\};$$

$n^k = |\tilde{J}_t^{k,-}|$ — число элементов в подмножестве $\tilde{J}_t^{k,-}$.

Определим

$$G^k_{t,i} = G_i - \sum_{j \in \tilde{J}^{k,1}_t} g_{ij}x_j,\ i = 1, ..., M,\ k = 1, ..., K;$$

$$\bar{Z}^{k,-}_{t,i} = \left\{ j \in \tilde{J}^{k,-}_{t,i} | g_{ij} < 0 \right\},\ \bar{Z}^{k,0}_{t,i} = \left\{ j \in \tilde{J}^{k,-}_{t,i} | g_{ij} = 0 \right\},$$

$$\bar{Z}^{k,'+}_{t,i} = \left\{ j \in \tilde{J}^{k,-}_{t,i} | g_{ij} > 0 \right\},\ i = 1, ..., M,\ k = 1, ..., K;$$

$$D^k_{t,i} = G^k_{t,i} - \sum_{j \in \bar{Z}^{k,+}_t} g_{ij}x_j,\ R^k_{t,i} = G^k_{t,i} + \sum_{j \in \bar{Z}^{k,-}_t} g_{ij}x_j, \quad (18)$$

$$i = 1, ..., M,\ k = 1, ..., K;$$

$$\tilde{V}^k = \left\{ q_j, j \in \tilde{J}^{k,-}_t | q_j > 0 \right\}; \quad (19)$$

$$\tilde{U}^{k,1}_t = \left\{ q_j, j \in \tilde{J}^{k,-}_t | q_{j_1} \geqslant q_{j_2} \geqslant ... \geqslant q_{j_p} \geqslant ... \geqslant |q_{j_{n^n}}| \right\},$$

$$k = 1, ..., K; \quad (20)$$

$$\tilde{U}^{k,2}_t = \left\{ q_j, j \in \tilde{J}^{k,-}_t \| q_{j_1} | \geqslant |q_{j_2}| \geqslant ... \geqslant |q_{j_2}| \geqslant ... \geqslant |q_{j_{n^n}}| \right\},$$

$$k = 1, ..., K; \quad (21)$$

$$\tilde{U}^{k,3}_t = \left\{ q_j, j \in \tilde{J}^{k,-}_t \| w_{j_1} | \geqslant |w_{j_2}| \geqslant ... \geqslant |w_{j_2}| \geqslant ... \geqslant w_{j_{n^n}} \right\},$$

$$k = 1, ..., K. \quad (22)$$

Здесь $|w_j| = |q_j|, j \in \tilde{J}^{k,-}_t$ — абсолютные значения соответствующих величин;

$$|w_j| = |q_j| + \sum_{i \in \left(\bar{Z}^{k,-}_{t,i} \cup \bar{Z}^{k,+}_{t,i} \right)} |g_{ij}|,\ j \in \tilde{J}^{k,-}_t.$$

Назовем набор $x_j = 1, j \in \tilde{J}^{k,1}_t;\ x_j = 0, j \in \tilde{J}^{k,0}_t$ — частичным планом в k-м подмножестве вариантов $k = 1, ..., K$. Любой набор значений n^k булевых переменных $j \in \tilde{J}^{k,-}_t$, каждая из которых может принимать значение 0 или 1, назовем дополняющим планом задачи (16), (17) в k-м подмножестве вариантов. Обозначим $\tilde{I}^{k,+}$ и $\tilde{I}^{k,-}$ соответственно подмножества активных и неактивных ограничений в k-м подмножестве вариантов.

Утверждение 2. Если в k-м подмножестве вариантов есть p-е ограничение, в котором все коэффициенты $g_{pj} = 0, j \in \tilde{J}^{k,0}_t$, то это p-е ограничение является неактивным, т. е. $p \in \tilde{I}^{k,-}$, и в k-м подмножестве вариантов может быть исключено из дальнейшего рассмотрения. Если справедливо неравенство $D^k_{t,p} \leqslant 0$, то p-е ограничение в k-м подмножестве вариантов справедливо для любых значений переменных $j \in \tilde{J}^{k,-}_t$ и, следовательно, является неактивным и может в дальнейшем не рассматриваться.

Следствие утверждения 2. Если справедлива система неравенств $D^k_{t,i} \leqslant 0\ \forall i \in \tilde{I}^{k,+}$, то в оптимальном решении дополняющего плана в k-м подмножестве вариантов должны быть выбраны следующие значения переменных:

$$x_j = 1, j \in \bar{Y}^{k,+};\ x_j = 0, j \in \bar{Y}^{k,-}. \quad (23)$$

Утверждение 3. Если в k-м подмножестве вариантов есть p-е ограничение, для которого справедливо неравенство $R^k_{t,p} > 0$, то это p-е ограничение не выполнится ни при каких значениях переменных $j \in \tilde{J}^{k,0}_t$ в k-м подмножестве вариантов. Следовательно, $\tilde{\Omega}^k_t = \varnothing$ и k-е подмножество вариантов может быть исключено из дальнейшего рассмотрения.

Утверждение 4. Если для некоторого значения x_j, $j \in \tilde{J}^{k,+}_t$ хотя бы для одного из ограничений $i \in \tilde{I}^{k,+}$ справедливо неравенство

$$R^k_{t,i} - g_{ij} > 0,$$

то в k-м подмножестве вариантов i-е ограничение не может быть выполнено при значении $x_j = 1$ при любых значениях других переменных $l \in \{\tilde{J}^{k,+}_t / j\}$ и, следовательно, в допустимом решении для k подмножества вариантов должно быть принято $x_j = 0$.

Утверждение 5. Если в k-м подмножестве вариантов хотя бы для одного из ограничений $i \in \tilde{I}^{k,+}$ справедливо неравенство

$$G^k_{t,i} + \sum_{j \in \left(\bar{Z}^{k,-}_{t,i} / l \right)} g_{ij}x_j > 0,$$

$$G^k_{t,i} + \sum_{j \in \left(\bar{Z}^{k,-}_{t,i} / l \right)} g_{ij}x_j + g_{il}x_l \leqslant 0, \quad (24)$$

то в допустимом решении для k подмножества вариантов должно быть принято $x_l = 1, l \in \tilde{J}^{k,-}_t \cap \bar{Z}^{k,-}_{t,i}$.

Утверждение 6. Если для всех ограничений $i \in \tilde{I}^{k,+}$ и какой-либо переменной $x_j = 1, j \in \tilde{J}^{k,-}_t$ справедливы условия

$$\tilde{V}^{1,k} = \left\{ x_j = 1, j \in (\tilde{J}^{k,-}_t \cap \bar{Z}^{k,+}_{k,i}) | q_j > 0; \right.$$

$$\left. \sum_{j \in \left(\tilde{J}^{k,-}_t \cap \bar{Z}^{k,+}_{k,i} \right)} g_{ij} \leqslant G^k_{t,i} \in \tilde{I}^{k,+} \right\}, \quad (25)$$

то в оптимальном дополняющем плане в k-м подмножестве вариантов значение $x_j = 1$.

Утверждение 7. Если хотя бы для одного из ограничений $p \in \tilde{I}^{k,+}$ и одной из переменных $x_l = 1$, $l \in \tilde{J}_t^{k,-}$ справедливы условия

$$\tilde{V}^{2,k} = \left\{ x_l = 1,\ l \in (\tilde{J}_t^{k,-} \cap \bar{Z}_{k,p}^{k,-}) | q_j < 0; \right.$$

$$\left. \sum_{j \in \left(\tilde{J}_t^{k,-} \cap \bar{Z}_{k,p}^{k,-}\right)} g_{ij} \leqslant G_{t,p}^k, \quad \sum_{j \in \left(\tilde{J}_t^{k,-} \cap \bar{Z}_{k,p}^{k,-}\right)} g_{pj} - g_{pl} > G_{t,p}^k \right\},$$

то в допустимом дополняющем плане подмножества вариантов $\tilde{\Omega}_t^k$ должно быть принято $x_l = 1$, $l \in (\tilde{J}_t^{k,-} \cap \bar{Z}_{k,p}^{k,-})$.

Утверждение 8. Верхняя граница оптимального решения для k-го подмножества вариантов определяется по формуле

$$\xi(F|\tilde{\Omega}_t^k) = \sum_{j \in \tilde{J}_t^{k,1}} q_j + \sum_{j \in \bar{Y}^{k,+}} q_j. \qquad (26)$$

Выделение областей, не содержащих допустимых планов

Рассмотрим итеративные процессы выделения в подмножестве $\tilde{\Omega}_t^k$ областей, не содержащих допустимых планов, а также исключения из рассмотрения неактивных ограничений $\tilde{I}^{k,-}$.

Алгоритм 1. На каждом шаге итеративного процесса выполняем представленные далее вычисления.

Шаг 1. Рассматриваем все активные ограничения $i \in \tilde{I}^{k,\prime+}$.

Если для некоторого ограничения p выполняются условия утверждения 2, то k-е подмножество вариантов не содержит допустимых решений, т. е. $\tilde{\Omega}_t^k = \varnothing$, где значение $R_{t,p}^k$ вычисляется по формуле (18), оно исключается из дальнейшего рассмотрения и алгоритм завершает свою работу.

Если для некоторого ограничения p выполняются условия утверждения 1, то это p-е ограничение исключается из множества ограничений:

$$\tilde{I}^{k,-} = (\tilde{I}^{k,-} \cup p),\ p \in \tilde{I}^{k,\prime+} = (\tilde{I}^{k,\prime+}/p).$$

Переходим к шагу 2.

Шаг 2. Если выполняются условия следствия утверждения 1, т. е. $\tilde{I}^{k,\prime+} = \varnothing$, то полагаем значения переменных (23), и алгоритм завершает свою работу. В противном случае переходим к шагу 3.

