ISSN 1816-0301

ИНФОРМАТИКА

3(51)

ИЮЛЬ-СЕНТЯБРЬ
2016

Адрес редакции:
220012, Минск,
ул. Сурганова, 6, к. 305
тел. (017) 284-26-22
e-mail: rio@newman.bas-net.by
http://uiip.bas-net.by

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ
НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК БЕЛАРУСИ

ИНФОРМАТИКА

ЕЖЕКВАРТАЛЬНЫЙ НАУЧНЫЙ ЖУРНАЛ

Издается с января 2004 г.

№ 3(51) • июль-сентябрь 2016

СОДЕРЖАНИЕ

СТАТЬИ ПО МАТЕРИАЛАМ СЕДЬМОЙ МЕЖДУНАРОДНОЙ НАУЧНОЙ КОНФЕРЕНЦИИ «ТАНАЕВСКИЕ ЧТЕНИЯ»

Редактор Г.Б. Гончаренко
Корректор А.А. Михайлова
Компьютерная верстка О.Б. Бутевич

Сдано в набор 09.08.2016. Подписано в печать 05.09.2016.
Формат 60×84 1/8. Бумага офсетная. Гарнитура Таймс. Ризография.
Усл. печ. л. 14,2. Уч.-изд. л. 13,9. Тираж 60 экз. Заказ 5.

Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий № 1/274 от 04.04.2014.
ЛП № 02330/444 от 18.12.13.
Ул. Сурганова, 6, 220012, Минск.

THE UNITED INSTITUTE OF INFORMATICS PROBLEMS
OF THE NATIONAL ACADEMY OF SCIENCES OF BELARUS

INFORMATICS

PUBLISHED QUATERLY

Issued since 2004

№ 3(51) • Jule-September 2016

CONTENTS

ARTICLES ON THE MATERIALS OF SEVENTH INTERNATIONAL SCIENTIFIC CONFERENCE «TANAYEVSKY READING»

ОБРАБОТКА СИГНАЛОВ, ИЗОБРАЖЕНИЙ И ТЕКСТА

УДК 004.912

С.Ф. Липницкий

МОДЕЛИРОВАНИЕ СМЫСЛОВОГО СОДЕРЖАНИЯ ТЕКСТА НА ОСНОВЕ СЛИЯНИЯ КОММУНИКАТИВНЫХ ФРАГМЕНТОВ

Предлагается математическая модель процесса пересказа содержания тестовых сообщений на основе слияния коммуникативных фрагментов. Формально определяются понятия таких фрагментов, а также вербально-ассоциативных сетей в качестве моделей знаний о предметной области и пересказываемых текстах. Приводится описание алгоритма пересказа содержания текста.

Введение

Проблема пересказа содержания текстовых документов возникает при решении задач аналитико-синтетической обработки информации (например, при индексировании текстов, их аннотировании и реферировании). Традиционно синтез текста рассматривается как процесс последовательной генерации морфем, лексем, синтаксических фраз и, наконец, предложений, т. е. предполагается, что в иерархии этих языковых элементов каждый следующий складывается из предыдущих по известным синтаксическим правилам. Однако в монографии [1] показано, что основой использования языка человеком является его языковая память. Согласно этой концепции предложения при синтезе строятся из готовых хранящихся в памяти компонентов, названных коммуникативными фрагментами. Такие фрагменты не образуются по синтаксическим правилам, а извлекаются из памяти целиком.

Сложность моделирования и алгоритмизации процесса пересказа текста на основе слияния коммуникативных фрагментов заключается в их динамичности. Динамичность связана с изменчивостью границ фрагментов в предложениях, а также с лексическим и синтаксическим многообразием их представления.

В данной статье предлагается формальная модель, позволяющая алгоритмизировать процесс интерпретации содержания текстовых документов путем слияния коммуникативных фрагментов.

1. Формализация понятия коммуникативного фрагмента

В работе [1] под коммуникативными фрагментами понимаются «отрезки речи различной длины», которые человек использует при синтезе предложений. Эти отрезки «хранятся в памяти говорящего в качестве стационарных частиц его языкового опыта». С целью алгоритмизации синтеза текстов на основе слияния коммуникативных фрагментов, а также создания системы лингвистических словарей, используемых при синтезе, формализуем понятие коммуникативного фрагмента.

1.1. Определение коммуникативного фрагмента

Рассмотрим произвольное предложение $\pi = a_1\, a_2 \ldots a_n$ из тематического корпуса текстов Ct [2]. Подцепочку $f = a_1 a_2 \ldots a_m$ цепочки π назовем коммуникативным фрагментом, если значения информативности вербально-ассоциативной связи между словами этой подцепочки удовлетворяют следующим двум условиям:

– для любых индексов i, j, таких, что $i \geq 1$, $j \leq m$, $i < j$, для значений информативности $I_{Ct}^{a_i a_j}$ вербально-ассоциативной связи между словами a_i и a_j выполняется соотношение $I_{Ct}^{a_i a_j} \geq I_{Ct}^{00}$, где I_{Ct}^{00} – пороговое значение информативности;

– существует хотя бы одно слово $a_r \in \{a_1, a_2, ..., a_m\}$, такое, что справедливо соотношение $I_{Ct}^{a_r a_{m+1}} < I_{Ct}^{00}$.

Информативность I_{Ct}^{ab} вербально-ассоциативной связи между словами *a* и *b* в тематическом корпусе текстов *Ct* вычисляется по формуле

$$I_{Ct}^{ab} = \frac{n_{Ct}^{ab} + \sum\limits_{\substack{c \in Par_a,\, c \neq a \\ d \in Par_b,\, d \neq b}} n_{Ct}^{cd} + \sum\limits_{\substack{r \in Syn_a,\, r \neq a \\ s \in Syn_b,\, s \neq b}} (n_{Ct}^{rs} + \sum\limits_{\substack{p \in Par_r,\, p \neq r \\ q \in Par_s,\, q \neq s}} n_{Ct}^{pq})}{n_{Cf}^{ab} + \sum\limits_{\substack{c \in Par_a,\, c \neq a \\ d \in Par_b,\, d \neq b}} n_{Cf}^{cd} + \sum\limits_{\substack{r \in Syn_a,\, r \neq a \\ s \in Syn_b,\, s \neq b}} (n_{Cf}^{rs} + \sum\limits_{\substack{p \in Par_r,\, p \neq r \\ q \in Par_s,\, q \neq s}} n_{Cf}^{pq})}, \quad (1)$$

где n_{Ct}^{ab}, n_{Ct}^{cd}, n_{Ct}^{rs} и n_{Ct}^{pq} – абсолютные частоты совместной встречаемости слов *a* и *b*, *c* и *d*, *r* и *s*, а также *p* и *q* соответственно в одном и том же предложении тематического корпуса текстов Ct_i; n_{Cf}^{ab}, n_{Cf}^{cd}, n_{Cf}^{rs} и n_{Cf}^{pq} – абсолютные частоты их совместного появления в предложениях полного корпуса текстов *Cf* [2]; Par_a – множество всех словоизменений слова *a*; Syn_a – множество всех его синонимов. Для хранения указанных словоизменений и синонимов используются следующие лингвистические словари:

– словарь словоизменительных парадигм $Dic_{par} = \{(a, Par_a) \mid a \in W_{Cf}, a \in Par_a\}$, где Par_a – множество всех словоизменений слова *a*, W_{Cf} – множество всех словоформ из корпуса текстов *Cf*. В словаре Dic_{par} для каждой словоформы представлены все ее словоизменения;

– словарь синонимичных словоформ, состоящий из совокупностей синонимичных слов: $Dic_{syn}^{a} = \{(a, Syn_a) \mid a \in W_{Cf}, a \in Syn_a\}$, где Syn_a – множество всех синонимов слова *a*.

Оба словаря формируются «вручную» экспертом-лингвистом информационной системы.

Для хранения информации о частотах совместной встречаемости слов в одном и том же предложении полного корпуса текстов *Cf* и всех тематических корпусов будем использовать словарь вербально-ассоциативных пар слов $Dic_{ab} = \{\langle (a, b),\ n_{Cf}^{ab},\ n_{Ct_1}^{ab},\ n_{Ct_2}^{ab},\ \ldots,\ n_{Ct_n}^{ab} \rangle \mid a, b \in W_{Cf},\ n_{Cf}^{ab} \neq 0,\ n_{Ct_i}^{ab} \neq 0,\ i = \overline{1, n}\}$.

Словарь Dic_{ab} создается программно следующим образом. Формируется множество W_{Cf}^{ab} всех пар слов (*a*, *b*), таких, что слова *a* и *b* из каждой пары содержатся хотя бы в одном предложении полного корпуса текстов *Cf*. Для каждой пары слов вычисляется частота n_{Ct}^{ab} ее появления в корпусе *Cf*, а также определяются частоты $n_{Ct_i}^{ab}$ ($i = \overline{1, n}$) появления пары слов (*a*, *b*) в тематических корпусах текстов Ct_i. На основе этих данных формируются кортежи вида $\langle (a, b), n_{Cf}^{ab}, n_{Ct_1}^{ab}, n_{Ct_2}^{ab}, \ldots, n_{Ct_n}^{ab} \rangle$, являющиеся компонентами создаваемого словаря Dic_{ab}.

Обозначим через *F* множество всех коммуникативных фрагментов в полном корпусе текстов *Cf*. Рассмотрим совокупность коммуникативных фрагментов $Dic_f = \{\langle f, I_{Ct_1}^{f}, I_{Ct_2}^{f}, \ldots, I_{Ct_n}^{f} \rangle \mid f \in F\}$, где $I_{Ct_i}^{f}$ ($i = \overline{1, n}$) – значение информативности коммуникативного фрагмента *f* в тематическом корпусе текстов Ct_i:

$$I_{Ct_i}^{f} = \frac{\sum\limits_{a \in f} I_{Ct_i}^{a}}{\sqrt{\sum\limits_{a \in f} (I_{Ct_i}^{a})^2}}. \quad (2)$$

Множество Dic_f – это словарь коммуникативных фрагментов. В формуле (2) информативность $I_{Ct_i}^{a}$ слова *a* в тематическом корпусе текстов Ct_i вычисляется по формуле, аналогичной выражению (1) из статьи [2]:

$$I^a_{Ct_i} = \frac{n^a_{Ct_i} + \sum\limits_{b\in Par_a,\, b\neq a} n^b_{Ct_i} + \sum\limits_{c\in Syn_a,\, c\neq a} (n^c_{Ct_i} + \sum\limits_{d\in Par_c,\, d\neq c} n^d_{Ct_i})}{n^a_{Cf} + \sum\limits_{b\in Par_a,\, b\neq a} n^b_{Cf} + \sum\limits_{c\in Syn_a,\, c\neq a} (n^c_{Cf} + \sum\limits_{d\in Par_c,\, d\neq c} n^d_{Cf})}. \qquad (3)$$

Для вычисления значений информативности $I^a_{Ct_i}$ используется словарь словоформ $Dic_a = \{\langle a,\ n^a_{Cf}\ ,\ n^a_{Ct_1}\ ,\ n^a_{Ct_2}\ ,\ \ldots\ ,\ n^a_{Ct_n}\rangle |\ a \in W_{Cf}\}$, где n^a_{Cf} и $n^a_{Ct_i}$ – абсолютные частоты появления словоформы *a* соответственно в полном и *i*-м тематическом корпусах текстов.

Формируется словарь Dic_a программно аналогично словарю Dic_{ab} путем последовательного вычисления частот n^a_{Cf} и $n^a_{Ct_i}$ ($i = \overline{1, n}$) появления словоформ в полном и тематических корпусах текстов.

Рассмотрим процесс создания словаря Dic_f коммуникативных фрагментов. Он формируется в два этапа. На первом этапе каждое предложение $\pi = a_1 a_2 \ldots a_l$ ($l \geq 2$) входного текста разбивается на коммуникативные фрагменты в полном соответствии с их формальным определением. Вначале формируется множество всех пар слов вида (a_r, a_j) из совокупности слов $\{a_1, a_2, \ldots, a_j\}$, причем в каждой паре $r < j$. Для каждой такой пары проверяется справедливость неравенства $I^{a_r a_j}_{Ct} < I^{00}_{Ct}$, где I^{00}_{Ct} – пороговое значение информативности. Если неравенство выполняется при некотором значении j, то согласно упомянутому выше определению получаем коммуникативный фрагмент. Далее процедура повторяется аналогичным образом. В результате работы алгоритма получаем предложение в виде цепочки коммуникативных фрагментов $\pi = f_1 f_2 \ldots$.

На втором этапе формируется список лексикографически упорядоченных коммуникативных фрагментов и для каждого фрагмента вычисляется множество значений информативности $\{ I^f_{Ct_1}\ ,\ I^f_{Ct_2}\ ,\ \ldots\ ,\ I^f_{Ct_n}\ \}$ по формулам (2) и (3).

1.2. Отношение контаминации

Как отмечается в работе [1], различные коммуникативные фрагменты *контаминируются*, т. е. объединяются, перемещаются друг в друга, образуя новые языковые единицы. Для формализации понятия контаминации введем в рассмотрение следующее отношение.

Определим на множестве *F* строгий порядок (транзитивное и антирефлексивное отношение) $\prec_F$, такой, что для любых коммуникативных фрагментов (непустых цепочек) $f_1, f_2 \in F$ соотношение $f_1 \prec_F f_2$ выполняется тогда и только тогда, когда фрагмент f_1 получается из фрагмента f_2 путем исключения из него одного или более слов. Строгий порядок $\prec_F$ назовем *отношением контаминации* на множестве *F*.

Например, пусть f_1 – это коммуникативный фрагмент «национальный университет», а f_2 – фрагмент «национальный технический университет». В данном случае фрагмент f_1 получен из фрагмента f_2 исключением из него слова «технический».

1.3. Классы контаминированных коммуникативных фрагментов

Пусть $\{ K^F_i \mid i = \overline{1, l}\ \}$ – множество подмножеств множества *F*, такое, что $F = \bigcup\limits_{i=1}^{l} K^F_i$. Множества K^F_i будем называть классами контаминированных коммуникативных фрагментов, если все они (т. е. при любом $i = \overline{1, l}$) являются максимально совершенными (по терминологии из монографии [3], определение 4.4). Это означает, что при синтезе пересказа содержания текста возможна взаимозаменяемость коммуникативных фрагментов внутри одного класса. Более того, согласно теореме 4.3 Хаусдорфа [3] для любого коммуникативного фрагмента из множества *F* существует свой класс, элементом которого данный фрагмент является.

2. Моделирование знаний о предметной области

Эффективность систем аналитико-синтетической обработки текстовой информации существенным образом зависит от их интеллектуальности, т. е. способности работать не только с данными, но и знаниями об объектах и явлениях предметной области. При автоматизации процесса пересказа текста необходимые знания накапливаются в базе знаний о предметной области. Построим модель представления таких знаний.

2.1. Вербально-ассоциативное отношение коммуникативных фрагментов предметной области

Обозначим через *Ft* множество всех коммуникативных фрагментов произвольного тематического корпуса текстов $Ct \in Cf$. Определим на множестве *Ft* отношение толерантности Δ (рефлексивное и симметричное бинарное отношение), такое, что пара (f, g) любых фрагментов из множества *Ft* является элементом отношения Δ, т. е. $(f, g) \in \Delta$ тогда и только тогда, когда фрагменты *f* и *g* из этой пары содержатся хотя бы в одном предложении корпуса *Ct*. Отношение Δ будем называть *вербально-ассоциативным отношением коммуникативных фрагментов предметной области*, определяемой тематическим корпусом текстов *Ct*.

Вербально-ассоциативное отношение Δ – это отношение вербально-ассоциативной связи коммуникативных фрагментов в тематическом корпусе текстов *Ct*.

Пусть *f* и *g* – произвольные коммуникативные фрагменты предметной области, определяемой тематическим корпусом текстов *Ct*. Для вычисления силы вербально-ассоциативной связи между фрагментами *f* и *g* будем использовать следующую формулу из статьи [4]:

$$I_{Ct}^{fg} = \frac{\sum\limits_{a \in f,\, b \in g} I_{Ct}^{ab}}{\sqrt{\sum\limits_{a \in f,\, b \in g} (I_{Ct}^{ab})^2}}. \qquad (4)$$

В выражении (5) информативность I_{Ct}^{ab} вычисляется по формуле (1).

2.2. Дискурсивная сочетаемость коммуникативных фрагментов

Текст как связную последовательность предложений, обладающую семантическим единством, в лингвистике отождествляют с понятием дискурса [5]. Для получения «хороших» предложений при их синтезе из коммуникативных фрагментов будем использовать отношение дискурсивной сочетаемости таких фрагментов. Понятие этого отношения введем следующим образом.

Определим на множестве *Ft* всех коммуникативных фрагментов в тематическом корпусе текстов *Ct* антирефлексивное бинарное отношение Λ, такое, что для любых фрагментов $f, g \in Ft$ соотношение $(f, g) \in \Lambda$ выполняется тогда и только тогда, когда в некотором тексте $T \in Ct$ существует предложение π, в котором коммуникативный фрагмент *f* непосредственно предшествует фрагменту *g*. Отношение Λ будем называть *отношением дискурсивной сочетаемости коммуникативных фрагментов* в тематическом корпусе текстов *Ct*.

Дискурсивно-сочетаемыми являются, например, следующие пары коммуникативных фрагментов: «используется при» и «капитальном строительстве», «отложен ежегодный визит» и «председателя объединения».

Упорядоченные пары $(f, g) \in \Lambda$ будем хранить в специальном списке – словаре дискурсивно-сочетаемых коммуникативных фрагментов:

$$Dic_{fg} = \{(f, g) \mid f \in Ft, g \in Ft, (f, g) \in \Lambda\}. \qquad (5)$$

Основой для формирования словаря Dic_{fg} являются несовпадающие предложения полного корпуса текстов, представленные в виде цепочек, которые состоят из коммуникативных фрагментов.

2.3. Вербально-ассоциативная сеть предметной области

Рассмотрим граф вербально-ассоциативного отношения Δ. Пометим каждую вершину *f* этого графа значением информативности I_{Ct}^{f} коммуникативного фрагмента (с учетом синонимии и словоизменения), а каждое ребро (*f, g*) – значением информативности I_{Ct}^{fg} вербально-ассоциативной связи фрагментов *f* и *g* (также учитывая синонимию и словоизменения). Пусть (*f, g*) – произвольное ребро этого графа. Если (*f, g*) ∈ Λ, то для всех таких пар (*f, g*) вершины *f* и *g* соединим дугой, направленной от *f* к *g*. Обозначим полученный смешанный граф через Net_{Ct}.

Граф Net_{Ct} назовем *вербально-ассоциативной сетью предметной области*, определяемой тематическим корпусом текстов *Ct*.

Сеть Net_{Ct} является моделью поискового образа тематического корпуса текстов *Ct*. Построим этот поисковый образ в виде множества коммуникативных фрагментов, вербально-ассоциативных пар таких фрагментов, в которой фрагментам приписаны значения их информативности, а парам – значения информативности вербально-ассоциативных связей между их фрагментами:

$$\text{ПО}_{Ct} = \{(f, I_{Ct}^{f}), \ldots, ((g,h), I_{Ct}^{gh}), \ldots, ((\overrightarrow{p,q}), I_{Ct}^{pq}), \ldots \mid I_{Ct}^{f} > I_{Ct}^{0}; I_{Ct}^{gh}, I_{Ct}^{pq} > I_{Ct}^{00}\}, \quad (6)$$

где I_{Ct}^{0}, I_{Ct}^{00} – пороговые значения информативности коммуникативного фрагмента и вербально-ассоциативной связи этих фрагментов соответственно, стрелка над парой фрагментов (*p, q*) означает, что (*p, q*) ∈ Λ.

2.4. Базовые и связующие коммуникативные фрагменты

В зависимости от информативности будем различать базовые и связующие коммуникативные фрагменты предметной области, определяемой тематическим корпусом текстов *Ct*. Обозначим через I_{Ct}^{0} пороговое значение информативности коммуникативного фрагмента. Тогда коммуникативный фрагмент *f* будем называть *базовым*, если значение его информативности I_{Ct}^{f} удовлетворяет неравенству $I_{Ct}^{f} \geq I_{Ct}^{0}$. Если же $I_{Ct}^{f} < I_{Ct}^{0}$, то фрагмент *f* назовем *связующим*. Связующим, например, является коммуникативный фрагмент «предлагается новый подход к решению проблемы», а базовым – фрагмент «принятия решений в условиях неопределенности».

Обозначим через $Ft_{\text{баз.}}$ множество всех базовых коммуникативных фрагментов, а через $Ft_{\text{св.}}$ – множество всех связующих. Тогда множество всех коммуникативных фрагментов предметной области – это объединение множеств базовых и связующих фрагментов, т. е. $Ft = Ft_{\text{баз.}} \cup Ft_{\text{св.}}$.

3. Моделирование знаний о пересказываемом тексте

Основой для моделирования знаний о пересказываемом тексте является вербально-ассоциативная сеть предметной области, тематический корпус которой релевантен этому тексту. С использованием сети предметной области строится вербально-ассоциативная сеть пересказываемого текста.

3.1. Вербально-ассоциативная сеть пересказываемого текста

Пусть *Q* ∈ *Cf* – некоторый текст, а *Ct* – релевантный ему тематический корпус текстов. Вычислим информативность каждого предложения π текста *Q* по формуле, являющейся аналогом выражения (1):

$$I_{Q}^{\pi} = I_{Ct}^{\pi} = \frac{\sum\limits_{a \in \pi} I_{Ct}^{a}}{\sqrt{\sum\limits_{a \in \pi} (I_{Ct}^{a})^{2}}}. \quad (7)$$

Обозначим через I_Q^0 некоторое пороговое значение информативности предложений текста Q. Если $I_Q^\pi \geq I_Q^0$, то предложение π будем считать информативным. Исключив из текста все неинформативные предложения, т. е. такие, для которых $I_Q^\pi < I_Q^0$, получим кортеж информативных предложений $T = \langle \pi_1, \pi_2, \ldots, \pi_m \rangle$. Каждое предложение π_i ($i = \overline{1, m}$) представим в виде цепочки коммуникативных фрагментов $\pi_i = f_1 f_2 \ldots$. Обозначим через Δ_T сужение вербально-ассоциативного отношения Δ на множество F_T всех коммуникативных фрагментов кортежа предложений T, т. е. $\Delta_T = \Delta \cap (F_T \times F_T)$, а через Λ_T – сужение отношения дискурсивной сочетаемости коммуникативных фрагментов Λ на это же множество. Тогда вербально-ассоциативную сеть пересказываемого текста Q построим следующим образом.

Рассмотрим вербально-ассоциативную сеть предметной области *Net*. Исключим из сети *Net* все ребра (f, g), такие, что $(f, g) \notin \Delta_T$, и все дуги (r, s), для которых $(r, s) \notin \Lambda_T$. Исключим также из сети *Net* инцидентные исключенным ребрам и дугам вершины. Полученный граф назовем *вербально-ассоциативной сетью пересказываемого текста Q*.

3.2. Поиск релевантного тематического корпуса текстов

Текст $Q \in Cf$ – это запрос на поиск релевантного ему тематического корпуса текстов. Исключим из всех поисковых образов тематических корпусов текстов значения информативности коммуникативных фрагментов и вербально-ассоциативных пар таких фрагментов, т. е. преобразуем выражение (6) к виду

$$\text{ПО}_{Ct_i} = \{f, \ldots, (g, h), \ldots, (\overrightarrow{p, q}), \ldots \mid f, g, h, p, q \in Ct_i\}. \quad (8)$$

Аналогично представим поисковое предписание, т. е. поисковый образ текста Q:

$$\text{ПП}_Q = \{f, \ldots, (g, h), \ldots, (\overrightarrow{p, q}), \ldots \mid f, g, h, p, q \in Q\}. \quad (9)$$

Реализуем процесс поиска тематического корпуса текстов, используя в качестве критерия выдачи косинус угла между векторами поискового предписания и поискового образа документа в евклидовом пространстве E. Этот критерий применяется в большинстве известных информационных систем.

Обозначим через W множество всех различных коммуникативных фрагментов, вербально-ассоциативных пар фрагментов и упорядоченных пар фрагментов вида $(\overrightarrow{p, q})$, входящих в поисковые образы всех тематических корпусов текстов. Пусть их количество равно n. Лексикографически упорядочим все элементы множества W, т. е. представим W в виде кортежа $W = \langle w_1, w_2, \ldots, w_n \rangle$. Для каждого тематического корпуса текстов $Ct \in Cf$ построим вектор его поискового образа в пространстве E: $\mathbf{F}_{Ct} = (p_1, p_2, \ldots, p_n)$, где $p_i = 1$, если элемент w_i входит в этот поисковый образ, в противном случае $p_i = 0$. Аналогично представим вектор поискового предписания, построенного для запроса Q: $\mathbf{F}_Q = (q_1, q_2, \ldots, q_n)$. Тогда для вычисления меры близости между векторами $\mathbf{F}_{Ct}$ и $\mathbf{F}_Q$ воспользуемся критерием выдачи

$$\cos\varphi = \frac{\mathbf{F}_{Ct}\mathbf{F}_Q}{|\mathbf{F}_{Ct}||\mathbf{F}_Q|} = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^{\,2}}\sqrt{\sum_{i=1}^{n} q_i^{\,2}}}. \quad (10)$$

При реализации информационной системы критерий (10) целесообразно преобразовать следующим образом. Пусть l – количество совпавших элементов поискового образа ПО_{Ct} и поискового предписания ПП_Q, n_{Ct} – количество элементов в множестве ПО_{Ct}, а n_Q – их количество в множестве ПП_Q. Тогда критерий (10) примет вид

$$\cos\varphi = \frac{l}{\sqrt{n_{Ct} n_Q}}. \qquad (11)$$

Результатом поиска будет тематический корпус текстов *Ct*, для которого значение $\cos\varphi$ является наибольшим из всех значений, таких, что $\cos\varphi \geq \eta_0$.

3.3. Фрагментно-слотовый шаблон предложения

Пусть имеется предложение $\pi = f_1 f_2 \ldots f_l$. Цепочку, полученную из предложения π заменой его базовых коммуникативных фрагментов слотами («пустыми» фрагментами), будем называть *фрагментно-слотовым шаблоном предложения* π.

Фрагментно-слотовые шаблоны предложений создаются в автоматизированном режиме: сначала на основе специально подготовленных текстов программно формируется совокупность фрагментно-слотовых шаблонов, а затем они корректируются экспертом-лингвистом информационной системы. При синтезе предложения слоты заменяются коммуникативными фрагментами. При этом может учитываться отношение контаминации.

3.4. Фрагментно-слотовый шаблон текста

Пусть имеется текст в виде кортежа предложений $T = \langle \pi_1, \pi_2, \ldots, \pi_m \rangle$, такой, что каждому предложению π_i ($i = \overline{1, m}$) соответствует его фрагментно-слотовый шаблон H_i. Рассмотрим кортеж фрагментно-слотовых шаблонов предложений $SH_T = \langle H_1, H_2, \ldots, H_m \rangle$. В качестве характеристики связности фрагментно-слотовых шаблонов предложений текста T определим на множестве SH_T антирефлексивное бинарное отношение Ω_T, элементами которого являются пары соседних фрагментно-слотовых шаблонов предложений из множества Sh_T, т. е. $\Omega_T = \{(H_i, H_{i+1}) | i = \overline{1, m-1}\}$. Отношение Ω_T назовем *фрагментно-слотовым шаблоном текста T*.

Вершины-слоты соседних фрагментно-слотовых шаблонов предложений текста могут быть помечены символом «′» или символом «″». Это означает, что после заполнения первого слота коммуникативным фрагментом f' второй слот должен быть заполнен коммуникативным фрагментом f'', таким, что выполняется соотношение $f'' \prec_F f'$, т. е. f' и f'' принадлежат некоторому классу контаминированных коммуникативных фрагментов.

3.5. Фрагментно-слотовый шаблон предметной области

Для формирования фрагментно-слотового шаблона предметной области необходимо предварительно подготовить множество $\{T_i \mid i = \overline{1, r}\}$ некоторых «хороших» текстов. Обозначим через Ω_{T_i} фрагментно-слотовый шаблон текста T_i. Тогда объединение множеств $\Omega_{Ct} = \bigcup_{i=1}^{r} \Omega_{T_i}$ назовем *фрагментно-слотовым шаблоном предметной области*, определяемой тематическим корпусом текстов *Ct*.

4. Описание алгоритма пересказа содержания текста

Пусть $T \in Cf$ – некоторый текст, Ct – релевантный ему тематический корпус текстов, а $Q = \langle \pi_1, \pi_2, \ldots, \pi_m \rangle$ – кортеж информативных предложений этого текста. Алгоритм пересказа содержания текста T работает следующим образом.

Из вербально-ассоциативной сети предметной области Net_{Ct} исключим все вершины, которые соответствуют базовым коммуникативным фрагментам, отсутствующим в предложениях кортежа Q. Удалим также из сети Net_{Ct} инцидентные исключенным вершинам ребра и дуги. Полученный граф обозначим через Net_T^+.

Из фрагментно-слотового шаблона предметной области Ω_{Ct} сформируем множество начальных фрагментно-слотовых шаблонов предложений $Form_1 = \{H_1, H_2, \ldots\}$. Построим для каждого шаблона из множества $Form_1$ его вербально-ассоциативную сеть Net_{H_1}. Сформируем множество всех орцепей $Req^{12}_{H_1}$ длиной 1 и 2 графа Net_{H_1}. Элементами множества $Req^{12}_{H_1}$ являются орцепи вида f_1s_1, s_2g_2, $f_3s_3g_3$, где f_1s_1 – конечные фрагменты предложения, а s_2g_2 – начальные; s_1, s_2 и s_3 – слоты; f_3 и g_3 – коммуникативные фрагменты, не являющиеся начальными и конечными в фрагментно-слотовом шаблоне H_1. Орцепи из множества $Req^{12}_{H_1}$ являются запросами на поиск релевантных орцепей в графе Net^{+}_{T} с целью заполнения найденных слотов коммуникативными фрагментами. Орцепи графов Net^{+}_{T} и Net_{H_1} считаем совпавшими, если совпали соответствующие коммуникативные фрагменты. Например, совпавшими являются орцепи *fsg* и *frg*, где *r* – коммуникативный фрагмент, заполняющий слот *s*. Если все слоты фрагментно-слотового шаблона заполнены, то полученное в результате предложение включаем в некоторое множество *Sen*. Далее описанную процедуру повторяем для всех остальных шаблонов из множества $Form_1$.

В сформированном множестве *Sen* ищем релевантное графу Net^{+}_{T} предложение π_l, полученное из шаблона H_l. Оно является началом формируемого пересказа текста.

Далее создаем множество $Form_2$, состоящее из всех фрагментно-слотовых шаблонов предложений *H*, таких, что $(H_l, H) \in \Omega_T$. Процессы заполнения слотами шаблонов из множеств $Form_2$, $Form_3$ и т. д. аналогичны такой процедуре для шаблонов из множества $Form_1$.

П р и м е р . Рассмотрим следующий текстовый фрагмент из статьи [5]:

В ходе интерпретации воссоздается мысленный мир, в котором, по презумпции интерпретатора, автор конструировал дискурс и в котором описывается реальное или нереальное положение дел. При этом анализ дискурса предполагает наличие языкового инструментария, при котором исследователь обращается не только к собственным лингвистическим знаниям, но также и общему фоновому знанию о реальном мире, поскольку в процессах понимания и порождения речи взаимодействуют все базы данных, хранящиеся в когнитивном аппарате человека. В основном анализу подвергаются не отдельные слова, а более крупные объединения (предложения или даже целые тексты), так как известно, что трансляция смысла ведется с помощью именно текстов. Именно поэтому текст стал объектом исследования отдельного направления языкознания, лингвистики текста, которое стремится выйти за рамки предложения. Дискурс может членироваться на высказывания, в то время как существуют другие объединения, которые складываются из последовательных предложений, например текст.

На начальном этапе генерации пересказа данного текста формируется множество фрагментно-слотовых шаблонов предложений $Form_1$: $Form_1$ = {⟨обсуждается проблема / ◊ ⟩, ⟨рассматриваются вопросы / ◊ / путем использования ◊ ⟩, ⟨речь идет о / ◊ ⟩, в работе / приведены результаты / ⟩,...}, где символом «/» обозначены разделители между коммуникативными фрагментами, а символом «◊» – слоты.

После построения для всех фрагментно-слотовых шаблонов их вербально-ассоциативных сетей и поиска коммуникативных фрагментов для заполнения слотов выбранного шаблона получим начальное предложение пересказа исходного текста в виде цепочки коммуникативных фрагментов: *Рассматриваются вопросы / конструирования дискурса / путем использования / лингвистических знаний / о реальном мире.*

На последующих этапах синтеза выходного текста процесс поиска фрагментно-слотовых шаблонов предложений и заполнения их слотов повторяется аналогичным образом. В результате получим следующие предложения сформированного пересказа текста: *Для изучения дискурса / возникло направление / в языкознании / – / лингвистика текста. При конструировании / анализируются фрагменты / данного дискурса. Элементами дискурса / являются высказывания.*

Заключение

Модель пересказа содержания текста на основе слияния коммуникативных фрагментов может быть использована при решении следующих задач:

– индексирование текстовых документов и запросов на поиск информации. В индексируемом тексте выявляются коммуникативные фрагменты. Поисковый образ проиндексированного текста или запроса на поиск информации состоит из информативных фрагментов, каждому соответствует значение информативности;

– автоматическое реферирование и аннотирование текста. Одним из этапов при реализации этих процессов является построение шаблона входного текста. На основе этого шаблона генерируется выходной текст.

Список литературы

1. Гаспаров, Б.М. Язык, память, образ. Лингвистика языкового существования / Б.М. Гаспаров. – М. : Новое литературное обозрение, 1996. – 352 с.

2. Липницкий, С.Ф. Индексирование текстовой информации на основе моделирования вербальных ассоциаций / С.Ф Липницкий // Информатика. – 2012. – № 3. – С. 94–102.

3. Шрейдер, Ю.А. Равенство. Сходство. Порядок / Ю.А. Шрейдер. – М. : Наука, 1971. – 256 с.

4. Липницкий, С.Ф. Модель представления знаний в информационных системах на основе вербальных ассоциаций / С.Ф Липницкий // Информатика. – 2011. – № 4. – С. 21–28.

5. Темнова, Е.В. Современные подходы к изучению дискурса / Е.В. Темнова // Язык, сознание, коммуникация : сб. статей. – М. : МАКС Пресс, 2004. – Вып. 26. – С. 24–32.

Поступила 20.05.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: lipn@newman.bas-net.by

S.F. Lipnitsky

MODELING THE TEXT CONTENT BY MERGING COMMUNICATIVE FRAGMENTS

Mathematical model of the process of retelling the content of the test messages by the convergence of communicative fragments is proposed. Formally defined concepts such fragments, as well as the verbal-associative networks as models of domain knowledge and retell the texts. Algorithm description of the content of the text presented.

УДК 621.391

О.Г. Шевчук, В.Ю. Цветков

НОРМАЛИЗАЦИЯ КОНТУРНЫХ ЛИНИЙ ПО ТОЛЩИНЕ НА ОСНОВЕ АНАЛИЗА ЛОКАЛЬНЫХ ОРИЕНТАЦИЙ ИХ ФРАГМЕНТОВ

Предлагается метод нормализации двухконцевых контурных линий по толщине на бинарных изображениях, основанный на анализе локальных ориентаций их фрагментов. Проводится сравнение предложенного метода с известными методами скелетизации. Показывается, что предложенный метод превосходит известные методы скелетизации по быстродействию и качеству.

Введение

После обработки полутоновых изображений такими операторами выделения краев, как Канни [1], Робертс [2] и др., формируются бинарные изображения, которые содержат множество линий, образованных связанными совокупностями единичных пикселов. Получаемые таким образом линии имеют толщину, как правило, в несколько пикселов, что приводит к ошибкам их параметризации и последующей идентификации. Решить данную проблему можно путем минимизации толщины линий. Для этого могут использоваться известные методы скелетизации объектов [3–7]. В результате работы данных методов происходит удаление избыточных контурных пикселов и формируются контурные линии или скелет объекта толщиной в один пиксел. Для обработки бинарных изображений широко применяются методы скелетизации Зонга – Суня [4], шаблонный [8], волновой [9], Щепина [10]. Они являются итеративными и имеют высокую вычислительную сложность. Ориентация на площадные объекты с учетом итеративного характера обработки делает данные методы неэффективными для скелетизации контурных линий, имеющих толщину в несколько пикселов. Быстрая нормализация контурных линий по толщине возможна за счет неитеративного анализа локальных ориентаций небольших фрагментов контурных линий размером три пиксела и удаления избыточных контурных пикселов в окрестности Мура.

Целью настоящей работы является разработка быстрого метода нормализации контурных линий по толщине на основе анализа локальных ориентаций их фрагментов.

1. Метод нормализации контурных линий по толщине на основе масок

Предлагается метод нормализации двух концевых контурных линий по толщине на основе анализа локальных ориентаций их фрагментов, образованных смежными контурными пикселами, и удаления избыточных контурных пикселов. Метод отличается от известных методов скелетизации, использующих многократную обработку пикселов, однократным анализом каждого пиксела в результате квантования по ориентации фрагментов контурной линии с помощью масок 2×2 пиксела, определением избыточных контурных пикселов в этих фрагментах и их удалением, что позволяет повысить скорость и качество контурной обработки.

Исходными данными для метода нормализации контурных линий по толщине на основе анализа локальных ориентаций их фрагментов являются координаты $\{X(n)\}_{(n=\overline{0,N_L-1})}$, $\{Y(n)\}_{(n=\overline{0,N_L-1})}$ контурных пикселов выделенных линий, координаты $X_e=\|x_1(n),x_2(n)\|_{(n=\overline{0,N_L-1})}$, $Y_e=\|y_1(n),y_2(n)\|_{(n=\overline{0,N_L-1})}$ их концевых точек и бинарная матрица $P=\|p(x,y)\|_{(y=\overline{0,Y-1},x=\overline{0,X-1})}$ образов выделенных контурных линий, где $n=\overline{0,N_L-1}$ – порядковый номер линии; N_L – количество выделенных линий; $p(x,y)=\{0,1\}$ – значение пиксела в бинарной матрице образов; $y=\overline{0,Y-1}$, $x=\overline{0,X-1}$ – координаты пиксела; X, Y – размер матрицы образов; $X(n)=\|x(n,i)\|_{(i=\overline{0,N_p(n)-1})}$, $Y(n)=\|y(n,i)\|_{(i=\overline{0,N_p(n)-1})}$ – координаты пикселов n-й линии;

$i=\overline{0,N_p(n)-1}$ – порядковый номер пиксела n-й линии; $N_p(n)$ – количество пикселов в n-й линии. Контурные пикселы формируются в результате сегментации бинарного изображения методом выращивания областей (Region Growing, RG) [11]. Каждому контурному пикселу присваивается номер линии, которой он принадлежит.

Алгоритм нормализации n-й выделенной линии состоит из следующих шагов:

Шаг 1. Инициализация матрицы $H(n)=\|h(n,i)\|_{(i=\overline{1,N_p(n)-1})}$ состояний контурных пикселов n-й линии, где $h(n,i)=\begin{cases}0, & \text{если пиксел должен быть удален;}\\ 1, & \text{если пиксел не обработан, } h(n,i)\leftarrow 1 \text{ при } i=\overline{0,N_p(n)-1};\\ 2, & \text{если пиксел обработан.}\end{cases}$

Шаг 2. Определение ориентации $o(n)$ линии с помощью выражения

$$o(n)=\frac{y_2(n)-y_1(n)}{x_2(n)-x_1(n)}.$$

Шаг 3. Формирование бинарных масок $M_1=\|m_1(g,j)\|_{(g=\overline{0,1},\,j=\overline{0,1})}$, $M_2=\|m_2(g,j)\|_{(g=\overline{0,1},\,j=\overline{0,1})}$ размером 2×2 пиксела для квантования по ориентации фрагментов линии, где

$$m_1(0,0)=\begin{cases}0 \text{ при } o(n)<0,\\ 1 \text{ при } o(n)\ge 0;\end{cases}\quad m_1(0,1)=\begin{cases}1 \text{ при } o(n)<0,\\ 0 \text{ при } o(n)\ge 0;\end{cases}\quad m_1(1,0)=1,\ m_1(1,1)=1;$$

$$m_2(0,0)=\begin{cases}1 \text{ при } o(n)<0,\\ 0 \text{ при } o(n)\ge 0;\end{cases}\quad m_2(0,1)=\begin{cases}0 \text{ при } o(n)<0,\\ 1 \text{ при } o(n)\ge 0;\end{cases}\quad m_2(1,0)=1,\ m_2(1,1)=1.$$

Примерами масок являются матрицы $\begin{bmatrix}0 & 1\\ 1 & 1\end{bmatrix}$ и $\begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}$.

Шаг 4. Инициализация счетчика цикла $i\leftarrow 0$ обработки пикселов n-й линии.

Шаг 5. Начало цикла анализа пикселов n-й линии.

Обработка i-го элемента матрицы $H(n)$ состояний согласно выражению

$$\begin{cases}\big(h(n,i)=1\big)\Rightarrow y_0\leftarrow y(n,i),\ x_0\leftarrow x(n,i), \text{ переход на шаг 6;}\\ \big(h(n,i)=2\big)\Rightarrow \text{ переход на шаг 12;}\\ \big(h(n,i)=0\big)\Rightarrow \text{ удаляются } y(n,i),\ x(n,i) \text{ и } h(n,i),\ N_p(n)=N_p(n)-1, \text{ переход на шаг 13,}\end{cases}$$

где y_0, x_0 – координаты опорного пиксела.

Шаг 6. Формирование матриц $X_R=\|x_R(q)\|_{(q=\overline{0,2})}$, $Y_R=\|y_R(q)\|_{(q=\overline{0,2})}$ координат и $I_R=\|i_R(q)\|_{(q=\overline{0,2})}$ индексов связанных пикселов с использованием маски M_1, элементы которых вычисляются с помощью выражений

$$\begin{cases}\exists b\Big(b\in\big[0,N_p(n)-1\big]\Big)\left(\begin{array}{l}\big(m_1(g,j)\,p(y_0+g,x_0+j)=1\big)\wedge\big((y_0+g)=y(n,b)\big)\wedge\\ \wedge\big((x_0+j)=x(n,b)\big)\wedge\big(h(n,b)=1\big)\wedge(b\ne i)\end{array}\right)\Rightarrow\\ \Rightarrow\Big(y_R(K_q)\leftarrow(y_0+g)\Big),\Big(x_R(K_q)\leftarrow(x_0+j)\Big),\Big(i_R(K_q)\leftarrow b\Big) \text{ при } o(n)<0;\\ \exists b\Big(b\in\big[0,N_p(n)-1\big]\Big)\left(\begin{array}{l}\big(m_1(g,j)\,p(y_0-g,x_0-j)=1\big)\wedge\big((y_0-g)=y(n,b)\big)\wedge\\ \wedge\big((x_0-j)=x(n,b)\big)\wedge\big(h(n,b)=1\big)\wedge(b\ne i)\end{array}\right)\Rightarrow\\ \Rightarrow\Big(y_R(K_q)\leftarrow(y_0-g)\Big),\Big(x_R(K_q)\leftarrow(x_0-j)\Big),\Big(i_R(K_q)\leftarrow b\Big) \text{ при } o(n)\ge 0;\end{cases}$$

$$\left(K_q \leftarrow K_q + 1\right)$$

при $g = \overline{0,1}$, $j = \overline{0,1}$, где K_q – количество найденных связанных пикселов (при инициализации шага 6 $K_q \leftarrow 0$).

Если $K_q = 0$, осуществляется формирование матриц X_R, Y_R и индексов I_R связанных пикселов с использованием маски M_2, элементы которых вычисляются с помощью выражений

$$\begin{cases} \exists b\left(b \in \left[0, N_p(n)-1\right]\right)\left(\begin{matrix}\left(m_2(g,j)\, p(y_0-g, x_0-j)=1\right)\wedge\left(\left(y_0-g\right)=y(n,b)\right)\wedge \\ \wedge\left(\left(x_0-j\right)=x(n,b)\right)\wedge\left(h(n,b)=1\right)\wedge\left(b\neq i\right)\end{matrix}\right)\Rightarrow \\ \Rightarrow\left(y_R\left(K_q\right)\leftarrow\left(y_0-g\right)\right),\left(x_R\left(K_q\right)\leftarrow\left(x_0-j\right)\right),\left(i_R\left(K_q\right)\leftarrow b\right) \text{ при } o(n)<0; \\ \exists b\left(b \in \left[0, N_p(n)-1\right]\right)\left(\begin{matrix}\left(m_2(g,j)\, p(y_0+g, x_0+j)=1\right)\wedge\left(\left(y_0+g\right)=y(n,b)\right)\wedge \\ \wedge\left(\left(x_0+j\right)=x(n,b)\right)\wedge\left(h(n,b)=1\right)\wedge\left(b\neq i\right)\end{matrix}\right)\Rightarrow \\ \Rightarrow\left(y_R\left(K_q\right)\leftarrow\left(y_0+g\right)\right),\left(x_R\left(K_q\right)\leftarrow\left(x_0+j\right)\right),\left(i_R\left(K_q\right)\leftarrow b\right) \text{ при } o(n)\geq 0; \end{cases}$$

$$\left(K_q \leftarrow K_q + 1\right)$$

при $g = \overline{0,1}$, $j = \overline{0,1}$.

При $K_q = 1..3$ осуществляется переход на шаг 7, при $K_q = 0$ – переход на шаг 11.

Шаг 7. Формирование матрицы $O_R = \left\|o_R(q)\right\|_{q=\overline{0,K_q}}$ локальных ориентаций контурных фрагментов, образованных опорным пикселом и пикселами, связанными с ним. Локальная ориентация $o_R(q)$ каждого фрагмента с координатами (y_0, x_0) и $\left(x_R(q), y_R(q)\right)$ вычисляется с помощью выражения

$$o_R(q) = \frac{y_R(q) - y_0}{x_R(q) - x_0}$$

при $q = \overline{0, K_q - 1}$.

Шаг 8. Определение состояний $h\left(n, i_R(q)\right)$ связанных пикселов на основе анализа их локальных ориентаций O_R относительно ориентации линии $o(n)$ в соответствии с выражением

$$\begin{cases} \left(\left(o_R(q)=0\right)\vee\left(o_R(q)\to\infty\right)\right)\Rightarrow\left(h\left(n,i_R(q)\right)\leftarrow\begin{cases}1 \text{ при } \left\|o_R(q)\right|-\left|o(n)\right\|\leq 0{,}1\\ 0 \text{ при } \left\|o_R(q)\right|-\left|o(n)\right\|> 0{,}1\end{cases}\right), \\ \left(o_R(q)=1\right)\Rightarrow\left(h\left(n,i_R(q)\right)\leftarrow\begin{cases}1 \text{ при } \left(o(n)\neq 0\right)\wedge\left(\left|o(n)\right|<<\infty\right)\\ 0 \text{ при } \left(o(n)=0\right)\vee\left(\left|o(n)\right|\to\infty\right)\end{cases}\right) \end{cases}$$

при $q = \overline{0, K_q - 1}$.

Шаг 9. Проверка наличия разрывов в линии.

Если выполняется условие

$$\left(K_q = 2\right)\wedge\left(\sum_{q=0}^{K_q-1} h\left(n, i_R(q)\right) = 0\right),$$

то для устранения разрыва линии осуществляется переход на шаг 10, иначе – переход на шаг 11.

Шаг 10. Устранение разрывности.

Для устранения разрывности линии в матрицы $Y(n)$, $X(n)$ и $H(n)$ добавляются новые элементы в соответствии с выражениями

$$N_p(n) \leftarrow N_p(n)+1;$$

$$y(n, N_p(n)-1) \leftarrow \begin{cases} y_0 \text{ при } |o(n)| <= 0,5, \\ y_0+1 \text{ при } |o(n)| > 0,5; \end{cases}$$

$$x(n, N_p(n)-1) \leftarrow \begin{cases} x_0-1 \text{ при } o(n)=[-2;0], \\ x_0 \text{ при } |o(n)| > 2, \\ x_0+1 \text{ при } o(n)=(0;2]; \end{cases}$$

$$h(n, N_p(n)-1) \leftarrow 1.$$

Осуществляется переход на шаг 11.

Шаг 11. В соответствии с порядковым номером координат i-го опорного пиксела y_0, x_0 матриц $Y(n)$ и $X(n)$ устанавливается состояние данного пиксела $h(n,i) \leftarrow 2$ в стеке $H(n)$.

Шаг 12. Установка значения счетчика $i \leftarrow i+1$.

Шаг 13. Анализ значения счетчика.

При $i > (N_p(n)-1)$ осуществляется выход из алгоритма, иначе – переход на шаг 5.

В результате выполнения метода происходят поиск и удаление избыточных пикселов. Избыточными являются пикселы, исключение которых уменьшает толщину контура до одного пиксела, не разрывая его.

2. Оценка эффективности метода нормализации контурных линий по толщине

Разработанный алгоритм реализован на языке C++ с использованием библиотеки OpenCV 2.4.10. Для сравнительной оценки работы алгоритма реализован наиболее известный алгоритм скелетизации Зонга – Суня. Эксперимент проведен на компьютере со следующими техническими характеристиками: процессор Intel(R) Core(TM) i7-4700HQCPU @ 2,40 ГГц; ОЗУ – 6 ГБ; тип системы – 64-разрядная операционная система Windows 8.1.

Для первичного тестирования алгоритма нормализации контурных линий по толщине использованы следующие линии:

– искусственные, построенные и повернутые в графическом редакторе;

– выделенные на изображении, полученном с помощью фотокамеры, и повернутые в графическом редакторе.

В качестве критериев эффективности алгоритмов использованы дисперсия формфактора и время нормализации линии. Формфактор F_i линии для i-го угла поворота изображения вычисляется как отношение размера линии r_i, определяемого по известным координатам ее конечных точек, к длине линии s_i, определяемой суммой образующих ее контурных точек [12]:

$$F_i = r_i / s_i .$$

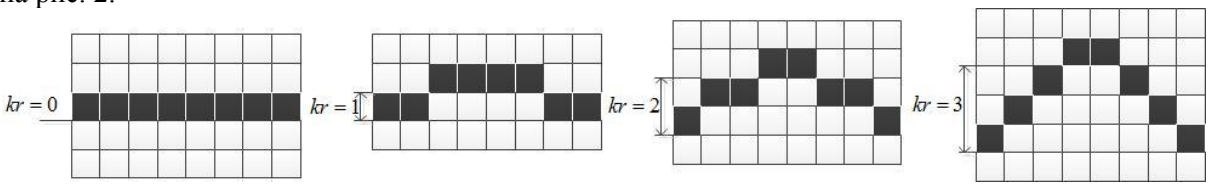
В отличие от известных методов оценки качества выделения линий (метода Прэтта [13], метрики ошибок Баддели [14], ро-коэффициента Григореску [15]) формфактор имеет существенно меньшую вычислительную сложность, что позволяет строить быстрые алгоритмы обработки изображения на его основе.

Дисперсия формфактора рассчитывается для множества изображений одной и той же линии при повороте на различные углы с помощью выражения

$$D = \frac{\sum_{i=1}^{n} \left(F_i - \overline{F} \right)^2}{n},$$

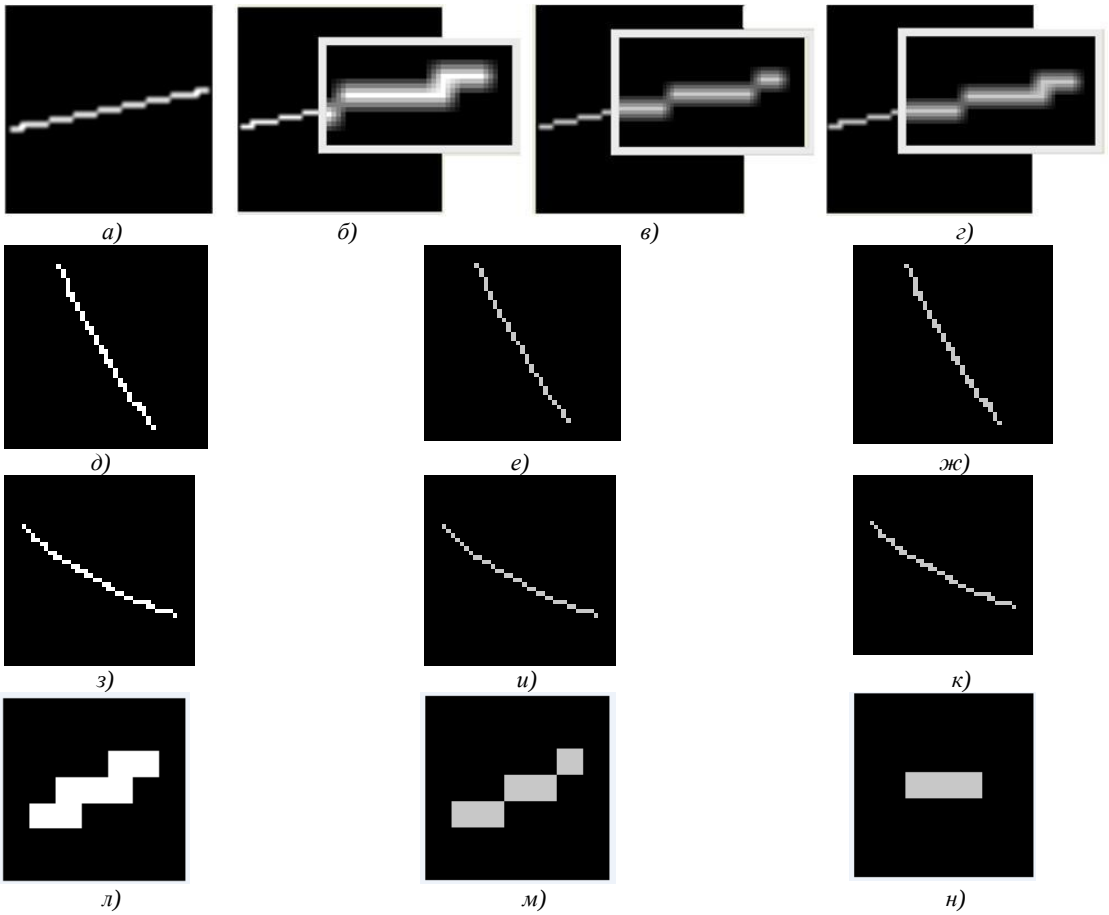
где n – количество линий; $\overline{F}$ – значение формфактора линии, усредненное по углам поворота изображения.

Для искусственно созданных линий использованы размеры 5, 11, 15, 25, 41, 65 и 101 пиксел с кривизной $kr = 0..3$ (рис. 1) относительно идеальной прямой линии. Каждая линия поворачивалась на угол от 0 до 180° относительно центра изображения. Примеры исходной линии и линии, обработанной разработанным алгоритмом и алгоритмом Зонга – Суня, представлены на рис. 2.



а) *б)* *в)* *г)*

Рис. 1. Кривизна линии длиной восемь пикселов: *а)* $kr = 0$; *б)* $kr = 1$; *в)* $kr = 2$; *г)* $kr = 3$



а) *б)* *в)* *г)*

д) *е)* *ж)*

з) *и)* *к)*

л) *м)* *н)*

Рис. 2. Результаты работы алгоритма нормализации линии по толщине: *а)* исходная линия 1 длиной 41 пиксел при $k = 0$ и повороте на 11°; *б)* исходная линия 1 с выделенным проблемным участком; *в)* результат обработки линии 1 разработанным алгоритмом; *г)* результат обработки линии 1 алгоритмом Зонга – Суня; *д)* исходная линия 2 длиной 41 пиксел при $k = 2$ и повороте на 120°; *е)* результат обработки линии 2 разработанным алгоритмом; *ж)* результат обработки линии 2 алгоритмом Зонга – Суня; *з)* исходная линия 3 длиной 41 пиксел при $k = 3$ и повороте на 150°; *и)* результат обработки линии 3 разработанным алгоритмом; *к)* результат обработки линии 3 алгоритмом Зонга – Суня; *л)* исходная линия 4 длиной пять пикселов при повороте на 26°; *м)* результат обработки линии 4 разработанным алгоритмом; *н)* результат обработки линии 4 алгоритмом Зонга – Суня

Из рис. 2 следует, что алгоритм Зонга – Суня оставляет больше избыточных пикселов, а это влияет на вычисление формфактора. На рис. 2, *л–н* показано, что для алгоритма Зонга – Суня характерны искажения линий.

Графики зависимости дисперсии формфактора от кривизны линии различной длины приведены на рис. 3. Для линии длиной пять пикселов использована кривизна $kr = 0..1$ пиксела, для линии длиной 11 пикселов – $kr = 0..2$ пиксела, для линий длиной 15, 25, 41, 65 и 101 пиксел – $kr = 0..3$ пиксела.

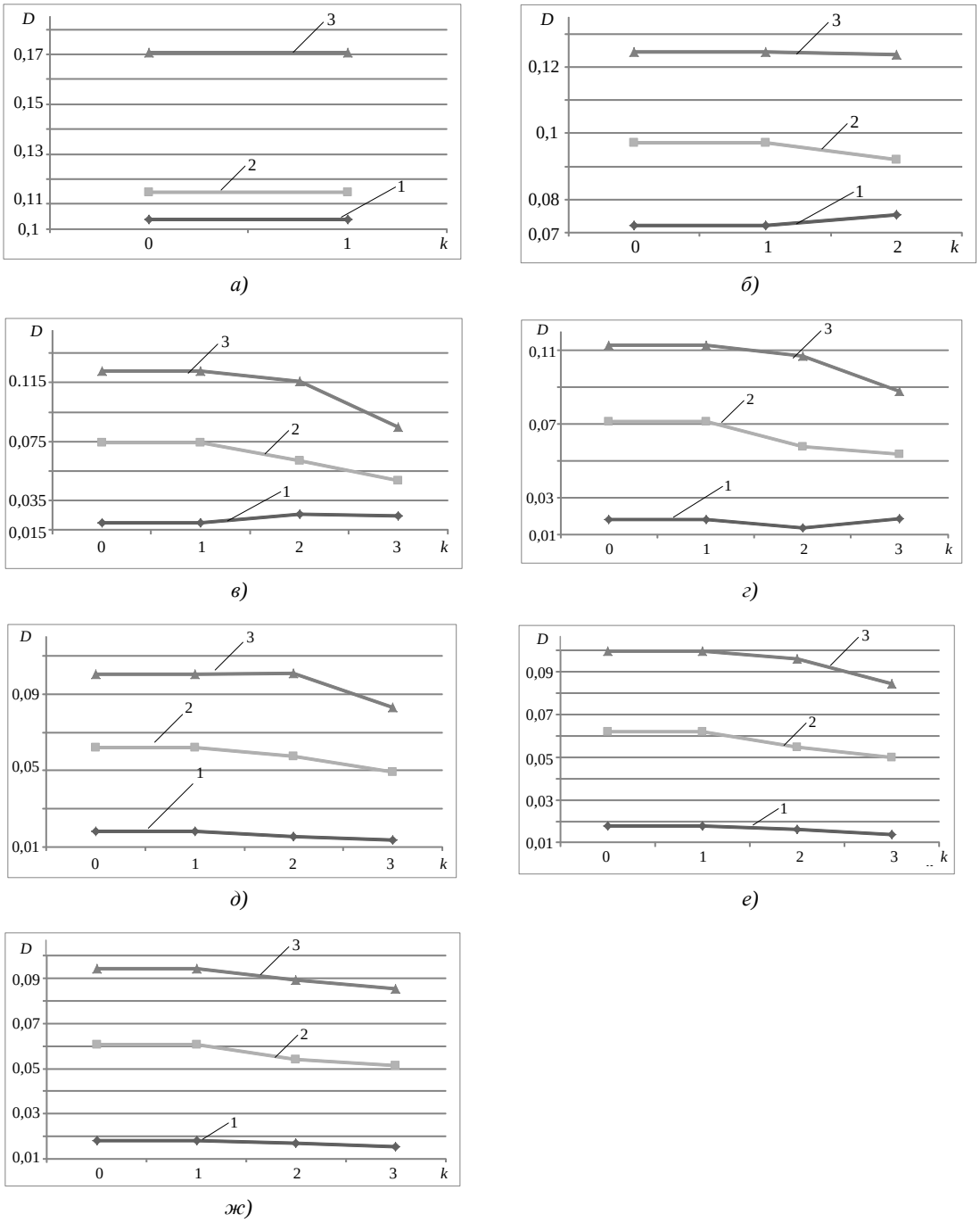
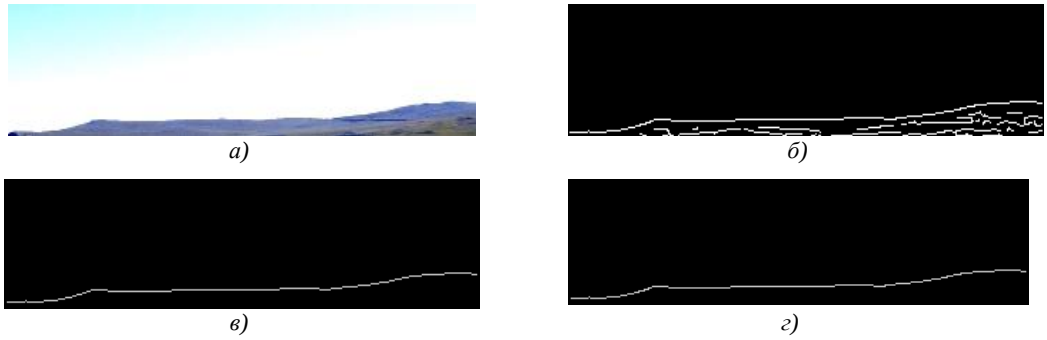


Рис. 3. Зависимости дисперсии формфактора от кривизны линии при длине линии:
а) 5 пикселов; *б)* 11 пикселов; *в)* 15 пикселов; *г)* 25 пикселов; *д)* 41 пиксел; *е)* 65 пикселов; *ж)* 101 пиксел;
1 – при использовании алгоритма нормализации линии по толщине;
2 – при использовании алгоритма Зонга – Суня;
3 – без применения алгоритмов скелетизации

Из рис. 3 следует, что использование методов нормализации и скелетизации уменьшает дисперсию формфактора при повороте искусственной линии. При применении разработанного алгоритма нормализации дисперсия D формфактора в зависимости от длины и кривизны линии в 1,6–4,2 раза меньше по сравнению с алгоритмом скелетизации Зонга – Суня и в 3,4–7,8 раз меньше по сравнению с вариантом без использования алгоритмов нормализации или скелетизации. С увеличением кривизны k линии дисперсия D для алгоритма нормализации линий по толщине незначительно увеличивается или уменьшается в зависимости от длины линии, а при использовании алгоритма Зонга – Суня или варианта без использования алгоритмов нормализации и скелетизации увеличивается в среднем на 10 % для искусственных линий.

Результаты работы алгоритма нормализации контурных линий по толщине на реальном изображении представлены на рис. 4.



а) *б)* *в)* *г)*

Рис. 4. Результаты нормализации линии реального изображения по толщине:
а) тестовое изображение; *б)* тестовое изображение, обработанное алгоритмом Канни; *в)* результат обработки линии разработанным алгоритмом; *г)* результат обработки линии алгоритмом Зонга – Суня

Приведем значения дисперсии формфактора линии реального изображения (рис. 4, *а*) при повороте изображения в графическом редакторе на угол от 0 до 180° относительно центра изображения с шагом в 15°, полученные с помощью следующих методов:

алгоритма нормализации линий по толщине – 0,019 484;

алгоритма Зонга – Суня – 0,103 612;

без нормализации – 0,166 342.

Видно, что разработанный алгоритм нормализации контурных линий по толщине для реальных изображений улучшает значение формфактора в 5,3 раза по сравнению с алгоритмом Зонга – Суня и в 8,5 раза по сравнению с вариантом без использования алгоритмов нормализации и скелетизации линий.

Оценка среднего времени (усреднение по 180 углам поворота изображения) выполнения разработанного алгоритма по сравнению с алгоритмом Зонга – Суня и среднее время сегментации (усреднение по 180 углам поворота изображения) для искусственной линии приведена в табл. 1 (алгоритм нормализации контурных линий по толщине используется после сегментации, а алгоритм Зонга – Суня – до).

Таблица 1

Среднее время выполнения алгоритмов нормализации и сегментации
для искусственной линии различной длины при повороте изображения

Длина линии, пиксел	Алгоритм нормализации линий по толщине		Алгоритм Зонга – Суня	
	Время выполнения, мс	Время сегментации, мс	Время выполнения, мс	Время сегментации, мс
5	0,057 0065	0,038 7545	0,110 769	0,039 522
11	0,101 445	0,062 543 67	0,355 2903	0,063 474
15	0,130 6015	0,078 612	0,593 886	0,080 837
25	0,195 1233	0,116 095 25	1,560 834	0,118 144
41	0,293 3203	0,179 705	4,119 6445	0,183 963
65	0,445 7818	0,290 381	10,051 333	0,292 878
101	0,668 9235	0,477 1175	24,203 958	0,481 981

Из табл. 1 видно, что среднее время сегментации для обоих алгоритмов примерно одинаково, при этом среднее время выполнения разработанного алгоритма уменьшается в 1,9–36 раз в зависимости от длины линии по сравнению с алгоритмом Зонга – Суня при обработке искусственной линии. В результате общее среднее время выполнения разработанного алгоритма уменьшается до 21,5 раза по сравнению с алгоритмом Зонга – Суня.

В табл. 2 приведено среднее время (усреднение по 12 углам поворота изображения) выполнения алгоритма сегментации и алгоритмов нормализации и скелетизации для линии изображения на рис. 4, *а*.

Таблица 2

Среднее время выполнения алгоритмов сегментации, нормализации и скелетизации линии реального изображения при его повороте

Время, мс	Алгоритм нормализации линий по толщине	Алгоритм Зонга – Суня
Нормализации	3,203 058	92,422 05
Сегментации	1,756 553	1,980 582

Из табл. 2 следует, что среднее время сегментации для обоих алгоритмов примерно одинаково, при этом среднее время выполнения разработанного алгоритма до 29 раз меньше по сравнению с алгоритмом Зонга – Суня для линии реального изображения, а общее среднее время выполнения до 19 раз меньше.

Для оценки работы алгоритма нормализации линий по толщине в реальных условиях использованы полутоновые изображения размером 3920×2204 пиксела, полученные с помощью поворачиваемой фотокамеры. В качестве критериев оценки применимы стабильность сегментации линий при повороте камеры (определяемая числом выделяемых линий при различных углах поворота) и время обработки изображения:

$$F(\alpha) = \left(f(\alpha_i) \cdot 100\,\% \right) / f(\alpha_1),$$

где $f(\alpha_1)$ – множество прямых линий с заданным значением формфактора на эталонном изображении при повороте камеры на угол $\alpha_1 = 0$; $f(\alpha_i)$ – множество прямых линий с заданным значением формфактора на изображении i при повороте камеры на угол α_i°. Линии на изображениях выделены по значению формфактора в пределах 0,8–1,2.

На рис. 5 показаны зависимости стабильности сегментации линий на изображении от угла поворота камеры при использовании алгоритма нормализации линий по толщине и алгоритма Зонга – Суня и без использования каких-либо алгоритмов нормализации. Из рисунка следует, что в зависимости от угла поворота камеры использование разработанного алгоритма повышает стабильность $F(\alpha)$ количества выделенных прямых линий по заданному формфактору в 1,04–1,62 раза по сравнению с алгоритмом Зонга – Суня и в 1,52–2,3 раза по сравнению с вариантом без нормализации или скелетизации линий (количество выделенных прямых линий увеличивается до 186 раз).

Оценка среднего времени (усреднение по шести-семи углам поворота изображения) выполнения алгоритма сегментации и нормализации изображений, полученных с помощью поворачивающейся фотокамеры, представлена в табл. 3.

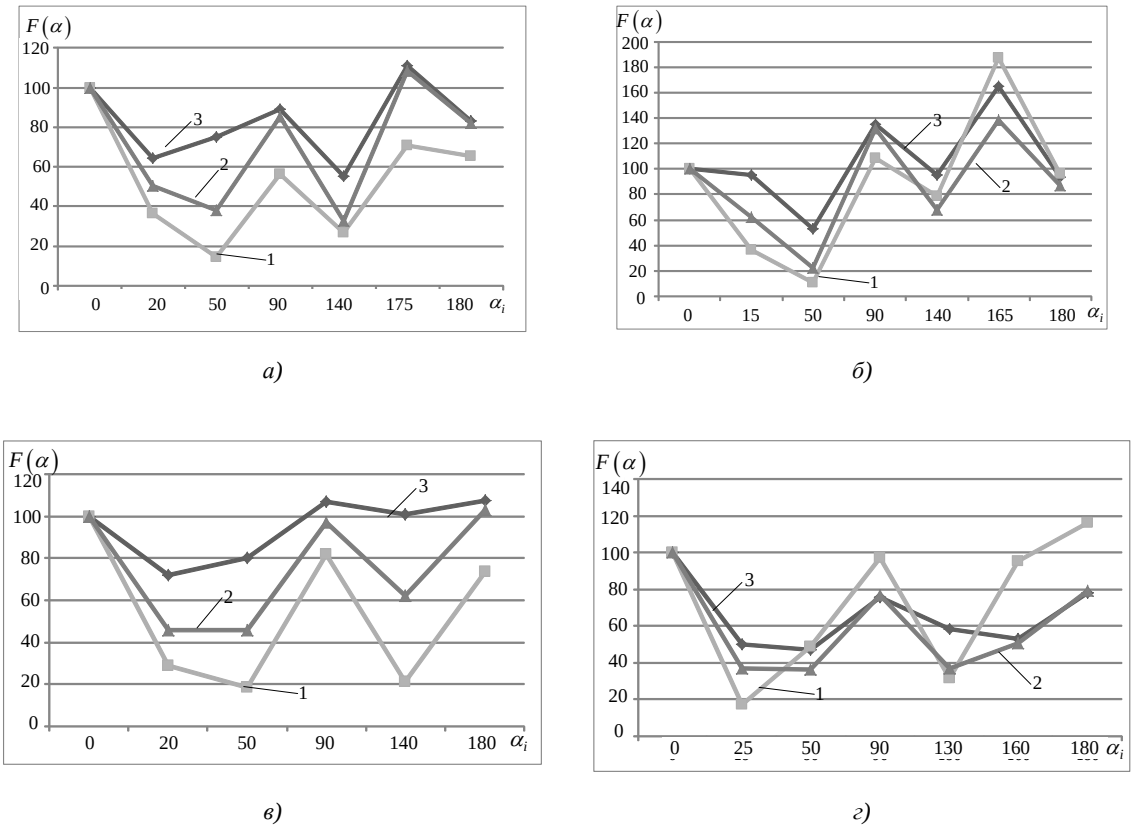


Рис. 5. Зависимости числа линий от угла поворота изображений:
а) для изображения 1; *б)* для изображения 2; *в)* для изображения 3; *г)* для изображения 4;
1 – без применения алгоритмов скелетизации;
2 – при использовании алгоритма Зонга – Суня;
3 – при использовании алгоритма нормализации линии по толщине

Таблица 3

Время выполнения алгоритмов сегментации, нормализации и скелетизации линий при повороте реального изображения размером 3920×2204 пиксела

Изображение 1		Время, мс	**Изображение 3**		Время, мс
Алгоритм Зонга – Суня	Сегментация	32,594	Алгоритм Зонга – Суня	Сегментация	10,320
	Скелетизация	244,094		Скелетизация	233,619
Алгоритм нормализации	Сегментация	34,209	Алгоритм нормализации	Сегментация	10,845
	Скелетизация	8,408		Скелетизация	2,300
Изображение 2		Время, мс	**Изображение 4**		Время, мс
Алгоритм Зонга – Суня	Сегментация	14,191	Алгоритм Зонга – Суня	Сегментация	39,870
	Скелетизация	160,262		Скелетизация	507,012
Алгоритм нормализации	Сегментация	14,912	Алгоритм нормализации	Сегментация	40,953
	Скелетизация	5,336		Скелетизация	6,477

В табл. 3 показано, что среднее время выполнения разработанного алгоритма в 29– 101 раз меньше времени выполнения алгоритма Зонга – Суня при примерно одинаковом среднем времени сегментации. В результате общее среднее время выполнения разработанного алгоритма уменьшается до 18,5 раз по сравнению с алгоритмом Зонга – Суня.