Шаг 3. Для каждого из активных ограничений $i \in \tilde{I}^{k,+}$ проверяем справедливость условий утверждений 3—5.

Если для некоторой переменной x_j, $j \in \tilde{J}_t^{k,+}$ выполняются условия утверждения 3 в виде (22), то полагаем $x_j = 0$, $\tilde{J}_t^{k,-} = (\tilde{J}_t^{k,-}/j)$, $\tilde{J}_t^{k,\prime\prime+} = (\tilde{J}_t^{k,-} \cup j)$.

Если для некоторой переменной $l \in \tilde{J}_t^{k,-} \cap \bar{Z}_{t,i}^{k,-}$ справедливы условия (24), то полагаем $x_l = 1$, $\tilde{J}_t^{k,-} = (\tilde{J}_t^{k,-}/l)$, $\tilde{J}_t^{k,\prime\prime+} = (\tilde{J}_t^{k,-} \cup l)$.

Если для некоторой переменной $j \in \tilde{J}_t^{k,-} \cap \bar{Z}_{t,i}^{k,\prime+}$ условия утверждения 5, т. е. справедливы условия (25), то полагаем

$$x_j = 1,\ j \in \tilde{J}_t^{k,-},\ \tilde{J}_t^{k,-} = (\tilde{J}_t^{k,-}/j),$$
$$\tilde{J}_t^{k\prime\prime,+} = (\tilde{J}_t^{k,-} \cup j).$$

Если $\tilde{J}_t^{k,\prime\prime-} = \varnothing$, то переходим к шагу 4. В противном случае переходим к шагу 1.

Шаг 4. Полагаем

$$\tilde{J}_{t+1}^{k,-} = \tilde{J}_t^{k,-},\ \tilde{J}_{t+1}^{k,+} = \tilde{J}_t^{k,\prime+},$$
$$\tilde{I}_{t'+1}^{k,\prime+} = \tilde{I}_t^{k,\prime+},\ \tilde{I}_{t+1}^{k,\prime-} = \tilde{I}_t^{k,\prime-}.$$

Алгоритм завершает свою работу.

Через некоторое число итераций алгоритма будут определены значения некоторого количества переменных задачи и исключены из рассмотрения ряд несущественных ограничений. В результате работы алгоритма будет построено новое подмножество вариантов $\tilde{\Lambda}_t^k \subseteq \tilde{\Omega}_t^k$.

Ветвление. Разбиение на подмножества

Алгоритм 2. Для подмножества вариантов $\tilde{\Omega}_t^k$ вычисляем верхнюю границу целевой функции по формуле (26). Выбираем индекс переменной $\rho \in \tilde{J}_t^{k,-}$, стоящий первым в одной из последовательностей $\tilde{U}_t^{k,1}$, $\tilde{U}_t^{k,2}$ или $\tilde{U}_t^{k,3}$, определенных выражениями (20)—(22).

Подмножество вариантов $\tilde{\Omega}_t^k$ разбиваем на два подмножества

$$\tilde{\Omega}_t^{k,1} = \{\tilde{\Omega}_t^k | x_\rho = 1\},\ \tilde{\Omega}_t^{k,2} = \{\tilde{\Omega}_t^k | x_\rho = 0\}.$$

Для вновь построенных подмножеств вариантов $\tilde{\Omega}_t^{k,1}$ и $\tilde{\Omega}_t^{k,2}$ могут быть либо вычислены по формуле (26) значения $\xi(F|\tilde{\Omega}_t^{k,1})$ и $\xi(F|\tilde{\Omega}_t^{k,2})$, либо в начале алгоритмом 1 следует выполнить исключение областей, не содержащих допустимых и оптимальных планов, а затем вычислить для вновь построенных $\tilde{\Lambda}_t^{k,1}$ и $\tilde{\Lambda}_t^{k,2}$

подмножеств верхнюю границу целевой функции по формуле (26).

Вычислительная схема алгоритма. Алгоритм решения задачи представляет собой итеративный процесс, каждая итерация которого предусматривает выполнение следующих шагов.

Алгоритм 3. Обозначим $\tilde{\Omega}_0^0$ — множество переменных задачи, удовлетворяющих системе ограничений (17);

$$\tilde{J}_0^{0,1} = \tilde{J}_0^{0,0} = \varnothing,\ \tilde{J}_0^{0,-} = \tilde{J}\{1, ..., j, ..., n\};$$
$$\tilde{I}^{0,+} = \{1, ..., i, ..., M\},\ \tilde{I}^{0,-} = \varnothing.$$

Пусть $\bar{F}_t^K$ — наилучшие из решений на t-м шаге решения. Вычислим верхнюю границу целевой функции для подмножества вариантов $\tilde{\Omega}_0^0$ по формуле (26). Положив $k = t = 0$, $\tilde{\Omega}_0^0 = -\infty$ переходим к шагу 1.

Шаг 1. Среди всех перспективных подмножеств вариантов $k = 1, ..., K$ выберем подмножество $\tilde{\Omega}_t^\gamma$ с наилучшим значением верхней границы

$$\xi(F|\tilde{\Omega}_t^\delta) = \max_{1 \leqslant k \leqslant K} \xi(F|\tilde{\Omega}_t^k). \qquad (27)$$

Если $\xi(F|\tilde{\Omega}_t^\delta) \leqslant \bar{F}_t^K$, то решение задачи получено и алгоритм заканчивает свою работу. В противном случае выбираем подмножество вариантов $\tilde{\Omega}_t^\delta$ $(k = \delta)$ и переходим к шагу 2.

Шаг 2. Для подмножества вариантов $\tilde{\Omega}_t^{k=\gamma}$ алгоритмом 1 выполняем итеративные процессы исключения подобластей, не содержащих допустимых и оптимальных решений, и построение подмножества вариантов $\tilde{\Lambda}_t^{k=\gamma} \subseteq \tilde{\Omega}_t^{k=\gamma}$.

Если $\tilde{\Lambda}_t^{k=\gamma} = \varnothing$, то исключаем $\tilde{\Omega}_t^{k=\gamma}$ из рассмотрения, проводим переиндексацию и уменьшение числа подмножеств рассматриваемых вариантов $k = 1, ..., K := (K - 1)$, и переходим к шагу 1. Если $\tilde{\Lambda}_t^{k=\gamma} = \varnothing$, то вычисляем $\xi(F|\tilde{\Lambda}_t^{k=\gamma} = \tilde{\Omega}_t^{k=\gamma})$ по формуле (26). В случае если $\tilde{J}_t^{k,1} \cup \tilde{J}_t^{k,0} = \tilde{J}$ и $\tilde{J}_t^{k,-} = \varnothing$, то полагаем $\xi(\bar{F}|\tilde{\Lambda}_t^\gamma = \tilde{\Omega}_t^\gamma) = \xi(F|\tilde{\Lambda}_t^{k=\gamma} = \tilde{\Omega}_t^{k=\gamma})$. Если $\xi(\bar{F}|\tilde{\Lambda}_t^\gamma = \tilde{\Omega}_t^\gamma) > \bar{F}_t^K$, то полагаем $\bar{F}_t^K = \xi(\bar{F}|\tilde{\Lambda}_t^\gamma = \tilde{\Omega}_t^\gamma)$. Переходим к шагу 3.

Если $\tilde{\Lambda}_t^{k=\gamma} = \varnothing$ и $\tilde{J}_t^{k,-} = \varnothing$, то переходим к шагу 3.

Шаг 3. Выбираем подмножество вариантов с наилучшим значением верхней границы $\tilde{\Omega}_t^\delta$ по формуле (27). Выполним алгоритмом 2 разбиение этого подмножества на два различных подмножества, вычислим по формуле (26) значения $\xi(F|\tilde{\Omega}_t^{\delta,1})$ и $\xi(F|\tilde{\Omega}_t^{\delta,2})$.

Исключаем из рассмотрения множество вариантов $\tilde{\Omega}_t^\delta$, вводим новые подмножества $\tilde{\Omega}_t^{\delta,1}$ и $\tilde{\Omega}_t^{\delta,2}$, увеличиваем K на 1 и проводим переиндексацию всех подлежащих рассмотрению подмножеств вариантов. Полагаем $k = K$, $t = (t + 1)$. Выбираем подмножество вариантов $\tilde{\Omega}_t^k = \arg\max[\xi(F|\tilde{\Omega}_t^{\delta,1})$ и $\xi(F|\tilde{\Omega}_t^{\delta,2})]$. Переходим к шагу 2.

Через некоторое число шагов алгоритма либо будет получено оптимальное решение задачи, либо установлен факт о несовместности исходной системы ограничений.