Заключение

Разработаны метод и алгоритм нормализации контурных линий по толщине на бинарных изображениях на основе анализа локальных ориентаций их фрагментов. Показано, что для искусственных линий, построенных и повернутых в графическом редакторе, а также линии, выделенной на изображении, которое получено с помощью фотокамеры и повернуто в графическом редакторе, предложенный метод обеспечивает уменьшение дисперсии форм-фактора линий до 4,2 и 5,3 раза по сравнению с методом скелетизации Зонга – Суня и до 7,8 и 8,5 раза по сравнению с вариантом без использования методов нормализации или скелетизации соответственно. Предложенный метод по сравнению с методом Зонга – Суня позволяет уменьшить время обработки до 36 и 29 раз для искусственных линий и линий, выделенных на реальных изображениях. Установлено, что для линий, выделенных на фотографиях с поворачиваемой камеры, предложенный метод обеспечивает повышение стабильности сегментации линий до 1,6 раза по сравнению с методом скелетизации Зонга – Суня и до 2,3 раза по сравнению с вариантом без использования методов нормализации или скелетизации. При этом предложенный метод по сравнению с методом Зонга – Суня позволяет уменьшить время обработки до 100 раз.

Список литературы

1. Canny, J. A Computational Approach to Edge Detection / J. Canny // IEEE Trans. Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8, no. 6. – P. 679–698.

2. Roberts, L. Machine Perception of 3D Solids / L. Roberts // Optical and Electro-optical Information Processing. – 1965. – Vol. 1. – P. 159–197.

3. Lam, L. Thinning Methodologies – a Comprehensive Survey / L. Lam, S.-W. Lee, C.Y. Suen // IEEE Trans. on Pattern Analysis and Machine Intelligence. – 1992. – Vol. 14, no. 9. – P. 869–885.

4. Chen, W. Improved Zhang – Suen thinning algorithm in binary line drawing applications / W. Chen // IEEE International Conference on Systems and Informatics 2012. – Yantai, 2012. – P. 1947–1950.

5. Абламейко, С.В. Полутоновое утоньшение цветного изображения / С.В. Абламейко, А.М. Недзьведь // Цифровая обработка изображений : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1998. – № 2. – С. 41–51.

6. Nedzved, A. Thinning of the Gray-Scale and Color Images by Sequential Analysis of the Binary Layers / A. Nedzved, S. Ablameyko // Pattern Recognition and Image Analysis. – 2000. – Vol. 10, no. 2. – P. 226–235.

7. Color Thinning with Applications to Biomedical Images / A. Nedzved [et al.] // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – 2001. – Vol. 2124. – P. 256–263.

8. Молчанова, В.С. Решение задачи топологического утончения объектов бинарного растра с использованием специализированных агентов / В.С. Молчанова, И.С. Грунский // Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ–2013). – 2013. – Vol. 12. – С. 743–747.

9. Клубков, И.М. Применение волнового алгоритма для нахождения скелета растрового изображения / И.М. Клубков // Вестник ДГТУ. – 2001. – Т. 1, № 1(7). – С. 126–133.

10. Щепин, Е.В. К топологическому подходу в анализе изображений / Е.В. Щепин, Г.М. Непомнящий // Геометрия, топология и приложения : межвузовский сб. науч. тр. / М-во высшего и средн. спец. образования РСФСР. – 1990. – С. 13–25.

11. Shih, F.Y. Automatic seeded region growing for color image segmentation / F.Y. Shih, S. Cheng // Image and Vision Computing. Newark. – 2005. – No. 23. – P. 877–886.

12. Бородина, О.Г. Выделение изолированных прямых линий на изображениях с использованием форм-фактора / О.Г. Бородина, В.Ю. Цветков // Изв. СПбГЭТУ «ЛЭТИ». – 2015. – № 1. – С. 41–45.

13. Прэтт, У. Цифровая обработка изображений / У. Прэтт. – М. : Мир, 1982. – 312 с.

14. Baddeley, A.J. An error metric for binary images / A.J. Baddeley // Robust Computer Vision: Quality of Vision Algorithms. – Amsterdam : Centre for mathematics and computer science, 1992. – P. 59–78.

15. Grigorescu, C. Contour detection based on nonclassical receptive field inhibition / C. Grigorescu // IEEE Trans. on Image Processing. – 2003. – Vol. 12, no. 7. – P. 729–739.

16. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М. : Техносфера, 2005. – 1007 с.

Поступила 13.06.2016

Белорусский государственный университет информатики и радиоэлектроники,
Минск, ул. П. Бровки, 4
e-mail: vtsvet@bsuir.by

A.G. Shauchuk, V.Yu. Tsviatkou

NORMALIZATION OF CONTOUR LINES IN THICKNESS BASED ON ANALYSIS OF LOCAL ORIENTATION OF THEIR FRAGMENTS

We propose a method of normalization in thickness of the double-ended contour lines based on the analysis of local orientations of fragments for binary images. Comparison of the proposed method with known methods of thinning is held. It is shown that the proposed method is superior to the known methods of thinning on speed and quality.

УДК 004.942

В.М. Артемьев, А.О. Наумов, Л.Л. Кохан

АДАПТИВНАЯ ФИЛЬТРАЦИЯ КОМПЛЕКСИРОВАННЫХ ИЗМЕРЕНИЙ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Приводится методика синтеза адаптивных фильтров на основе метода наименьших квадратов при комплексированных измерениях, позволяющих решать задачи при отсутствии априорной информации о моделях полезных сигналов и характеристиках шумов измерений. Дается сравнительная оценка точности фильтрации с результатами фильтра Калмана.

Введение

Повышение точности фильтрации и надежности при отказах датчиков достигается комплексированием результатов измерений одних и тех же параметров совокупностью измерителей, которые могут быть построены на различных физических принципах. Комплексирование широко используется в ряде технических систем, например в навигации, управлении подвижными объектами, атомной энергетике и др. Статистическая теория комплексированной фильтрации разработана достаточно полно и отображена в литературе, например [1]. Разработанные методы для своей реализации требуют знания моделей и априорных статистических характеристик измеряемых параметров и помех. Однако в некоторых случаях такие данные исследователю недоступны, что приводит к необходимости решения задач в условиях неопределенности. Для этого может применяться детерминистский подход, использующий метод наименьших квадратов [2], в котором вместо статистических данных применяется лишь эмпирическая информация и информация, извлекаемая из текущих результатов наблюдений. Такой подход расширяет область применения фильтров за счет их универсальности. Однако его недостатком является ухудшение точности фильтрации из-за снижения объема информации о характеристиках воздействий. Чтобы частично компенсировать этот недостаток, применяют адаптивную фильтрацию, использующую дополнительную информацию о воздействиях, которую получают в процессе обработки текущих измерений. По данному направлению имеется ряд публикаций, связанных с обработкой сигналов в адаптивных антенных решетках [3, 4], системах связи, фильтрации речевых сигналов и др. [5]. В этих работах авторы исследуют структуры решетчатых или трансверсальных фильтров с конечной памятью и задача сводится к нахождению параметров фильтра заданной структуры на основе использования классического квадратичного функционала невязки.

Вызывает интерес решение более общей задачи синтеза адаптивных рекуррентных фильтров с нахождением их структуры, которое можно осуществить с помощью рекуррентного метода наименьших квадратов (РМНК) [6]. Для этого предлагается расширить структуру критерия оптимальности путем добавления квадратичных слагаемых, придающих фильтру как адаптивные, так и рекуррентные свойства. Задача решается с учетом комплексированных измерений полезного сигнала.

1. Исходные положения

Предполагается, что фильтрации подлежит случайная скалярная последовательность x_k, где $k = 0, 1, 2\ldots$ есть дискретное время. Измерения осуществляются посредством N датчиков, совокупность сигналов на выходах которых можно представить N-мерным вектором $x_k \cdot \mathbf{1}$, $\mathbf{1} = [1,...,1]^T$, где верхний индекс T обозначает операцию транспонирования. Вектор сигналов датчиков представляется в виде $x_k H\mathbf{1}$, где $N \times N$ матрица H с элементами $h_{i,j}$ ($i = \overline{1,\ N}$, $j = \overline{1,N}$) отображает характеристики датчиков и связей между ними. Диагональные элементы

матрицы $h_{i,i}$ являются коэффициентами чувствительности датчиков. Наблюдения осуществляются со случайными ошибками в виде аддитивных шумов с нулевым математическим ожиданием $\mathbf{v}_k = \left[v_{1,k}, v_{2,k}, ..., v_{\mathrm{N},k} \right]^T$, в результате чего вектор наблюдений $\mathbf{z}_k = \left[z_{1,k}, z_{2,k}, ..., z_{\mathrm{N},k} \right]^T$ имеет вид

$$\mathbf{z}_k = H\mathbf{1}x_k + \mathbf{v}_k. \tag{1}$$

Считается, что модель формирования полезного сигнала x_k и его характеристики априори неизвестны. Кроме того, неизвестной полагается и ковариационная матрица шумов $\mathbf{v}_k$.

Адаптивная фильтрация должна обеспечить текущие рекуррентные оценки скалярного полезного сигнала x_k по результатам измерений (1) в условиях отмеченной выше априорной неопределенности. Для этого используется РМНК, в основе которого лежит выбор подходящего функционала потерь. Его главной составляющей остается классическая квадратичная невязка $\left(\mathbf{z}_k - H\mathbf{1}x_k\right)^T \left(\mathbf{z}_k - H\mathbf{1}x_k\right)$. Если ограничиться лишь этой составляющей, то теряется возможность получения рекуррентных свойств оценок, поэтому в состав функционала предлагается ввести дополнительные квадратичные составляющие, способные обеспечить как рекуррентные, так и адаптивные свойства фильтру. Одним из возможных вариантов функционала J_k может быть следующий:

$$J_k\left(\hat{x}_k, \hat{a}_k\right) = \left(\mathbf{z}_k - H\mathbf{1}\hat{x}_k\right)^T \left(\mathbf{z}_k - H\mathbf{1}\hat{x}_k\right) + \left(\hat{x}_k - \hat{a}_k\hat{x}_{k-1}\right)^2 + \alpha\left(\hat{a}_k - \hat{a}_{k-1}\right)^2. \tag{2}$$

В выражении (2) первое слагаемое задается невязкой решения, второе определяет согласование оценки $\hat{x}_k$ с оценкой на предыдущем шаге $\hat{x}_{k-1}$ и неизвестным сглаживающим коэффициентом $\hat{a}_k$, подлежащим оценке наряду с $\hat{x}_k$. Третье слагаемое обеспечивает сглаживание оценки этого коэффициента с размерным эмпирическим коэффициентом регуляризации α, дающим возможность получения стабильного решения.

2. Уравнения адаптивного фильтра

Оптимальные текущие оценки $\hat{x}_k$ и $\hat{a}_k$ находятся из условия минимума критерия (2). Первоначально полагаем, что ограничения на решения отсутствуют. Тогда необходимые условия оптимальности определяются уравнениями

$$\frac{\partial J_k\left(\hat{x}_k, \hat{a}_k\right)}{\partial \hat{x}_k} = 0; \; \frac{\partial J_k\left(\hat{x}_k, \hat{a}_k\right)}{\partial \hat{a}_k} = 0.$$

После выполнения дифференцирования приходим к равенствам

$$-\mathbf{1}^T H^T \left(\mathbf{z}_k - H\mathbf{1}\hat{x}_k\right) + \left(\hat{x}_k - \hat{a}_k\hat{x}_{k-1}\right) = 0;$$
$$-\hat{x}_{k-1}\left(\hat{x}_k - \hat{a}_k\hat{x}_{k-1}\right) + \alpha\left(\hat{a}_k - \hat{a}_{k-1}\right) = 0,$$

из которых находим следующие уравнения фильтра:

$$\hat{x}_k = \frac{\hat{a}_k}{1 + h_0^2}\hat{x}_{k-1} + \frac{1}{1 + h_0^2}\mathbf{b}^T\mathbf{z}_k; \tag{3}$$

$$\hat{a}_k = \frac{\alpha}{\alpha + \hat{x}_{k-1}^2}\hat{a}_{k-1} + \frac{\hat{x}_{k-1}}{\alpha + \hat{x}_{k-1}^2}\hat{x}_k. \tag{4}$$

В уравнениях (3), (4) использованы обозначения

$$h_0^2 = \mathbf{1}^T H^T H \mathbf{1} = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} h_{i,j} \right)^2 ; \tag{5}$$

$$\mathbf{b} = H\mathbf{1} = \left[b_1, b_2, ..., b_N \right]^T , \; b_i = \sum_{j=1}^{N} h_{i,j} . \tag{6}$$

В результате сомножитель $\mathbf{b}^T \mathbf{z}_k$ будет скалярным комплексированным входным воздействием фильтра.

После подстановки в формулу (4) значения $\hat{x}_k$ из (3) находим окончательное выражение для оценки $\hat{a}_k$:

$$\hat{a}_k = \frac{\alpha\left(1+h_0^2\right)}{\alpha\left(1+h_0^2\right)+h_0^2\hat{x}_{k-1}^2}\hat{a}_{k-1} + \frac{\hat{x}_{k-1}}{\alpha\left(1+h_0^2\right)+h_0^2\hat{x}_{k-1}^2}\mathbf{b}^T\mathbf{z}_k . \tag{7}$$

Выражения (3) и (7) являются рекуррентными уравнениями, в которых значения $\hat{x}_{k-1}$ и $\hat{a}_{k-1}$ определены на предыдущем шаге решения. Они отображают адаптивный характер фильтрации, поскольку их параметры не зависят от априорных сведений о модели полезного сигнала и характеристик шумов, а определяются на основе текущих измерений $\mathbf{z}_k$. Уравнение для коэффициента $\hat{a}_k$ автономно, поэтому результаты его предварительного решения используются при решении уравнения (3).

Учет комплексирования происходит посредством коэффициента h_0^2 и вектора $\mathbf{b}$, связанных с параметрами матрицы датчиков соотношениями (5) и (6).

Уравнения фильтра (3) и (7) получены в предположении отсутствия ограничений. В то же время из структуры уравнения (3) следует, что его решение будет устойчивым, если выполняются ограничения $0 < \dfrac{\hat{a}_k}{1+h_0^2} < 1$. Отсюда получаем ограничение на решение уравнения (7):

$$0 < \hat{a}_k < 1 + h_0^2 , \tag{8}$$

которое приводит к тому, что фильтр становится, с одной стороны, нелинейным, а с другой – квазиоптимальным, поскольку ограничение использовано после минимизации критерия (2), а не в ее процессе.

Решение уравнения (7) будет устойчивым при выполнении неравенства $0 < \dfrac{\alpha\left(1+h_0^2\right)}{\alpha\left(1+h_0^2\right)+h_0^2\hat{x}_{k-1}^2} < 1$ для любой величины эмпирического коэффициента регуляризации α в интервале от нуля до бесконечности. Его величина влияет на ошибку фильтрации и первоначально может быть принята равной единице. Дальнейшее ее уточнение проводится путем моделирования с оценкой точности фильтрации.

3. Сравнение адаптивного фильтра с фильтром Калмана

Качество работы синтезированного квазиоптимального адаптивного фильтра определяется величиной функционала потерь (2). Однако при статистическом подходе качество фильтрации оценивается величиной дисперсии ошибки фильтрации. В линейном случае оптимальным по этому критерию является фильтр Калмана, обеспечивающий минимальные значения дисперсий ошибок фильтрации. Сравнение дисперсий ошибок фильтра Калмана и адаптивного фильтра представляет интерес при заданной модели полезного сигнала и статистических харак-

теристиках помех, что дает возможность оценки потерь из-за априорной неопределенности при переходе от статистического подхода к детерминистскому.

Проведем такое сравнение для следующего конкретного случая. Пусть формирование полезного сигнала x_k осуществляется посредством модели в виде стохастического конечно-разностного уравнения первого порядка:

$$x_k = a_x x_{k-1} + w_k, \tag{9}$$

где постоянный параметр модели $0 < a_x < 1$, а w_k есть дискретный белый шум с нулевым математическим ожиданием и постоянной дисперсией σ_w^2. Можно показать, что в установившемся режиме параметры модели (9) связаны с дисперсией сигнала x_k соотношением [7]

$$\sigma_w^2 = \sigma_x^2 \left(1 - a_x^2\right). \tag{10}$$

Параметр a_x связан с длительностью корреляции τ_x, которая определяется шириной прямоугольной функции, аппроксимирующей корреляционную функцию, соотношением

$$a_x = \frac{\tau_x}{\tau_x + 1}. \tag{11}$$

Полагаем, что полезный сигнал x_k измеряется посредством N одинаковых, независимых друг от друга датчиков с коэффициентами чувствительности $h_{i,i} = h$, т. е. $N \times N$ матрица наблюдений $H = hI$, где I – единичная матрица. В этом случае коэффициент (5) принимает значение

$$h_0^2 = Nh^2,$$

а вектор (6) имеет вид

$$\mathbf{b} = h\mathbf{1}.$$

Скалярное комплексированное входное воздействие фильтра с учетом (1) и (6)

$$\mathbf{b}^T \mathbf{z}_k = x_k h_0^2 + h v_{0,k}, \tag{12}$$

где скалярный комплексированный шум наблюдения на входе фильтра равен сумме

$$v_{0,k} = \sum_{i=1}^{N} v_{i,k}.$$

Предположим, что составляющие $v_{i,k}$ вектора шума являются дискретными белыми шумами с нулевыми математическими ожиданиями и одинаковой дисперсией σ_v^2. Тогда дисперсия шума $v_{0,k}$ определяется выражением

$$\sigma_{0,v}^2 = N\sigma_v^2. \tag{13}$$

Дисперсия σ_v^2 шума наблюдений в каждом канале фильтра определяется исходя из отношения сигнала к шумам $q = \sigma_x / \sigma_v$ при заданных значениях q и σ_x.

Для рассматриваемого примера выражения (3) и (7) принимают следующие значения:

$$\hat{x}_k = \frac{\hat{a}_k}{1 + h_0^2} \hat{x}_{k-1} + \frac{1}{1 + h_0^2} \mathbf{b}^T \mathbf{z}_k; \tag{14}$$

$$\hat{a}_k = \frac{\alpha\left(1+h_0^2\right)}{\alpha\left(1+h_0^2\right)+h_0^2\hat{x}_{k-1}^2}\hat{a}_{k-1} + \frac{\hat{x}_{k-1}}{\alpha\left(1+h_0^2\right)+h_0^2\hat{x}_{k-1}^2}\mathbf{b}^T\mathbf{z}_k. \qquad (15)$$

Уравнение (15) должно решаться с учетом ограничений (8).

Дисперсия σ_e^2 ошибки фильтрации $e_k = x_k - \hat{x}_k$ в силу нелинейного характера уравнений фильтра находится путем моделирования с получением результатов методом статистических испытаний. Ниже приводятся результаты оценки точности фильтрации в установившемся режиме при следующих параметрах: $\alpha = 1$; $\sigma_x^2 = 400$; $\tau_x = 10, 20, 50$; $q = 3, 5, 10$.

Усреднение проводилось по данным 10^3 испытаний.

На рис. 1 изображен фрагмент реализации входного сигнала (штрихпунктирная линия), а также результаты обработки фильтром Калмана (штриховая линия) и адаптивным фильтром наименьших квадратов (сплошная линия).

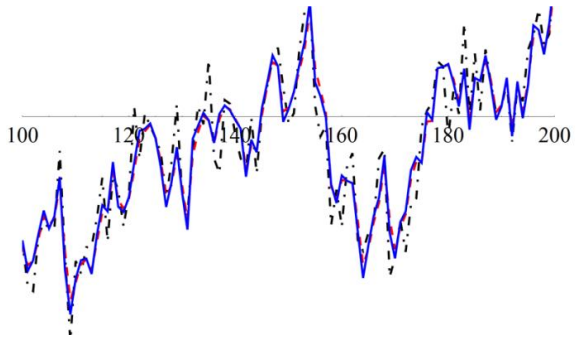
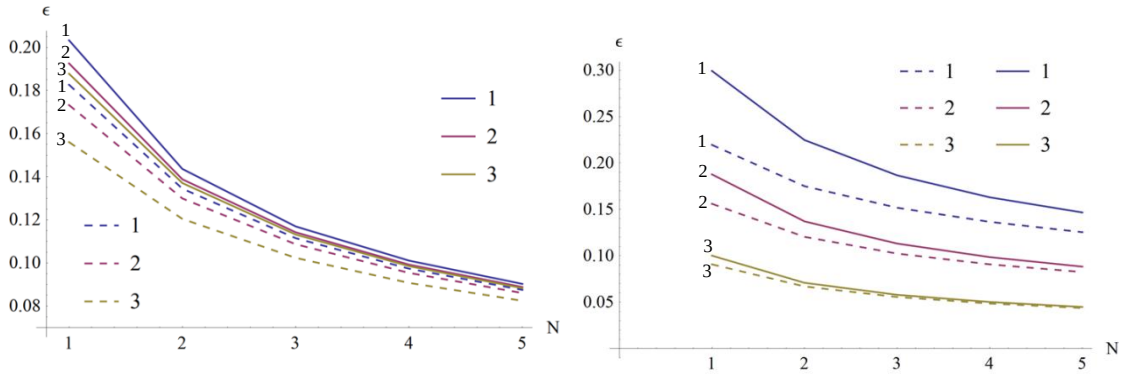


Рис. 1. Фрагмент реализации результатов фильтрации при $\tau_x = 50,\ q = 5$

На рис. 2, *а* сплошными линиями показана зависимость относительной величины среднеквадратического отклонения (СКО) ошибки фильтрации $\varepsilon = \sigma_e/\sigma_x$ от числа каналов N измерений входного сигнала при различных значениях длительности корреляции τ_x при $q = 5$, а на рис. 2, *б* – зависимость ε от N при различных значениях q и $\tau_x = 50$.



а) *б)*

Рис. 2. Относительные значения СКО ошибок фильтрации: *а)* при различных длительностях корреляции: 1) $\tau_x = 10$, 2) $\tau_x = 20$, 3) $\tau_x = 50$ и $q = 5$; *б)* при различных отношениях сигнала к шумам: 1) $q = 3$, 2) $q = 5$, 3) $q = 10$ и $\tau_x = 50$

На рис. 2 штриховыми линиями показаны результаты для фильтра Калмана, которые могут служить нижней границей ошибок. Увеличение ошибки фильтрации адаптивного фильтра по сравнению с этим случаем вызвано неучетом априорной информации. Исследования показывают, что выбор коэффициента $\alpha = 1$ обеспечивает ошибку фильтрации, близкую к минимальной.

Заключение

Полученные результаты позволяют утверждать, что адаптивные фильтры наименьших квадратов при комплексированных измерениях могут быть использованы при отсутствии априорной информации о моделях полезных сигналов и шумов наблюдений. В этом смысле данные фильтры универсальны по сравнению с фильтром Калмана и могут иметь широкое применение. Однако неучет априорной информации приводит к увеличению ошибок фильтрации по сравнению с фильтром Калмана. В то же время нахождение параметров адаптивного фильтра наименьших квадратов менее трудоемко, чем у фильтра Калмана, что упрощает их реализацию в реальном масштабе времени.

Список литературы

1. Справочник по теории автоматического управления / под. ред. А.А. Красовского. – М. : Наука, 1987. – 711 с.

2. Степанов, А.О. Основы теории оценивания с приложениями к задачам обработки навигационной информации : в 3 ч. / А.О. Степанов. – СПб. : ГНЦ РФ ЦНИИ «Электроприбор», 2009. – Ч. 1. : Введение в теорию оценивания. – 496 с.

3. Veen, van B.D. Beam forming: A versatile approach to spatial filtering / B.D. van Veen, K.M. Bukley // IEEE Acoust., Speech, Signal Processing Mag. – 1988. – Vol. 5. – P. 4–24.

4. A Fast Least-Squares Algorithms for Linearly Constrained Adaptive Filtering / G.B. Maurice [et al.] // IEEE Trans. Signal Processing. – 1996. – Vol. 44, no. 5. – P. 1168–1174.

5. Адаптивные фильтры / под. ред. П.Н. Гранта, К.Ф. Коуэна. – М. : Мир, 1988. – 388 с.

6. Линник, Ю.В. Метод наименьших квадратов и основы математико-статистической теории обработки изображений / Ю.В. Линник. – М. : Физматгиз, 1962. – 349 с.

7. Артемьев, В.М. Оптимальная линейная совмещенная фильтрация случайных последовательностей на основе рекуррентного метода наименьших квадратов / В.М. Артемьев, А.О. Наумов, Л.Л. Кохан // Информатика. – 2015. – № 1. – С. 8–16.

Поступила 15.06.2016

Институт прикладной физики
НАН Беларуси,
Минск, Академическая, 16
e-mail: naumov@iaph.bas-net.by

V.M. Artemiev, A.O. Naumov, L.L. Kokhan

ADAPTIVE FILTERING THE MULTISENSORY MEASUREMENTS BY LEAST-SQUARE METHOD

The least-square methodology is given consider adaptive filtering for multisensory measurements in case of uncertain prior knowledge about signal models and measurement noise parameters. The accuracy comparison with Kalman filtering is presented.

ЗАЩИТА ИНФОРМАЦИИ

УДК 621.9.048.7

В.Л. Пивоваров, В.Ф. Голиков

СПОСОБ ФОРМИРОВАНИЯ ОБЩЕГО КРИПТОГРАФИЧЕСКОГО КЛЮЧА ДЛЯ СЛАБО СОВПАДАЮЩИХ БИНАРНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Рассматривается способ формирования общей секретной бинарной последовательности по открытому каналу связи. В этом способе не используются известные однонаправленные функции; он сводится к итерационному устранению несовпадающих битов в исходных бинарных последовательностях абонентов, изготовленных специальным образом с определенным процентом несовпадений. Предлагается метод криптоанализа данного способа, основанный на наличии отклонения априорного распределения вероятностей инвертирования битов в исходных бинарных последовательностях абонентов от равномерного. Показывается, что часть битов в итоговой секретной последовательности может быть определена с высокой вероятностью.

Введение

В работе [1] и патенте [2] предложен способ формирования идентичных бинарных последовательностей (БП) без применения односторонних функций, который может быть использован для решения задачи распределения ключей криптографической системы между двумя абонентами, имеющими не защищенный от прослушивания канал связи и не обладающими общим секретом. В основе данного способа лежат следующие теоретические предпосылки [3].

Если абоненты A и B независимо друг от друга сгенерируют случайные БП X_A и X_B и сравнят их, то количество несовпадающих битов (в дальнейшем – ошибок) есть величина случайная с математическим ожиданием, равным 50 %. При этом чем больше размер БП, тем меньше среднее отклонение процента ошибок от 50 %. Известно [4], что при 50 % ошибок их невозможно удалить (или исправить), не раскрывая все совпадающие биты. Вместе с тем в работах [3, 4] предлагаются методы устранения ошибок для подобных задач при небольшом количестве (1–2 %) ошибок. В [5, 6] показано, что эффективность этих методов резко падает с ростом процента ошибок.

В работе [1] показано, что если удастся создать две БП, процент ошибок в которых гарантированно не равен 50 % (либо меньше, либо больше хотя бы на 1–2 %), то существующие ошибки можно удалить, «пожертвовав» при этом значительным количеством совпадающих битов. Такие БП названы слабо совпадающими. При согласовании этих БП потери совпадающих секретных битов очень большие. Поэтому предлагается использовать исходные БП, длина которых на несколько порядков больше длины общей итоговой БП. Расчеты, проведенные в [1], показывают, что если выбрать длину исходной БП 10^6 битов, то можно сформировать итоговую БП размером около 60 битов. Однако наиболее проблемным в этой задаче является вопрос получения исходных БП с указанными свойствами.

В [1] исходные БП формируются абонентами из несекретной общей БП путем независимого секретного инвертирования в ней битов в количестве примерно 50 %. Действительно, если в несекретной БП секретно инвертировать ровно половину битов, выбранных случайным образом, то такая БП превращается в секретную, так как в ней любой бит известен с вероятностью 0,5. Для того чтобы сформированная таким образом БП оставалась слабо совпадающей, предложено в одной БП инвертировать чуть менее половины битов, а в другой – чуть более.

1. Алгоритм формирования секретной БП

В целом рассматриваемый алгоритм формирования секретной БП содержит следующие шаги:

1. Абонент A генерирует базовую БП X_0 длиной n и посылает ее абоненту B.

2. Затем абоненты A и B задаются количеством инвертируемых битов r_A и r_B, где $0 \le r_A \le n$, $0 \le r_B \le n$, и генерируют независимо друг от друга случайные секретные последовательности чисел соответственно S_A и S_B. При этом $S_A = \{S_1^a, S_2^a, \dots, S_{r_a}^a\}$, $S_B = \{S_1^b, S_2^b, \dots, S_{r_b}^b\}$, где $S_i^a \in \{1, 2, \dots, n\}$, $S_i^b \in \{1, 2, \dots, n\}$, $S_i^a \ne S_j^a$, $i, j = 1, 2, \dots, r_A$ для S_A, $S_i^b \ne S_j^b$, $i, j = 1, 2, \dots, r_b$ для S_B.

3. Далее абоненты A и B в соответствии с полученными номерами битов S_i^a и S_i^b инвертируют эти биты в X_0 и получают последовательности X_A и X_B.

После того как сформированы последовательности X_A и X_B, абоненты A и B выполняют следующие шаги:

4. Согласованно разбивают свои последовательности X_A и X_B на пары битов либо по порядку, либо случайным образом.

5. Вычисляют четности каждой пары битов и обмениваются ими.

6. Сравнивая четности пар своей БП с полученными четностями, находят пары с несовпадающими четностями и удаляют их из последовательностей.

7. В оставшихся парах удаляют по одному биту по договоренности.

8. Из оставшихся битов, количество которых n_1, образуют промежуточные последовательности X_{A1} и X_{B1} путем сдвига всех бит влево до полного заполнения образовавшихся после удаления пар и битов вакансий.

9. Повторяют шаги 4–8 назначенное число раз (N) до получения X_{AN} и X_{BN}.

Полученные последовательности X_{AN} и X_{BN} и будут формировать криптографический ключ.

В работе [1] также проводится исследование для выбора оптимальных значений r_A и r_B с целью обеспечения доли несовпадающих битов, близкой к 50 %. Так, для n = 100 000 оптимальными значениями будут r_A= 43 800, r_B= 38 700 либо r_A = 38 700, r_B= 43 800. Такие значения r_A и r_B обеспечат долю несовпадающих битов, равную примерно 0,48.

Поскольку вероятности, с которыми каждый бит в исходных БП равен или противоположен соответствующему биту базовой последовательности, отличаются от 0,5, то это свойство может быть использовано для криптоанализа.

2. Анализ обнаруженной уязвимости

Будем считать, что все процедуры формирования итоговой последовательности известны криптоаналитику, как и вся информация, циркулирующая по открытому каналу связи.

На этапе формирования исходных БП криптоаналитику известна базовая последовательность X_0; на этапе удаления несовпадающих битов – номера битов исходной последовательности, вошедших в ту или иную пару, четности пар битов абонентов A и B, номера битов в парах, остающихся в БП после сравнения четностей, номера битов, удаленных из каждой неудаленной пары. Таким образом, криптоаналитику, пассивно наблюдающему процесс формирования идентичных БП, известны номера битов X_0, вошедших в итоговую БП, а также их новые позиции в этой последовательности. Однако значения этих битов криптоаналитику неизвестны, поскольку часть из них подвергалась инвертированию. Доля инвертированных битов зависит от выбранных значений r_A и r_B. При значениях r_A и r_B, обеспечивающих долю несовпадающих битов примерно 0,48, доля инвертированных битов у абонента A равняется в среднем примерно 0,387 и является случайной величиной. Следовательно, в итоговой последовательности в каждой сотне битов содержится в среднем 38 битов, значения которых противоположны этим же битам в X_0, и 62 бита, значения которых равны значениям этих же битов в X_0, но позиции совпадающих и противоположных битов криптоаналитику неизвестны. Таким образом, для раскрытия итоговой последовательности криптоаналитику требуется определить, какие биты в итоговой последовательности инвертированы относительно исходной, а какие – нет. Для этого как раз и предлагается использовать свойство, что вероятности, с которыми каждый бит итоговой последовательности равен или противоположен соответствующему биту исходной последовательности, отличаются от 0,5.

Априори криптоаналитику для каждой пары базовых БП известны четности и вероятности инвертирования битов. Значения этих вероятностей до начала устранения шибок близки к 0,5, но могут уточняться по мере получения информации о четности пар битов на каждом шаге алгоритма. Для этого можно использовать формулу условной вероятности Байеса

$$P(H_i/A) = \frac{P(H_i)P(A/H_i)}{\sum P(H_i)P(A/H_i)}, \quad (1)$$

которая позволяет уточнять вероятности событий после получения результатов опытов.

Пусть X_0 – базовая БП длиной n; X_A и X_B – сформированные БП на основе X_0; r_A, r_B – количество инвертируемых битов для X_A и X_B соответственно; C_0 – значение четности пары битов для последовательности X_0 (0 либо 1); C_1 – значение четности пары битов для последовательности X_A (0 либо 1) после инвертирования (на первом шаге). Тогда в формуле (1) рассматриваются гипотеза H (количество инвертированных битов в паре последовательности X_A), событие A (четность пары битов после инвертирования не изменилась, т. е. $C_0 = C_1$) и событие $\bar{A}$ (противоположное событию A).

Поясним указанные величины:

H_1 – гипотеза «оба бита в первой паре последовательности X_A были инвертированы относительно базовой последовательности X_0»;

H_2 – гипотеза «первый бит в первой паре последовательности X_A был инвертирован, второй остался таким же относительно базовой последовательности X_0»;

H_3 – гипотеза «второй бит в первой паре последовательности X_A был инвертирован, первый остался таким же относительно базовой последовательности X_0»;

H_4 – гипотеза «оба бита в первой паре последовательности X_A остались такими же относительно базовой последовательности X_0».

Априорные вероятности перечисленных гипотез следующие:

$$P(H_1) = p \times p;$$

$$P(H_2) = p \times (1-p);$$

$$P(H_3) = (1-p) \times p;$$

$$P(H_4) = (1-p) \times (1-p),$$

где $p = r_A/n$ – вероятность того, что какой-либо бит последовательности X_A был инвертирован.

Пусть после инвертирования и оглашения четностей для некой пары битов оказалось, что $C_0 = C_1$, т. е. имеет место событие A. Это возможно только в том случае, если в данной паре инвертированы 0 или 2 бита.

Следовательно, условные вероятности этого события при справедливости гипотез H_1 или H_4 определяются выражениями $P(A|H_1) = 1$, $P(A|H_4) = 1$. Очевидно, что событие $C_0 = C_1$ невозможно при справедливости гипотез H_2 или H_3, т. е. $P(A|H_2) = 0$, $P(A|H_3) = 0$. Тогда уточненные значения вероятностей выдвинутых гипотез H_1 и H_4 примут вид

$$P(H_1/A) = \frac{P(H_1)P(A/H_1)}{P(H_1)P(A/H_1) + P(H_2)P(A/H_2) + P(H_3)P(A/H_3) + P(H_4)P(A/H_4)} =$$

$$= \frac{P(H_1)P(A/H_1)}{P(H_1)P(A/H_1) + P(H_4)P(A/H_4)} = \frac{p^2}{p^2 + (1-p)^2};$$

$$P(H_4/A) = \frac{P(H_4)P(A/H_4)}{P(H_1)P(A/H_1) + P(H_2)P(A/H_2) + P(H_3)P(A/H_3) + P(H_4)P(A/H_4)} =$$

$$= \frac{P(H_4)P(A/H_4)}{P(H_1)P(A/H_1) + P(H_4)P(A/H_4)} = \frac{(1-p)^2}{p^2 + (1-p)^2},$$

соответствующие вероятности гипотез H_2 и H_3 окажутся равными $P(H_2/A) = 0, P(H_3/A) = 0$, а, следовательно, исключатся из дальнейшего рассмотрения. Таким образом, для следующего шага алгоритма априорные вероятности выдвинутых гипотез

$$P(H_1) = \frac{p^2}{p^2+(1-p)^2},\ P(H_2) = 0,\ P(H_3) = 0,\ P(H_4) = \frac{(1-p)^2}{p^2+(1-p)^2}.$$

Из полученных выражений видно, что если бы априорные вероятности гипотез H_1 и H_4 равнялись 0,5, то и апостериорные вероятности остались бы такими же. Отклонение их значений от равновероятных позволяет уточнять вероятности гипотез. Например, если $p = 0{,}45$, $1 - p = 0{,}55$, то уточненные вероятности гипотез H_1 и H_4 будут иметь значения $P(H_1) = 0{,}4$, $P(H_4) = 0{,}6$.

Пусть после инвертирования и оглашения четностей для некой пары битов оказалось, что $C_0 \neq C_1$. Такое событие возможно только в том случае, если в данной паре инвертирован 1 бит.

Следовательно, условные вероятности этого события при справедливости гипотез H_2 или H_3 имеют значения $P(\bar{A}|H_2) = 1$, $P(\bar{A}|H_3) = 1$. Очевидно, что событие $C_0 \neq C_1$ невозможно при справедливости гипотез H_1 или H_4, т. е. $P(\bar{A}|H_1) = 0$, $P(\bar{A}|H_4) = 0$. Тогда уточненные значения вероятностей выдвинутых гипотез H_1 и H_4 примут вид

$$P(H_2/\bar{A}) = \frac{P(H_2)P(\bar{A}/H_2)}{P(H_1)P(\bar{A}/H_1) + P(H_2)P(\bar{A}/H_2) + P(H_3)P(\bar{A}/H_3) + P(H_4)P(\bar{A}/H_4)} =$$

$$= \frac{P(H_2)P(\bar{A}/H_2)}{P(H_2)P(\bar{A}/H_2) + P(H_3)P(\bar{A}/H_3)} = \frac{p(1-p)}{p(1-p) + p(1-p)} = 0{,}5;$$

$$P(H_3/\bar{A}) = \frac{P(H_3)P(\bar{A}/H_3)}{P(H_1)P(\bar{A}/H_1) + P(H_2)P(\bar{A}/H_2) + P(H_3)P(\bar{A}/H_3) + P(H_4)P(\bar{A}/H_4)} =$$

$$= \frac{P(H_3)P(\bar{A}/H_3)}{P(H_2)P(\bar{A}/H_2) + P(H_3)P(\bar{A}/H_3)} = \frac{p(1-p)}{p(1-p) + p(1-p)} = 0{,}5.$$

Соответствующие вероятности гипотез H_1 и H_4 окажутся равными $P(H_1/\bar{A}) = 0$, $P(H_4/\bar{A}) = 0$, а следовательно, исключатся из дальнейшего рассмотрения. Поэтому для следующего шага алгоритма априорные вероятности выдвинутых гипотез

$$P(H_1) = 0,\ P(H_2) = 0{,}5,\ P(H_3) = 0{,}5,\ P(H_4) = 0.$$

Таким образом, для пары, имеющей $C_0 \neq C_1$, уточнение вероятностей гипотез H_2 и H_3 не происходит.

Проводя аналогичные вычисления на каждом шаге алгоритма формирования идентичных бинарных последовательностей (см. разд. 1, шаги 4–8), используя вместо C_i уже новые значения, полученные из последовательностей X_{A1}, X_{A2} и т. д., и вместо $P(H_i)$ – значения $P(H_i/A)$, полученные на предыдущем шаге, можно значительно уточнить вероятность, с которой биты в итоговой последовательности были инвертированы относительно битов в исходной последовательности на тех же позициях, и тем самым понизить конфиденциальность итоговой последовательности.

Рассмотрим результаты данного криптоанализа на конкретном примере. Пусть абонент *A* сформировал последовательность

X_0 = [1(1) 1(2) 0(3) 0(4) 0(5) 0(6) 0(7) 0(8) 0(9) 1(10) 1(11) 0(12) 1(13) 0(14) 0(15) 0(16) 1(17) 1(18) 0(19) 1(20) 1(21) 0(22) 0(23) 0(24) 0(25) 1(26) 1(27) 1(28) 1(29) 1(30) 1(31) 1(32) 1(33) 1(34) 1(35) 1(36) 0(37) 0(38) 0(39) 1(40) 1(41) 1(42) 0(43) 1(44) 0(45) 1(46) 1(47) 0(48) 0(49) 0(50) 0(51) 0(52) 1(53) 1(54) 0(55) 0(56) 1(57) 1(58) 0(59) 1(60) 1(61) 1(62) 1(63) 1(64) 1(65) 1(66) 1(67) 0(68) 1(69) 1(70) 1(71) 0(72) 0(73) 0(74) 0(75) 0(76) 1(77) 1(78) 1(79) 0(80) 1(81) 0(82) 0(83) 1(84) 0(85) 1(86) 0(87) 1(88) 0(89) 1(90) 0(91) 0(92) 1(93) 0(94) 0(95) 1(96) 1(97) 1(98) 0(99) 0(100) 1(101) 1(102) 0(103) 0(104) 1(105) 1(106) 0(107) 1(108) 0(109) 0(110) 1(111) 1(112) 0(113) 0(114) 1(115) 0(116) 1(117) 0(118) 1(119) 0(120) 0(121) 0(122) 1(123) 0(124) 1(125) 0(126) 1(127) 1(128) 1(129) 0(130) 0(131) 0(132) 1(133) 0(134) 0(135) 1(136) 1(137) 1(138) 0(139) 0(140) 0(141) 0(142) 1(143) 1(144) 1(145) 1(146) 1(147) 0(148) 1(149) 0(150) 0(151) 0(152) 1(153) 1(154) 0(155) 1(156) 0(157) 1(158) 1(159) 0(160) 1(161) 1(162) 0(163) 0(164) 0(165) 1(166) 0(167) 1(168) 0(169) 0(170) 1(171) 1(172) 0(173) 1(174) 0(175) 0(176) 1(177) 0(178) 1(179) 0(180) 0(181) 1(182) 1(183) 1(184) 0(185) 1(186) 1(187) 1(188) 0(189) 0(190) 1(191) 1(192) 1(193) 1(194) 0(195) 1(196) 0(197) 0(198) 0(199) 0(200) 0(201) 1(202) 0(203) 0(204) 1(205) 1(206) 1(207) 0(208) 0(209) 0(210) 1(211) 0(212) 1(213) 1(214) 0(215) 0(216) 0(217) 0(218) 1(219) 1(220)]

длиной 220 битов, r_A= 91, r_B = 84. После инвертирования битов абоненты получили следующие последовательности:

X_A = [0(1) 1(2) 0(3) 0(4) 0(5) 1(6) 0(7) 1(8) 0(9) 1(10) 1(11) 1(12) 1(13) 0(14) 1(15) 0(16) 1(17) 0(18) 0(19) 0(20) 0(21) 0(22) 1(23) 1(24) 1(25) 0(26) 1(27) 1(28) 0(29) 1(30) 0(31) 0(32) 1(33) 1(34) 1(35) 1(36) 0(37) 1(38) 0(39) 0(40) 0(41) 1(42) 0(43) 0(44) 0(45) 1(46) 0(47) 1(48) 0(49) 1(50) 1(51) 1(52) 1(53) 1(54) 1(55) 0(56) 1(57) 1(58) 0(59) 1(60) 1(61) 1(62) 1(63) 1(64) 1(65) 1(66) 1(67) 0(68) 0(69) 1(70) 0(71) 1(72) 0(73) 0(74) 1(75) 1(76) 1(77) 1(78) 1(79) 1(80) 1(81) 0(82) 0(83) 1(84) 0(85) 0(86) 0(87) 1(88) 0(89) 0(90) 0(91) 0(92) 1(93) 0(94) 0(95) 0(96) 1(97) 1(98) 0(99) 0(100) 1(101) 1(102) 0(103) 1(104) 0(105) 1(106) 1(107) 0(108) 0(109) 1(110) 1(111) 1(112) 1(113) 1(114) 1(115) 1(116) 1(117) 1(118) 0(119) 1(120) 0(121) 0(122) 0(123) 0(124) 0(125) 0(126) 1(127) 1(128) 1(129) 0(130) 1(131) 1(132) 1(133) 1(134) 0(135) 0(136) 1(137) 1(138) 0(139) 0(140) 1(141) 0(142) 1(143) 0(144) 1(145) 1(146) 0(147) 0(148) 1(149) 1(150) 1(151) 0(152) 0(153) 0(154) 0(155) 0(156) 1(157) 1(158) 1(159) 1(160) 0(161) 1(162) 0(163) 1(164) 0(165) 1(166) 0(167) 0(168) 1(169) 1(170) 0(171) 0(172) 0(173) 0(174) 0(175) 0(176) 0(177) 1(178) 0(179) 1(180) 1(181) 0(182) 1(183) 1(184) 0(185) 0(186) 0(187) 1(188) 0(189) 0(190) 0(191) 1(192) 0(193) 1(194) 1(195) 1(196) 0(197) 1(198) 0(199) 1(200) 0(201) 1(202) 1(203) 1(204) 1(205) 1(206) 0(207) 1(208) 0(209) 1(210) 1(211) 1(212) 0(213) 1(214) 0(215) 0(216) 0(217) 1(218) 0(219) 1(220)],

X_B = [0(1) 1(2) 0(3) 1(4) 1(5) 1(6) 0(7) 0(8) 1(9) 0(10) 1(11) 1(12) 1(13) 0(14) 1(15) 0(16) 1(17) 0(18) 0(19) 1(20) 1(21) 1(22) 1(23) 0(24) 0(25) 0(26) 1(27) 0(28) 1(29) 1(30) 1(31) 1(32) 1(33) 0(34) 1(35) 0(36) 0(37) 1(38) 0(39) 1(40) 1(41) 1(42) 0(43) 1(44) 1(45) 1(46) 1(47) 0(48) 0(49) 0(50) 1(51) 0(52) 1(53) 1(54) 1(55) 0(56) 0(57) 0(58) 1(59) 1(60) 0(61) 1(62) 0(63) 1(64) 0(65) 0(66) 1(67) 0(68) 0(69) 1(70) 0(71) 0(72) 0(73) 1(74) 0(75) 1(76) 0(77) 1(78) 0(79) 1(80) 0(81) 0(82) 1(83) 1(84) 1(85) 0(86) 0(87) 1(88) 0(89) 0(90) 1(91) 1(92) 1(93) 0(94) 0(95) 1(96) 1(97) 0(98) 1(99) 0(100) 1(101) 1(102) 0(103) 0(104) 1(105) 0(106) 1(107) 1(108) 1(109) 0(110) 0(111) 0(112) 1(113) 1(114) 1(115) 1(116) 0(117) 1(118) 1(119) 1(120) 0(121) 1(122) 0(123) 0(124) 1(125) 0(126) 0(127) 1(128) 0(129) 0(130) 1(131) 0(132) 1(133) 0(134) 0(135) 1(136) 0(137) 1(138) 0(139) 0(140) 1(141) 0(142) 1(143) 0(144) 1(145) 1(146) 1(147) 0(148) 1(149) 1(150) 1(151) 0(152) 1(153) 1(154) 1(155) 1(156) 0(157) 1(158) 1(159) 1(160) 1(161) 1(162) 0(163) 0(164) 1(165) 1(166) 0(167) 0(168) 0(169) 1(170) 1(171) 0(172) 1(173) 0(174) 0(175) 0(176) 1(177) 0(178) 1(179) 0(180) 0(181) 0(182) 0(183) 1(184) 1(185) 0(186) 0(187) 1(188) 1(189) 0(190) 1(191) 1(192) 1(193) 1(194) 0(195) 1(196) 0(197) 0(198) 0(199) 0(200) 0(201) 1(202) 1(203) 1(204) 1(205) 1(206) 0(207) 0(208) 0(209) 1(210) 1(211) 1(212) 1(213) 0(214) 0(215) 0(216) 0(217) 0(218) 0(219) 1(220)].

Для удобства каждый бит последовательностей X_0, X_A, X_B пронумеруем, биты в последовательностях будем разбивать на пары по порядку, а удалять из пары будем биты на первой позиции.

Далее по формуле (1) вычислим значения $P(H_1/A)$, $P(H_2/A)$, $P(H_3/A)$, $P(H_4/A)$ для каждой пары битов последовательности X_A.

Зная значения p, C_i, $P(H_1/A)$, $P(H_2/A)$, $P(H_3/A)$, $P(H_4/A)$, легко можно получить вероятности того, что второй бит в каждой паре равен 1 (0 – с противоположной вероятностью):

P = [0,5(2) 0,3322(4) 0,5(6) 0,5(8) 0,6677(10) 0,5(12) 0,3322(14) 0,5(16) 0,5(18) 0,5(20) 0,5(22) 0,3322(24) 0,6677(26) 0,6677(28) 0,5(30) 0,6677(32) 0,6677(34) 0,6677(36) 0,5(38) 0,5(40) 0,5(42) 0,5(44) 0,6677(46) 0,3322(48) 0,5(50) 0,3322(52) 0,6677(54) 0,5(56) 0,6677(58) 0,6677(60) 0,667723(62) 0,6677(64) 0,6677(66) 0,3322(68) 0,5(70) 0,3322(72) 0,3322(74) 0,3322(76) 0,6677(78) 0,5(80) 0,3322(82) 0,6677(84) 0,5(86) 0,6677(88) 0,5(90) 0,3322(92) 0,3322(94) 0,5(96) 0,6677(98) 0,3322(100) 0,6677(102) 0,5(104) 0,5(106) 0,6677(108) 0,5(110) 0,6677(112) 0,3322(114) 0,5(116) 0,5(118) 0,3322(120) 0,3322(122) 0,5(124) 0,5(126) 0,6677(128) 0,3322(130) 0,3322(132) 0,5(134) 0,5(136) 0,6677(138) 0,3322(140) 0,5(142) 0,5(144) 0,6677(146) 0,5(148) 0,5(150) 0,5(152) 0,6677(154) 0,5(156) 0,5(158) 0,5(160) 0,5(162) 0,5(164) 0,6677(166) 0,5(168) 0,3322(170) 0,6677(172) 0,5(174) 0,3322(176) 0,3322(178) 0,3322(180) 0,6677(182) 0,6677(184) 0,5(186) 0,5(188) 0,3322(190) 0,5(192) 0,5(194) 0,5(196) 0,5(198) 0,5(200) 0,6677(202) 0,3322(204) 0,6677(206) 0,3322(208) 0,5(210) 0,5(212) 0,5(214) 0,3322(216) 0,5(218) 0,5(220)].

Значения вычисляем только для вторых битов в каждой паре, так как первые биты на каждой итерации удаляются из последовательности.

Проделывая аналогичные вычисления на каждом шаге работы алгоритма, получаем следующие данные:

Шаг 1

X_{A1} = [1(2) 1(10) 1(12) 0(14) 0(16) 0(18) 0(22) 0(32) 1(38) 1(48) 1(54) 0(56) 1(58) 1(66) 0(68) 1(70) 1(88) 0(90) 0(92) 0(94) 1(102) 1(106) 1(110) 1(112) 1(114) 1(116) 0(124) 0(140) 0(142) 0(144) 1(146) 1(150) 0(152) 0(154) 0(156) 1(160) 0(168) 0(176) 1(178) 1(180) 1(188) 1(202) 1(204) 1(206) 1(210) 1(212) 1(214) 0(216)];

X_{B1} = [1(2) 0(10) 1(12) 0(14) 0(16) 0(18) 1(22) 1(32) 1(38) 0(48) 1(54) 0(56) 0(58) 0(66) 0(68) 1(70) 1(88) 0(90) 1(92) 0(94) 1(102) 0(106) 0(110) 0(112) 1(114) 1(116) 0(124) 0(140) 0(142) 0(144) 1(146) 1(150) 0(152) 1(154) 1(156) 1(160) 0(168) 0(176) 0(178) 0(180) 1(188) 1(202) 1(204) 1(206) 1(210) 1(212) 0(214) 0(216)];

P_1 = [0,6677(10) 0,3322(14) 0,5(18) 0,6677(32) 0,3322(48) 0,3322(56) 0,8015(66) 0,6677(70) 0,3322(90) 0,1984(94) 0,6677(106) 0,6677(112) 0,3322(116) 0,3322(140) 0,5(144) 0,6677(150) 0,6677(154) 0,5(160) 0,3322(176) 0,1984(180) 0,6677(202) 0,5(206) 0,5(212) 0,3322(216)].

Шаг 2

X_{A2} = [0(14) 0(18) 0(32) 0(56) 1(66) 1(70) 0(90) 1(112) 1(116) 0(140) 0(144) 1(150) 0(176) 1(180) 1(202) 1(206)];

X_{B2} = [0(14) 0(18) 1(32) 0(56) 0(66) 1(70) 0(90) 0(112) 1(116) 0(140) 0(144) 1(150) 0(176) 0(180) 1(202) 1(206)];

P_2 = [0,3322(18) 0,5(56) 0,8902(70) 0,8015(112) 0,5(140) 0,6677(150) 0,3322(180) 0,6677(206)].

Шаг 3

X_{A3} = [0(18), 0(140), 1(150), 1(206)];

X_{B3} = [0(18), 0(140), 1(150), 1(206)];

P_3 = [0,1322(18); 0,5(140); 0,8677(150); 0,8015(206)].

Алгоритм произвел полную синхронизацию последовательностей за три такта. За эти три такта удалось сильно подкорректировать вероятности того, что какой-либо бит был инвертирован в итоговой последовательности. Таким образом, можно утверждать, что бит на позиции 1 в итоговой последовательности (бит 18 в базовой последовательности) с вероятностью 0,1322 равен 1, бит на позиции 2 в итоговой последовательности (бит 140 в базовой последователь-

ности) с вероятностью 0,5 равен 1, бит на позиции 3 в итоговой последовательности (бит 150 в базовой последовательности) с вероятностью 0,8677 равен 1 и бит на позиции 4 в итоговой последовательности (бит 206 в базовой последовательности) с вероятностью 0,8015 равен 1. Для большинства битов удалось с высокой точностью определить их настоящее значение, однако есть биты, для которых вероятность того, что они равны 1, будет 0,5.

Заключение

Предложенный в настоящей работе криптоанализ позволяет с большой вероятностью раскрыть некоторые биты в итоговой последовательности, из-за чего анализируемый способ формирования идентичных БП требует дополнительных усилий для обеспечения секретности полученной последовательности.

Список литературы

1. Абдольванд, Ф. Устранение ошибок в бинарных последовательностях при формировании криптографического ключа без использования однонаправленных функций / Ф. Абдольванд, В.Ф. Голиков // Информационные системы и технологии : материалы VI Междунар. науч. конф., Минск, 24–25 нояб. 2010 г. – Минск : БГУ, 2010. – С. 34–37.

2. Способ распределения криптографического ключа между абонентами : пат. 17856 Респ. Беларусь : МПК 04L 9/08 (2006.01) / В.Ф. Голиков; опубл. 19.07.11 / Нац. центр интеллектуал. собственности. – 2011.

3. Брассар, Ж. Современная криптология / Ж. Брассар. – М. : Полимед, 1999. – 373 с.

4. Боумейстер, Д. Физика квантовой информации / Д. Боумейстер, А. Экерт, А. Цайлингер. – М. : Постмаркет, 2002. – 276 с.

5. Голиков, В.Ф. Оценка потерь конфиденциальности при неклассических способах формирования криптографического ключа / В.Ф. Голиков, Ф. Абдольванд // Информатика. – 2011. – № 2(30). – С. 104–110.

6. Голиков, В.Ф. Эффективность устранения ошибок в бинарных последовательностях при разнесенном формировании криптографического ключа / В.Ф. Голиков, Ф. Абдольванд // Доклады БГУИР. – 2010. – № 6(52). – С. 107–112.

Поступила 24.03.2016

Белорусский национальный
технический университет,
Минск, пр. Независимости, 65
e-mail: vadim.pif@gmail.com
vgolikov@bntu.by

V.L. Pivovarov, U.F. Holikau

METHOD OF GENERATING COMMON CRYPTOGRAPHIC KEYS FOR LOOSLY COINCIDENT BINARY SEQUENCES

The method of forming a common secret binary sequence between using an open communication channel is considered. The method is not based on common unidirectional functions and results in iterative elimination of distinct bits in the initial binary sequences with a certain percentage of mismatches, intentionally made by subscribers themselves. The cryptanalysis technique of this method based on the use of the deviation of aprior distribution of probabilities of inverting bits in the original binary sequences of subscribers from uniform distribution is proposed. Part of the bits in the final secret sequence can be identified accurately enough.

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ

УДК 004.056.53, 681.32

С.С. Заливако, А.А. Иванюк

ОБЗОР МЕТОДОВ АКТИВНОЙ ИДЕНТИФИКАЦИИ ЦИФРОВЫХ УСТРОЙСТВ

Рассматриваются существующие методы активной идентификации цифровых устройств и приводится обоснование необходимости использования таких методов компаниями – проектировщиками интегральных схем. Представляется описание методов, основанных на модифицирующих преобразованиях цифрового конечного автомата и протоколах асимметричного шифрования. Анализируются преимущества и недостатки методов активной идентификации и предлагаются пути решения проблем, актуальных в настоящее время.

Введение

Процесс производства современных интегральных схем (ИС) состоит из двух этапов: проектирования и изготовления. На первом этапе разработчик, как правило, при помощи системы автоматизированного проектирования (САПР) создает HDL-описание, производит RTL- и технологический синтез, связывание абстрактных компонентов с определенными физическими ресурсами кристалла, проектирование топологии ИС, размещение компонентов в логические ячейки, соединение компонентов внутренними трассировочными ресурсами, а также параметрическое моделирование созданного проектного описания. На втором этапе, получив проектное описание ИС, производитель осуществляет изготовление полупроводниковых пластин и их обработку, фотолитографию, разделение пластин на кристаллы, монтаж в корпус, герметизацию, электрические измерения, выходной контроль (тестирование), маркировку и упаковку. В результате получается готовая ИС, которая может быть самостоятельным цифровым устройством (ЦУ) либо системой на кристалле, а также использоваться как компонент более сложной системы.

В настоящее время стоимость средств производства ИС с использованием современных технологических процессов (от 45 нм и меньше) оценивается в несколько миллиардов долларов США [1]. В связи с этим большое количество компаний избрало горизонтальную бизнес-модель [2] для выхода на рынок полупроводниковых устройств. Данная тенденция привела к тому, что за последние 20 лет значительная часть продаж ИС (около 30 %) приходится на компании, не обладающие собственными производственными мощностями (рис. 1). Таким образом, горизонтальная модель ведения бизнеса позволяет небольшим компаниям – проектировщикам ИС конкурировать с такими крупными производителями, как Intel, Samsung, Sony, Texas Instruments и др. Другой важной тенденцией, обусловленной текущим состоянием рынка полупроводниковых устройств, является разработка IP-компонент [3] (Intellectual Property – IP) компаниями – поставщиками САПР.

В связи с тем что компании без собственных производственных мощностей доказали свою состоятельность и заняли значительную часть рынка, они, разумеется, столкнулись с новыми проблемами, связанными с нелегальным копированием их проектов ИС и подделкой изготовленных полупроводниковых устройств [4]. Законодательство не позволяет защитить права интеллектуальной собственности на разработанные такими компаниями IP-компоненты. Это отчасти привело к тому, что доля поддельных (или нелегально изготовленных) ИС составляет сегодня порядка 5 % [5], а также к ежегодным потерям порядка 4 млрд долл. Существует ряд критических приложений для ИС (системы для обработки персональных данных, медицинская электроника, вооружение и др.), для которых характерны более жесткие требования к неклонируемости, защите от нелегального копирования и обратного проектирования. Соответственно,

для перечисленных выше типов полупроводниковых устройств наличие данной проблемы представляет собой серьезную угрозу для безопасности.

Рис. 1. Объемы продаж компаний – производителей полупроводниковых устройств [1]

Одним из эффективных методов защиты проектных описаний от несанкционированного копирования и клонирования является идентификация аппаратного обеспечения (Hardware Metering), которая впервые была рассмотрена в работах [6, 7]. Под термином «идентификация» понимается считывание данных с аппаратного обеспечения, которое помогает распознать устройство и доказать принадлежность к обладателю прав интеллектуальной собственности на него. Изначально это понятие было определено для пассивной идентификации аппаратного обеспечения (Passive Hardware Metering) в качестве протокола безопасности, который позволяет компании-проектировщику осуществить распознавание изготовленного ЦУ. Особенностью активной идентификации аппаратного обеспечения (Active Hardware Metering) является возможность контроля ЦУ (ограничения или отключения некоторых его функций) после его изготовления, что позволяет решать проблемы, характерные для горизонтальной бизнес-модели.

1. Классификация методов идентификации аппаратного обеспечения

Методы идентификации аппаратного обеспечения (рис. 2) в настоящее время делятся на два больших класса: пассивные и активные [8]. Методы пассивной идентификации позволяют уникально идентифицировать IP-компоненты, входящие в состав ЦУ, или ЦУ целиком. Термин «пассивный» означает, что владелец прав на проектное описание может установить, является ли ЦУ подлинным, но не может изменить или в дальнейшем контролировать его функциональность. Методы пассивной идентификации, в свою очередь, также подразделяются на два подкласса: пассивные функциональные и нефункциональные. Нефункциональные методы основаны на уникальной идентификации ЦУ, которая не зависит от его функциональности. Вместе с тем функциональные методы генерируют уникальный идентификатор, в основе которого лежит определенная функция ЦУ. Оба класса методов могут использовать как воспроизводимые, так и неклонируемые (невоспроизводимые) идентификаторы.

В отличие от пассивной идентификации методы активной идентификации предоставляют владельцу прав собственности на ЦУ или его проектное описание возможность включения или отключения функциональности ЦУ для предотвращения его несанкционированного использования. В соответствии с определением методы активной идентификации похожи на методы пассивной функциональной идентификации, поскольку используют некую функцию ЦУ в качестве блокирующей схемы, которая позволяет ЦУ работать в обычном или ограниченном (неактивном) режиме. Особенностью блокирующей схемы в методах активной идентификации является то, что она проектируется зависимой от идентификатора ЦУ и, следовательно, попытки изменения значения идентификатора приведут к нарушению функционирования ЦУ. Методы активной идентификации, в свою очередь, подразделяются на два подкласса: закрытые

и открытые. В закрытых методах для построения блокирующей схемы применяются только средства той IP-компоненты, которую планируется защитить. В открытых методах совместно с внутренними средствами используются также дополнительные (внешние) модули или устройства.

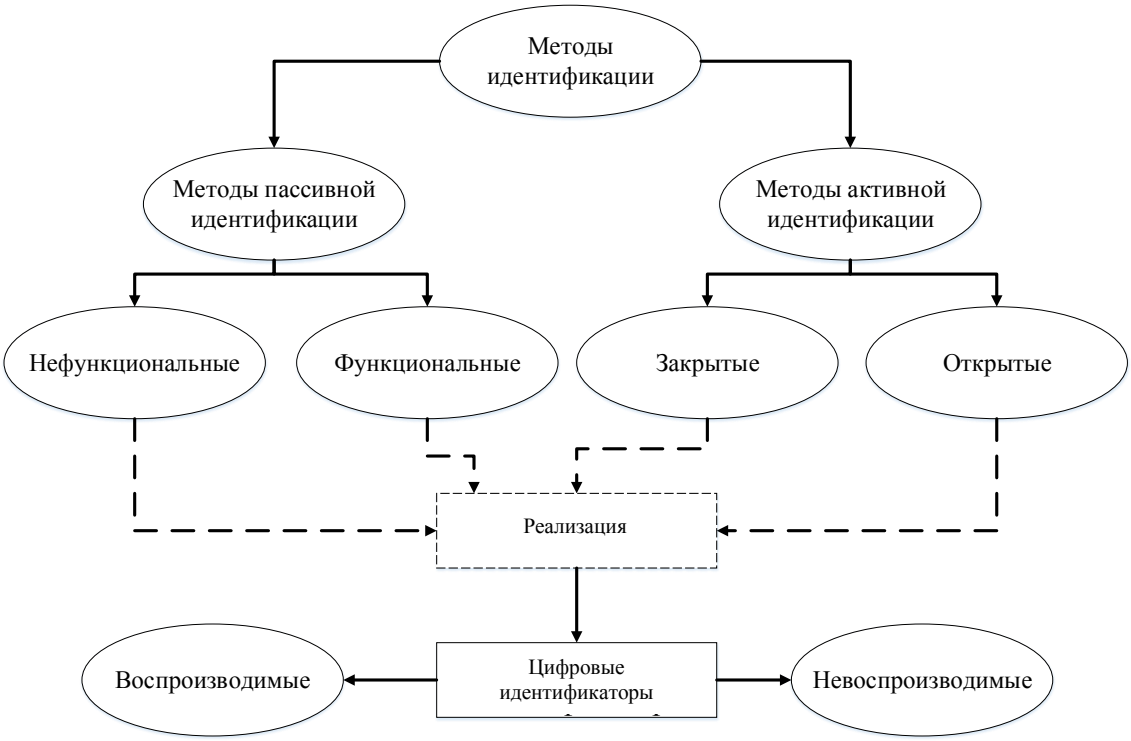


Рис. 2. Классификация методов идентификации аппаратного обеспечения

Рассмотрим подробнее методы активной идентификации.

2. Методы активной идентификации цифровых устройств

Основной особенностью методов активной идентификации является возможность активного контроля над ЦУ после изготовления. Благодаря данной особенности все методы активной идентификации применяют два базовых компонента для практической реализации: блокирующую схему и источник энтропии для генерирования секретных криптографических ключей, которые обладают свойствами уникальности, неклонируемости, случайности, а также не изменяются при внешнем воздействии на одном и том же ЦУ. Текущие реализации методов активной идентификации в качестве блокирующей схемы используют либо модифицированный цифровой конечный автомат (ЦКА) [9–11], либо схему на базе комбинационной логики [12]. Применение физически неклонируемых функций (ФНФ) [13] позволяет генерировать уникальные, неклонируемые, невоспроизводимые и надежные идентификаторы ЦУ, которые могут быть использованы в качестве секретных ключей. Блокирующая схема проектируется зависимой от ключей, генерируемых ФНФ, что позволяет ей быть уникальной для каждого ЦУ. Таким образом, даже обладание секретной информацией для разблокирования одного ЦУ не даст злоумышленнику существенных преимуществ для взлома другого ЦУ, изготовленного по тому же проекту.

Общая схема активной идентификации ЦУ изображена на рис. 3 [9], в качестве блокирующей схемы используется ЦКА, а в качестве источника энтропии – ФНФ. Как правило, модель активной идентификации включает две стороны, взаимодействующие между собой: компанию, проектирующую ЦУ и владеющую правами интеллектуальной собственности на

проект, и компанию-изготовитель, которой передается необходимое для производства ИС проектное описание.

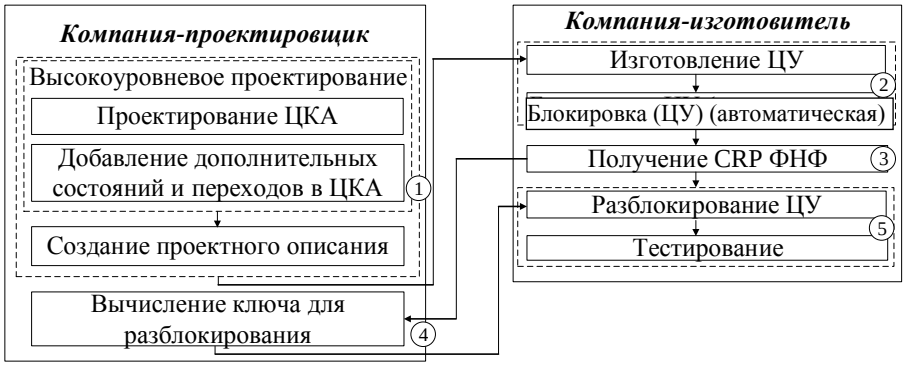


Рис. 3. Общая схема функционирования методов активной идентификации

Рассмотрим этапы проектирования и изготовления ЦУ:

1. Проектировщик создает начальное HDL-описание и проектирует структуру ЦКА как основу для блокирующей схемы. Далее разработчик расширяет структуру исходного ЦКА с помощью добавления фиктивных состояний и (или) переходов. Таким образом, получается расширенный ЦКА (РЦКА), который и является блокирующей схемой. После этого в результате синтеза создается готовое проектное описание, которое отправляется компании – изготовителю ЦУ.

2. Компания-изготовитель производит заказанное проектировщиком число копий ЦУ, каждое из которых содержит неклонируемый идентификатор, реализованный, как правило, с помощью слабой ФНФ совместно с кодами коррекции ошибок [14]. Для повышения надежности ФНФ эталонные пары «запрос – ответ» (Challenge Response Pairs – CRP) сохраняются заранее в энергонезависимой памяти, а затем используются для корректировки нестабильных ответов. На этом этапе возникает риск, что производитель, зная проектное описание ЦУ, может произвести большее число нелегальных копий. Именно поэтому функциональность изготовленного ЦУ изначально находится в заблокированном состоянии и разблокировать его может только обладатель прав интеллектуальной собственности на проектное описание, т. е. компания-проектировщик.

3. Далее производитель извлекает информацию о CRP с помощью изготовленного ЦУ и отправляет их компании-проектировщику. Поскольку структура РЦКА зависит от ФНФ, функциональность ЦУ может быть восстановлена тогда и только тогда, когда известны и структура РЦКА, и множество CRP ФНФ.

4. Проектировщик вычисляет лицензионный ключ, который способен восстановить функциональность ЦУ на основании CRP, предоставленных производителем. Вычисленный ключ применим только к конкретному ЦУ, поскольку информация о CRP не может быть воспроизведена на другом ЦУ, реализованном по такому же проекту, и, следовательно, является уникальной и неклонируемой.

5. На последнем этапе компания-изготовитель осуществляет разблокировку готового ЦУ с использованием лицензионного ключа, предоставленного проектировщиком, а также тестирование функциональности ЦУ с целью подтверждения его работоспособности.

Таким образом, методы активной идентификации дают возможность предотвратить нелегальное копирование проектных описаний ЦУ сторонней компанией-изготовителем.

2.1. Закрытые методы активной идентификации

Особенность закрытых методов активной идентификации заключается в том, что блокирующая схема формируется с помощью создания фиктивных состояний и (или) переходов ЦКА [9, 10].

Пусть первоначально спроектированный ЦКА содержит M состояний, что может быть реализовано с использованием как минимум $K = \lceil \log_2(M) \rceil$ триггеров. Тогда для формирования РЦКА требуется добавить в исходный ЦКА дополнительно M_1 состояний, что приведет к суммарному расходу $K_2 = \lceil \log_2(M + M_1) \rceil$ триггеров. Добавочные состояния должны быть дополнены переходами, которые сохранят связность графа, соответствующего РЦКА, т. е. каждое из добавленных состояний должно быть достижимо из уже имеющихся в ЦКА. Таким образом, число добавочных триггеров может быть вычислено как $K_1 = K_2 - K$. Отметим, что экспоненциальный рост числа состояний в РЦКА вызовет только линейный рост числа триггеров, поэтому можно добавить столько состояний, чтобы $M_2 >> M$.

ЦУ должно иметь в своем составе реализацию ФНФ для генерирования кода случайного начального состояния, который является ответом на заранее определенный запрос, формируемый компанией-проектировщиком в виде тестового вектора. Поскольку для реализации РЦКА требуется K_2 триггеров, то размерность ответа ФНФ должна составлять K_2 бит.

После инициализации ФНФ генерирует код начального состояния в РЦКА (рис. 4). На рисунке состояния первоначально спроектированного (оригинального) ЦКА обозначены черным цветом, а добавочные состояния – белым. Кодирование состояний РЦКА осуществляется с помощью битовой карты, номера бит в которой определяются случайным образом компанией-проектировщиком: биты, обозначенные на рис. 4 черным цветом, предназначены для кодирования состояний первоначального ЦКА, а белым – для кодирования остальных (добавочных) состояний соответственно. Все добавочные состояния таковы, что если пользователь не знает структуры РЦКА, то он не может выйти из этих состояний и достичь начального состояния (S_0) оригинального ЦКА. При этом состояния S_0 можно достичь тогда и только тогда, когда код состояния S_x окажется среди кодов состояний первоначально спроектированного ЦКА, однако вероятность такого события крайне мала: $P_g = 2^{-K_1}$ и $2^{K_2} >> 2^K$. Следовательно, $2^{-K_1} \approx 0$.