4. Иллюстративный пример

Рассмотрим задачу Fuzzy-линейного булевого программирования

$$F = \bar{C}_1 x_1 \oplus \bar{C}_2 x_2 \oplus \bar{C}_3 x_3 \to \max,$$

в условиях ограничений

$$\bar{A}_{11}x_1 \oplus \bar{A}_{12}x_2 \oplus \bar{A}_{13}x_3 \,\bar{\leqslant}_{\mathrm{Re}}\, \bar{B}_1;$$
$$2\bar{A}_{21}x_1 \oplus (-3\bar{A}_{22})x_2 \oplus \bar{A}_{23}x_3 \,\bar{\leqslant}_{\mathrm{Re}}\, \bar{B}_2.$$

Функции принадлежности Fuzzy-множеств коэффициентов критерия оптимальности, левых частей ограничений задачи и граничных значений заданы LR-представлениями треугольного вида $\{a_1, m, a_2\}_{LR}$:

$$\bar{C}_1 = \{1, 2, 4\}_{LR},\ \bar{C}_2 = \{-1, 1, 2\}_{LR},\ \bar{C}_3 = \{-1, 0, 3\}_{LR};$$

$$\bar{A}_{11} = \{0, 2, 3\}_{LR},\ \bar{A}_{12} = \{0, 1, 3\}_{LR},$$
$$\bar{A}_{13} = \{0, 2, 4\}_{LR};\ \bar{B}_1 = \{7, 8, 9\}_{LR};$$

$$\bar{A}_{21} = \{0{,}5, 1, 2\}_{LR},\ \bar{A}_{22} = \{1, 1, 2\}_{LR},\ \bar{A}_{23} = \{1, 2, 5\}_{LR};$$
$$\bar{B}_2 = \{3, 6, 10\}_{LR}.$$

При построении детерминированного эквивалента потребуем выполнение детерминированных ограничений в виде

$$a_{1p}(\bar{A}_{i1})x_1 + a_{1p}(\bar{A}_{i2})x_2 +$$
$$+ a_{1p}(\bar{A}_{i3})x_3 \leqslant a_{1p}(\bar{B}_i),\ p = 1, 2,\ i = 1, 2;$$
$$m(\bar{A}_{i1})x_1 + m(\bar{A}_{i2})x_2 + m(\bar{A}_{i3})x_3 \leqslant m(\bar{B}_i),\ i = 1, 2.$$

Трехкритериальную задачу

$$a_{1p}(\bar{C}_1)x_1 + a_{1p}(\bar{C}_2)x_2 + a_{1p}(\bar{C}_3)x_3 \to \max,\ p = 1, 2;$$
$$m(\bar{C}_1)x_1 + m(\bar{C}_2)x_2 + m(\bar{C}_3)x_3 \to \max,$$

решим введением компромиссного критерия в виде линейной свертки критериев вида

$$\Phi = \sum_{j=1}^{3} \left\{ \beta_1 \frac{1}{2} [a_j(\bar{C}_j) + a_j(\bar{C}_j)] + \beta_2 m(\bar{C}_j) \right\} x_j \to \max.$$

Здесь $\beta_1 = 0{,}3$, $\beta_2 = 0{,}7$ — весовые коэффициенты.

Следовательно, детерминированным эквивалентом рассматриваемой задачи является задача линейного булевого программирования с одним критерием оптимальности и шестью ограничениями:

$$\Phi = 6{,}45x_1 - 0{,}85x_2 + 0{,}6x_3 \to \max, \qquad (28)$$

$$\varphi_{11} = 0x_1 + 0x_2 + 0x_3 \leqslant 7;\ \varphi_{12} = 2x_1 + x_2 + 2x_3 \leqslant 8;$$
$$\varphi_{13} = 3x_1 + 3x_2 + 4x_3 \leqslant 9;$$

$$\varphi_{21} = x_1 - 3x_2 + x_3 \leqslant 3;\ \varphi_{22} = 2x_1 - 3x_2 + 2x_3 \leqslant 6;$$
$$\varphi_{23} = 4x_1 - 6x_2 + 5x_3 \leqslant 10.$$

Легко заметить, что все ограничения, кроме ограничения φ_{13}, выполняются при любых значениях переменных, и только ограничение

$$\varphi_{13} = 3x_1 + 3x_2 + 4x_3 \leqslant 9 \qquad (29)$$

является существенным.

Решение задачи (28), (29) тривиально: $x_1 = x_3 = 1$, $x_2 = 0$.

Следовательно, Fuzzy-множества ограничений задачи имеют следующий вид:

$$\bar{\Psi}_1 = \{0, 4, 7\}_{LR} \,\bar{\leqslant}_{\mathrm{Re}}\, \bar{B}_1 = \{7, 8, 9\}_{LR};$$
$$\bar{\Psi}_2 = \{2, 4, 8\}_{LR} \,\bar{\leqslant}_{\mathrm{Re}}\, \bar{B}_2 = \{3, 6, 10\}_{LR}.$$

Fuzzy-множество целевой функции в оптимальном решении — $\bar{F} = \{1, 6, 18\}_{LR}$.

Заключение

На основе различных методов сравнения и определения безусловных и относительных условий предпочтения нечетких множеств для широкого класса функций принадлежности граничных значений и коэффициентов левых частей ограничений и целевых функций предложены различные виды детерминированных эквивалентов задачи Fuzzy-линейного булевого программирования. Их анализ при решении практических задач позволит выбрать соответствующую детерминированную модель в соответствии с предпочтениями лица, принимающего решение, и экспертов в целях достижения максимального эффекта и обеспечения надежности выполнения всех ограничений.

Учитывая большое число ограничений детерминированных эквивалентов этих задач в виде моделей линейного булевого программирования, рассмотрены эффективные методы их решения. Такие методы уже на начальных этапах решения (при исключении областей, не содержащих допустимых и оптимальных планов) позволяют установить либо факт несовместности исходной системы ограничений и перейти к более грубой модели, либо определить значения большого числа переменных задачи.

Встречающиеся в практических приложениях задачи нечеткого булевого линейного программирования, как правило, небольшого размера. Этот факт означает, что даже на современных персональных компьютерах решение их описанным алгоритмом не потребует больших затрат вычислительных ресурсов. Предложенный алгоритм при односторонней схеме ветвления может использоваться и для получения приближенного решения.

Описанные в работе алгоритмы могут быть использованы на контроллерах и микропроцессорах, в системах управления конкретными системами, а также служить основой при создании программных комплексов подготовки принятия решений.

Список литературы

1. **Язенин А. В.** Нечеткое математическое программирование. Калинин: Изд-во Калининского гос. ун-та, 1986. 60 с.
2. **Rommelfanger H.-J.** FULPAL 2.0 — An Interactive Algorithm for Solving Multicriteria Fuzzy Linear Programs Controlled by Aspiration Levels // Methods of multickriteria decision theory / Eds. D. Schegert. Lamprecht: Pfalzakademie, 1995. P. 21—34.
3. **Kumar A., Kaur J., Singh P.** A new method for solving fully fuzzy linear programming problems // Applied Mathematical Modelling. 2011. Vol. 35. N. 2. P. 817—823.
4. **Kumar A., Kaur J., Singh P.** Fuzzy optimal solution of fully fuzzy linear programming problems with inequality constraints // International Journal of Applied Mathematics and Computer Sciences. 2010. Vol. 6. P. 37—41.
5. **Kumar A., Kaur J., Singh P.** Fuzzy linear programming problems with fuzzy parameters // Journal of Advanced Research in Scientific Computing. 2010. Vol. 2. P. 1—12.
6. **Herrera F., Verdegay J. L.** Three models of fuzzy integer linear programming // European Journal of Operational Research. 1995. Vol. 83. P. 581—593.
7. **Allahviranloo T., Lotfi F. H., Kiasary M. K., Kiani N. A., Alizadeh L.** Solving fully fuzzy linear programming problem by the ranking functhion // Applied Mathematical Sciences. 2008. Vol. 2. P. 19—32.
8. **Williams H. P.** Logic and Integer Programming (International Series in Operations Research & Management Science). Dorbrecht—Heidelberg—London—New-York: Springer, 2009. 155 p.
9. **Maleki R., Tata M., Mashinchi M**. Linear programming with fuzzy variables // Fuzzy Sets and Systems. 2000. Vol. 109. P. 21—33.
10. **Зак Ю. А.** Принятие решений в условиях размытых и нечетких данных. М.: Либроком, 2012. 349 с.
11. **Inguichi M., Ramik J.** Possibilistic linear programming: a brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem // Fuzzy Sets and System. 2000. N. 11. P. 3—28.
12. **Barth P.** A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization. Saarbrücken: Max-Plank-Institut für Informatik, 1995. 12 p.
13. **Зак Ю. А.** Критерии и методы сравнения нечетких множеств // Системные исследования и информационные технологии. 2013. № 3. С. 58—68.
14. **Зак Ю. А.** Множество Парето для критериев эффективности, представленных стохастическими и нечеткими данными // Искусственный интеллект и принятие решений. 2014. № 2. С. 95—107.

15. **Zack Yu. A.** Mathematical models of trade-off schemes in multicriteria mathematical programming problems with fuzzy constraints // Cybernetics and systems analysis. 2010. Vol. 46, N. 5. P. 755—771.
16. **Зак Ю. А.** Четкий эквивалент задачи Fuzzy-линейного программирования // Проблемы управления и информатики. 2011. № 1. С. 84—101.
17. **Корбут А. А., Финкельштейн Ю. Ю.** Дискретное программирование. М.: Наука, 1969. 368 с.
18. **Карманов В. Г.** Математическое программирование. М.: Наука, 1986. 288 с.
19. **Зак Ю. А.** Об одном классе задач целочисленного линейного программирования с булевыми переменными // Автоматика и вычислительная техника. 1977. № 6. С. 26—33.

Yu. A. Zack, Consultant, e-mail: yuriy_zack@hotmail.com, Aachen, Deutschland

Algorithms Fuzzy-Linear Boolean Programming

Considered setting deterministic equivalents and methods for solving Boolean Fuzzy- linear programming in an environment where the parameters of the objective function and constraints, as well as the permissible limit values of the constraint vector problem — fuzzy sets with given trapezoidal membership functions and the triangular form. Solution to the problem is sought in the form of a vector of Boolean variables. Methods for comparing and ranking fuzzy sets, as well as original algorithms for solving linear programming boolean branch and bound method.

Keywords: *fuzzy sets, Fuzzy-Boolean linear programming, deterministic equivalent, comparison and ranking of fuzzy sets, branch and bound method*

References

1. **Jazenin A. V.** *Nechetkoe matematicheskoe programmirovanie*. Kalinin: Izd-vo Kalininsk. gos. un-ta, 1986. 60 p.
2. **Rommelfanger. H.-J.** FULPAL 2.0 — An Interactive Algorithm for Solving Multicriteria Fuzzy Linear Programs Controlled by Aspiration Levels. *Methods of multickriteria decision theory*. Eds. D. Schegert. Lamprecht: Pfalzakademie, 1995. P. 21—34.
3. **Kumar A., Kaur J., Singh P.** A new method for solving fully fuzzy linear programming problems. A*pplied Mathematical Modelling*. 2011. Vol. 35. N. 2. P. 817—823.
4. **Kumar A., Kaur J., Singh P.** Fuzzy optimal solution of fully fuzzy linear programming problems with inequality constraints. *International Journal of Applied Mathematics and Computer Sciences*. 2010. Vol. 6. P. 37—41.
5. **Kumar A., Kaur J., Singh P**. Fuzzy linear programming problems with fuzzy parameters. *Journal of Advanced Research in Scientific Computing*. 2010. Vol. 2. P. 1—12.
6. **Herrera F., Verdegay J. L.** Three models of fuzzy integer linear programming. *European Journal of Operational Research*. 1995. Vol. 83. P. 581—593.
7. **Allahviranloo T., Lotfi F. H., Kiasary M. K., Kiani N. A., Alizadeh L.** Solving fully fuzzy linear programming problem by the ranking functhion. *Applied Mathematical Sciences*. 2008. Vol. 2. P. 19—32.
8. **Williams H. P**. *Logic and Integer Programming* (International Series in Operations Research & Management Science). Dorbrecht-Heidelberg-London- New-York: Springer, 2009. 155 p.
9. **Maleki R., Tata M., Mashinchi M.** Linear programming with fuzzy variables. *Fuzzy Sets and Systems*. 2000. Vol. 109. P. 21—33.
10. **Zack Yu. A.** Prinyatije reshenij v uslovijach razmitich i nechetkich dannich. M.: Librokom, 2012. 349 p.
11. **Inguichi M., Ramik J.** Possibilistic linear programming: a brief review of fuzzy mathematical programming and a comparison with stochastic programming in portfolio selection problem. *Fuzzy Sets and System*. 2000. N. 11. P. 3—28.
12. **Barth P.** *A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization*. Saarbrücken: Max-Plank-Institut für Informatik, 1995. 12 p.
13. **Zack Yu. A.** Kriterii i metodi sravnenija nechetkich mnozhestv. *Sistemnije issledovanija i informazionnije technologii*. 2013. N. 3. P. 58—68.
14. **Zack Yu. A.** Mnozhestvo Pareto dlya kriterijev, predstavlennich stochasticheskimi i nechetkimi dannimi. *Iskusstvennij intellekt i prinyatije reshenij*. 2014. N. 2. P. 95—107.
15. **Zack Yu. A.** Mathematical models of trade-off schemes in multicriteria mathematical programming problems with fuzzy constraints. *Cybernetics and systems analysis*. 2010. Vol. 46. N. 5. P. 755—771.
16. **Zack Yu. A.** Chetkij determinirovannij ekvivalent zadachi Fuzzy-linejnogo programmirovanija. *Problemi upravlenija i informatiki*. 2011. N. 1. P. 84—101.
17. **Korbut A. A., Finkelshtein Yu. Yu**. *Diskretnoje programmirovanije*. M.: Nauka, 1969. 368 p.
18. **Karmanov V. G**. *Matematicheskoje programmirovanije*. M.: Nauka, 1986. 288 p.
19. **Zack Yu. A**. Ob odnom klasse sadach zelochislennogo linejnogo programmirovanija s bulevimi peremennimi. *Avtomatika i vichislitelnajaja technika*.1977. N. 6. P. 26—33.

УДК 629.73.02; 629.73.05/.06; 535.643

И. О. Жаринов, д-р техн. наук, доц., зав. каф., Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики (НИУ ИТМО), руководитель учебно-научного центра, Санкт-Петербургское ОКБ "Электроавтоматика" им. П. А. Ефимова, e-mail: igor_rabota@pisem.net,
О. О. Жаринов, канд. техн. наук, доц., Санкт-Петербургский государственный университет аэрокосмического приборостроения (ГУАП)

Оценка инструментальной погрешности косвенного измерения координат цвета в цветовой модели данных, применяемой в авионике

Рассмотрена задача оценки инструментальной погрешности косвенного измерения координат цвета в цветовой модели RGB при проведении экспериментов с жидкокристаллическими матрицами. Показано, что на значение погрешности косвенного измерения оказывают влияние приборные погрешности измерительного оборудования, принятый разработчиками матрицы стандарт баланса белого цвета и измеряемые прямым способом значения координат цвета и цветности. Рассмотрены два способа косвенного измерения компонентов кодов RGB: на основе прямого измерения координат цвета и прямого измерения координат цветности с компонентом относительной яркости. Получены в аналитическом виде выражения для оценки абсолютной и относительной составляющих инструментальной погрешности косвенного измерения обоих способов измерений. Приведены результаты компьютерного моделирования — графики зависимости максимумов абсолютной погрешности косвенного измерения компонентов кода RGB для обоих способов измерения и размеров приборных погрешностей на уровне существующей сегодня и перспективной аппаратур.

Ключевые слова: *косвенное измерение, инструментальные погрешности, координаты цвета, координаты цветности, модель RGB, оценка*

Введение

При разработке бортовых систем индикации пилотируемых летательных аппаратов актуальной является задача исследования эргономических свойств бортовых средств отображения информации [1—3]. Эргономические исследования предполагают, в первую очередь, получение оценок визуальных характеристик восприятия пилотом изображения (яркость, яркостной контраст), оптических характеристик (коэффициенты диффузного и зеркального отражения) жидкокристаллического (ЖК) экрана бортового индикатора и др.

Ранее [4] авторами было показано, что решение задачи выбора координат цветности элементов изображения, индицируемых на ЖК экране бортового индикатора класса МФЦИ (многофункциональный цветной индикатор [5]) и обладающих повышенными визуальными характеристиками восприятия для человека, сопряжено с необходимостью "измерения" компонентов управляющих кодов *RGB* (*R — Red*, *G — Green*, *B — Blue*), с использованием которых подготовлено функциональное программное обеспечение МФЦИ.

Прямое измерение компонентов кода *RGB* сегодня невозможно, так как существующая измерительная аппаратура (колориметры, спектроколориметры, спектрорадиометры) осуществляет измерение только координат цвета в системе *XYZ* и/или координат цветности (*x*, *y*) в системе *XY*, по которым впоследствии оценивают десятичный код компонентов цветовой модели *RGB*. Таким образом, на практике реализуется исключительно косвенный способ измерения компонентов кода *RGB* путем прямого измерения координат цвета и/или координат цветности и последующего математического расчета.

Цель статьи заключается в представлении широкому кругу читателей результатов исследований, в процессе выполнения которых были получены аналитические выражения для оценки относительной и абсолютной инструментальных погрешностей косвенного измерения компонентов кодов *RGB*.

1. Принцип формирования изображения в бортовых системах индикации

Изображение на экране МФЦИ формируется путем отображения в различных цветах пикселей ЖК экрана. Совокупности пикселей, структурированные определенным образом, образуют знаки, мнемосимволы, линии, секторы, дуги, окружности с заливкой и без и т. д., индицируемые на экране МФЦИ. Цвет отображаемых пикселей определяется углами поворота жидких кристаллов в красном, зеленом и синем компонентах цвета модели данных *RGB*, поляризующих задний или боковой подсвет белого спектра свечения, создаваемый лампами подсвета ЖК экрана.

Число отображаемых на экране МФЦИ оттенков каждого цвета определяется разрядностью данных ЖК панели и особенностями построения тракта видеоОЗУ графического контроллера МФЦИ. Так, для ЖК панели с 6-битным форматом данных по компонентам *RGB* возможно отображение 2^6 оттенков для каждого из основных следующих цветов:

- красного цвета (двоичный позиционный код управления углом поворота жидкого кристалла каждого пикселя $R_0...R_5$ = {000000}, {000001}, ..., {100000}, ..., {111111});
- зеленого цвета (двоичный позиционный код управления углом поворота жидкого кристалла каждого пикселя $G_0...G_5$ = {000000}, {000001}, ..., {100000}, ..., {111111});
- синего цвета (двоичный позиционный код управления углом поворота жидкого кристалла каждого пикселя $B_0...B_5$ = {000000}, {000001}, ..., {100000}, ..., {111111}),

а также их 2^{18} комбинаций, образующих полную цветовую палитру ЖК матрицы. Аналогично для 8-битного *RGB*-формата данных ЖК панели $\{R_0...R_7, G_0...G_7, B_0...B_7\}$ число отображаемых на экране МФЦИ цветов и оттенков составит 2^{24}.

Функциональная схема тракта формирования изображения в графическом контроллере (графическом модуле) МФЦИ приведена на рис. 1, *а*; пример графического изображения, индицируемого на ЖК экране МФЦИ, — на рис. 1, *б*.

Канал формирования изображения состоит из трех банков видеоОЗУ; элементов программируемой логической интегральной схемы (ПЛИС); ПЗУ знаков и цветов, программируемого пользователем извне; шинного формирователя (ШФ), необходимого для поддержки межмодульного интерфейса графического контроллера; ПЗУ загрузки ПЛИС и передатчика интерфейса LVDS (*Low Voltage Digital Signal*) — интерфейса ЖК панели. Очевидно, для воспроизведения всей гаммы цветовой палитры на экране МФЦИ разрядность данных видеоОЗУ и передатчика LVDS должна совпадать или превышать разрядность формата данных ЖК панели для определения углов поворота жидких кристаллов.