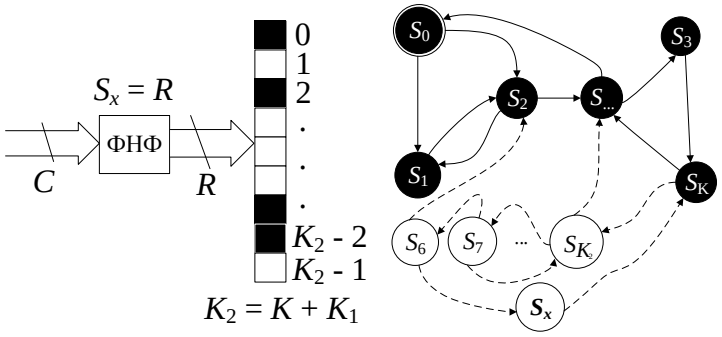


Рис. 4. Схема активной идентификации с использованием избыточных состояний (S_x обозначает начальное состояние, $0 \le x \le K_2$)

РЦКА проектируется таким образом, чтобы путь из каждого добавочного состояния в состояние S_0 был единственным и, соответственно, осуществление его подбора было бы затруднено. Следовательно, злоумышленник не сможет восстановить функциональность ЦУ без знания реальной структуры РЦКА, которая является секретной и не передается производителю на этапе 2 (см. общую схему методов активной идентификации). Таким образом, путь из состояния S_x в S_0 (последовательность кодов переходов) и является ключом, который уникален для каждого изготовленного ЦУ, поскольку выбор начального состояния РЦКА (S_x) полностью зависит от кода, сгенерированного ФНФ.

Другой закрытый метод активной идентификации, основанный на применении ЦКА в качестве блокирующей схемы, был предложен в работе [11]. В отличие от метода, описанного выше, генерируются не дополнительные состояния ЦКА, а копии, дублирующие некоторые существующие состояния с соответствующими переходами. Таким образом, использование описываемого метода влечет за собой меньшие затраты на генерирование дублирующих состояний, но усложняет механизм работы с ФНФ.

Пусть некоторое состояние S_i выбрано для дублирования и создана копия S_{ij}. Тогда все переходы из состояния S_i и в него также дублируются для S_{ij}. Как и в предыдущем методе, в качестве источника энтропии была выбрана ФНФ, запрос для которой (C) является кодом состояния S_i, а часть ответа (i_c) используется в качестве кода перехода в копию этого состояния (S_{ij}). В свою очередь, вторая часть ответа ФНФ (seed) применяется как инициализирующее значение в блоке коррекции, чтобы сгенерировать код перехода, соответствующего выходу из состояния S_{ij} (o_c). Допустим, начальным состоянием было выбрано S_i, тогда после инициализации ЦУ ФНФ сгенерирует код перехода в состояние S_{ij}, код выхода из которого известен только проектировщику. Таким образом, для выхода из этого состояния требуется сгенерировать код выходного перехода, алгоритм построения которого основан на второй части ответа ФНФ и ключе (key) (рис. 5), т. е. $o_c = h(\text{seed}, \text{key})$, где h – хеш-функция, реализуемая блоком коррекции. Только знание ответа ФНФ и секретного ключа позволит сгенерировать код перехода, позволяющего выйти из состояния-копии и таким образом вернуть работоспособность ЦУ.

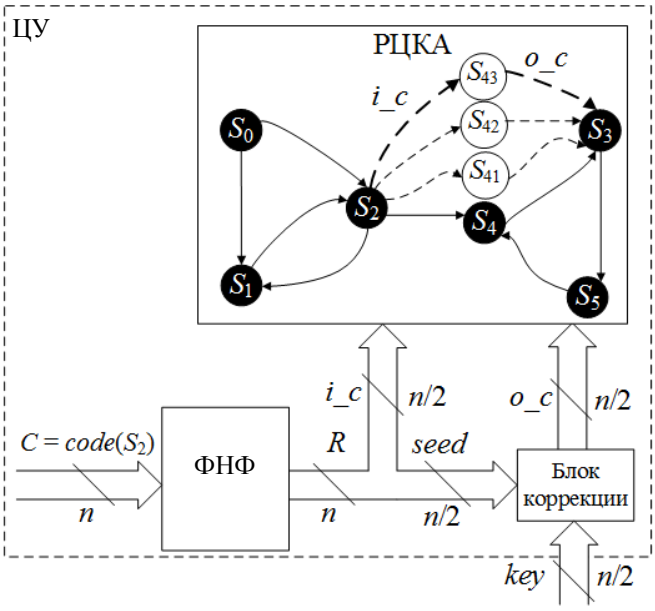


Рис. 5. Схема активной идентификации с использованием копий исходных состояний

Для пояснения описанной выше схемы приведем пример с использованием конкретных числовых значений. Допустим, что в ФНФ поступил запрос «01001101», который является кодом состояния S_2, выбранного в качестве начального состояния ЦКА. Далее ФНФ генерирует ответ «11001010», поэтому после инициализации ЦУ i_c = «1100». Не нарушая общности, предположим, что блок коррекции в данном случае реализует функцию исключающего ИЛИ (XOR), однако он может использовать и более сложное преобразование значений seed и key, например, с помощью адаптивного сигнатурного анализатора [15]. Тогда, для того чтобы сгенерировать код выходного перехода из состояния S_{43} (o_c = «0110»), необходимо подать в блок коррекции в качестве ключа значение «1100»: действительно, операция исключающего ИЛИ для ключа и второй части ответа ФНФ («1010») в результате даст необходимое значение, которое применимо для осуществления перехода в состояние S_3. Отметим, что количество бит в ответе ФНФ должно быть достаточным для невозможности применения метода полного перебора вариантов в качестве криптографической атаки, т. е. время подбора правильного кода выходного перехода должно быть достаточно велико для применения на практике [16]. Таким образом, описанный метод расходует меньшее количество аппаратурных ресурсов, но требует ФНФ большей разрядности и надежный блок коррекции, реализующий нелинейную хеш-функцию, затрудняющую криптографические атаки на секретные ключи.

2.2. Открытые методы активной идентификации

В отличие от закрытых открытые методы активной идентификации используют алгоритмы асимметричного шифрования для проектирования блокирующей схемы, что, соответственно, требует наличия внешних ключей для осуществления работы протокола. Впервые такой метод был упомянут в работе [12]. Общая схема предложенного метода представлена на рис. 6.

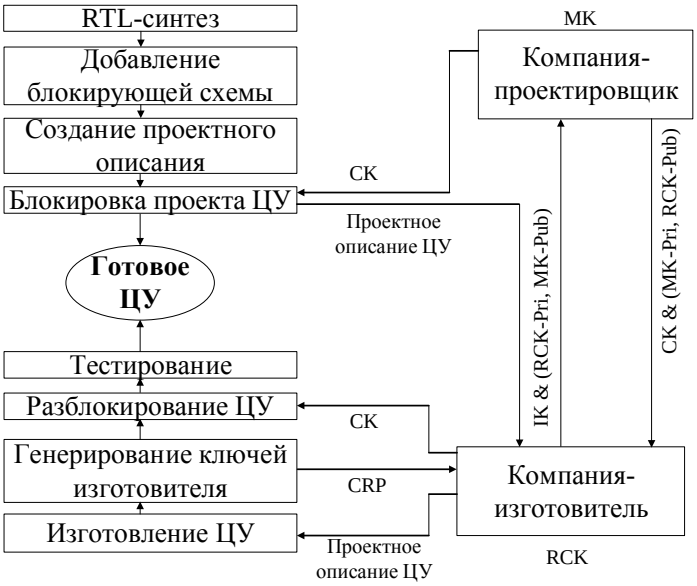


Рис. 6. Схема активной идентификации с использованием алгоритма асимметричного шифрования

В соответствии с алгоритмами криптографии с открытым ключом проектировщику необходимо сгенерировать пару ключей (Master Keys – MK). Закрытый ключ (MK-Pri) должен держаться в секрете и не передаваться ни при каких обстоятельствах, а открытый ключ (MK-Pub) передается по незащищенным каналам и поэтому известен компании-изготовителю. Как правило, эта пара ключей генерируется для каждого ЦУ с использованием программного обеспечения, которое может генерировать пары ключей для определенного протокола шифрования. Блокирующая схема реализована на уровне RTL и встроена в проектное описание ЦУ.

Согласно предложенному методу блокирующая схема проектируется с помощью элементов исключающего ИЛИ, соединенных с регистром хранения общего ключа (Common Key – CK). Только наличие правильного CK переводит ЦУ в нормальный режим работы, в противном случае поведение ЦУ может быть непредсказуемым. Для предотвращения похищения CK на стороне проектировщика он генерируется случайным образом для каждого конкретного ЦУ. После внедрения блокирующей схемы и генерирования всех необходимых ключей (MK, CK) проектное описание отправляется компании-изготовителю.

Для активации ЦУ производитель обязан сгенерировать пару секретных ключей (Random Chip Keys – RCK) асимметричного шифрования (закрытый ключ RCK-Pri и открытый ключ RCK-Pub) после изготовления каждого ЦУ с помощью встроенной ФНФ. Как правило, ответы, сгенерированные ФНФ, используются в качестве инициализирующего значения для программного обеспечения, генерирующего пары секретных ключей асимметричного шифрования. Далее производитель осуществляет шифрование персональной информации (Input Key – IK) ключом RCK-Pri и использует MK-Pub для подписи. Таким образом, компания-проектировщик может аутентифицировать изготовителя с помощью имеющихся MK-Pri и RCK-Pub соответственно. В результате успешной аутентификации компания-проектировщик отправляет CK, зашифрованный MK-Pri и подписанный RCK-Pub. Аналогичным образом компания-изготовитель расшифровывает CK, осуществляет разблокирование и тестирование ЦУ и отправляет его компании-проектировщику.

Поскольку ключи на стороне производителя генерируются встроенной ФНФ, то они будут уникальны для каждого ЦУ, что исключает применение одной и той же лицензионной информации для других ЦУ, изготовленных по одному проектному описанию.

2.3. Преимущества и недостатки существующих методов активной идентификации

Методы активной идентификации показали себя эффективным и многообещающим решением в борьбе против пиратства и нелегального копирования проектных описаний ЦУ, поскольку злоумышленнику требуется одновременно решить две проблемы: осуществить взлом блокирующей схемы и получить информацию из источника энтропии (как правило, это ФНФ). Такая необходимость возникает по той причине, что блокирующая схема изначально проектируется зависимой от множества CRP ФНФ. Это делает практически невозможным изготовление дополнительных копий ЦУ даже при условии, что попытка взлома была успешно осуществлена на одном из них. Соответственно, продажа таких ЦУ не принесет большой прибыли компании-изготовителю, так как взлом одного из них является достаточно трудоемким процессом и даже в результате успеха не дает значительных преимуществ при попытках взлома других копий, произведенных по одинаковому проектному описанию.

Тем не менее существующие методы активной идентификации обладают двумя существенными недостатками. Во-первых, проектирование блокирующей схемы с использованием избыточных состояний и (или) переходов в ЦКА несет в себе дополнительные аппаратурные затраты, которые недопустимы в условиях ограничения на применение ресурсов и потребляемой мощности. В случае использования протоколов асимметричного шифрования для реализации блокирующей схемы также применяются значительные аппаратурные ресурсы, усложняется проектное описание ЦУ и возникает необходимость генерирования и хранения нескольких пар секретных ключей для поддержания работоспособности протокола. Перспективным является применение блокирующих схем, основанных на комбинационной логике [17].

Во-вторых, фиксированная блокирующая схема делает определенное устройство уязвимым к возможным взломам и, соответственно, последующему изучению проектного описания на корректно функционирующем ЦУ. Для решения этой проблемы авторами предлагается схема повторного лицензирования (рис. 7), основанная на применении реконфигурируемых ФНФ (РФНФ) [18] вместо фиксированных классических [19].

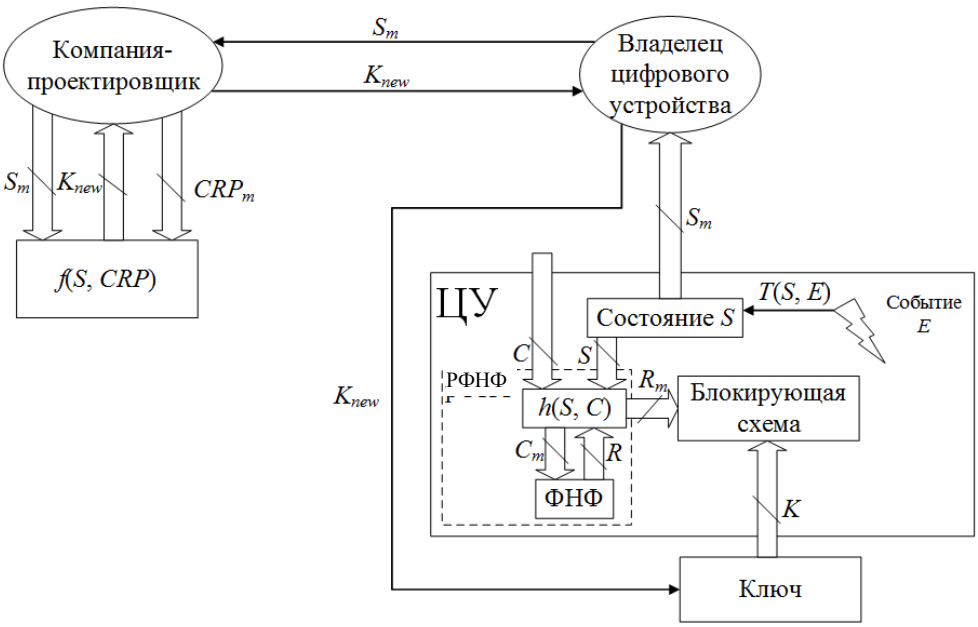


Рис. 7. Схема повторного лицензирования, основанная на методах активной идентификации

Цифровое устройство проектируется таким образом, чтобы оно содержало в себе компоненты методов активной идентификации: блокирующую схему и ФНФ в качестве источника энтропии. К ФНФ предъявляется требование реконфигурируемости: ответы на запросы становятся зависимыми не только от вариаций технологического процесса, но и от некоторого состояния (конфигурации) S, определяемого параметрами ФНФ (например, для случая мультиарбитральной ФНФ [20] номерами арбитров, которые используются для генерирования ответа; конфигурацией симметричных путей; идентификаторами, формируемыми метастабильными арбитрами). Таким образом, ответ на один и тот же запрос для конфигурации S будет отличаться от ответа для модифицированной конфигурации S_m.

Генерирование CRP осуществляется при помощи аппаратной хеш-функции h: $C_m = h(S, C)$ и $R_m = h(S, R)$, где C, R, C_m, R_m – запрос и ответ ФНФ до хеширования и после соответственно. Поскольку блокирующая схема является зависимой от ФНФ, ее реконфигурация осуществляется также в соответствии с состоянием S. Как и во всех методах активной идентификации, описанных ранее, изначально ЦУ находится в заблокированном состоянии и для его разблокирования компании-проектировщику необходимо получить CRP. Таким образом, для конкретной конфигурации ФНФ S и множества CRP формируется ключ $K = f(S, \text{CRP})$, где f – функция генерирования ключа на основании информации о ФНФ и блокирующей схеме. Для подготовки устройства к повторному лицензированию проектировщику необходимо сгенерировать N возможных ключей $K_1, K_2, \ldots, K_N$, где N – максимально возможное число случаев повторного лицензирования.

Протокол взаимодействия между компанией-проектировщиком и владельцем ЦУ при повторном лицензировании состоит из следующих этапов:

1. ЦУ в конфигурации S было разблокировано с помощью ключа K, но произошло событие E (ключ украден третьим лицом, срок его действия истек или владелец осуществил попытку изменения лицензионной информации), которое сделало ключ недействительным, поскольку была принудительно изменена конфигурация $S_m = T(S, E)$, где T – функция преобразования состояния S, срабатывающая по наступлении события E.

2. Поскольку устройство после реконфигурирования является заблокированным в силу недействительности ключа, владелец ЦУ вынужден обратиться к компании-проектировщику с просьбой о повторном предоставлении лицензионного ключа K_{new}. Для этого он отправляет по защищенному каналу текущее состояние S_m.

3. После получения от владельца необходимой информации проектировщик осуществляет генерирование нового ключа $K_{new} = f(S_m, \text{CRP}_m)$, где CRP_m – пары «запрос – ответ» ФНФ для состояния S_m.

4. Новый ключ отправляется владельцу по защищенному каналу. Далее осуществляется разблокирование ЦУ с помощью K_{new}. Таким образом, ЦУ переводится вновь в работоспособное состояние до наступления следующего события E и соответственно повторного лицензирования.

Текущие реализации методов активной идентификации в качестве блокирующей схемы используют ЦКА, реконфигурацию которого невозможно осуществить после синтеза. В связи с этим разработка реконфигурируемых блокирующих схем, работающих совместно с РФНФ, является неразрешенной и актуальной в настоящее время проблемой в области активной идентификации ЦУ.

Заключение

В статье рассмотрены существующие методы активной идентификации ЦУ, основанные на применении как цифровых конечных автоматов, так и протоколов асимметричного шифрования в качестве блокирующей схемы. Предложены варианты реализации блокирующих схем, которые проектируются зависимыми от пар «запрос – ответ» физически неклонируемой функции, что делает затруднительными криптографические атаки на один из компонентов схем активной идентификации ЦУ. Проанализированы также преимущества и недостатки существующих подходов. Предложены возможные пути преодоления проблем, не имеющих решения в настоящее время.

Список литературы

1. Wafer fabrication yield learning and cost analysis based on in-line inspection / I. Tirkela [et al.] // Intern. J. of Production Research. – 2016. – Vol. 54, no. 1. – P. 1–13.

2. Koushanfar, F. Hardware metering: A survey / F. Koushanfar // Introduction to Hardware Security and Trust / M. Tehranipoor, C. Wang (eds.). – N. Y. : Springer, 2012. – Ch. 5. – P. 103–122.

3. Choosing an Intellectual Property Core [Electronic resource]. – 2002. – Mode of access : http://wiki.prplfoundation.org/w/images/9/9e/Choosing_an_Intellectual_ Property_Core.pdf. – Date of access : 17.02.2016.

4. Tehranipoor, M. Counterfeit Integrated Circuits. Detection and Avoidance / M. Tehranipoor, U. Guin, D. Forte. – Switzerland : Springer International Publishing, 2015. – 269 p.

5. Integrated Circuit Security Threats and Hardware Assurance Countermeasures / K.M. Goertzel [et al.] // CrossTalk. – 2013. – Vol. 26, no. 6. – P. 33–38.

6. Koushanfar, F. Intellectual property metering / F. Koushanfar, G. Qu, M. Potkonjak // 4th Intern. Workshop Information Hiding (IH'01); ed. I. Moskowitz; Pittsburg, USA, April 25–27, 2001. – Pittsburg, 2001. – P. 81–95.

7. Koushanfar, F. Hardware Metering / F. Koushanfar, G. Qu // Proc. IEEE Design Automation Conference (DAC'01), Scottsdale, USA, June 18–22, 2001 / ACM, New York. – Scottsdale, 2001. – P. 490–493.

8. Koushanfar, F. Integrated circuits metering for piracy protection and digital rights management: an overview / F. Koushanfar // Great Lakes Symp. on VLSI (GLSVLSI'11), Salt Lake City, USA, May 3–4, 2012 / ACM, New York. – Salt Lake City, 2011. – P. 449–454.

9. Alkabani, Y. Active hardware metering for intellectual property protection and security / Y. Alkabani, F. Koushanfar // USENIX Security Symposium (SS'07), Boston, USA, August 6–10, 2007 / Addison Wesley, Indianapolis. – Boston, 2007. – P. 291–306.

10. Koushanfar, F. Provably secure active IC metering techniques for piracy avoidance and digital rights management / F. Koushanfar // IEEE Trans. Inf. Forensics and Security. – Vol. 7, no. 1. – P. 51–63.

11. Alkabani, Y. Remote activation of ICs for piracy prevention and digital right management / Y. Alkabani, F. Koushanfar, M. Potkonjak // EEE/ACM Intern. Conf. on Comp.-Aided Design (ICCAD'07), San Jose, USA, Nov. 4–8, 2007 / ACM, New York. – San Jose, 2007. – P. 674–677.

12. Ending piracy of integrated circuits / J. Roy [et al.] // Computer. – 2010. – Vol. 43, no. 10. – P. 30–38.

13. Ярмолик, В.Н. Физически неклонируемые функции / В.Н. Ярмолик, Ю.Г. Вашинко // Информатика. – 2011. – № 2. – С. 92–103.

14. Maes, R. PUFKY: A Fully Functional PUF-Based Cryptographic Key Generator / R. Maes, A. Van Herrewege, I. Verbauwhede // Cryptographic Hardware and Embedded Systems (CHES'12), Leuven, Belgium, Sept. 9–12, 2012 / Springer, New York. – Leuven, 2012. – P. 302–319.

15. Иванюк, А.А. Проектирование контролепригодных цифровых устройств / А.А. Иванюк, В.Н. Ярмолик. – Минск : Бестпринт, 2006. – 295 с.

16. How secure is AES against brute force attacks? [Electronic resource]. – 2012. – Mode of access : http://www.eetimes.com/document.asp?doc_id=1279619. – Date of access : 23.02.2016.

17. Solving the Third-Shift Problem in IC Piracy With Test-Aware Logic Locking / S.M. Plaza [et al.] // IEEE Trans. on Comput.-Aided Design of Integrated Circuits and Syst. – 2015. – Vol. 34, no. 6. – P. 961–971.

18. Reconfigurable physical unclonable functions – enabling technology for tamper-resistant storage / K. Kursawe [et al.] // IEEE Intern. Workshop on Hardw.-Orient. Secur. and Trust (HOST'09), San Francisco, USA, July 27, 2009 / IEEE, New York. – San Francisco, 2009. – P. 22–29.

19. Заливако, С.С. Схема удаленного контроля для активного измерения цифровых устройств / С.С. Заливако, А.А. Иванюк // Информационные технологии и системы 2014 (ИТС 2014) : материалы Междунар. науч. конф., БГУИР, Минск, Беларусь, 29 окт. 2014 г. / редкол. : Л.Ю. Шилин [и др]. – Минск : БГУИР, 2014. – С. 73–74.

20. Multi-valued arbiters for quality enhancement of PUF responses on FPGA implementation / S.S. Zalivaka [et al.] // Special Session on Cyber-Physical Systems and Security, in Proc. 21st IEEE Asia and South Pacific Design Automation Conf. (ASP-DAC 2016), Macao, China, 26–28 Jan., 2016 / IEEE, New York. – Macao, 2016. – P. 533–538

Поступила 07.06.2016

Белорусский государственный университет информатики и радиоэлектроники,
Минск, ул. П. Бровки, 6
e-mail: zalivako@bsuir.by,
ivaniuk@bsuir.by

S.S. Zalivaka, A.A. Ivaniuk

ACTIVE METERING OF DIGITAL DEVICES: AN OVERVIEW

The paper presents existing active hardware metering approaches. The motivation of using active metering approaches by fabless integrated circuits design companies is given. The finite state machine based and asymmetrical cryptography based methods are presented. The advantages and disadvantages of existing active metering approaches are analyzed. The potential solution for modern technique issues are proposed.

СТАТЬИ ПО МАТЕРИАЛАМ СЕДЬМОЙ МЕЖДУНАРОДНОЙ НАУЧНОЙ КОНФЕРЕНЦИИ «ТАНАЕВСКИЕ ЧТЕНИЯ»

УДК 004.43

П.Н. Бибило

СХЕМНАЯ РЕАЛИЗАЦИЯ VHDL-ОПИСАНИЙ СИСТЕМ НЕ ПОЛНОСТЬЮ ОПРЕДЕЛЕННЫХ БУЛЕВЫХ ФУНКЦИЙ

Предлагаются синтезируемые VHDL-модели систем не полностью определенных булевых функций. Приводятся результаты экспериментов по различным способам схемной реализации VHDL-описаний таких систем функций комбинационными схемами в библиотеке проектирования заказных СБИС и FPGA.

Введение

Современные синтезаторы осуществляют высокоуровневый синтез логических схем заменой описания каждой VHDL-конструкции функциональным описанием соответствующей логической подсхемы. После этапа высокоуровневого синтеза выполняются этапы оптимизации и отображения в заданный технологический базис логических элементов [1]. Результаты синтеза в значительной мере зависят от вида исходного VHDL-описания, подаваемого на вход синтезатора. В результате синтеза по исходным VHDL-описаниям частичных булевых функций получаются комбинационные схемы, моделями поведения которых являются системы полностью определенных функций. Известно, что использование при синтезе логических схем неопределенных булевых функций может приводить к более экономичным схемам [2].

В настоящей работе предлагается способ описания на языке VHDL матричных форм систем не полностью определенных (частичных) булевых функций, приводятся примеры синтезируемых VHDL-моделей систем частичных булевых функций и сообщается о результатах экспериментов по схемной реализации VHDL-описаний таких систем функций. Эксперименты показали достаточно большую эффективность использования VHDL-описаний частичных функций вместо соответствующих описаний полностью определенных булевых функций.

1. Частичные булевы функции и их VHDL-описания

На практике достаточно часто приходится проектировать функциональные блоки комбинационной логики, характеризующиеся неопределенностью поведения для некоторых (многих) наборов значений входных сигналов. К таким блокам относятся кодовые преобразователи, комбинационные части конечных автоматов, цепочки последовательных арифметических VHDL-операторов с операндами, которым присущи неполные диапазоны значений и т. д. Математическими моделями этих блоков являются системы частичных булевых функций. Области неопределенности таких функций могут занимать почти всю область их определения, тогда говорят о слабоопределенной булевой функции или системе функций. Эффективность использования VHDL-моделей частичных функций была изучена в работе [3], где описана методика выделения последовательных VHDL-конструкций и замены таких конструкций эквивалентными по поведению матричными представлениями систем частичных булевых функций, при этом функции задавались на наборах значений аргументов.

В данной работе предлагается VHDL-модель для системы частичных функций, заданных в интервальной матричной форме.

Под *векторной* булевой функцией $\boldsymbol{f}(\boldsymbol{x})$ будем понимать упорядоченную систему частичных булевых функций $\boldsymbol{f}(\boldsymbol{x})=(f^1(\boldsymbol{x}),\ldots,f^m(\boldsymbol{x}))$, значениями векторных функций на элементах $\boldsymbol{x}^*$ булева пространства аргументов вектора $\boldsymbol{x}=(x_1,\ldots,x_n)$ являются m-компонентные тро-

ичные векторы $\boldsymbol{f}(\boldsymbol{x}^*)$. Пример интервального матричного задания частичной векторной функции $\boldsymbol{f}(\boldsymbol{x}) = (f^1(\boldsymbol{x}), f^2(\boldsymbol{x}))$ представлен в табл. 1. Частичная булева функция $f^1(\boldsymbol{x})$ может быть задана парой дизъюнктивных нормальных форм (ДНФ): множество $M^1_{f^1}$ задается в виде $D^1_{f^1} = \bar{x}_1 x_3 x_4 \vee x_1 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4$; множество $M^0_{f^1}$ – в виде $D^0_{f^1} = x_1 x_2 x_3 x_4 \vee x_2 x_3 \bar{x}_4$. Аналогично в виде пары ДНФ может быть задана частичная булева функция $f^2(\boldsymbol{x})$. Интервальная матричная форма системы частичных функций состоит из троичной матрицы T^x задания элементарных конъюнкций и троичной матрицы T^f вхождений конъюнкций в ДНФ, представляющих области нулевых и единичных значений функций f^j компонент векторной функции $\boldsymbol{f}(\boldsymbol{x})$. Неопределенное значение «–» элемента t_{ij} матрицы T^x не означает, что функция f^j имеет неопределенное значение на наборах, которые порождает троичный вектор строки i матрицы T^x. Например, на пересечении первой строки и первого столбца матрицы T^f находится неопределенное значение «–», однако это не означает, что на наборах 0001, 0011, 0101, 0111 функция f^1 не определена. Рассмотрев вторые строки матриц T^x, T^f, можно убедиться, что на наборах 0011, 0111 функция f^1 принимает единичное значение. Если же все строки матрицы T^x являются попарно ортогональными, то в строках матрицы T^x задаются значения 0,1, «–» компонентных частичных функций f^j векторной функции $\boldsymbol{f}(\boldsymbol{x})$.

Таблица 1

Интервальная форма системы частичных булевых функций

T^x				T^f	
x_1	x_2	x_3	x_4	f^1	f^2
0	–	–	1	–	1
0	–	1	1	1	–
1	–	0	0	1	–
–	–	1	0	–	0
0	0	0	0	1	0
1	1	1	1	0	1
–	1	1	0	0	–

Частичная функция $f^1(\boldsymbol{x})$ реализуется частичной функцией $f^2(\boldsymbol{x})$ (обозначается $f^1(\boldsymbol{x}) \prec f^2(\boldsymbol{x})$), если и только если $D^1_{f^1} \to D^1_{f^2}$, $D^0_{f^1} \to D^0_{f^2}$, где через $\to$ обозначена операция логической импликации. Для полностью определенных булевых функций отношение реализации является отношением равенства. Отношение реализации векторных функций сводится к выполнению отношения реализации между компонентными функциями: если каждая компонентная функция $f^{j*}(\boldsymbol{x})$, j = 1, …, m, векторной функции $f^*(\boldsymbol{x}) = (f^{1*}(\boldsymbol{x}), \ldots, f^{m*}(\boldsymbol{x}))$ реализует $(f^j \prec f^{j*})$ соответствующую компонентную функцию f^j векторной функции $\boldsymbol{f} = (f^1, \ldots, f^m)$, то тогда векторная функция $\boldsymbol{f}^*$ реализует векторную функцию $\boldsymbol{f}$. В этом случае векторная функция $\boldsymbol{f}^*$ называется *доопределением* векторной функции $\boldsymbol{f}$. Подробное описание различных форм представлений систем полностью определенных и частичных булевых функций дано в работе [3].

В языке VHDL при синтезе логических схем используется девятизначный алфавит на базе перечислимого типа *std_logic*. Массив (вектор) значений типа *std_logic* обозначается как *std_logic_vector* [1]. На базе типов *std_logic*, *std_logic_vector* предлагается строить VHDL-модели частичных функций. Дело в том, что одно из девяти значений данного типа называется *don't care* (неопределенное значение), обозначается как «–» и предназначается для оптимизации при логическом синтезе. Однако в литературе не показано, каким именно образом значение

don't care используется при составлении VHDL-описаний частичных функций и как использование неопределенности влияет на результаты синтеза по VHDL-описаниям частичных булевых функций.

Предлагаемое VHDL-описание интервальной формы системы частичных функций состоит из двух частей. В первой части с помощью логических операторов записываются ДНФ $D^1_{f^i}$, $D^0_{f^i}$, вторая часть – это вызов VHDL-функции *value_f*, формирующей значения функций системы. Описание функции *value_f* приводится в VHDL-пакете *chast_pack*. В листинге 1 задан пример VHDL-модели системы функций из табл. 1. Если при моделировании системы частичных функций хотя бы в одном векторе *F* значений выходных сигналов будет содержаться символ 'X', то это означает, что на соответствующем наборе значений входных сигналов система функций задана противоречиво.

Листинг 1. VHDL-описание системы частичных функций (табл. 1), заданных в интервальной матричной форме

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
package chast_pack is
constant N : natural := 4;
constant M : natural := 2;
Function value_f (constant M : natural;
f_nul : std_logic_vector; f_ed :std_logic_vector )
return  std_logic_vector ;
end chast_pack;
package body chast_pack is
Function value_f (constant M : natural;
f_nul : std_logic_vector; f_ed  : std_logic_vector )
return  std_logic_vector  is
variable z : std_logic_vector (1 to M);
variable e : std_logic_vector (1 to 2);
begin
for i in f_nul'range loop
    e := (f_nul(i), f_ed(i));
  case e is
    when "10" => z(i):='0';
    when "01" => z(i):='1';
    when "00" => z(i):='-';
    when others  => z(i):='X'; -- error
  end case;
end loop;
return z ;
end value_f ;
end;

LIBRARY ieee;
USE ieee.std_logic_1164.all;
use work.chast_pack.all;
ENTITY EXAMPLE1 IS
PORT (x1, x2, x3, x4 : IN std_logic;
F : OUT std_logic_vector (1 to M));
end;
ARCHITECTURE BEH OF EXAMPLE1 IS
signal f_0 : std_logic_vector (1 to M);
signal f_1 : std_logic_vector (1 to M);
```

```
BEGIN
f_0(1) <= (x1 and x2 and x3 and x4) or
          (x2 and x3 and not x4);
f_1(1) <= (not x1 and x3 and x4) or
          (x1 and not x3 and not x4) or
          (not x1 and not x2 and not x3 and not x4);
f_0(2) <= (x3 and not x4) or
          (not x1 and not x2 and not x3 and not x4);
f_1(2) <= (not x1 and x4) or
          (x1 and x2 and x3 and x4);
F <= value_f (M, f_0, f_1);
end;
```

Если же функции системы заданы на наборах значений аргументов (табл. 2), то VHDL-модель становится менее компактной (листинг 2).

Таблица 2

Задание системы частичных булевых функций на наборах значений аргументов

B^x				T^f	
x_1	x_2	x_3	x_4	f^1	f^2
0	0	0	0	1	0
0	0	0	1	–	1
0	0	1	0	–	0
0	0	1	1	1	1
0	1	0	1	–	1
0	1	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	–
1	0	1	0	–	0
1	1	0	0	1	–
1	1	1	0	0	0
1	1	1	1	0	1

Предполагается, что на остальных четырех наборах 0100, 1001, 1011, 1101 булева пространства, не приведенных в табл. 2, значения частичных функций f^1, f^2 не определены.

Листинг 2. VHDL-описание системы частичных функций, заданных на наборах (табл. 2)

```
Library ieee;
use ieee.std_logic_1164.all;
ENTITY EXAMPLE2 IS
PORT (x : IN std_logic_vector(1 to 4);
F : OUT std_logic_vector (1 to 2));
end;
ARCHITECTURE BEH OF EXAMPLE2 IS
begin
F <= "10" when x = "0000" else
     "-1" when x = "0001" else
     "-0" when x = "0010" else
     "11" when x = "0011" else
     "-1" when x = "0101" else
     "00" when x = "0110" else
     "11" when x = "0111" else
```

```
      "1-" when x = "1000" else
      "-0" when x = "1010" else
      "1-" when x = "1100" else
      "00" when x = "1110" else
      "01" when x = "1111" else
      "--";
end;
```

Предлагаемые VHDL-модели систем частичных функций являются синтезируемыми (по ним синтезатор может построить комбинационные схемы) и позволяют проводить проверку отношения реализации частичных функций полностью определенными функциями (либо логическими схемами, реализующими полностью определенные булевы функции) путем формальной верификации, например, используя систему верификации *FormalPro* [4].

2. Оптимизация представлений систем не полностью определенных булевых функций

Основными направлениями оптимизации представлений систем частичных булевых функций являются минимизация в классе ДНФ, т. е. оптимизация двухуровневых представлений, оптимизация многоуровневых представлений на основе разложения Шеннона (BDD(Binary Decision Diagram)-оптимизация) [5, 6] и декомпозиция [7, 8]. Теоретические методы по указанным направлениям оптимизации различных форм представлений систем частичных функций достаточно хорошо изучены [2, 3]. Влияние способов логической оптимизации на результаты логического синтеза в библиотеке проектирования отечественных заказных СБИС экспериментально исследовано в работах [3, 9] для систем полностью определенных функций. Эксперименты над программами логической оптимизации на потоке практических примеров показали, что для различных комбинационных схем целесообразно использовать различные оптимизационные процедуры (минимизацию в классе ДНФ, BDD-оптимизацию, декомпозицию) [9]. Однако BDD-оптимизация наиболее часто приводила к схемам, требующим меньшую площадь кристалла заказной СБИС, даже если сравниваться с результатами использования эффективной программы ESPRESSO [10] совместной минимизации систем полностью определенных функций в классе ДНФ.

BDD-оптимизация булевых функций основана на разложении Шеннона. *Разложением Шеннона* булевой функции $f(\boldsymbol{x}) = f(x_1, \ldots, x_n)$ по переменной x_i называется представление $f(\boldsymbol{x})$ в виде

$$f(\boldsymbol{x}) = \bar{x}_i f_0 \vee x_i f_1. \qquad (1)$$

Если исходная функция $f(\boldsymbol{x})$ является частичной, то и коэффициенты f_0, f_1 разложения являются частичными функциями. Каждый из коэффициентов $f_0 = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots x_n)$, $f_1 = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots x_n)$ может быть разложен по одной из переменных из множества $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n\}$. Процесс разложения коэффициентов заканчивается, когда все n переменных будут использованы для разложения. В процессе разложения либо на последнем шаге разложения коэффициенты вырождаются до констант 0, 1, «–». На каждом шаге разложения выполняется сравнение на равенство полученных коэффициентов и из множества равных коэффициентов остается один. Под диаграммой двоичного выбора (BDD) понимается ориентированный ациклический граф, задающий последовательные разложения Шеннона булевой функции $f(x_1, \ldots, x_n)$ по всем ее переменным $x_1, \ldots, x_n$ при заданном порядке (перестановке) переменных, по которым проводятся разложения. Функциональная вершина, соответствующая функции f, называется корнем. Под *сложностью (размером) BDD* будем понимать число функциональных вершин BDD.

Далее для векторной булевой функции $\boldsymbol{f}(\boldsymbol{x})$ будут рассматриваться BDD, которые построены по общей для всех компонентных функций $f^i(\boldsymbol{x})$ перестановке переменных. В резуль-

тате BDD-оптимизации [3] матричная форма системы частичных функций заменяется графом BDD, представляющим систему полностью определенных булевых функций. Такая BDD имеет *m* корневых и две листовые вершины 0, 1, которые обычно дублируются для упрощения изображения графа. BDD, реализующая систему частичных функций (см. табл. 1), изображена на рис. 1. VHDL-описание BDD дано в листинге 3. Каждая функциональная вершина BDD, за исключением вершин-переменных либо их инверсий, задается соответствующей формулой разложения Шеннона.

Листинг 3. VHDL-описание BDD (рис. 1)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY EXAMPLE3 IS
PORT (x1,x2,x3,x4,x5 : IN std_logic;
F : OUT std_logic_vector (1 to 2));
end;
ARCHITECTURE BDD OF EXAMPLE3 IS
signal p1: std_logic;
BEGIN
p1 <= not x1 and x4;
F(1) <= (not x3) or (x3 and p1);
F(2) <= (not x3 and p1) or x3 and x4;
end;
```

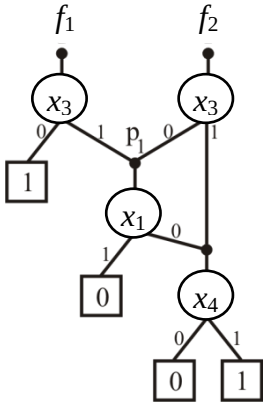


Рис. 1. BDD, реализующая систему частичных булевых функций

3. Экспериментальные исследования

В этом разделе приводятся результаты экспериментов по схемной реализации систем частичных булевых функций на заказных СБИС (ASIC) и на программируемых логических интегральных схемах типа FPGA. В качестве программы совместной минимизации векторных частичных функций в классе ДНФ использовалась программа [11], позволяющая минимизировать общее число элементарных конъюнкций, на которых задана минимизированная векторная полностью определенная функция, реализующая исходную частичную векторную функцию.

Эксперимент 1. Синтез схем кодовых преобразователей в библиотеке проектирования заказной СБИС

Синтез осуществлялся по двум видам исходных описаний: матричным формам систем частичных функций и BDD-представлениям полностью определенных функций, построенным в результате оптимизации исходных матричных форм систем функций в классе многоуровневых представлений. Исходные системы частичных функций, заданные на наборах значений аргументов, и BDD представлялись на языке VHDL. Исходные VHDL-описания матричных форм систем частичных функций поступали на вход синтезатора LeonardoSpectrum [1], который по этим описаниям строил схемы в библиотеке проектирования заказной СБИС.

Влияние различных режимов синтеза на сложность получаемых логических схем и их быстродействие подробно исследовано для синтезатора LeonardoSpectrum в работе [12]. При экспериментах в данной работе синтез (optimize) всех схем в LeonardoSpectrum осуществлялся с устранением иерархии (hierarchy flatten) в исходном описании. Режим синтеза (optimize effort) выполнялся по опции (effort standard), при установке которой синтезатор осуществляет несколько итераций оптимизации и технологического отображения на основе различных методов [12]. Для увеличения быстродействия синтезированной схемы выполнялась ее перестройка согласно установленной опции (optimize timing).

Состав библиотеки логических элементов приведен в [3]. Полученные схемы сравнивались со схемами, построенными по VHDL-описаниям полностью определенных систем функций, полученных с помощью программы оптимизации BDD для систем частичных функций.

Результаты эксперимента 1 для практических примеров Verg1 и Verg2 систем частичных функций (таблиц для хранения микропрограмм команд микропроцессоров) представлены в табл. 3. Каждая система частичных функций была задана на наборах значений аргументов двумя булевыми матрицами B^x, B^f, т. е. троичная матрица T^f выродилась до булевой матрицы B^f. Первая булева матрица B^x имела размерность $n \times k$ (n столбцов, k строк), вторая матрица B^f значений функций имела размерность $m \times k$. В табл. 3 используются следующие обозначения: n – число аргументов; m – число функций; k – число двоичных наборов в задании системы функций; L – число элементов схемы; S_{ASIC} – суммарная площадь всех элементов схемы (далее площадь схемы), размещаемых на кристалле заказной СБИС; τ – задержка схемы, нс.

Затем пара булевых матриц была интерпретирована как система совершенных ДНФ (СДНФ) полностью определенных булевых функций, заданных в матричной форме, и проведен синтез по оптимизированной BDD. Заметим, что синтезатор LeonardoSpectrum не смог построить логическую схему по исходному VHDL-описанию, интерпретируемому как система полностью определенных функций, поэтому в табл. 3 отсутствуют данные по результатам синтеза по таким описаниям. При поиске лучшей перестановки последовательности переменных, по которым строится BDD, для примера Verg1 эвристический алгоритм испытал 500 перестановок, для более сложного примера Verg2 – 100 перестановок аргументов.

Результаты эксперимента 1 показывают, что использование BDD позволяет уменьшить площадь схемы в несколько раз по сравнению со схемной реализацией исходного неоптимизированного VHDL-описания системы частичных функций. Кроме того, использование частичной определенности функций, даже без BDD-оптимизации, позволяет в полтора-два раза уменьшить площадь схем.

Таблица 3

Результаты эксперимента 1

Пример	n	m	k	Интерпретация пары булевых матриц	Схемная реализация матричной формы			Схемная реализация минимизированных BDD		
					S_{ASIC}	L	τ	S_{ASIC}	L	τ
Verg1	17	1	003	Система частичных булевых функций	8940	2183	17,79	**3741**	**991**	**12,15**
				Система полностью определенных булевых функций	Синтез не выполнен			5562	1488	17,38
Verg2	18	3	129	Система частичных булевых функций	8884	2272	20,29	**5809**	**1508**	**12,59**
				Система полностью определенных булевых функций	Синтез не выполнен			7670	1924	20,39

Эксперимент 2. Схемная реализация систем псевдослучайных частичных булевых функций в библиотеке проектирования заказной СБИС

Генерируемые случайным образом частичные векторные функции представлялись в матричной форме, пример которой дан в табл. 1. При генерации варьировались значения: α – чис-

ло определенных (0,1) значений в строке матрицы T^x; β – число определенных значений в строке матрицы T^f. Матричные представления конвертировались в VHDL-описания, после чего синтезировались логические схемы. Затем исходные матричные представления минимизировались в классах ДНФ и BDD, преобразовывались в VHDL-описания, по которым синтезировались комбинационные схемы. Результаты эксперимента 2 приведены в табл. 4. Анализ результатов показывает, что для систем псевдослучайных частичных функций предварительная совместная минимизация в классе ДНФ чаще приводит к схемам меньшей площади. Величина площади уменьшается при уменьшении чисел определенных элементов в матрицах T^x и T^f. При небольшом ($\alpha = 4$) числе определенных элементов в процессе BDD-оптимизации частичные функции системы вырождаются (доопределяются) до констант 0,1.

Эксперимент 3. Схемная реализация систем псевдослучайных частичных функций на FPGA

Эксперимент 3 отличается от эксперимента 2 только тем, что в качестве технологического базиса была выбрана программируемая логическая схема (ПЛИС) типа FPGA Spartan 3-1000, в качестве примеров систем функций были взяты те же примеры Gen1 – Gen8, что и в эксперименте 2. Синтез осуществлялся с помощью синтезатора XST, входящего в состав ISE (САПР фирмы XILINX) при опциях, установленных по умолчанию. Результаты эксперимента 3 представлены в табл. 5, где в столбцах LUT приводится число программируемых элементов FPGA, требуемых для реализации комбинационной схемы, через τ обозначена задержка схемы, нс; k_{min} – число конъюнкций в минимизированной системе ДНФ.

Таблица 4

Результаты эксперимента 2 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных частичных функций			Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)			
			S_{ASIC}	L	τ	S_{ASIC}	L	τ	S_{ASIC}	L	τ	Размер BDD
Gen1	16	4	483875	1351	8,76	**246815**	**662**	**6,36**	605737	1609	9,48	1486
Gen2	12	4	474908	1320	8,63	**167880**	**448**	**5,70**	336837	874	7,56	983
Gen3	8	4	343304	917	10,74	**11260**	**33**	3,39	14620	43	**3,33**	75
Gen4	4	4	454268	1164	11,36	Доопределение функций до констант			Доопределение функций до констант			
Gen5	16	8	527271	1495	12,23	**373257**	**1041**	**9,23**	822492	2072	10,57	2161
Gen6	12	8	464479	1299	11,65	**256334**	**687**	**8,17**	274542	716	8,75	911
Gen7			387793	1057	11,64	1836	**7**	2,22	**1735**	**7**	**2,08**	8
Gen8			258583	687	0,69	Доопределение функций до констант			Доопределение функций до констант			

Таблица 5

Результаты эксперимента 3 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных частичных функций		Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)		
			LUT	τ	LUT	τ	k_{min}	LUT	τ	Размер BDD
Gen1	16	4	891	**3,67**	**406**	**3,73**	212	926	4,88	1486
Gen2	12	4	846	**3,66**	**279**	3,78	166	575	4,33	983
Gen3	8	4	663	3,77	**24**	2,30	27	26	2,56	75
Gen4	4	4	98	3,43	Доопределение функций до констант			Доопределение функций до констант		
Gen5	16	8	1047	3,85	**569**	**3,79**	244	1440	4,49	2161

Окончание табл. 5

Пример	α	β	Схемная реализация исходных частичных функций		Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)		
			LUT	τ	LUT	τ	k_{min}	LUT	LUT	τ
Gen6	12	8	956	3,89	**418**	**3,86**	227	489	4,25	911
Gen7	8	8	662	4,00	**3**	**1,00**	10	**3**	**1,00**	8
Gen8	4	8	Доопределение функций до констант		Доопределение функций до констант			Доопределение функций до констант		

Эксперимент 4. Схемная реализация систем псевдослучайных полностью определенных функций на FPGA

Системы псевдослучайных полностью определенных функций в эксперименте 4 генерировались в виде систем ДНФ, заданных в матричной форме парой матриц T^x и B^f, где T^x – троичная матрица, строки которой задают троичные векторы, соответствующие элементарным конъюнкциям ДНФ функций системы; B^f – булева матрица, в которой единичные значения элементов в строке отмечают факт вхождения соответствующей элементарной конъюнкции в ДНФ функций системы [2]. Результаты эксперимента 4 приведены в табл. 6. Анализ результатов показывает, что предварительная минимизация в классе ДНФ псевдослучайных систем полностью определенных функций практически не улучшает (а процедура BDD-оптимизации ухудшает) результаты последующего синтеза в базисе FPGA в отличие от их значительной эффективности [3, 9] при синтезе примеров схем (benchmarks), взятых из практики проектирования.

Таблица 6

Результаты эксперимента 4 (n = 16, m = 16, k_{mp} = 256)

Пример	α	β	Схемная реализация исходных функций (полностью определенных функций)		Схемная реализация минимизированных ДНФ (полностью определенных функций)			Схемная реализация минимизированных BDD (полностью определенных функций)		
			LUT	τ	LUT	τ	k_{min}	LUT	τ	Размер BDD
Gen9	16	4	**803**	**3,91**	**803**	3,92	256	6919	4,32	9479
Gen10	12	4	787	**3,82**	**783**	3,88	256	10384	4,39	13700
Gen11	8	4	**688**	**3,89**	695	4,00	255	18029	4,66	23227
Gen12	6	4	628	**3,72**	**541**	3,76	235	5359	4,83	7951
Gen13	4	4	57	3,18	**54**	**3,13**	77	59	3,23	203

Заключение

Доопределение частичных булевых функций при оптимизации позволяет в несколько раз уменьшить сложность логических схем при проектировании заказных СБИС и структур FPGA по сравнению с реализацией матричных форм функций, интерпретируемых как системы полностью определенных булевых функций. В отличие от случая промышленных примеров систем ДНФ предварительная совместная минимизация в классе ДНФ матричных форм систем псевдослучайных частичных функций в данном эксперименте позволила получить при синтезе лучшие результаты по сравнению с предварительной оптимизацией их BDD-представлений.

Список литературы

1. Бибило, П.Н. Системы проектирования интегральных схем на основе языка VHDL. StateCAD, ModelSim, LeonardoSpectrum / П.Н. Бибило. – М. : СОЛОН-Пресс, 2005. – 384 с.

2. Закревский, А.Д. Логический синтез каскадных схем / А.Д. Закревский. – М. : Наука, 1981. – 416 с.

3. Бибило, П.Н. Применение диаграмм двоичного выбора при синтезе логических схем / П.Н. Бибило. – Минск : Беларус. навука, 2014. – 231 с.

4. Лохов, А. Функциональная верификация СБИС в свете решений Mentor Graphics / А. Лохов // Электроника: наука, технология, бизнес. – 2004. – № 1. – С. 58–62.

5. Yang, S. BDS: a BDD-based logic optimization system / S. Yang, M. Ciesielski // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. – 2002. – Vol. 21. – No. 7. – P. 866–876.

6. Stojkovich, S. Determining Assignment of Incompletely Specified Boolean Functions for Compact Representations by Binary Decision Diagrams / S. Stojkovich, M. Stanković, R. Stanković // 10th Intern. Workshop on Boolean Problems, Freiberg (Sachsen), 2012. – Freiberg (Sachsen), 2012. – P. 233–238.

7. Taghavi Afshord, S. An input variable partitioning algorithm for functional decomposition of a system of Boolean functions based on the tabular method / S. Taghavi Afshord, Yu.V. Pottosin, B. Arasteh // Discrete Applied Mathematics. – 2015. – No. 185. – P. 208–219.

8. Lee , R.-R. Bi-decomposing large Boolean functions via interpolation and satisfiability solving / R.-R. Lee , J.-H. R. Jiang , W.-L. Hung // Proc. of the 45th Annual Design Automation Conference, Anaheim, California, 8–13 June 2008. – N. Y., 2008. – P. 636–641.

9. Авдеев, Н.А. Эффективность логической оптимизации при синтезе комбинационных схем из библиотечных элементов / Н.А. Авдеев, П.Н. Бибило // Микроэлектроника. – 2015. – Т. 44, № 5. – С. 383–399.

10. Logic minimization algorithm for VLSI synthesis / K.R. Brayton [et al.]. – Boston : Kluwer Academic Publishers, 1984. – 193 p.

11. Торопов, Н.Р. Минимизация систем булевых функций в классе ДНФ / Н.Р. Торопов // Логическое проектирование. – Минск : Ин-т техн. кибернетики НАН Беларуси, 1999. – Вып. 4. – С. 4–19.

12. Авдеев, Н.А. Эффективность проектирования заказных схем в синтезаторе LeonardoSpectrum / Н.А. Авдеев, П.Н. Бибило // Современная электроника. – 2015. – № 1. – С. 58–61.

Поступила 26.05.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: bibilo@newman.bas-net.by

P.N. Bibilo

CIRCUIT IMPLEMENTATION OF VHDL-DESCRIPTIONS OF SYSTEMS OF PARTIAL BOOLEAN FUNCTIONS

Method for description of incompletely specified (partial) Boolean functions in VHDL is proposed. Examples of synthesized VHDL models of partial Boolean functions are presented; and the results of experiments on circuit implementation of VHDL descriptions of systems of partial functions. The realizability of original partial functions in logical circuits was verified by formal verification. The results of the experiments show that the preliminary minimization in DNF class and in the class of BDD representations for pseudo-random systems of completely specified functions does not improve practically (and in the case of BDD sometimes worsens) the results of the subsequent synthesis in the basis of FPGA unlike the significant efficiency of these procedures for the synthesis of benchmark circuits taken from the practice of the design.

УДК 681.325

Н.А. Кириенко, Л.Д. Черемисинова

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ТЕХНОЛОГИЧЕСКИ НЕЗАВИСИМОЙ ОПТИМИЗАЦИИ ФУНКЦИОНАЛЬНЫХ ОПИСАНИЙ КМОП-СХЕМ

Исследуется влияние процедур технологически независимой оптимизации функциональных описаний, которые представлены системами булевых функций, на сложность и задержку схем, синтезированных из библиотечных элементов КМОП СБИС. Сравнивается эффективность предварительной минимизации и алгебраической декомпозиции систем булевых функций в классе ДНФ при оптимизации функциональных описаний средствами синтезатора LeonardoSpectrum. Экспериментальное исследование показало, что к наиболее существенному уменьшению сложности и задержки схем приводит последовательное выполнение на этапе технологически независимой оптимизации процедур минимизации систем булевых функций и алгебраической декомпозиции.

Введение

Задача синтеза состоит в представлении исходного функционального описания логической схемой из элементов технологической библиотеки изготовителя СБИС, каждый из которых характеризуется своей функцией и физическими характеристиками. Широко используемыми критериями оптимальности при синтезе являются площадь результирующего кристалла СБИС (функция от числа вентилей, транзисторов), быстродействие и (в последние годы) энергопотребление. В такой постановке задача синтеза выступает комбинаторно сложной. Процесс синтеза существенно упрощается за счет разбиения его на две стадии (как это делается практически во всех системах логического проектирования): технологически независимую оптимизацию и технологическое отображение.

На первом этапе исходное функциональное описание проектируемой схемы представляется в виде многоуровневой схемы из простых элементов (типа И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ). На втором этапе эта схема преобразуется в функционально эквивалентную схему из элементов целевой технологической библиотеки.

В настоящей работе исследуется степень влияния разных подходов к технологически независимой оптимизации функциональных описаний, представленных системами булевых функций, на площадь и задержку схем, синтезированных из библиотечных элементов КМОП СБИС с помощью средств синтезатора LeonardoSpectrum.

1. Технологически независимая оптимизация функциональных описаний

Результатом выполнения данного этапа является схема из вентилей (типа И, ИЛИ, НЕ). Цель этапа состоит в том, чтобы построить такой вариант представления схемы, который послужил бы хорошей отправной точкой для этапа технологического отображения в базис библиотечных элементов. В качестве количественных оценок эффективности проектирования на этом этапе используется суммарное число входных полюсов вентилей и задержка схемы.

Минимизация систем булевых функций в классе дизъюнктивных нормальных форм (ДНФ). Технологически независимая оптимизация, используемая в системах логического проектирования микроэлектронных устройств, включает в себя в качестве первого этапа, как правило, этап минимизации булевых функций. Выполнение этого этапа обусловлено тем, что минимизация позволяет сократить сложность исходного задания (иногда довольно существенно). Минимизация функционального описания ведется с учетом фактора сложности (оцениваемой по Квайну числом всех литералов полученной системы ДНФ). В результате выполнения этапа минимизации систем функций получается двухуровневая И-ИЛИ-схема (с использованием инверторов), в которой элементы имеют число входных полюсов, не привязанное к целевому технологическому базису.

Алгебраическая декомпозиция систем ДНФ булевых функций. Основным методом (используемым в большинстве САПР) построения многоуровневой схемы из вентилей, подлежащей переводу в технологический базис, является алгебраическая декомпозиция системы ДНФ [1], в основе которой лежит построение факторизованных форм (или факторизованных ДНФ) путем поиска факторов – общих частей конъюнкций или дизъюнкций системы ДНФ. Факторизованная форма, по сути, является скобочной формой задания многоуровневого представления ДНФ. Специальный термин вводится для обозначения того факта, что заданная система логических уравнений подготовлена для многоуровневой реализации, т. е. удовлетворяет требованиям, накладываемым технологическим базисом, в котором предполагается реализовать описываемую ею комбинационную схему. Преобразование системы ДНФ в факторизованную форму, которой соответствует многоуровневая реализация из вентилей с ограниченным числом входов, разбивается в программном комплексе проектирования КМОП-схем [2] на два этапа [1, 3]:

– совместную нетривиальную факторизацию системы ДНФ. На этом этапе выделяются факторы (в виде конъюнкций или ДНФ), которые имеют ограниченную длину (число литералов), диктуемую целевым технологическим КМОП-базисом, и входят в несколько конъюнкций или ДНФ;

– построение скобочных выражений ДНФ независимо для каждой из функций системы. Оно основано на итеративном вынесении общих литералов конъюнкций заданной ДНФ D за скобки путем декомпозиции вида $D = k(A) + B$, где D, A и B – ДНФ (дизъюнкции некоторого множества конъюнкций), а k – конъюнкция, состоящая из некоторого множества литералов, общих для всех конъюнкций из A.

В результате выполнения первого этапа получается многоуровневая И-ИЛИ-схема (с использованием инверторов), в которой могут быть элементы с числом входных полюсов, не привязанным к целевому технологическому базису. В многоуровневой схеме, являющейся результатом выполнения второго этапа, число входных полюсов всех элементов не превосходит максимума числа входов вентилей, диктуемого технологическим базисом.

2. Организация эксперимента по исследованию подходов к технологически независимой оптимизации

Синтез схем производился в базисе отечественной библиотеки заказных КМОП СБИС. В качестве средства проектирования схем использовался промышленный синтезатор LeonardoSpectrum (версия 2011a.4) [4], настроенный на целевую отечественную библиотеку КМОП СБИС. На вход синтезатора подавались VHDL-описания представлений двух- и многоуровневых И-ИЛИ-схем.

Проводилось сравнение результатов синтеза схем из библиотечных элементов, полученных с помощью синтезатора LeonardoSpectrum, исходя из четырех представлений одних и тех же тестовых систем булевых функций из набора Berkeley PLA test set [5]:

1) двухуровневые И-ИЛИ-схемы, соответствующие исходным описаниям систем булевых функций; в этом случае оптимизация проектируемой схемы производилась только средствами синтезатора LeonardoSpectrum (исследуемый вариант 1);

2) двухуровневые И-ИЛИ-схемы, полученные после совместной минимизации систем булевых функций с помощью программы [6], реализующей модифицированный метод ESPRESSO (вариант 2);

3) многоуровневые И-ИЛИ-схемы, полученные после совместной минимизации систем булевых функций и последующей совместной нетривиальной факторизации полученной системы ДНФ (вариант 3);

4) многоуровневые И-ИЛИ-схемы из элементов с ограниченным числом входных полюсов, полученные после совместной минимизации систем булевых функций и последующей алгебраической декомпозиции (вариант 4).

Представления схем по вариантам 2–4 были получены с помощью программного комплекса проектирования цифровых интегральных КМОП-микросхем с пониженным энергопотреблением ELS [2]. На рис. 1 и 2 показаны окна, демонстрирующие функциональные возможности комплекса в части технологически независимой оптимизации логических схем. Исследуемые ва-

рианты 2–4 технологически независимой оптимизации функциональных описаний реализованы в комплексе соответственно проектными операциями «Двухуровневая оптимизация», «Многоуровневая оптимизация» и «Синтез схем из вентилей».

Из трех возможных проектных операций двухуровневой оптимизации выбрана «Минимизация Espresso» (рис. 1), реализующая модифицированный метод Espresso совместной минимизации системы булевых функций. В результате выполнения этой операции получается оптимизированное двухуровневое И-ИЛИ-представление схемы.

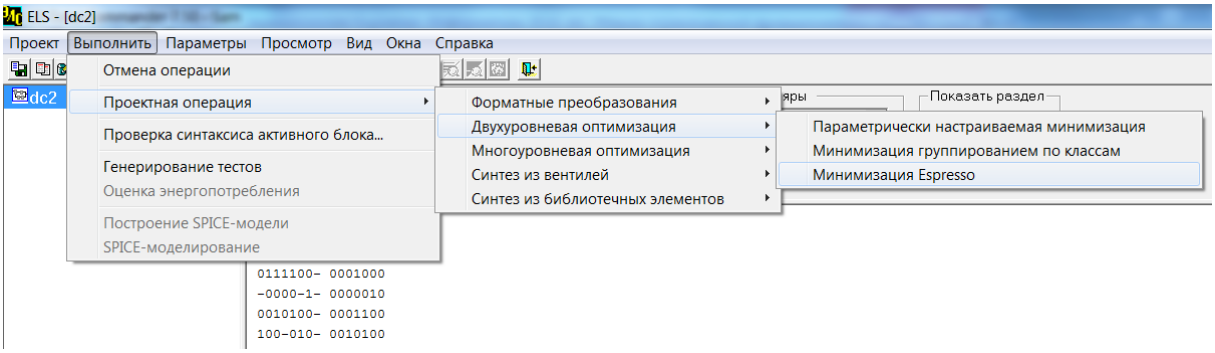


Рис. 1. Вызов процедуры совместной минимизации систем булевых функций

На рис. 2 представлено окно выбора одной из процедур выполнения «Многоуровневая оптимизация», а именно процедуры «Совместная факторизация» системы булевых функций. Данная процедура выделяет факторы (в виде конъюнкций или ДНФ), которые имеют ограниченную длину (число литералов), диктуемую целевым технологическим КМОП-базисом, и входят в несколько конъюнкций или ДНФ. В результате выполнения этой операции получается оптимизированное многоуровневое И-ИЛИ-представление схемы.

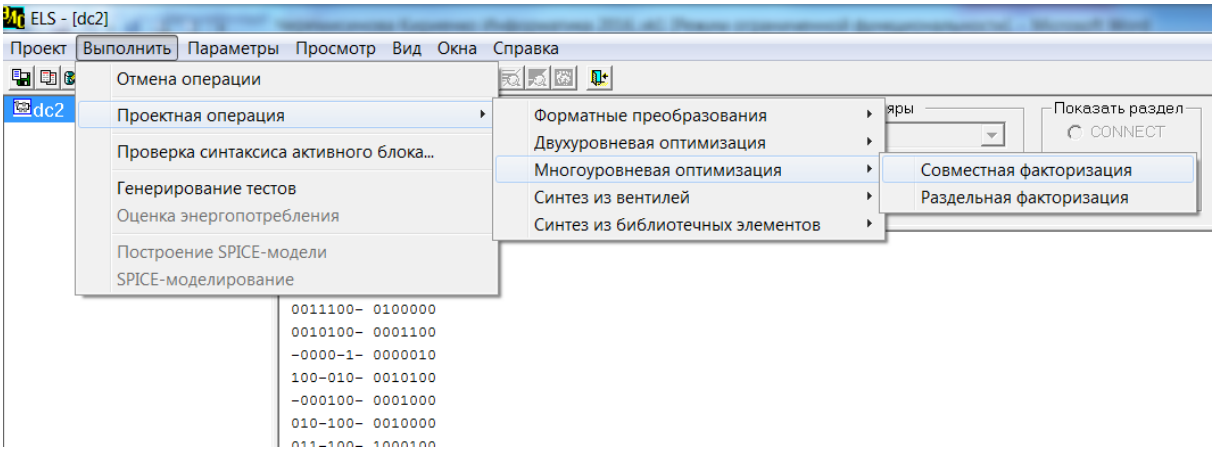


Рис. 2. Вызов процедуры совместной факторизации систем булевых функций

На рис. 3 показано окно вызова одной из процедур выполнения «Синтез из вентилей», а именно процедуры синтеза схем из многовходовых вентилей. В ходе выполнения этой процедуры осуществляется алгебраическая декомпозиция систем ДНФ булевых функций, состоящая из этапов совместной нетривиальной факторизации системы ДНФ и построения скобочных выражений ДНФ независимо для каждой из функций системы. В результате получается оптимизированное многоуровневое представление схемы, ориентированное на заданный отечественный технологический КМОП-базис, на который настроен программный комплекс ELS [2].

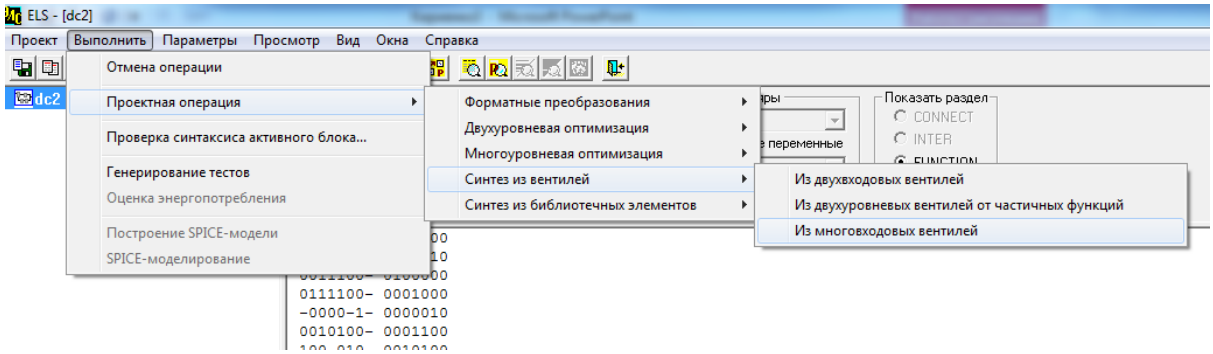


Рис. 3. Вызов процедуры синтеза схем из многовходовых вентилей

Полученные данные исследования различных вариантов технологически независимой оптимизации для каждой из тестовых схем приведены в таблице. В ходе исследования для каждого из тестовых примеров было получено четыре варианта схемных решений. Состав и последовательность вызова оптимизационных процедур для каждого исследуемого варианта оптимизации представлены на рис. 4.

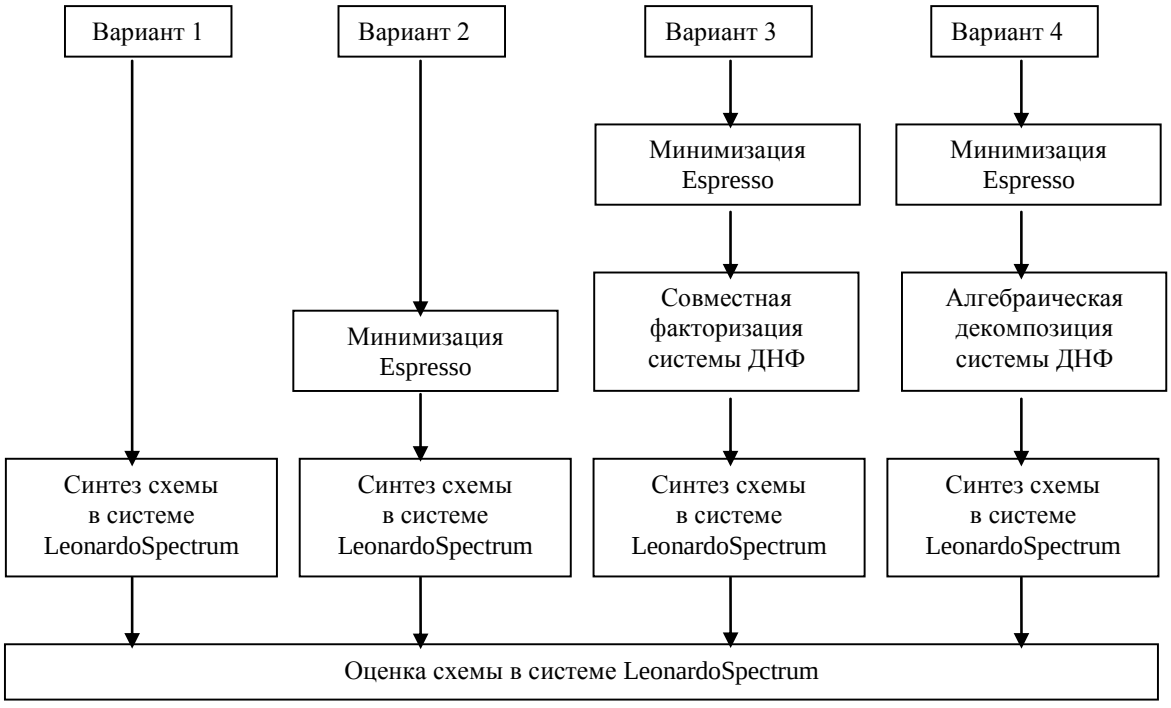


Рис. 4. Исследуемые варианты оптимизационных преобразований функциональных описаний схем

Как видно из рис. 4, для каждого из полученных четырех вариантов описаний, реализующих одну и ту же систему булевых функций, синтезируется схема в заданном библиотечном КМОП-базисе с помощью синтезатора. Оценка сложности синтезированных схем производится средствами LeonardoSpectrum: подсчитывается площадь схемы (рис. 5) и задержка критического пути (рис. 6). Площадь схемы подсчитывается (в условных единицах) как сумма площадей входящих в нее библиотечных элементов при размещении их на кристалле СБИС. Задержка критического (наиболее «длинного») пути измеряется в наносекундах.

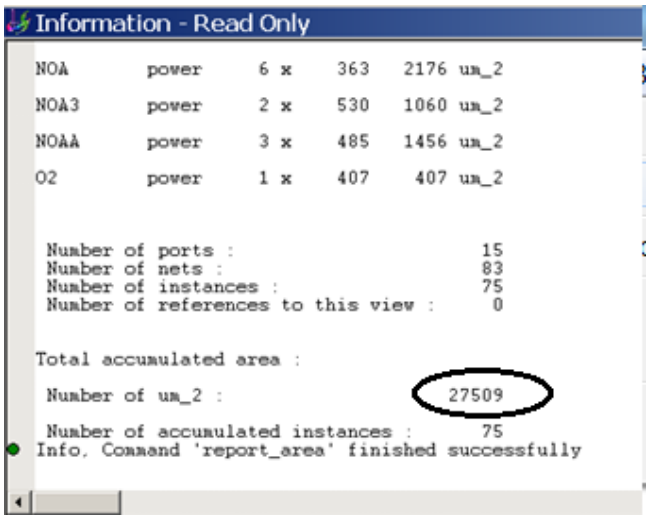


Рис. 5. Оценка площади синтезированной схемы

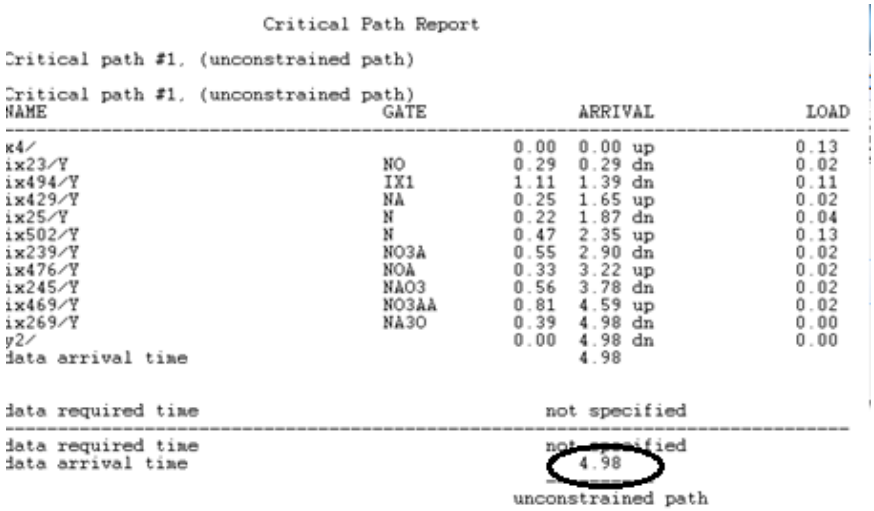


Рис. 6. Оценка задержки синтезированной схемы

3. Результаты исследования подходов к технологически независимой оптимизации

Для каждого тестового примера в таблице приведены следующие параметры:

– числа *n*, *m* и *k* аргументов, функций и конъюнкций тестовой системы ДНФ;

– величины s_1 и τ_1 площади и задержки схем, полученных средствами синтезатора LeonardoSpectrum по варианту 1;

– для вариантов 2–4 процент относительного выигрыша/проигрыша *i*-го варианта оптимизации по площади схемы относительно варианта 1: $\%s_i = ((s_1 – s_i)/s_1) \cdot 100\ \%$;

– для вариантов 2–4 процент относительного выигрыша/проигрыша *i*-го варианта оптимизации по задержке схемы относительно варианта 1: $\%\tau_i = ((\tau_1 – \tau_i)/\tau_1) \cdot 100\ \%$.