Система кодирования цвета в модели *RGB* предполагает способ задания каждого цвета или оттенка цвета в графическом контроллере МФЦИ в виде двоичного позиционного кода, в котором присутствуют группы разрядов, соответствующие компонентам основных цветов (красный, зеленый, синий). Значения кодов основных цветов могут находиться в произвольных пропорциях. При программировании графического контроллера МФЦИ коды *RGB* принято представлять в десятичной системе счисления. Система *XY* предполагает способ задания каждого цвета или оттенка цвета в виде пары вещественных (*x*, *y*)-координат на *XY*-плоскости.

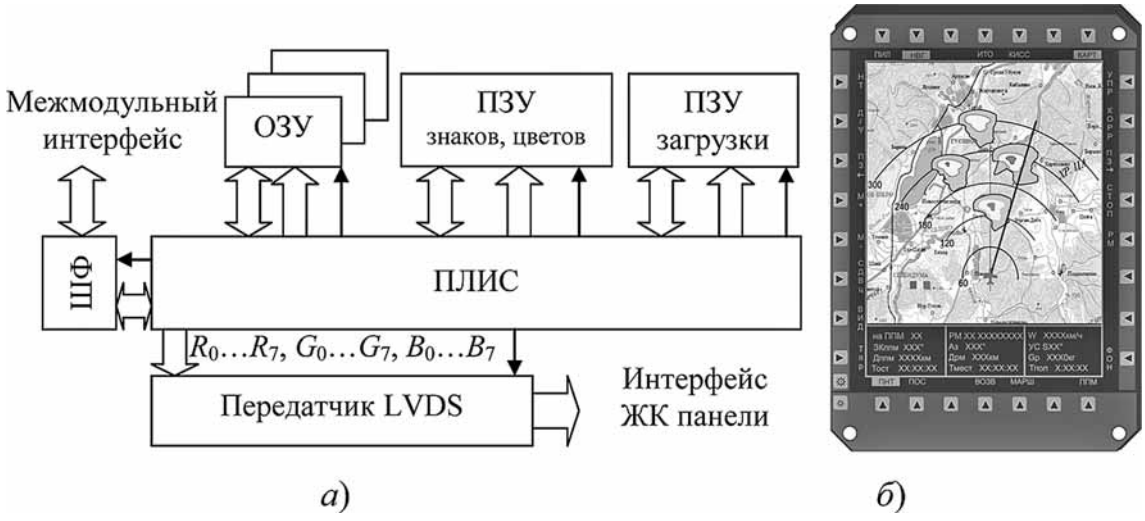

Рис. 1. Функциональная схема тракта формирования изображения в графическом контроллере (*а*) и пример индикационного кадра на ЖК экране МФЦИ (*б*)

Переход из одной системы кодирования цвета в другую осуществляется через цветовой треугольник Максвелла по правилам прямого $RGB \to XYZ$ и обратного $XYZ \to RGB$ преобразований Грассмана [4]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \tilde{X}_r & \tilde{X}_g & \tilde{X}_b \\ \tilde{Y}_r & \tilde{Y}_g & \tilde{Y}_b \\ \tilde{Z}_r & \tilde{Z}_g & \tilde{Z}_b \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{1}$$

$$\begin{bmatrix} \tilde{X}_r & \tilde{X}_g & \tilde{X}_b \\ \tilde{Y}_r & \tilde{Y}_g & \tilde{Y}_b \\ \tilde{Z}_r & \tilde{Z}_g & \tilde{Z}_b \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1},$$

где X, Y, Z — компоненты цвета в системе XYZ цветового треугольника Максвелла; X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b — компоненты цвета, определенные Международной комиссией по освещению (МКО) и используемые в качестве эталона для точного стандарта определения цвета; R, G, B — десятичный код цвета компонентов основных цветов в системе RGB. Компоненты X_r, Y_r, Z_r определяют правило преобразования кода RGB для эталонного значения красного цвета, компоненты X_g, Y_g, Z_g и X_b, Y_b, Z_b — для зеленого и синего цветов соответственно.

Переход от значений координат цвета XYZ треугольника Максвелла к (x, y)-координатам цветности на XY-плоскости осуществляется по следующим формулам [4]:

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}. \tag{2}$$

2. Принцип косвенного измерения координат цвета в цветовой модели *RGB*

Косвенное измерение координат цвета в цветовой модели RGB осуществляется двумя способами:

1) путем прямого измерения колориметром (спектроколориметром, спектрорадиометром) координат цвета в системе XYZ;

2) путем прямого измерения колориметром (спектроколориметром, спектрорадиометром) координат цветности (x, y) и прямого измерения компонента относительной яркости Y_{xy} и последующим вычислением на основе измерений значений кода RGB.

Вычисление кода RGB по результатам прямых измерений координат цвета XYZ (способ 1) осуществляется с использованием обратного преобразования Грассмана (1):

$$\begin{cases} \hat{R} = \tilde{X}_r \hat{X} + \tilde{X}_g \hat{Y} + \tilde{X}_b \hat{Z} \\ \hat{G} = \tilde{Y}_r \hat{X} + \tilde{Y}_g \hat{Y} + \tilde{Y}_b \hat{Z} \\ \hat{B} = \tilde{Z}_r \hat{X} + \tilde{Z}_g \hat{Y} + \tilde{Z}_b \hat{Z} \end{cases}, \tag{3}$$

где $\hat{R}$, $\hat{G}$, $\hat{B}$ — оцененные значения компонентов кода RGB, $\hat{X}$, $\hat{Y}$, $\hat{Z}$ — измеренные (оцененные по результатам серии измерений) значения компонентов координат цвета в системе XYZ.

При вычислении кодов RGB по формуле (3) необходимо учитывать, что измерительный прибор индицирует результаты измерений значений координат цвета $\hat{X}_{xy}$, $\hat{Y}_{xy}$, $\hat{Z}_{xy}$ в относительных единицах, нормированных по значению компонента Y, принимаемого за 100 % в точке белого, в связи с чем в систему (3) необходимо подставлять не сами значения $\hat{X}_{xy}$, $\hat{Y}_{xy}$, $\hat{Z}_{xy}$, а значения $\hat{X}$, $\hat{Y}$, $\hat{Z}$, подвергнутые денормировке: $\hat{X} = 255\hat{X}/100$, $\hat{Y} = 255\hat{Z}/100$, $\hat{Z} = 255\hat{Z}/100$.

Вычисление кодов RGB по результатам прямых измерений координат цветности (x, y) и измерения компонента относительной яркости Y_{xy} (способ 2) осуществляются следующим образом:

$$\begin{cases} \hat{R} = \tilde{X}_r \dfrac{\hat{x}\hat{Y}_{xy}}{y} + \tilde{X}_g \hat{Y}_{xy} + \tilde{X}_b \left(1 - \hat{x} - \hat{y}\right) \dfrac{\hat{Y}_{xy}}{\hat{y}} \\ \hat{G} = \tilde{Y}_r \dfrac{\hat{x}\hat{Y}_{xy}}{y} + \tilde{Y}_g \hat{Y}_{xy} + \tilde{Y}_b \left(1 - \hat{x} - \hat{y}\right) \dfrac{\hat{Y}_{xy}}{\hat{y}} \\ \hat{B} = \tilde{Z}_r \dfrac{\hat{x}\hat{Y}_{xy}}{y} + \tilde{Z}_g \hat{Y}_{xy} + \tilde{Z}_b \left(1 - \hat{x} - \hat{y}\right) \dfrac{\hat{Y}_{xy}}{\hat{y}} \end{cases}, \tag{4}$$

где $\hat{x}$, $\hat{y}$, $\hat{Y}_{xy}$ — измеренные (оцененные по результатам серии измерений) координаты цветности (x, y) и измеренное (оцененное) значение компонента относительной яркости Y_{xy}. Компонент $\hat{Y}_{xy}$ также подлежит денормировке.

В системе (4) использованы принятые в колориметрии соотношения, связывающие координаты цвета и координаты цветности в различных цветовых пространствах [6]:

$$X = \frac{xY_{xy}}{y}, \; Y = Y_{xy}, \; Z = (1 - x - y)\frac{Y_{xy}}{y}. \tag{5}$$

3. Погрешности косвенного измерения координат цвета в модели *RGB*

Из теории метрологии известно, что оценка относительной ε_ψ и абсолютной $\Delta\psi$ инструментальной погрешности косвенного измерения величины ψ, вычисляемой по формуле $\psi = f(\xi_1, \xi_2, ..., \xi_\zeta)$, где $f(\cdot)$ — функция произвольного вида, с использованием результатов прямых измерений параметров $\xi_1, \xi_2, ..., \xi_\zeta$, осуществляется следующим образом:

$$\varepsilon_\psi = \sqrt{\sum_{i=1}^{\zeta}\left(\left.\frac{\partial \ln f\left(\xi_1,\xi_2,...,\xi_\zeta\right)}{\partial\xi_i}\right|_{\xi_i=\overline{\xi}_i}\Delta\xi_i\right)^2}\,100\%, \qquad \Delta\psi = \frac{\varepsilon_\psi\overline{\psi}}{100\%}, \tag{6}$$

где ζ — число аргументов функции $f(\cdot)$; $\partial\ln f(\xi_1, \xi_2, ..., \xi_\zeta)/\partial\xi_i$ — частные производные натурального логарифма от функции $f(\cdot)$ по аргументам ξ_i; $\Delta\xi_i$ — абсолютная погрешность прямого измерения параметра ξ_i; $\overline{\psi}$, $\overline{\xi}_i$ — средние значения параметров, измеряемых косвенным и прямым способом соответственно.