Если *i*-й вариант оптимизации по площади или задержке схемы выигрывает у варианта по площади или задержке схемы, соответствующий параметр ($\%s_i$ или $\%\tau_i$) будет иметь положительное значение, в противном случае (в случае проигрыша) он будет отрицательным.

В таблице жирным курсивом отмечены выигрышные варианты оптимизации (по площади и задержке схемы) для каждого тестового примера, а в последней строке для каждого варианта оптимизации приведено число тестовых примеров, для которых использование соответствующего варианта позволило получить лучшую схему по каждому из оцениваемых критериев (площадь и задержка).

Результаты экспериментального исследования влияния
логической оптимизации на площадь и задержку синтезированных схем

Имя схемы	Параметры схемы (n, m, k)	Оптимизация средствами Leonardo Spectrum (вариант 1)		Минимизация Espresso (вариант 2)		Совместная факторизация (вариант 3)		Алгебраическая декомпозиция (вариант 4)	
		s_1	τ_1	$\%s_2$	$\%\tau_2$	$\%s_3$	$\%\tau_3$	$\%s_4$	$\%\tau_4$
b9	16, 5, 123	***26148***	***3,92***	–15,6	–17,1	–8,6	–14,5	–6,7	–21,2
br1	12, 8, 34	32615	6,54	19,5	16,4	29,2	***22,2***	***38,5***	13,6
br2	12, 8, 35	22632	6,44	5,1	***5,1***	20,3	–1,6	***22,3***	–15,7
dc2	8, 7, 58	32375	4,73	7,5	***12,3***	***16,1***	–1,5	15,0	–5,3
dist	8, 5, 256	110060	7,97	20,0	18,3	26,4	10,3	***30,2***	***25,1***
dk48	15, 17, 148	30082	7,44	41,3	***41,0***	48,7	39,4	***49,5***	39,2
f51m	8, 8, 256	276350	11,17	***91,0***	56,2	89,1	***67,0***	90,0	65,1
gary	15, 11, 446	107627	***6,66***	5,2	–22,4	8,8	–41,4	***11,8***	–31,8
in0	15, 11, 138	133786	9,39	19,4	***30,5***	33,5	15,0	***36,1***	24,8
in2	19, 10, 137	141336	7,77	1,8	***11,5***	22,2	4,1	***33,2***	3,6
life	9, 1, 512	24641	5,54	***2,7***	***11,6***	–26,2	9,7	–12,0	–11,7
max1024	10, 6, 1024	423288	10,77	54,1	***16,8***	53,0	4,5	***55,2***	11,9
m1	6, 12, 32	21773	5,34	–2,9	14,0	–1,8	3,9	***8,5***	***16,3***
m2	8, 16, 96	85960	7,51	28,9	***3,5***	27,9	–7,9	***30,0***	–3,3
m3	8, 16, 128	97120	7,86	15,9	***10,7***	***18,6***	–0,6	14,3	0,8
m4	8, 16, 256	222447	11,01	34,9	***25,5***	43,6	17,2	***45,9***	15,8
mlp4	8, 8, 256	164967	8,06	36,9	***23,3***	40,7	6,3	***43,8***	21,8
root	8, 5, 256	55750	6,47	***25,8***	23,8	22,0	24,3	23,3	***30,9***
tms	8, 16, 30	46465	7,36	–10,5	2,7	***3,6***	***14,7***	–3,9	9,8
z9sym	9, 1, 420	59896	9,9	***28,2***	38,7	19,7	35,2	23,8	***44,9***
Выигрыш:		**1**	**2**	**4**	**11**	**3**	**3**	**12**	**4**

По результатам эксперимента можно сделать следующие выводы:

1. Использование при синтезе схем из элементов КМОП-библиотеки предварительного этапа технологически независимой оптимизации по варианту 4 позволило получить наибольший выигрыш по площади синтезируемых схем (в 12 случаях из 20). Значения относительных выигрышей по площади находятся в пределах от 8,5 до 55,2 %. При этом полученные схемы проигрывают по быстродействию схемам, полученным по варианту 2 (4 выигрыша против 11). Значения относительных выигрышей по быстродействию находятся в пределах от 16,3 до 44,9 %.

2. Использование при синтезе схем из элементов КМОП-библиотеки предварительного этапа оптимизации по варианту 3 также позволяет в некоторых случаях (в 3 из 20) уменьшить

площадь схем и повысить их быстродействие (в 3 случаях из 20). Значения относительных выигрышей по площади находятся в пределах от 3,6 до 18,6 %. Значения относительных выигрышей по быстродействию находятся в пределах от 14,7 до 67,0 %.

3. Использование при синтезе схем из элементов КМОП-библиотеки предварительного этапа оптимизации по варианту 2 (минимизация Espresso) продемонстрировало эффективность этого метода для повышения быстродействия схемы. Получен выигрыш в 11 случаях из 20. Метод также позволяет в некоторых случаях (в 4 из 20) уменьшить площадь схем. Значения относительных выигрышей по площади находятся в пределах от 2,7 до 91,0 %. Значения относительных выигрышей по быстродействию находятся в пределах от 3,5 до 41,0 %.

4. Только в одном случае из 20 не получено уменьшение площади КМОП-схемы при выполнении предварительной технологически независимой оптимизации (с помощью вариантов 2–4). Здесь реализованные в LeonardoSpectrum процедуры оптимизации позволили получить для примера b9 лучший результат.

В целом выполнение процедур совместной минимизации систем булевых функций, совместной нетривиальной факторизации системы ДНФ, а также алгебраической декомпозиции системы ДНФ позволяет в большинстве случаев улучшить качество проектируемых схем с помощью промышленного синтезатора LeonardoSpectrum. Проигрыш вариантов 3 и 4 варианту 2 по быстродействию получаемых схем объясняется тем, что при оптимизации схем по вариантам 3 и 4 критерий быстродействия принимается во внимание в последнюю очередь, а в первую очередь рассматриваются критерии сокращения площади и энергопотребления. При синтезе схем по варианту 2 критерий быстродействия, наоборот, является более приоритетным.

Заключение

Проведены сравнительные исследования влияния процедур технологически независимой оптимизации функциональных описаний на сложность и задержку схем, синтезированных из библиотечных элементов с помощью промышленного синтезатора LeonardoSpectrum. Показано, что использование процедур минимизации булевых функций и алгебраической декомпозиции систем булевых функций приводит к существенному выигрышу в сложности и задержке результирующих схем.

Список литературы

1. Черемисинова, Л.Д. Синтез и оптимизация комбинационных структур СБИС / Л.Д. Черемисинова. – Минск, 2007.

2. Автоматизация логического синтеза КМОП-схем с пониженным энергопотреблением / П.Н. Бибило [и др.] // Программная инженерия. – 2013. – № 8. – С. 35–41.

3. Черемисинова, Л.Д. Синтез многоуровневых логических схем с учетом энергопотребления / Л.Д. Черемисинова, Н.А. Кириенко // Информационные технологии. – 2013. – № 3. – С. 8–14.

4. Бибило, П.Н. Системы проектирования интегральных схем на основе языка VHDL / StateCAD, ModelSim, LeonardoSpectrum / П.Н. Бибило. – М. : Солон-Пресс, 2005. – 384 с.

4. Berkeley PLA test set [Electronic resource]. – 2015. – Mode of access : http://www1.cs.columbia.edu/~cs6861/sis/espresso-examples. – Date of access : 25.02.2015.

6. Черемисинов, Д.И. Минимизация двухуровневых КМОП-схем с учетом энергопотребления / Д.И. Черемисинов, Л.Д. Черемисинова // Информационные технологии. – 2011. – № 5. – С. 17–23.

Поступила 31.05.2016

Объединенный институт проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: kir@newman.bas-net.by,
cld@newman.bas-net.by

N.A. Kirienko, L.D. Cheremisinova

ANALYSIS OF EFFECTIVENESS OF TECHNOLOGY INDEPENDENT OPTIMIZATION OF CMOS CIRCUIT FUNCTIONAL DESCRIPTIONS

The influence of procedures of technology independent optimization of functional descriptions on complexity and delay of circuits made of CMOS VLSI library elements is investigated. The effectiveness of the optimization methods implemented in the synthesizer LeonardoSpectrum, the methods of minimization of Boolean functions systems in DNF and algebraic decomposition was compared. Experimental analysis was shown that sequential fulfilment of procedures of minimization of Boolean functions systems and algebraic decomposition during the technology independent optimization stage results in the great reduction of the circuit complexity and delay.

УДК 519.8

И.В. Рубанов, М.С. Баркетов, М.Я. Ковалев

МЕТОДЫ ПОИСКА НЕСКОЛЬКИХ РЕШЕНИЙ СИСТЕМЫ РАЗНОСТНЫХ И ИНТЕРВАЛЬНЫХ ОГРАНИЧЕНИЙ

Рассматриваются методы поиска решений систем линейных неравенств специального вида, состоящих из разностных неравенств с двумя переменными и интервальных ограничений с одной переменной. Для решения таких систем предлагаются два подхода, с помощью которых могут быть найдены два экстремальных («максимальное» и «минимальное») решения, а также некоторые другие решения. Первый подход основан на методе Фурье – Моцкина, второй – на представлении системы в виде сети ограничений.

Введение

Пусть имеется сеть пересекающихся маршрутов и n объектов (например, самолетов), каждый из которых совершает непрерывное движение по какому-либо определенному маршруту. Пусть известно время движения каждого объекта от начала маршрута до каждой точки пересечения с другими маршрутами и заданы ограничения на минимальное расстояние по времени сближения объектов, а также допустимые временные интервалы моментов начала движения каждого объекта. Требуется составить допустимое расписание начала движения объектов.

Данная задача может быть сведена к решению системы дизъюнктивно объединенных пар неравенств [1]. Одним из способов решения этой системы может являться какой-либо метод перебора, когда перебираются все возможные варианты выбора неравенства из каждой дизъюнктивной пары. В процессе такого перебора получается система линейных неравенств вида

$$\begin{cases} x_i - x_j \le a_{ji}, \quad (i, j) \in IJ \subseteq N \times N; \\ x_i^{\min} \le x_i \le x_i^{\max}, \quad i \in N, \end{cases} \tag{1}$$

где $N = \{1, \ ..., \ n\}$.

Переменные x_i и x_j выражают моменты начала движения соответствующего объекта. Разностное ограничение $x_i - x_j \le a_{ji}$ гарантирует, что объекты начнут движение так, чтобы между проходами ими точки пересечения маршрутов был гарантированный минимальный промежуток времени. Интервальное ограничение $x_i^{\min} \le x_i \le x_i^{\max}$ задает время начала движения объекта i. Система (1) не обязательно является полной в том смысле, что она может включать не все возможные разностные и интервальные ограничения.

Задача (1) представляет собой известный класс систем линейных неравенств, называемый DBM – Difference-bounds matrices (см., например, [2]), метод решения которых может быть небольшой модификацией [2] хорошо известных методов решения обычных систем разностных неравенств (difference constraits), основанных на поиске кратчайших путей в построенной по (1) сети ограничений (см., например, [3–6]). Рассматриваемый в [2] метод позволяет за время $O(n^3)$ найти решение (1), соответствующее одной из экстремальных точек многогранника ограничений, в которой значения каждой из переменных являются максимально возможными.

В настоящей статье рассматриваются методы поиска нескольких решений (1), возможно, не принадлежащих границам исходного многогранника. Задачи поиска таких решений могут быть сведены к инкрементальным, позволяющим находить решения системы, к которой добавляются новые ограничения (при условии, что решение исходной системы известно), за время, меньшее, чем потребовалось бы для решения системы с исходными и новыми ограничениями.

В статье [7] рассматриваются инкрементальные методы решения системы разностных неравенств с добавленными ограничениями за время $O(m+n\log n)$, где m – общее количество ограничений. Приведенная оценка является оценкой сложности одной из реализаций алгоритма Дийкстры, используемого в предлагаемых в [7] методах.

В настоящей работе предлагаются два метода решения системы DBM, при помощи которых могут быть найдены новые значения переменных x_j при вынужденном изменении значения одной из них (x_i) без изменения набора ограничений при условии, что значение x_i находится в подходящем вычисляемом интервале. Для каждого значения переменной из данного интервала может быть найдено допустимое решение задачи с соответствующим значением данной переменной. Один из предлагаемых алгоритмов основан на универсальном методе Фурье – Моцкина и включает как проверку на совместность системы (1), так и построение решений. Первое решение заданной системы может быть найдено за время $O(n^3)$, некоторые другие ее решения при уже найденном одном – за время $O(n^2)$. Другой метод основан на представлении системы в виде сети ограничений и позволяет найти два начальных решения за время работы алгоритма поиска кратчайших путей от одной вершины к остальным, выявляющего циклы отрицательной длины, например, за время $O(n^3)$. Этот метод позволяет найти и другие решения, верхняя оценка вычислительной сложности построения которых так же, как для метода, приведенного в [7], не выше оценок для алгоритма Дийкстры (например, $O(m+n\log n)$).

1. Алгоритм Фурье – Моцкина

Одним из известных методов решения систем линейных неравенств является метод Фурье – Моцкина [8]. Он заключается в построении последовательности систем-следствий из данной системы, причем каждая последующая система содержит на одну переменную меньше. Если известно допустимое решение какой-либо системы из данной последовательности, то можно построить допустимое решение для предыдущей системы. Более подробно суть метода можно изложить следующим образом.

Пусть дана система m линейных неравенств с n неизвестными $x_1, x_2, ..., x_n$. Выберем любую переменную x_i и запишем все содержащие ее неравенства так, чтобы коэффициент при этой переменной равнялся +1 и она находилась в левой части неравенства:

$$\begin{cases} x_i \le f_1(X_i), \\ ..., \\ x_i \le f_{m_1}(X_i), \\ x_i \ge f_{m_1+1}(X_i), \\ ..., \\ x_i \ge f_{m_1+m_2}(X_i), \\ ... \end{cases} \qquad (2)$$

Здесь X_i – вектор переменных системы за исключением i-й, m_1 неравенств имеют тип «$\le$» и m_2 неравенств имеют тип «$\ge$». Далее содержащие x_i неравенства из (2) перепишем в виде

$$\begin{cases} f_{m_1+1}(X_i) \le x_i \le f_1(X_i), \\ \ldots, \\ f_{m_1+1}(X_i) \le x_i \le f_{m_1}(X_i), \\ f_{m_1+2}(X_i) \le x_i \le f_1(X_i), \\ \ldots, \\ f_{m_1+2}(X_i) \le x_i \le f_{m_1}(X_i), \\ \ldots \end{cases} \quad (3)$$

Система (3) содержит все возможные сочетания право- и левосторонних относительно x_i ограничений. Если теперь удалить рассматриваемую переменную и переписать (3) в виде

$$\begin{cases} f_{m_1+1}(X_i) \le f_1(X_i), \\ \ldots, \\ f_{m_1+1}(X_i) \le f_{m_1}(X_i), \\ f_{m_1+2}(X_i) \le f_1(X_i), \\ \ldots, \\ f_{m_1+2}(X_i) \le f_{m_1}(X_i), \\ \ldots, \end{cases} \quad (4)$$

то вместе с остальными, изначально не содержавшими x_i неравенствами получится система-следствие, при любом решении которой, если оно будет найдено, можно будет поместить значение x_i в интервал из (4) с наибольшей нижней и наименьшей верхней границами, дополнив набор значений переменных до решения исходной системы. Если одно из рассматриваемых неравенств приводится к виду $\text{const}_1 \le \text{const}_2$ при $\text{const}_1 > \text{const}_2$, то исходная система несовместна. Применяя описанные действия к получившейся системе (4), исключим следующую переменную и получим очередную систему-следствие. После $n-1$ шагов будут удалены все переменные, кроме одной, для которой будет получен интервал изменения в явном виде. Выбрав ее значение, подставим его в систему, получившуюся на предыдущем шаге, и получим в явном виде диапазон для предыдущей переменной. Так повторяем до первой из удаленных переменных. В настоящей статье этап удаления переменных будем называть обратным ходом алгоритма, этап подстановки – прямым.

Простое описание алгоритма Фурье – Моцкина на примере его применения для решения небольших задач линейного программирования дано в монографии [4]. В общем случае этот метод является вычислительно сложным, поскольку на этапе удаления переменных может возникнуть $(m/2)^{2^n}$ неравенств. Поэтому он применяется для небольших систем или систем специального вида.

Далее приводится оценка сложности получения нескольких возможных решений методом Фурье – Моцкина для специфичной системы DBM.

1.1. Обратный ход алгоритма

Рассмотрим случай, в котором система (1) является полной, т. е. в нее входят все возможные разностные и интервальные ограничения. Их количество равно $2C_2^n + 2n = n^2 + n$.

Выберем в рассматриваемой системе для исключения любую переменную x_{i_1}. Она встретится $n-1$ раз с коэффициентом $+1$ и $n-1$ раз с коэффициентом -1 во всех неравенствах с двумя переменными. Добавив по одному случаю, вносимому интервальными ограничениями, получим n случаев появления выбранной переменной с каждым знаком. Матрицу коэффици-

ентов полной системы, включающую только неравенства с рассматриваемой переменной, можно записать следующим образом:

$$\begin{array}{cccc} -1 & 0 & \ldots & 1 \\ 0 & -1 & \ldots & 1 \\ & \ldots & & \\ 0 & 0 & \ldots & 1 \\ 1 & 0 & \ldots & -1 \\ 0 & 1 & \ldots & -1 \\ & \ldots & & \\ 0 & 0 & \ldots & -1 \end{array} \qquad (5)$$

Для исключения переменной по алгоритму Фурье – Моцкина комбинирование лево- и правосторонних относительно x_i неравенств представим как попарное сложение неравенств с коэффициентами $+1$ и -1 при исключаемой переменной. Такая модификация метода предложена, например, в статье [9]. Возможных пар складываемых неравенств, в одно из которых переменная входит с коэффициентом $+1$, а в другое – с коэффициентом -1, будет n^2.

После сложения в левых частях результирующих неравенств либо не останется переменных, либо останется одна с коэффициентом $+1$ или -1, либо две с коэффициентами $+1$ и -1 (вторые из переменных в складываемых неравенствах в случае их присутствия будут с противоположными исключаемой знаками), т. е. вид системы не изменится. Возможные комбинации строк коэффициентов и их суммы таковы:

$$\begin{array}{cccc} -1 & 0 & \ldots & 1 \\ + & & & \\ 1 & 0 & \ldots & -1 \\ = & & & \\ 0 & 0 & \ldots & 0 \end{array} \qquad \begin{array}{cccc} -1 & 0 & \ldots & 1 \\ + & & & \\ 0 & 0 & \ldots & -1 \\ = & & & \\ -1 & 0 & \ldots & 0 \end{array} \qquad \begin{array}{cccc} -1 & 0 & \ldots & 1 \\ + & & & \\ 0 & 1 & \ldots & -1 \\ = & & & \\ -1 & 1 & \ldots & 0 \end{array} \qquad \ldots \qquad (6)$$

Результирующих неравенств будет n^2, из них n будут с нулевыми коэффициентами, т. е. число неравенств с ненулевыми коэффициентами равно $n^2 - n$. Среди остальных, не содержащих переменную x_{i_1} и не подвергшихся суммированию, будет также $n^2 - n$ неравенств. Далее необходимо рассмотреть неравенства, включающие одинаковую пару переменных с одинаковыми знаками и оставить из таких неравенств наиболее сильное (с минимальной правой частью). В итоге в системе останется $n^2 - n = (n-1)^2 + (n-1)$ неравенств.

Можно хранить информацию о системе в виде матрицы A_{ij}^n, $0 \le i \le n$, $0 \le j \le n$, $i \ne j$, где a_{ij} – значение правой части ограничения с коэффициентом -1 при переменной x_i и коэффициентом $+1$ при переменной x_j ($i = 0$ ($j = 0$) в случае отсутствия переменной с коэффициентом -1 ($+1$)). В случае отсутствия этого ограничения в системе используется специальный символ. Таким образом, вместо суммирования, последующего поиска и удаления нужных строк матриц коэффициентов в представлении (5) можно оперировать элементами матриц правых частей. Пусть переменные удаляются с n-й по первую. Тогда для удаления n-й переменной полной системы потребуется просуммировать каждый j-й элемент строки n с каждым i-м элементом столбца с тем же номером в A_{ij}^n. Из n^2 получившихся значений n элементов $a_{nk} + a_{kn}$ (соответствующих результирующим строкам с нулевыми коэффициентами в представлении (5)) проверяются на неотрицательность для выявления несовместности, а каждый из остальных $n^2 - n$ элементов $a_{in} + a_{nj}$ сравнивается с элементом a_{ij} матрицы A_{ij}^n и минимальный из них записывается в матрицу A_{ij}^{n-1}. Для удаления следующей переменной процесс повторяется

с A_{ij}^{n-1} и далее итеративно до получения матрицы A_{ij}^{1}. Таким образом, значения a_{ij}^{k-1} матрицы A_{ij}^{k-1} можно получить из A_{ij}^{n} по рекуррентной формуле

$$a_{ij}^{k-1} = \min\left(a_{ij}^{k}, a_{ik}^{k} + a_{kj}^{k}\right), \tag{7}$$

известной как формула для расчета элементов матрицы кратчайших расстояний по алгоритму Флойда – Уоршалла, а совместность определяется по отсутствию в A_{ij}^{k} отрицательных диагональных элементов, что может быть использовано в каком-либо доказательстве сетевого метода (см. разд. 2) решения системы (1).

Очевидно, что при таком хранении данных случай полной системы будет самым вычислительно сложным, поскольку требуется проводить операции с каждым элементом матриц A_{ij}. Подсчитаем возможное количество операций, которое потребуется на обратный ход алгоритма. При исключении первой переменной для суммирования строк по матрице A_{ij}^{n} требуется n^2 операций, для проверки неотрицательности – n операций, для сравнивания и формирования следующей матрицы – $2\left(n^2 - n\right)$ операций, т. е. всего $n^2 + n + 2n^2 - 2n = 3n^2 - n$ операций. Для исключения второй переменной потребуется $3\left(n-1\right)^2 - \left(n-1\right)$ операций и т. д. Всего на обратный ход алгоритма, т. е. на исключение всех переменных, кроме одной, потребуется следующее количество элементарных операций:

$$\left[3n^2 - n\right] + \left[3\left(n-1\right)^2 - \left(n-1\right)\right] + \ldots + 4 = \sum_{i=2}^{n}\left(3i^2 - i\right) = n^3 + n^2 - 2\,. \tag{8}$$

Таким образом, вычислительная сложность обратного хода алгоритма равна $\mathrm{O}(n^3)$.

1.2. Прямой ход алгоритма

После обратного хода метод Фурье – Моцкина работает следующим образом. Пусть переменные удалялись с n-й по первую. Из диапазона последней оставшейся переменной x_1 выбирается произвольное значение (см. общее описание метода в начале разд. 1) и подставляется в полученную на предпоследнем этапе систему с двумя переменными. Количество неравенств в ней $2^2 + 2 = 6$, и значение переменной x_1 требуется подставить в $2(2-1) = 2$ неравенства с ненулевыми коэффициентами при обеих переменных. В двух из получившихся неравенств предпоследняя исключавшаяся переменная присутствует с коэффициентом $+1$ (среди них нужно найти одно с минимальной правой частью) и в двух – с коэффициентом -1 (среди них аналогично нужно найти одно). Найденные неравенства определяют ее диапазон. Если произвольное значение может быть выбрано за одну операцию, преобразование правых частей после подстановки потребует две операции, а поиск минимальных правых частей – четыре операции, тогда общее количество операций равно семи.

Далее выбирается произвольное значение из полученного для предпоследней переменной интервала, значения последней и предпоследней переменных подставляются в $2(3-1) = 4$ неравенства, содержащие ненулевые коэффициенты при двух переменных (одной из подставляемых и переменной с искомым диапазоном), отыскиваются два неравенства с минимальными правыми частями среди $2 \cdot 3 = 6$ неравенств с ненулевыми коэффициентами при искомой переменной, всего $1 + 2(3-1) + 2 \cdot 3 = 11$ и т. д. На завершающем этапе прямого хода выбирается произвольное значение из диапазона для первой исключавшейся переменной x_n.

Можно воспользоваться матрицами A_{ij}^{k} правых частей, полученными на этапе обратного хода. Тогда на шаге $k-1$ прямого хода каждое из $k-1$ выбранных значений x_i^0 переменных x_i потребуется прибавить к элементам ik, $i \neq 0$, матрицы A_{ij}^{k} и выбрать среди них минимальное значение, получив верхнюю границу диапазона переменной x_k, далее отнять x_i^0 от элементов

$ki, i \neq 0$, выбрать среди них минимальное значение и взять его со знаком минус, получив нижнюю границу диапазона x_k.

Вычислительная сложность прямого хода алгоритма определяется следующим образом:

$$1+\left[2(2-1)+2\cdot 2\right]+1+\left[2(3-1)+2\cdot 3\right]+\ldots+\left[2(n-1)+2n\right]+1= \\ =n+2\sum_{i=2}^{n}(2i-1)=n+2n^2-2. \tag{9}$$

Суммируя с (8) и прибавив n^2+n операций на построение начальной матрицы A_{ij}^n, получаем вычислительную сложность метода Фурье – Моцкина для системы (1):

$$n^2+n+n^3+n^2-2+2n^2+n-2=n^3+4n^2+2n-4. \tag{10}$$

Вычислительная сложность решения при таком хранении данных для неполной системы, как уже говорилось ранее, может быть только меньше. Поэтому справедливо следующее

Утверждение. *Вычислительная сложность решения системы разностных неравенств с дополнительными ограничениями на минимумы и максимумы переменных описанным методом Фурье – Моцкина – не более* $O(n^3)$.

Рассмотренный в этом разделе метод можно использовать для построения нескольких решений системы (1), возможных при подходящих условиях. К примеру, пусть на этапе прямого хода для построения решения выбирались на каждом шаге минимальные значения переменных из определяемых для них на предыдущих шагах допустимых интервалов. Если теперь значение одной из переменных x_i увеличить в пределах допустимого интервала, потребуется переопределить интервалы и значения переменных на шагах от $i+1$ до n. Очевидно, что верхняя оценка сложности такого повторного построения решения будет не выше $O(n^2)$. Если значение x_i выходит за пределы допустимого интервала, можно попытаться повторно выбрать значения переменных на предыдущих шагах. В разд. 2 будет доказано, что существуют два решения системы (1), одно из которых состоит из минимально возможных среди любых решений значений переменных, второе – из максимально возможных. Таким образом, если строить решение, выбирая на каждом шаге минимальное (максимальное) значение каждой переменной, то ни одно из значений не сможет быть понижено (повышено) в каком-либо повторном решении.

Нетрудно видеть, что сложность метода Фурье – Моцкина остается невысокой и для решения более общей по отношению к (1) задачи «unit two variable per inequality» (UTVPI), представляющей собой систему неравенств, в каждом из которых присутствует не более двух переменных с коэффициентами +1 или –1, так как одна из переменных в каждом из двух комбинируемых неравенств всегда удаляется и среди результирующих неравенств остается мало независимых. Применение для такой задачи алгоритма Фурье – Моцкина так же, как и для рассматриваемой задачи (1), может быть способом построения нескольких ее решений. В работе [9] предлагается реализация метода Фурье – Моцкина для поиска целочисленных решений задачи UTVPI.

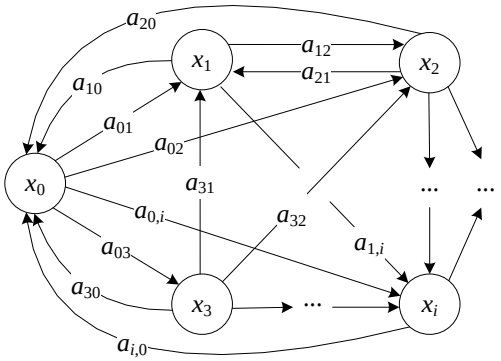
2. Сетевой метод

Добавим в (1) фиктивную переменную x_0 и будем искать решение системы

$$\begin{cases} x_i - x_j \le a_{ji}, \\ x_0 = 0, \end{cases} \quad (i,j) \in IJ_0 \subseteq N_0 \times N_0, \tag{11}$$

эквивалентной (1), где $N_0 = \{0,...,n\}$, $a_{0,i} = x_i^{\max}, ..., a_{i,0} = -x_i^{\min}$. Не ограничивая общности, предположим, что для каждой переменной указаны конечные границы ее значений, т. е. в формулировке (11) заданы по два неравенства с переменной x_0.

Создадим взвешенный ориентированный граф G, в котором каждой вершине поставим в соответствие переменную x_i (будем обозначать такую вершину как i), а каждой дуге – разностное ограничение $x_j - x_i \le a_{ij}$ системы (11) так, чтобы дуга была направлена от вершины i к вершине j, вес дуги – a_{ij} (рисунок).



Сеть ограничений, построенная по системе (1)

Далее сформулируем в виде теоремы известный результат о системе DBM [2], которую докажем аналогично построению в [5] для обычных систем разностных неравенств. Данная формулировка легко позволяет получить сразу два решения системы (1).

Теорема. *Система разностных и интервальных ограничений совместна тогда и только тогда, когда в G отсутствуют циклы отрицательной длины. В случае ее совместности величины $l_{0,i}$ ($-l_{0,i}$), каждая из которых равна кратчайшему расстоянию от вершины 0 до вершины i (от вершины i к вершине 0) в G, образуют решение системы.*

Доказательство. Если в G отсутствуют циклы отрицательной длины, то величины $l_{0,i}$ будут конечны и $x_i = l_{0,i}$ будут удовлетворять всем ограничениям (11) в силу того, что по свойству кратчайших расстояний кратчайшее расстояние $l_{0,j}$ от вершины 0 до вершины j будет всегда меньше, чем кратчайшее расстояние $l_{0,i}$ от вершины 0 до вершины i, непосредственно предшествующей j, плюс длина соединяющей их дуги a_{ij}:

$$\begin{aligned} & l_{0,j} \le l_{0,i} + a_{ij} \Leftrightarrow l_{0,j} - l_{0,i} \le a_{ij}, \\ & ..., \\ & l_{0,i} \le 0 + a_{0,i} \Leftrightarrow l_{0,i} - 0 \le a_{0,i}, \\ & \left(l_{0,0} = 0\right) \le l_{0,i} + a_{i,0} \Leftrightarrow 0 - l_{0,i} \le a_{i,0}, \\ & ... \end{aligned} \qquad (12)$$

Если цикл отрицательной длины в G присутствует, то, складывая неравенства, соответствующие дугам этого цикла, получим их несовместную линейную комбинацию

$$\left(x_i - x_{i_1} + x_{i_1} - x_{i_2} + x_{i_2} ... + x_{i_k} - x_i = 0\right) \le a_{i_1,i} + a_{i_2,i_1} + ... + a_{i,i_k} < 0.$$

Аналогично построениям (2) величины $-l_{i,0}$, каждая из которых равна кратчайшему расстоянию от вершины i до 0, взятому со знаком минус, также будут решениями системы в случае отсутствия в ней циклов отрицательной длины:

$$\begin{aligned}
&-l_{j,0} \ge -l_{i,0} - a_{ji} \Leftrightarrow l_{j,0} - l_{i,0} \le a_{ji} \Leftrightarrow x_i - x_j \le a_{ji},\\
&...,\\
&\left(-l_{0,0} = 0\right) \ge -l_{i,0} - a_{0,i} \Leftrightarrow -l_{i,0} - 0 \le a_{0,i} \Leftrightarrow x_i - x_0 \le a_{0,i},\\
&-l_{i,0} \ge 0 - a_{i,0} \Leftrightarrow 0 + l_{i,0} \le a_{i,0} \Leftrightarrow x_0 - x_i \le a_{i,0},\\
&...
\end{aligned} \qquad (13)$$

■

Отметим также, что известны другие подходы к решению системы разностных неравенств, которые могут быть адаптированы к решению системы (1). Добавив по одной дополнительной неотрицательной переменной («приведенной стоимости») к каждому неравенству, можно свести систему к системе равенств. Известно (см., например, [6]), что существует набор значений переменных (при этом все приведенные стоимости должны быть неотрицательны), удовлетворяющий данной системе равенств, тогда и только тогда, когда в соответствующем графе нет циклов отрицательной длины. Также в случае разрешимости системы известен метод построения такого набора значений переменных [6].

В случае неполной системы, если для какой-либо переменной x_i отсутствуют интервальные ограничения, в соответствующем графе G может отсутствовать путь из вершины 0 в вершину i и (или) обратный путь. Не нарушая общности метода, для такой переменной можно ввести в систему фиктивные неравенства $x_i - 0 \le M \wedge 0 - x_i \le M$, а в граф – дуги $a_{0,i}$, $a_{i,0}$ с заведомо большим весом M. Величину M можно определить как $M \ge \sum \left| a_{ij} \right| : a_{ij} < 0, \quad i, j = \overline{0,n}$, для того, чтобы гарантировать неотрицательную длину возникающих циклов.

Сформулируем несколько свойств области допустимых решений рассматриваемой системы. Учитывая некоторые из утверждений, можно строить решения (1) при возможном вынужденном изменении значения какой-либо переменной.

Перед этим сформулируем в виде леммы и докажем утверждение, которое, как и теорема, будет использоваться при доказательстве некоторых свойств.

Допустим, что переменные x_i, $i \ne 0$, приняли значения x_i^0. Назовем x^0-расстоянием между вершинами i и j на каком-либо пути в графе G величину

$$s_{i,j}^0 = x_j^0 - x_{i_k}^0 + x_{i_k}^0 - x_{i_{k-1}}^0 + ... + x_{i_1}^0 - x_i^0 = x_j^0 - x_i^0. \qquad (14)$$

Максимальным x^0-расстоянием между вершинами i и j на каком-либо пути будет длина этого пути:

$$s_{i,j} = a_{i_k,j} + a_{i_{k-1},i_k} + ... + a_{i,i_1}, \qquad (15)$$

где a_{ij} – длина дуги (i, j).

Лемма. Набор $x_1^0, x_2^0, ...$ значений переменных является решением (11) тогда и только тогда, когда x^0-расстояния между любой парой вершин на всех путях между ними не будут превышать длины этих путей: $s_{i,j}^0 \le s_{i,j}, \forall i, j$.

Доказательство. Достаточность этого утверждения очевидна: если x^0-расстояние не превышает длины пути для любой пары вершин, то это справедливо и для пар, входящих в разностные ограничения. Теперь докажем необходимость. Допустим, что набор $x_1^0, x_2^0, ...$ является решением системы и существует пара индексов i, j, для которой на каком-либо пути от i до j

выполняется $s^0_{i,j} > s_{i,j}$. Просуммировав все неравенства вдоль этого пути, получим неравенство-следствие $s^0_{i,j} \le s_{i,j}$, что является противоречием. ■

Свойство 1. $l_{0,i} = \min\limits_j \left(x_i^{\max}, x_j^{\max} + l_{j,i} \right), \quad -l_{i,0} = \max\limits_j \left(x_i^{\min}, x_j^{\min} - l_{i,j} \right), \quad i, j = \overline{1,n}$.

Доказательство. Рассмотрим путь минимальной длины от вершины 0 до вершины i, проходящий через другую вершину j (возможно, совпадающую с i). Тогда путь минимальной длины от вершины 0 до вершины i – это путь минимальной длины среди всех указанных путей через вершину j по всем j. Первое равенство описывает это утверждение. Аналогично доказывается справедливость второго равенства. ■

Свойство 2. *Соотношения* $-l_{i,0} \le x_i \le l_{0,i}$ *влекут соотношения* $-a_{i,0} \le x_i \le a_{0,i}$.

Доказательство. Соотношение $-a_{i,0} \le -l_{i,0}$ следует из третьего соотношения в (13), а соотношение $l_{0,i} \le a_{0,i}$ – из второго соотношения в (12). ■

Свойство 3. *В каждой индивидуальной задаче (11) величины* $l_{0,i}$ *и* $-l_{i,0}$ *являются верхней и нижней границами значения переменной* x_i *соответственно*.