Способ косвенного измерения 1. Для компонентов кода *RGB*, измеряемых косвенным способом и вычисляемых в соответствии с системой уравнений (3), уравнения относительных ε_R, ε_G, ε_B и абсолютных ΔR, ΔG, ΔB инструментальных погрешностей косвенного измерения координат цвета основных цветов в цветовой модели *RGB* имеют следующий вид:

$$\begin{cases} \varepsilon_R = \sqrt{\left(\tilde{X}_r\Delta X\right)^2 + \left(\tilde{X}_g\Delta Y\right)^2 + \left(\tilde{X}_b\Delta Z\right)^2} \Big/ \left(\tilde{X}_r\overline{X} + \tilde{X}_g\overline{Y} + \tilde{X}_b\overline{Z}\right)^2 \cdot 100\% \\ \varepsilon_G = \sqrt{\left(\tilde{Y}_r\Delta X\right)^2 + \left(\tilde{Y}_g\Delta Y\right)^2 + \left(\tilde{Y}_b\Delta Z\right)^2} \Big/ \left(\tilde{Y}_r\overline{X} + \tilde{Y}_g\overline{Y} + \tilde{Y}_b\overline{Z}\right)^2 \cdot 100\% \\ \varepsilon_B = \sqrt{\left(\tilde{Z}_r\Delta X\right)^2 + \left(\tilde{Z}_g\Delta Y\right)^2 + \left(\tilde{Z}_b\Delta Z\right)^2} \Big/ \left(\tilde{Z}_r\overline{X} + \tilde{Z}_g\overline{Y} + \tilde{Z}_b\overline{Z}\right)^2 \cdot 100\% \end{cases} \tag{7}$$

$$\begin{cases} \Delta R = \varepsilon_R\left(\tilde{X}_r\overline{X} + \tilde{X}_g\overline{Y} + \tilde{X}_b\overline{Z}\right)/100\% \\ \Delta G = \varepsilon_G\left(\tilde{Y}_r\overline{X} + \tilde{Y}_g\overline{Y} + \tilde{Y}_b\overline{Z}\right)/100\% \\ \Delta B = \varepsilon_B\left(\tilde{Z}_r\overline{X} + \tilde{Z}_g\overline{Y} + \tilde{Z}_b\overline{Z}\right)/100\% \end{cases}, \tag{8}$$

где ΔX, ΔY, ΔZ — абсолютные погрешности (паспортные данные измерителя) прямых измерений координат цвета X, Y, Z в системе XYZ; $\overline{X}$, $\overline{Y}$, $\overline{Z}$ — средние значения измеренных параметров.

Анализ паспортных данных измерителей показывает, что $\Delta X = \Delta Y = \Delta Z$. Существующие измерительные приборы, например, спектроколориметр ТКА-ВД (ООО "Научно-техническое предприятие "ТКА", Россия, регистрационный номер 44179-10 Государственного реестра средств измерений Российской Федерации), спектроколориметр LabScan XE (*Hunter Associates Laboratory*, Inc., США, регистрационный номер 35842-07 Государственного реестра средств измерений Российской Федерации) и др., обеспечивают точность прямых измерений координат цвета в системе *XYZ* с абсолютной погрешностью в следующих пределах: $0{,}01 \leqslant \Delta X, \Delta Y, \Delta Z \leqslant 1$. В то же время ГОСТ Р 52870—2007 "Средства отображения информации коллективного пользования. Требования к визуальному отображению информации и способы измерения" рекомендует исследователям в экспериментах использовать измерительные приборы со следующей характеристикой абсолютной погрешности измерений координат цвета: $0{,}07 \leqslant \Delta X, \Delta Y, \Delta Z \leqslant 0{,}12$.

При выводе систем уравнений (7), (8) в соответствии с формулой (6) использовано правило логарифмического дифференцирования для вычисления частных производных $\partial\ln R/\partial X$, $\partial\ln R/\partial Y$, $\partial\ln R/\partial Z$, $\partial\ln G/\partial X$, $\partial\ln G/\partial Y$, $\partial\ln G/\partial Z$, $\partial\ln B/\partial X$, $\partial\ln B/\partial Y$, $\partial\ln B/\partial Z$.

Способ косвенного измерения 2. Для компонентов кода *RGB*, измеряемых косвенным способом и вычисляемых в соответствии с системой (4), уравнения для вычисления относительных $\varepsilon_R^{xyY_{xy}}$, $\varepsilon_G^{xyY_{xy}}$, $\varepsilon_B^{xyY_{xy}}$ и абсолютных $\Delta R^{xyY_{xy}}$, $\Delta G^{xyY_{xy}}$, $\Delta B^{xyY_{xy}}$ инструментальных погрешностей косвенного измерения координат цвета основных цветов в цветовой модели *RGB* имеют следующий вид:

$$\varepsilon_R^{xyY_{xy}} = \frac{1}{\tilde{X}_r\dfrac{\overline{x}\overline{Y}_{xy}}{\overline{y}} + \tilde{X}_g\overline{Y}_{xy} + \tilde{X}_b\left(1-\overline{x}-\overline{y}\right)\dfrac{\overline{Y}_{xy}}{\overline{y}}}\left(\left(\left(\tilde{X}_r\frac{\overline{Y}_{xy}}{\overline{y}} - \tilde{X}_b\frac{\overline{Y}_{xy}}{\overline{y}}\right)\Delta x\right)^2 + \left(\left(\tilde{X}_r\frac{\overline{x}\overline{Y}_{xy}}{-\overline{y}^2} + \tilde{X}_b\left(1-\overline{x}\right)\frac{\overline{Y}_{xy}}{-\overline{y}^2}\right)\Delta y\right)^2 + \right.$$

$$\left. + \left(\left(\tilde{X}_r\frac{\overline{x}}{\overline{y}} + \tilde{X}_g + \tilde{X}_b\left(1-\overline{x}-\overline{y}\right)\frac{1}{\overline{y}}\right)\Delta Y_{xy}\right)^2\right)^{1/2} 100\%; \tag{9}$$

$$\varepsilon_G^{xyY_{xy}} = \frac{1}{\tilde{Y}_r \frac{\bar{x}\bar{Y}_{xy}}{\bar{y}} + \tilde{Y}_g \bar{Y}_{xy} + \tilde{Y}_b (1-\bar{x}-\bar{y}) \frac{\bar{Y}_{xy}}{\bar{y}}} \left(\left(\left(\tilde{Y}_r \frac{\bar{Y}_{xy}}{\bar{y}} - \tilde{Y}_b \frac{\bar{Y}_{xy}}{\bar{y}} \right) \Delta x \right)^2 + \left(\left(\tilde{Y}_r \frac{\bar{x}\bar{Y}_{xy}}{-\bar{y}^2} + \tilde{Y}_b (1-\bar{x}) \frac{\bar{Y}_{xy}}{-\bar{y}^2} \right) \Delta y \right)^2 + \right.$$

$$\left. + \left(\left(\tilde{Y}_r \frac{\bar{x}}{\bar{y}} + \tilde{Y}_g + \tilde{Y}_b (1-\bar{x}-\bar{y}) \frac{1}{\bar{y}} \right) \Delta Y_{xy} \right)^2 \right)^{1/2} 100\%, \quad (10)$$

$$\varepsilon_B^{xyY_{xy}} = \frac{1}{\tilde{Z}_r \frac{\bar{x}\bar{Y}_{xy}}{\bar{y}} + \tilde{Z}_g \bar{Y}_{xy} + \tilde{Z}_b (1-\bar{x}-\bar{y}) \frac{\bar{Y}_{xy}}{\bar{y}}} \left(\left(\left(\tilde{Z}_r \frac{\bar{Y}_{xy}}{\bar{y}} - \tilde{Z}_b \frac{\bar{Y}_{xy}}{\bar{y}} \right) \Delta x \right)^2 + \left(\left(\tilde{Z}_r \frac{\bar{x}\bar{Y}_{xy}}{-\bar{y}^2} + \tilde{Z}_b (1-\bar{x}) \frac{\bar{Y}_{xy}}{-\bar{y}^2} \right) \Delta y \right)^2 + \right.$$

$$\left. + \left(\left(\tilde{Z}_r \frac{\bar{x}}{\bar{y}} + \tilde{Z}_g + \tilde{Z}_b (1-\bar{x}-\bar{y}) \frac{1}{\bar{y}} \right) \Delta Y_{xy} \right)^2 \right)^{1/2} 100\%, \quad (11)$$

$$\begin{cases} \Delta R^{xyY_{xy}} = \dfrac{\varepsilon_R^{xyY_{xy}} \left(\tilde{X}_r \dfrac{\bar{x}\bar{Y}_{xy}}{\bar{y}} + \tilde{X}_g \bar{Y}_{xy} + \tilde{X}_b (1-\bar{x}-\bar{y}) \dfrac{\bar{Y}_{xy}}{\bar{y}} \right)}{100\%} \\ \Delta G^{xyY_{xy}} = \dfrac{\varepsilon_G^{xyY_{xy}} \left(\tilde{Y}_r \dfrac{\bar{x}\bar{Y}_{xy}}{\bar{y}} + \tilde{Y}_g \bar{Y}_{xy} + \tilde{Y}_b (1-\bar{x}-\bar{y}) \dfrac{\bar{Y}_{xy}}{\bar{y}} \right)}{100\%} \\ \Delta B^{xyY_{xy}} = \dfrac{\varepsilon_B^{xyY_{xy}} \left(\tilde{Z}_r \dfrac{\bar{x}\bar{Y}_{xy}}{\bar{y}} + \tilde{Z}_g \bar{Y}_{xy} + \tilde{Z}_b (1-\bar{x}-\bar{y}) \dfrac{\bar{Y}_{xy}}{\bar{y}} \right)}{100\%} \end{cases}, \quad (12)$$

где Δx, Δy, ΔY_{xy} — абсолютные погрешности (паспортные данные измерителя) прямых измерений координат цветности (x, y) и компонента относительной яркости Y_{xy}; $\bar{x}$, $\bar{y}$, $\bar{Y}_{xy}$ — средние значения измеренных параметров.

Анализ паспортных данных существующих измерителей показывает, что $\Delta x \le \Delta y$, $0{,}0001 \leqslant \Delta x, \Delta y \leqslant 0{,}01$. В таблице приведены сравнительные значения паспортных данных абсолютных погрешностей измерений (x, y)-координат цветности для колориметров (спектроколориметров, спектрорадиометров) различных фирм-изготовителей [7—9].

При выводе уравнений (9)—(12) в соответствии с (6) использовано правило логарифмического дифференцирования для вычисления частных производных $\partial \ln R/\partial x$, $\partial \ln R/\partial y$, $\partial \ln R/\partial Y_{xy}$, $\partial \ln G/\partial x$, $\partial \ln G/\partial y$, $\partial \ln G/\partial Y_{xy}$, $\partial \ln B/\partial x$, $\partial \ln B/\partial y$, $\partial \ln B/\partial Y_{xy}$.

4. Исследование свойств абсолютных инструментальных погрешностей косвенного измерения

Для исследования свойств абсолютных инструментальных погрешностей косвенного измерения координат цвета в модели *RGB* была разработана специализированная компьютерная программа. Среда проектирования программы — MathCAD 15.0. Код программы содержит формульные зависимости (1),

Предел допустимой абсолютной погрешности измерения, ед.

Координата	Изготовитель измерительного прибора					
	Carry	N-Vision	ООО "НТП "ТКА"	Hunter Associates Laboratory	АНО НТЦСЭ "ИСЭП"	ФГУП "ВНИИОФИ"
x	0,003	0,007	0,005	0,005	0,006	0,0002
y	0,003	0,007	0,005	0,005	0,007	0,0004

(2), (5), (7)—(12). Основное тело программы представляет собой тройной цикл по значениям параметров кода *RGB*. Код каждого компонента основного цвета *R*, *G*, *B* изменяется в диапазоне [1...255] с шагом 1 бит, таким образом, что общее число итераций тройного цикла составляет 2^{24}—3, исключаются нулевые начальные состояния кодов основных цветов.

В теле цикла осуществляется расчет по формулам (2), (3), (5) координат цвета *X*, *Y*, *Z*, координат цветности (*x*, *y*) и значения относительной яркости Y_{xy}. Значения констант матрицы преобразования МКО приняты следующими: $X_r = 0{,}478$; $X_g = 0{,}299$; $X_b = 0{,}175$; $Y_r = 0{,}263$, $Y_g = 0{,}650$; $Y_b = 0{,}081$; $Z_r = 0{,}020$; $Z_g = 0{,}160$; $Z_b = 0{,}908$. Значения абсолютных погрешностей измерений Δx, Δy, ΔY_{xy}, ΔX, ΔY, ΔZ изменяются в программе в следующих пределах: $0{,}0001 \leqslant \Delta x$, $\Delta y \leqslant 0{,}01$, $0{,}01 \leqslant \Delta X$, ΔY, ΔZ, $Y_{xy} \leqslant 1$, что соответствует техническим характеристикам современной и перспективной измерительной аппаратуры, используемой в колориметрии [9, 10].

Для расчета погрешностей косвенного измерения координат цвета цветовой модели *RGB* в выражениях (7)—(12) в качестве средних значений результатов измерений использовались истинные значения параметров, получаемые на основании вычислений по формулам (1), (2), (5). Таким образом, в программе оценивались только инструментальные составляющие погрешности косвенного измерения для таких условий измерений, при которых в процессе измерений отсутствуют систематическая и случайная составляющие погрешности. Природа возникновения и способы снижения систематической и случайной составляющих погрешности в колориметрических измерениях рассмотрены в работе [9].

Опытным путем замечено, что значения (*x*, *y*)-координат цветности на графике цветностей *XY*-плоскости (рис. 2, см. третью сторону обложки) в пределах треугольника цветового охвата распределены неравномерно. В окрестностях (*x*, *y*)-координат вершин треугольника цветового охвата, в точках белого, желтого, голубого и пурпурного цветов наблюдается повышенная плотность сосредоточения (*x*, *y*)-координат, вследствие чего абсолютная погрешность косвенного измерения кодов *RGB* в каждой точке *XY*-плоскости оказывается непостоянной. На рис. 2 выделены семь областей, вписанных в треугольник цветового охвата, в пределах которых шаг изменения значений (*x*, *y*)-координат цветности при изменении кода *RGB* на единицу не превышает значения 0,0001 ед.

Координаты цветности вершин треугольника цветового охвата на *XY*-плоскости определяются из соотношений, зависящих от коэффициентов матрицы МКО прямого преобразования (1), следующим образом:

$$R\text{: } \{x_R = X_r/(X_r + Y_r + Z_r),\ y_R = Y_r/(X_r + Y_r + Z_r)\},$$
$$G\text{: } \{x_G = X_g/(X_g + Y_g + Z_g),\ y_G = Y_g/(X_g + Y_g + Z_g)\},$$
$$B\text{: } \{x_B = X_b/(X_b + Y_b + Z_b),\ y_B = Y_b/(X_b + Y_b + Z_b)\}.$$

Гистограммы $p\left(\Delta R^{xyY_{xy}}\right)$, $p\left(\Delta G^{xyY_{xy}}\right)$, $p\left(\Delta B^{xyY_{xy}}\right)$ распределения оценок абсолютных инструментальных погрешностей косвенного измерения кодов *RGB* для основных цветов приведены на рис. 3 (см. третью сторону обложки).

При вычислении значений погрешностей косвенных измерений компонентов кода *RGB* в тело тройного цикла программы введены временные переменные, в которых сохраняются результаты оценки максимумов значений абсолютных погрешностей, раздельно для каждого компонента кода цвета *R*, *G*, *B* $\left(\max\left\{\varepsilon_R^{xyY_{xy}}\right\}, \max\left\{\varepsilon_G^{xyY_{xy}}\right\}, \max\left\{\varepsilon_B^{xyY_{xy}}\right\},\ R,\ G,\ B \in [1, \ldots, 255]\right)$ при фиксированных параметрах погрешностей измерительных приборов. Результаты вычислений приведены на рис. 4—6 (см. четвертую сторону обложки) для первого и второго способов косвенного измерения. На всех рисунках принадлежность кривой к оцениваемому компоненту *RGB* выполнена в цвете: *R* — красный цвет, *G* — зеленый цвет, *B* — синий цвет кривой.

Вычисления проводились на персональном компьютере ASUS K56CB-X0391H со следующими характеристиками: процессор Intel(R) Core(TM) i5-3337U, 4 ядра, тактовая частота 1,8 ГГц, оперативная память 6 Гбайт под управлением операционной системы Windows 8.1. Время вычисления одного значения максимума (одной точки) абсолютной погрешности косвенного измерения кода одного цвета $\Delta R^{xyY_{xy}}$ (или $\Delta G^{xyY_{xy}}$, или $\Delta B^{xyY_{xy}}$) в эксперименте с заданными приборными погрешностями Δx, Δy, ΔY_{xy} составило 1 ч 32 мин. Время выполнения расчетов и построения программой гистограмм (см. рис. 3, третья сторона обложки) составило около 9 ч.

Заключение

В результате проведенного исследования были получены в аналитическом виде выражения для оценки относительной и абсолютной инструментальных погрешностей косвенного измерения кодов цветовой модели *RGB* двумя способами.

Как следует из анализа уравнений (7)—(12), погрешности косвенного измерения зависят не только от абсолютных погрешностей измерительных приборов,

но и от положения координат цветности измеряемого цвета на *XY*-плоскости и от матрицы преобразования МКО, заданной компонентами: X_r, X_g, X_b, Y_r, Y_g, Y_b, Z_r, Z_g, Z_b. Это означает, что в экспериментах по измерению косвенным способом координат цвета в модели *RGB* для ЖК матриц, выполненных по различным стандартам МКО или имеющих различные системы баланса белого цвета, например, D-75, D-65, D-55, D-50 и др., погрешности измерений координат цвета *RGB* в точках с равными (x, y)-координатами будут различными.

Анализ полученных данных показывает, что максимум абсолютной инструментальной погрешности косвенного измерения приходится на точки с максимальными значениями кодов *RGB*, в которых значения кодов одного или нескольких цветов равны 255 (используется 8-битная модель кодирования основных цветов). Такими точками являются точки красного, зеленого и синего цветов, точка белого цвета, точки желтого (красный с зеленым), голубого (зеленый с синим) и пурпурного (красный с синим) цветов.

Интересным фактом является то, что точки максимума относительной и абсолютной погрешности косвенного измерения не совпадают. Максимум относительной погрешности косвенного измерения приходится на точки минимума значения кодов *RGB*. Точки максимума относительной погрешности приходятся на следующие значения кодов *RGB*: (1, 255, 255) при измерении кода красного цвета; (255, 1, 255) при измерении кода зеленого цвета; (255, 255, 1) при измерении кода синего цвета. Значения относительной погрешности косвенных измерений кода цвета в модели *RGB* изменяются в пределах 12...183 %.

Анализ результатов исследования также показывает, что для обеспечения значений абсолютной инструментальной погрешности косвенных измерений кода цвета в модели *RGB* на приемлемом разработчикам авионики уровне [4], т. е. $\Delta R < 1$, $\Delta G < 1$, $\Delta B < 1$, погрешности измерительного прибора должны находиться в следующих пределах: Δx, $\Delta y \leqslant 0{,}0001$, $\Delta Y_{xy} \leqslant 0{,}01$, что превосходит рекомендуемые сегодня требования ГОСТ Р 52870—2007. При малых значениях абсолютных погрешностей измерительного прибора Δx, Δy наибольший вклад в погрешность косвенного измерения кода *RGB* вносит погрешность ΔY_{xy}. Начиная со значений Δx, $\Delta y \geqslant 0{,}001$, значения $\max\left\{\varepsilon_R^{xy\,Y_{xy}}\right\}$, $\max\left\{\varepsilon_G^{xy\,Y_{xy}}\right\}$, $\max\left\{\varepsilon_B^{xy\,Y_{xy}}\right\}$ абсолютной инструментальной погрешности косвенного измерения существенно возрастают.

Сравнение рис. 4 и 6 показывает, что наибольшую точность косвенного измерения обеспечивает способ 1, в котором абсолютная инструментальная погрешность измерения компонентов кода *RGB* не превышает 3 ед. в актуальном диапазоне изменения значений абсолютных приборных погрешностей прямых измерений компонентов *XYZ*.

Результаты расчетов и моделирования показывают, что для рекомендуемых ГОСТ Р 52870—2007 значений абсолютной погрешности измерительного прибора на уровне $0{,}07 \leqslant \Delta Y_{xy} \leqslant 0{,}12$, Δx, $\Delta y \leqslant 0{,}001$, абсолютная погрешность косвенного измерения компонентов основных цветов кода *RGB* составит

- для способа 1 косвенного измерения компонентов кода *RGB*:
  - $\Delta R = 0{,}2$, $\Delta G = 0{,}17$, $\Delta B = 0{,}08$ при $\Delta X = \Delta Y = \Delta Z = 0{,}07$;
  - $\Delta R = 0{,}27$, $\Delta G = 0{,}22$, $\Delta B = 0{,}11$ при $\Delta X = \Delta Y = \Delta Z = 0{,}09$;
  - $\Delta R = 0{,}36$, $\Delta G = 0{,}29$, $\Delta B = 0{,}15$ при $\Delta X = \Delta Y = \Delta Z = 0{,}12$;
- для способа 2 косвенного измерения компонентов кода *RGB*:
  - $\Delta R \leqslant 2{,}8$, $\Delta G \leqslant 1{,}2$, $\Delta B \leqslant 4{,}6$ при $\Delta Y_{xy} \leqslant 0{,}075$, $\Delta x = \Delta y = 0{,}001$;
  - $\Delta R \leqslant 2{,}8$, $\Delta G \leqslant 1{,}2$, $\Delta B \leqslant 5$ при $\Delta Y_{xy} \leqslant 0{,}1$, $\Delta x = \Delta y = 0{,}001$;
  - $\Delta R \leqslant 2{,}8$, $\Delta G \leqslant 1{,}2$, $\Delta B \leqslant 5{,}5$ при $\Delta Y_{xy} \leqslant 0{,}125$, $\Delta x = \Delta y = 0{,}001$.

Список литературы

1. **Булгаков Д. Н.** Теоретические основы расчетно-экспериментальной оценки эргономического качества системы отображения информации в следящей системе // Моделирование и анализ данных. 2013. № 1. С. 88—96.

2. **Игнатова О.** Оценка характеристик систем отображения. Методы центра управления полетами // Электроника: Наука, Технология, Бизнес. 2012. № 6. С. 102—107.

3. **Костишин М. О., Жаринов И. О., Жаринов О. О.** Исследование визуальных характеристик средств отображения пилотажно-навигационных параметров и геоинформационных данных в авионике // Информационно-управляющие системы. 2014. № 4. С. 61—67.

4. **Жаринов И. О., Жаринов О. О.** Исследование распределения оценки разрешающей способности преобразования Грассмана в системах кодирования цвета, применяемых в авионике // Программная инженерия. 2014. № 8. С. 40—47.

5. **Жаринов И. О., Жаринов О. О.** Бортовые средства отображения информации на плоских жидкокристаллических панелях: учеб. пособие. Информационно-управляющие системы. СПб.: ГУАП, 2005. 144 с.

6. **Хорунжий М. Д.** Метод количественной оценки цветовых различий при восприятии цифровых изображений // Научно-технический вестник информационных технологий, механики и оптики. 2008. № 1. С. 136—144.

7. **Бушинский В. О., Воронов С. А., Панкратов В. И., Родионов В. Н.** Оценка погрешностей измерений спектроколориметра с интерференционными светофильтрами // Электроника и связь. 2012. № 3. С. 34—39.

8. **Михайлов О. М., Заргарьянц Г. С.** Интегральный дистанционный колориметр на основе трехцветной колориметрической системы КЗФ // Светотехника. 2008. № 3. С. 19—25.

9. **Стороженко А. И.** Оценка погрешностей визуальных и фотоэлектрических методов измерения координат цветности: автореф. дисс. ... канд. техн. наук. СПб., 2007. 21 с.

10. **Ложкин Л. Д.** Дифференциальная колориметрия в телевидении: автореф. дисс... д-ра техн. наук. СПб., 2014. 31 с.

I. O. Zharinov, Associate Professor, Chief of Department, Saint Petersburg National Research University of Information Technologies, Mechanics and Optics (University ITMO), Chief of learning-scientists center, SPb Scientific Design Bureau "Electroavtomatica" n. a. P. A. Efimova, e-mail: igor_rabota@pisem.net,
O. O. Zharinov, Associate Professor, Saint-Petersburg State University of Aerospace Equipment

The Evaluation of Hardware-Caused Inaccuracy of Indirect Measurements of Chromaticity Coordinates in Color Model Data Used in Avionics

A problem of evaluation of hardware-caused inaccuracy of indirect measurements of color coordinates in RGB (R — Red, G — Green, B — Blue) model during experiments with LCD matrixes is considered. It is shown that inaccuracy value of indirect measurement depends upon hardware-caused errors of used measurement equipment, the standard of white balance that was accepted by developers of LCD matrix, and also upon values of measured color coordinates or chromaticity coordinates. Two methods of indirect measurements of RGB codes were considered, namely: the one that based on color coordinates measurements and the other one that based on chromaticity coordinates measurements with additional measurement of relative brightness value. Analythical expressions for evaluation of both absolute and relative components of hardware-caused inaccuracy of indirect measurements for two considered methods were obtained. Results of computing simulation are presented, such as set of graphs, representing maximal values of absolute inaccuracy of indirect measurement of components of RGB codes vs hardware measurement error values for two considered methods with technical performances of modern hardware and perspective as well.

Keywords: *indirect measurement, hardware-caused inaccuracy, color coordinates, chromaticity coordinates, RGB model, evaluation*

References

1. **Bulgakov D. N.** Teoreticheskie osnovy raschetno-jeksperimental'noj ocenki jergonomicheskogo kachestva sistemy otobrazhenija informacii v sledjashhej sisteme. *Modelirovanie i analiz dannyh*. 2013. N. 1. P. 88—96.

2. **Ignatova O.** Ocenka harakteristik sistem otobrazhenija. Metody centra upravlenija poletami. *Jelektronika: Nauka, Tehnologija, Biznes*. 2012. N. 6. P. 102—107.

3. **Kostishin M. O., Zharinov I. O., Zharinov O. O.** Issledovanie vizual'nyh harakteristik sredstv otobrazhenija pilotazhno-navigacionnyh parametrov i geoinformacionnyh dannyh v avionike. *Informacionno-upravljajushhie sistemy*. 2014. N. 4. P. 61—67.

4. **Zharinov I. O., Zharinov O. O.** Issledovanie raspredelenija ocenki razreshajushhej sposobnosti preobrazovanija Grassmana v sistemah kodirovanija cveta, primenjaemyh v avionike. *Programmnaja inzhenerija*. 2014. N. 8. P. 40—47.

5. **Zharinov I. O., Zharinov O. O.** *Bortovye sredstva otobrazhenija informacii na ploskih zhidkokristallicheskih paneljah:* ucheb. posobie. Informacionno-upravljajushhie sistemy. SPb.: GUAP, 2005. 144 p.

6. **Horunzhij M. D.** Metod kolichestvennoj ocenki cvetovyh razlichij pri vosprijatii cifrovyh izobrazhenij. *Nauchno-tehnicheskij vestnik informacionnyh tehnologij, mehaniki i optiki*. 2008. N. 1. P. 136—144.

7. **Bushinskij V. O., Voronov S. A., Pankratov V. I., Rodionov V. N.** Ocenka pogreshnostej izmerenij spektrokolorimetra s interferencionnymi svetofil'trami. *Jelektronika i svjaz'*. 2012. N. 3. P. 34—39.

8. **Mihajlov O. M., Zargar'janc G. S.** Integral'nyj distancionnyj kolorimetr na osnove trehcvetnoj kolorimetricheskoj sistemy KZF. *Svetotehnika*. 2008. N. 3. P. 19—25.

9. **Storozhenko A. I.** Ocenka pogreshnostej vizual'nyh i fotojelektricheskih metodov izmerenija koordinat cvetnosti: avtoref. diss. ... kand. tehn. nauk. SPb., 2007. 21 p.

10. **Lozhkin L. D.** Differencial'naja kolorimetrija v televidenii: avtoref. diss. ... d-ra tehn. nauk. SPb., 2014. 31 p.

Указатель статей, опубликованных в журнале "Программная инженерия" в 2014 г.

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4

Технический редактор *Е. М. Патрушева*. Корректор *Е. В. Комиссарова*

Сдано в набор 02.10.2014 г. Подписано в печать 17.11.2014 г. Формат 60×88 1/8. Заказ PI1214
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**