Доказательство. В самом деле, сложив неравенства вдоль дуг кратчайшего пути от 0 до i, получим неравенство-следствие $x_i - x_0 \le l_{0,i}$. Значит, значение x_i не может быть больше $l_{0,i}$. Аналогично для $-l_{i,0}$. ■

Величину $l_{0,i}$ ($-l_{i,0}$) назовем *допустимым максимумом (минимумом)* переменной x_i. Согласно теореме набор значений, состоящий из всех $x_i = l_{0,i}$ ($x_i = -l_{i,0}$), является решением. Таким образом, если к (1) добавить целевую функцию с положительными коэффициентами при каждой переменной, то очевидно, что при $x_i = l_{0,i}$ ($x_i = -l_{i,0}$) она примет максимальное (минимальное) значение. Можно назвать *максимальным* (*минимальным*) такое решение системы (1). Ранее (см. введение и подразд. 1.2) упоминалось о существовании таких решений.

Свойство 4. *Если в допустимом решении системы (11) значение какой-либо переменной* x_i *равно ее допустимому максимуму (минимуму), то значения всех переменных* x_j, *лежащих на кратчайших путях из* 0 *в* i *(из* i *в* 0*), равны соответствующим допустимым максимумам (минимумам), а для первых (последних), находящихся на кратчайших путях* $(0,i)$ $((i,0))$ *переменных* x_j *будет выполняться* $x_j = l_{0,j} = x_j^{\max}$ ($x_j = -l_{j,0} = x_j^{\min}$).

Доказательство. Если какая-либо переменная x_i принимает значение $x_i^0 = l_{0,i}$ ($x_i^0 = l_{i,0}$), а для значения какой-либо находящейся на кратчайшем пути от 0 до i (от i до 0) переменной x_j выполняется $x_j^0 < l_{0,j}$ ($x_j^0 > -l_{j,0}$), тогда $x_i^0 - x_j^0 > l_{j,i}$ ($x_j^0 - x_i^0 > l_{i,j}$), что невозможно согласно лемме. Для первых (последних), находящихся на кратчайших путях переменных $l_{0,j} = a_{0,j} = x_j^{\max}$ ($l_{j,0} = a_{j,0} = -x_j^{\min}$). ■

Свойство 5. *Система (11) совместна только тогда, когда для каждой вершины* i *выполняется условие* $l_{0,i} \ge -l_{i,0}$.

Доказательство. Действительно, если для какой-либо вершины i выполняется неравенство $l_{0,i} < -l_{i,0}$, то $l_{0,i} + l_{i,0} < 0$ и существует цикл из вершины 0 в вершину 0 отрицательной длины, проходящий через вершину i, что противоречит условию сформулированной теоремы. Это следует также из свойства 3. ■

Если в графе, соответствующем системе (11), найдется цикл отрицательной длины и существуют пути $(0,i)$ и $(i,0)$ для какой-либо входящей в цикл вершины i, то для нее будет выполняться неравенство $l_{0,i} < -l_{i,0}$, поскольку $l_{0,i} = -\infty$, $-l_{i,0} = \infty$.

Свойство 6. *Решение системы (11) останется допустимым, если с повышением значения какой-либо переменной x_i повышать значения некоторых переменных x_j, предшествующих x_i на путях от 0 до i, при условии соблюдения верхних границ $l_{0,j}$, а также если с понижением значения какой-либо переменной x_i понижать значения некоторых переменных x_j, следующих за x_i на путях от i до 0, при условии соблюдения нижних границ $-l_{j,0}$.*

Это свойство аналогично свойству 3 следует из леммы.

Свойство 7. *Если в графе G из какой-либо вершины i (в какую-либо вершину i) исходит дуга только в 0 (входит дуга только из 0) и решением (11) будет какой-либо набор $x_1^0, x_2^0, ..., x_i^0, ...$, то значение x_i^0 можно заменить на любое значение в интервале $\left[x_i^{\min}; x_i^0\right]$ ($\left[x_i^0; x_i^{\max}\right]$) без потери совместности системы.*

Это свойство следует из свойства 6, а также становится очевидным при непосредственном рассмотрении неравенств (1).

Свойство 8. *Если какой-либо набор $x_1^0, x_2^0, ...$ значений переменных удовлетворяет разностным неравенствам и выполняется $-l_{i,0} \le x_i^0 \le l_{0,i}$, то этот набор является решением системы (11).*

Это свойство следует из свойства 2.

Свойство 9. *Если какой-либо набор $x_1^0, x_2^0, ..., x_i^0, ...$ является решением системы (11), то $x_1^0 + \delta, x_2^0 + \delta, ..., x_i^0 + \delta, ...$ является решением той же системы при выполнении условия $-l_{i,0} \le x_i^0 + \delta \le l_{0,i}$.*

Свойство 9 следует из свойства 8 и очевидного тождества $x_j - x_i = \left(x_j + \delta\right) - \left(x_i + \delta\right)$ для всех разностных неравенств. В работе [5] свойство разностных ограничений сохранять совместность при изменении значений переменных на общую величину в существующем решении сформулировано в виде отдельной леммы.

Свойство 10. *Если какой-либо набор $x_1^0, x_2^0, ..., x_i^0, ...$ является решением системы (11), то $x_1^0 + \delta, x_2^0 + \delta, ..., x_i^0 + \delta, ...$ является решением той же системы при любом δ, для которого выполняется условие $\max_i\left(-l_{i,0} - x_i^0\right) \le \delta \le \min_i\left(l_{0,i} - x_i^0\right)$.*

Это свойство следует из свойства 9.

Свойство 11. *Справедливы равенства* $\min_i\left(l_{0,i} - x_i^0\right) = \min_i\left(x_i^{\max} - x_i^0\right)$, $\min_i\left(x_i^0 + l_{i,0}\right) = \min_i\left(x_i^0 - x_i^{\min}\right)$.

Доказательство. Пусть имеется решение, такое, что $l_{i,0} \le x_i^0 \le l_{0,i}$ для всех i. Согласно свойству 10 при любом δ, таком, что $\max_i\left(-l_{i,0} - x_i^0\right) \le \delta \le \min_i\left(l_{0,i} - x_i^0\right)$, набор значений $x_i^0 + \delta$ будет решением системы. Следовательно, при любом таком δ будут выполняться соотношения $x_i^{\min} \le x_i^0 + \delta \le x_i^{\max}$, а значит, и соотношения $\min_i\left(l_{0,i} - x_i^0\right) \le \min_i\left(x_i^{\max} - x_i^0\right)$ и $\min_i\left(x_i^0 + l_{i,0}\right) \le \min_i\left(x_i^0 - x_i^{\min}\right)$. Если $\delta = \min_i\left(l_{0,i} - x_i^0\right)$ ($\delta = \max_i\left(-l_{i,0} - x_i^0\right)$), то для соответствующих x_i^0 будет выполняться $x_i^0 + \delta = l_{0,i}$ ($x_i^0 + \delta = -l_{i,0}$). Следовательно, согласно свойству 4 для каких-то x_j^0 будет выполняться $x_j^0 + \delta = x_j^{\max} = l_{0,j}$ ($x_j^0 + \delta = x_j^{\min} = -l_{j,0}$), т. е. $\min_i\left(l_{0,i} - x_i^0\right) \ge \min_i\left(x_i^{\max} - x_i^0\right)$ ($\min_i\left(x_i^0 + l_{i,0}\right) \ge \min_i\left(x_i^0 - x_i^{\min}\right)$). ■

Согласно теореме вычислительная сложность алгоритма поиска исходного решения системы DBM, основанного на представлении системы в виде сети ограничений, может быть сведена к сложности методов нахождения кратчайших путей от одной вершины ко всем остальным в графе с возможными отрицательными весами, выявляющих циклы отрицательной длины. Например, эту задачу можно решить с помощью алгоритмов Форда – Беллмана [3] или Флойда – Уоршалла [5], определяющих, в частности, наличие циклов отрицательной длины за время $O(n^3)$.

При изменении значения x_i^0 одной из переменных, если для нового ее значения $x_i^/$ выполняется $-l_{i,0} \le x_i^/ \le l_{0,i}$, строить новое решение можно, например, следующими способами:

1. Если $x_i^/ - x_i^0 = \delta \le \min\limits_j \left(x_j^{\max} - x_j^0\right)$ при увеличении x_i^0 ($x_i^0 - x_i^/ = \delta \le \min\limits_j \left(x_j^0 - x_j^{\min}\right)$ при уменьшении x_i^0), то все переменные можно увеличить (уменьшить) на δ (см. свойство 11). Очевидно, что сложность такого метода равна $O(n)$. Также из свойства 6 следует, что увеличивать (уменьшать) необходимо значения только тех переменных, которые предшествуют x_i на путях от 0 до i (следуют за x_i на путях от i до 0). Таким образом, предполагая, что количество таких переменных составляет в среднем половину от их общего количества, оценку сложности можно уменьшить вдвое.

2. Если условия для $x_i^/$ из первого способа не выполняются, можно применить следующий алгоритм:

Шаг 1. Увеличиваем (уменьшаем) $x_i^0, i = i_0,$ до требуемого значения $x_i^/$ и присваиваем соответствующей вершине графа G временную числовую метку, равную $x_i^/$.

Шаг 2. Рассматриваем значения переменных x_j^0 в вершинах графа G, смежных в ориентированном смысле на путях от 0 (до 0) (свойство 6), кроме самой вершины 0, с любой вершиной i из тех, которым присвоена временная метка. Если для какой-либо x_j не выполняется условие $x_j^0 - x_i^/ \le a_{ij}$ ($x_i^/ - x_j^0 \le a_{ji}$), то устанавливаем ее значение в $x_j^/ = x_i^/ + a_{ij}$ ($x_j^/ = x_i^/ - a_{ji}$) и присваиваем либо обновляем соответствующей ей вершине временную метку $x_j^/$. Далее метку $x_i^/$ делаем постоянной. Если в графе G присутствуют только постоянные метки, алгоритм останавливается; если нет, повторяем шаг 2.

Таким образом, при изменении значения x_i^0 одной из переменных x_i строится дерево допустимых x^0-расстояний между этой переменной и, в среднем, половиной из остальных. Сложность такого метода можно оценить как $O\left(\frac{m}{2} + \frac{n}{2}\log\frac{n}{2}\right)$ аналогично оценкам для алгоритма Дийкстры.

Данный способ является более универсальным, чем метод, рассмотренный в разд. 1, так как не зависит от порядка индексации переменных. При помощи приведенного алгоритма можно также скорректировать систему и найти ее решение, если $x_i^/ < -l_{i,0}$ ($x_i^/ > l_{0,i}$). Для этого на шаге 2 в число рассматриваемых вершин нужно также включить и вершину 0 и, если не выполняются условия $0 - x_i^/ \le a_{i,0}$ ($x_i^/ - 0 \le a_{0,i}$), увеличивать $a_{i,0}$ ($a_{0,i}$) до выполнения этих условий, т. е. при невыполнении $-l_{i,0} \le x_i^/ \le l_{0,i}$ для восстановления совместности системы достаточно расширить некоторые интервальные ограничения.

Способ 2 подходит для случая, когда желательны как можно меньшие изменения значений переменных. Если такого ограничения нет, то соблюсти совместность можно еще менее вычислительно сложным способом.

3. Пусть значение x_i^0 какой-либо переменной x_i увеличено (уменьшено) на величину $\delta \le l_{0,i} - x_i^0$ ($\delta \le x_i^0 + l_{i,0}$). Значения находящихся на путях от 0 до i (от i до 0) переменных x_j,

для которых выполняется условие $x_i^{/} - x_i^0 = \delta \le l_{0,j} - x_j^0$ ($x_i^0 - x_i^{/} = \delta \le x_j^0 + l_{j,0}$), увеличиваются (уменьшаются) на δ. Значения остальных, находящихся на путях от 0 до i (от i до 0) переменных x_k устанавливаются в $l_{0,k}$ ($-l_{k,0}$). Совместность системы при новом наборе значений переменных сохранится, так как значения переменных x_k, установленные в $l_{0,k}$ ($-l_{k,0}$), составляют часть максимального (минимального) решения и уменьшенные (увеличенные) по сравнению с максимальным (минимальным) решением значения $x_j^{/}$ переменных x_j, если они находятся на путях от k до 0 (от 0 до k), не влияют на совместность. С другой стороны, значения тех переменных, которые находятся на путях от 0 до k (от k до 0), равны $x_j^0 + \delta \ge x_k^0 - s_{j,k}^0 + \delta \Leftrightarrow x_k^0 - x_j^0 \le s_{j,k}^0$ ($x_j^0 - \delta \le x_k^0 + s_{k,j}^0 - \delta \Leftrightarrow x_j^0 - x_k^0 \le s_{k,j}^0$). Очевидно, что оценка сложности такого способа составляет $O(n)$.

4. Если в разностных неравенствах из (1) переменная x_i встречается только с коэффициентом +1 (-1), то очевидно (и по свойству 7), что значение x_i можно уменьшать до $x_i^{\min}$ (увеличивать до $x_i^{\max}$) без изменения значений остальных переменных.

Заключение

Для систем неравенств класса DBM разработаны два метода поиска решений, позволяющие получить несколько решений одной индивидуальной задачи и для большинства задач найти множества решений, составляющие непрерывные области. Один основан на методе Фурье – Моцкина, другой – на представлении системы в виде сети ограничений.

Полученные результаты позволяют сделать вывод о том, что с точки зрения универсальности и вычислительной сложности для решения описанного вида систем с большим количеством переменных предпочтительным является использование сетевого метода. Описанные методы могут быть использованы для решения практических задач и подзадач построения расписаний работ, непрерывного движения объектов по сети пересекающихся маршрутов и других важных вопросов.

Список литературы

1. Рубанов, И.В. Задача выбора маршрутов движения объектов при ограничении на сближение / И.В. Рубанов, М.С. Баркетов, М.Я. Ковалев // Танаевские чтения : докл. Междунар. науч. конф., Минск, 27–29 марта 2014 г. – Минск : ОИПИ НАН Беларуси, 2014. – С. 136–140.

2. Peron, M. An Abstract Domain Extending Difference-Bound Matrices with Disequality Constraints / M. Peron, N. Halbwachs. – Grenoble, France : Vérimag, 2006. – 15 p.

3. Bellman, R. On a Routing Problem / R. Bellman // Quarterly of Applied Mathematics. – 1958. – Vol. 16, no. 1. – P. 87–90.

4. Данциг, Дж. Линейное программирование, его применения и обобщения / Дж. Данциг. – М. : Прогресс, 1966. – 600 с.

5. Кормен, Т. Алгоритмы, построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест ; пер. с англ. под. ред. А. Шеня. – М. : МЦНМО, 2002. – 960 с.

6. Писарук, Н.Н. Исследование операций / Н.Н. Писарук. – Минск : БГУ, 2015. – 304 с.

7. Solving Systems of Difference Constraints Incrementally / G. Ramalingam [et al.] // Algorithmica. – 1999. – Vol. 23. – P. 261–275.

8. Fourier, J.B.J. Solution d'une question particulière du calcul des inégalités / J.B.J. Fourier. – France, 1826.

9. Герман, В.Н. Решение линейных ограничений над полем вещественных и рациональных чисел / В.Н. Герман // Кибернетика и системный анализ. – 2010. – № 4. – С. 123–133.

10. On Solving Boolean Combinations of UTVPI Constraints / Sanjit A. Seshia [et al.] // J. on Satisfability, Boolean Modeling and Computation. – 2007. – Vol. 3. – P. 67–90.

Поступила 13.07.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: irubanov@inbox.ru

I.V. Rubanov, M.S. Barketau, M.Y. Kovalyov

TWO METHODS OF SOLVING THE SYSTEM OF DIFFERENCE AND INTERVAL CONSTRAINTS

We develop two methods of solving DBM system. One method is based on Fourier – Motskin elimination scheme, it has complexity $O(n^3)$ for finding initial solve and complexity $O(n^2)$ for finding solve by changing one of variable value for some cases. The other method is based on the network of constraints approach, it has complexity $O(n^3)$ for finding initial solve and $O(n)$ or approximately $O\left(\frac{m}{2}+\frac{n}{2}\log\frac{n}{2}\right)$ for finding solve by changing one of variable value if it is limited by special bounds.

УДК 519.8

А.В. Кононов[1], И.Н. Лущакова[2]

ОПТИМАЛЬНОЕ ОБСЛУЖИВАНИЕ ТРЕБОВАНИЙ ДВУМЯ ПРИБОРАМИ ПРИ ЛИНЕЙНО УБЫВАЮЩИХ ФУНКЦИЯХ СТОИМОСТИ ВРЕМЕННЫХ ИНТЕРВАЛОВ

Рассматривается задача построения оптимального расписания обслуживания требований двумя параллельными приборами. В качестве целевой функции применяется линейная комбинация взвешенной суммы моментов завершения обслуживания требований и суммарной стоимости использования временных интервалов. В случае заданных для каждого из приборов линейно убывающих или постоянных последовательностей стоимостей временных интервалов предлагается точный псевдополиномиальный алгоритм динамического программирования.

Введение

В различных практических задачах качество расписания s обслуживания требований приборами может определяться целевой функцией вида $F_1(s) + F_2(s)$. Функция $F_1(s)$ – это какая-либо традиционная функция, рассматриваемая в теории расписаний, которая зависит от моментов завершения обслуживания требований, функция $F_2(s)$ – суммарная стоимость использования приборов в конкретные моменты времени. Например, стоимость заявки, выполненной во внеурочное время, будет выше. Стоимость доставки продуктов, заказанных в сетевом интернет-магазине, выше в вечернее время и выходные дни. Стоимость телевизионной рекламы зависит не только от длительности рекламного ролика, но и от времени суток трансляции ее в эфире. В облачных вычислениях разные вычислительные ресурсы (оборудование) имеют различную стоимость использования [1], причем пользователям могут предлагаться и различные схемы оплаты, зависящие от времени суток выполнения работ на удаленном сервере [2].

1. Постановка задачи и предварительные результаты

Разные задачи построения расписаний обслуживания множества требований одним прибором, минимизирующих функцию вида $F_1(s) + F_2(s)$, исследовались в работах [3, 4]. В настоящей статье обратимся к модели с параллельными приборами. Рассмотрим следующую задачу.

Множество $N = \{1, 2, ..., n\}$ требований необходимо обслужить на M параллельных приборах. Для каждого требования $i \in N$ заданы длительность его обслуживания $p_i > 0$, где p_i – целое число, и весовой коэффициент $w_i > 0$. Каждый прибор не может одновременно обслуживать более одного требования. Если в расписании s обслуживание требования i начинается на каком-либо приборе в момент времени $r_i(s)$, то оно протекает непрерывно на этом приборе и завершается в момент времени $C_i(s) = r_i(s) + p_i$. Будем рассматривать $F_1(s) = \sum_{i=1}^{n} w_i C_i(s)$.

Пусть горизонт планирования состоит из K интервалов единичной длины. Предположим, что $p_i \le K$ для каждого требования $i \in N$. Кроме того, предположим, что число K достаточно велико, так что все требования могут быть обслужены без прерываний на M приборах в пределах горизонта планирования. Интервал $(k-1, k]$ назовем k-м интервалом, $k = 1, 2, ..., K$. Введем следующие обозначения: π_k^L – стоимость использования k-го интервала времени прибором L, $1 \le L \le M$; $N_L(s)$ – множество требований, назначенных на прибор L в расписании s; $\overline{\pi}^L_{r_i(s), C_i(s)}$ – общая стоимость использования требованием $i \in N_L(s)$ временных интервалов для рассматриваемого горизонта планирования при расписании s, т. е.

$\overline{\pi}^L_{r_i(s),C_i(s)} = \overline{\pi}^L_{r_i(s),r_i(s)+p_i} = \pi^L_{r_i(s)+1} + \pi^L_{r_i(s)+2} + ... + \pi^L_{r_i(s)+p_i}$, где $r_i(s)$ – момент начала обслуживания требования i в расписании s. Тогда общая стоимость использования временных интервалов для рассматриваемого горизонта планирования при расписании s будет задана функцией $F_2(s) = \sum_{L=1}^{M} \sum_{i \in N_L(s)} \overline{\pi}^L_{r_i(s),C_i(s)}$.

Расписание s определяется указанием для каждого требования $i \in N$ прибора L, на котором оно будет обслужено, и момента $r_i(s)$ начала его обслуживания на приборе (либо, что то же самое, момента $C_i(s)$ окончания его обслуживания). Заметим, что при расписании s возможны простои приборов между интервалами времени, используемыми для обслуживания требований.

Необходимо построить расписание s^*, минимизирующее функцию

$$F_1(s) + F_2(s) = \sum_{i=1}^{n} w_i C_i(s) + \sum_{L=1}^{M} \sum_{i \in N_L(s)} \overline{\pi}^L_{r_i(s),C_i(s)}.$$

Обозначим эту задачу $PM / slotcost / \sum (w_i C_i + \overline{\pi}^L_{r_i,C_i})$. Аналогичная задача обслуживания требований одним прибором в работе [4] обозначена $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$.

Предположим, что стоимость использования временных интервалов не принимается во внимание. В этом случае задача $P2 / / \sum w_i C_i$ для двух параллельных приборов является NP-трудной [5]. Для задачи $PM / / \sum w_i C_i$ известен псевдополиномиальный алгоритм [6]. В монографии [7] отмечается, что задача $P / / \sum w_i C_i$ с произвольным числом приборов является NP-трудной в сильном смысле.

Для задачи $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$ в работах [3, 4] получены следующие результаты.

Теорема 1 [3]. *Задача* $1 / slotcost / \sum (C_i + \overline{\pi}_{r_i,C_i})$ *является NP-трудной в сильном смысле.*

Свойство 1 [3]. *Если последовательность* $\{\pi_k\}, k = 1, 2, ..., K,$ *неубывающая, то задача* $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$ *сводится к классической задаче* $1 / / \sum w_i C_i$.

Действительно, для минимизации регулярного критерия, каким является критерий $\sum w_i C_i$, требования должны быть обслужены как можно раньше, исключая простои на приборе. Этого же принципа в случае неубывающей последовательности $\{\pi_k\}, k = 1, 2, ..., K,$ следует придерживаться и с точки зрения минимизации общей стоимости использования временных интервалов. Поэтому приходим к выводу, что свойство, аналогичное свойству 1, может быть получено и для задачи $PM / slotcost / \sum (w_i C_i + \overline{\pi}^L_{r_i,C_i})$.

Теорема 2 [4]. *Если последовательность* $\{\pi_k\}, k = 1, 2, ..., K,$ *невозрастающая, то задача* $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$ *является NP-трудной в сильном смысле.*

Рассмотрим задачу $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$ при условии, что стоимость использования временных интервалов линейно убывает относительно индекса k, т. е. $\pi_k - \pi_{k+1} = \varepsilon, \ \varepsilon > 0,$ для $k = 1, 2, ..., K-1.$

Лемма 1 [4]. *Если последовательность* $\{\pi_k\}, k = 1, 2, ..., K,$ *линейно убывающая, то для задачи* $1 / slotcost / \sum (w_i C_i + \overline{\pi}_{r_i,C_i})$ *существует оптимальное расписание, в котором требования назначены на обслуживание в порядке* $\frac{w_1}{p_1} \geq \frac{w_2}{p_2} \geq ... \geq \frac{w_n}{p_n}$.

Лемма 2 [4]. *Если* $\frac{w_i}{p_i} > \varepsilon$, *то в оптимальном расписании требование* i *должно быть обслужено как можно раньше. Если* $\frac{w_i}{p_i} < \varepsilon$, *то в оптимальном расписании требование* i *должно быть обслужено как можно позже. Если* $\frac{w_i}{p_i} = \varepsilon$, *то не имеет значения, в каком интервале времени обслуживается требование* i.

На основании лемм 1 и 2 для задачи $1/slotcost/\sum(w_iC_i + \overline{\pi}_{r_i,C_i})$ в случае линейно убывающей последовательности стоимостей временных интервалов в [4] предложен алгоритм сложности $O(n\log n)$ операций.

2. Метод динамического программирования для двух приборов

Пусть $M = 2$. Предположим, что приборы имеют постоянные или линейно убывающие последовательности $\{\pi_k^1\}$ и $\{\pi_k^2\}$ стоимостей временных интервалов: $\pi_k^1 - \pi_{k+1}^1 = \varepsilon_1$ и $\pi_k^2 - \pi_{k+1}^2 = \varepsilon_2$ для $k = 1, 2, \ldots K-1$, причем $0 \le \varepsilon_1 \le \varepsilon_2$.

Разобьем множество N требований на три подмножества: $J_1 = \{i : \frac{w_i}{p_i} > \varepsilon_2\}$, $J_2 = \{i : \frac{w_i}{p_i} < \varepsilon_1\}$ и $J_3 = \{i : \varepsilon_1 \le \frac{w_i}{p_i} \le \varepsilon_2\}$. Для непустого множества J_j, $1 \le j \le 3$, полагаем $n_j = |J_j|$, $P_j = \sum_{i \in J_j} p_i$, в противном случае $n_j = 0, P_j = 0$.

Упорядочим и перенумеруем требования следующим образом: $J_1 = \{i_1, i_2, \ldots, i_{n_1}\}$ и $\frac{w_{i_1}}{p_{i_1}} \ge \frac{w_{i_2}}{p_{i_2}} \ge \ldots \ge \frac{w_{i_{n_1}}}{p_{i_{n_1}}} > \varepsilon_2$; $J_2 = \{k_1, k_2, \ldots, k_{n_2}\}$ и $\frac{w_{k_1}}{p_{k_1}} \le \frac{w_{k_2}}{p_{k_2}} \le \ldots \le \frac{w_{k_{n_2}}}{p_{k_{n_2}}} < \varepsilon_1$; $J_3 = \{l_1, l_2, \ldots, l_{n_3}\}$ и $\varepsilon_2 \ge \frac{w_{l_1}}{p_{l_1}} \ge \frac{w_{l_2}}{p_{l_2}} \ge \ldots \ge \frac{w_{l_{n_3}}}{p_{l_{n_3}}} \ge \varepsilon_1$.

Для решения задачи в случае $M = 2$ предлагается алгоритм динамического программирования *DPLinDec2*, состоящий из трех этапов.

На *этапе 1* алгоритм *DPLinDec2* назначает на обслуживание требования из множества J_1.

Пусть $J_1 \ne \varnothing$. Положим $\overline{p}_m^{(1)} = p_{i_1} + p_{i_2} + \ldots + p_{i_m}$ для $m = 1, 2, \ldots, n_1$; $P_1 = \overline{p}_{n_1}^{(1)}$. Рассмотрим подзадачу составления расписания для требований $i_1, i_2, \ldots, i_m$ из множества J_1. Предположим, что приборы начинают обслуживание требований в момент времени 0. Пусть $f_1(x,m)$ – наименьшее значение целевой функции при условии, что прибор 1 завершает обслуживание в момент времени x, $0 \le x \le \min\{P_1, K\}$. Тогда прибор 2 должен завершить обслуживание в момент времени $\overline{p}_m^{(1)} - x$. Функция $f_1(x,m)$ вычисляется рекурсивно с учетом того, что требование i_m обслуживается последним либо на приборе 1, либо на приборе 2:

$$f_1(x,m) = \min\{ f_1(x - p_{i_m}, m-1) + w_{i_m} x + \overline{\pi}^1_{x - p_{i_m}, x};$$

$$f_1(x, m-1) + w_{i_m}(\overline{p}_m^{(1)} - x) + \overline{\pi}^2_{\overline{p}_{m-1}^{(1)} - x, \overline{p}_m^{(1)} - x}\} \ .$$

Зададим начальные условия:

$$f_1(x,1)=\begin{cases} w_{i_1}p_{i_1}+\overline{\pi}^2_{0,p_{i_1}}, \text{ если } x=0,\\ w_{i_1}p_{i_1}+\overline{\pi}^1_{0,p_{i_1}}, \text{ если } x=p_{i_1},\\ +\infty \text{ в противном случае.}\end{cases}$$

Граничные условия определим следующим образом:

$$f_1(x,m)=+\infty \text{ для } x<0 \text{, либо } x>\min\{K,\overline{p}^{(1)}_m\} \text{, либо } \overline{p}^{(1)}_m-x>K.$$

Для каждого значения $x=0,\ 1,...,\min\{P_1,K\}$ будем хранить значение функции $f_1(x,n_1)$ и соответствующее ему расписание $\sigma'(x)$.

Если $J_1=\varnothing$, то полагаем $x=0,\ f_1(0,0)=0$ и $\sigma'(0)$ – пустое расписание.

Этап 1 алгоритма может быть выполнен за $O(n_1P_1)$ операций.

На *этапе 2* алгоритм *DPLinDec2* назначает на обслуживание требования из множества J_2.

Пусть $J_2\neq\varnothing$. Положим $\overline{p}^{(2)}_m=p_{k_1}+p_{k_2}+...+p_{k_m}$ для $m=1,2,...,n_2$; $P_2=\overline{p}^{(2)}_{n_2}$. Рассмотрим подзадачу составления расписания для требований $k_1,k_2,...,k_m$ из множества J_2. Пусть $f_2(y,m)$ – наименьшее значение целевой функции при условии, что прибор 2 начинает обслуживание некоторых из этих требований в момент времени y. Тогда прибор 1 начинает обслуживание оставшихся требований в момент времени $2K-y-\overline{p}^{(2)}_m$. Оба прибора завершают обслуживание требований в момент времени K. Функция $f_2(y,m)$ вычисляется рекурсивно с учетом того, что требование k_m обслуживается первым либо на приборе 2, либо на приборе 1:

$$f_2(y,m)=\min\{f_2(y+p_{k_m},m-1)+w_{k_m}(y+p_{k_m})+\overline{\pi}^2_{y,y+p_{k_m}},$$

$$f_2(y,m-1)+w_{k_m}(2K-y-\overline{p}^{(2)}_{m-1})+\overline{\pi}^1_{2K-y-\overline{p}^{(2)}_m,2K-y-\overline{p}^{(2)}_{m-1}}\}.$$

Зададим начальные условия:

$$f_2(y,1)=\begin{cases} w_{k_1}K+\overline{\pi}^2_{K-p_{k_1},K}, \text{ если } y=K-p_{k_1},\\ w_{k_1}K+\overline{\pi}^1_{K-p_{k_1},K}, \text{ если } y=K,\\ +\infty \text{ в противном случае.}\end{cases}$$

Граничные условия определим следующим образом:

$$f_2(y,m)=+\infty \text{ для } y>K \text{, либо } y<\max\{0,K-\overline{p}^{(2)}_m\} \text{, либо } 2K-y-\overline{p}^{(2)}_m<0.$$

Для каждого значения $y=K,\ K-1,...,\max\{0,K-P_2\}$ будем хранить значение функции $f_2(y,n_2)$ и соответствующее ему расписание $\sigma''(y)$.

Если $J_2=\varnothing$, то полагаем $y=K,\ f_2(K,0)=0$ и $\sigma''(K)$ – пустое расписание.

Этап 2 алгоритма может быть выполнен за $O(n_2P_2)$ операций.

Этап 3 алгоритма выполняется для всех возможных значений переменных x и y, $0\le x\le\min\{P_1,K\},\max\{0,K-P_2\}\le y\le K$.

Рассмотрим конкретные значения переменных x и y из указанных диапазонов. Сначала алгоритм проверяет существование допустимого расписания $\sigma(x,y)$, в котором требования множеств J_1 и J_2 выполняются в тех же интервалах времени и на тех же приборах, что

и в расписаниях $\sigma'(x)$ и $\sigma''(y)$. Расписание $\sigma(x,y)$ будет допустимым, если на каждом приборе в любой момент времени обслуживается не более одного требования. Чтобы убедиться, что расписание $\sigma(x,y)$ допустимо, достаточно проверить выполнение неравенства $P_1 - y \le x \le 2K - y - P_2$. Если расписание $\sigma(x,y)$ допустимо, алгоритм вставляет в него требования множества J_3.

Пусть $J_3 \neq \varnothing$. Положим $\overline{p}_m^{(3)} = p_{l_1} + p_{l_2} + ... + p_{l_m}$ для $m = 1, 2, ..., n_3$; $P_3 = \overline{p}_{n_3}^{(3)}$. Рассмотрим подзадачу составления расписания для требований $l_1,\ l_2, ..., l_m$ из множества J_3. Известно, что прибор 1 начинает обслуживание некоторых из этих требований в момент времени x и заканчивает в момент времени z, в то время как прибор 2 начинает обслуживание оставшихся требований в момент времени u. Пусть $f_3(x,y,z,u,m)$ – наименьшее значение целевой функции при условии, что прибор 1 завершает обслуживание части требований множества J_3 в момент времени z, а прибор 2 завершает обслуживание остальных требований множества J_3 в момент времени $u + \overline{p}_m^{(3)} - (z-x)$. Функция $f_3(x,y,z,u,m)$ вычисляется рекурсивно с учетом того, что l_m – это последнее требование из множества J_3, обслуженное либо на приборе 1, либо на приборе 2:

$$f_3(x,y,z,u,m) = \min\{ f_3(x,y,z-p_{l_m},u,m-1) + w_{l_m} z + \overline{\pi}^1_{z-p_{l_m},z};$$

$$f_3(x,y,z,u,m-1) + w_{l_m}(u + \overline{p}_m^{(3)} - (z-x)) + \overline{\pi}^2_{u+\overline{p}_{m-1}^{(3)}-(z-x),\,u+\overline{p}_m^{(3)}-(z-x)}\}.$$

Зададим начальные условия:

$$f_3(x,y,z,u,1) = \begin{cases} w_{l_1}(u + p_{l_1}) + \overline{\pi}^2_{u,u+p_{l_1}}, & \text{если } z = x \text{ и } u + p_{l_1} \le y, \\ w_{l_1}(x + p_{l_1}) + \overline{\pi}^1_{x,x+p_{l_1}}, & \text{если } z = x + p_{l_1} \le 2K - y - P_2, \\ +\infty \text{ в противном случае.} \end{cases}$$

Граничные условия определим следующим образом:

$$f_3(x,y,z,u,m) = +\infty \text{ для } z < x, \text{ либо } z > \min\{2K - y - P_2,\ x + \overline{p}_m^{(3)}\},$$
$$\text{либо } u + \overline{p}_m^{(3)} - (z-x) > y.$$

Таким образом, для допустимого расписания $\sigma(x,y)$ алгоритм *DPLinDec2* строит семейство расписаний $\tilde{\sigma}(x,y,z,u)$, $x \le z \le \min\{x + P_3, 2K - y - P_2\}$, $\max\{y - P_3, P_1 - x\} \le u \le y$. Каждому расписанию $\tilde{\sigma}(x,y,z,u)$ соответствует значение целевой функции $f_3(x,y,z,u,n_3)$.

Если $J_3 = \varnothing$, полагаем $z = x,\ u = y,\ f_3(x,y,x,y,0) = 0$. В этом случае соответствующее расписание $\tilde{\sigma}(x,y,x,y)$ совпадает с расписанием $\sigma(x,y)$.

Отметим, что согласно лемме 2 в оптимальном расписании все требования множества J_3, которые обслуживаются на приборе 2, должны быть обслужены как можно позже, т. е. прибор 2 должен начать их обслуживание в момент времени $u = y - (P_3 - (z-x))$ и завершить в момент времени y. Поэтому будем отбрасывать все расписания $\tilde{\sigma}(x,y,z,u)$, у которых $u \neq y - (P_3 - (z-x))$.

На каждом шаге этапа 3, определяемом конкретными значениями переменных x, y, z, будем хранить только текущее расписание $\tilde{\sigma}(x,y,z,y-(P_3-(z-x)))$ с минимальным значением целевой функции $f_1(x,n_1) + f_2(y,n_2) + f_3(x,y,z,y-(P_3-(z-x)),n_3)$, которое находится по всем

просмотренным к данному шагу значениям переменных x, y, z. По окончании этапа 3 алгоритм находит оптимальное расписание обслуживания требований множества N. Этап 3 алгоритма может быть выполнен за $O(n_3 P_1 P_2 P_3^2)$ операций.

Заметим, что если некоторые из подмножеств J_1, J_2, J_3 требований являются пустыми, то это отразится на общей вычислительной сложности алгоритма *DPLinDec2* (таблица).

Общая вычислительная сложность алгоритма *DPLinDec2*

Подмножества требований	Вычислительная сложность
$J_1 \neq \varnothing, J_2 \neq \varnothing, J_3 \neq \varnothing$	$O(n_1 P_1 + n_2 P_2 + n_3 P_1 P_2 P_3^2)$
$J_1 \neq \varnothing, J_2 \neq \varnothing, J_3 = \varnothing$	$O(n_1 P_1 + n_2 P_2 + P_1 P_2)$
$J_1 \neq \varnothing, J_2 = \varnothing, J_3 \neq \varnothing$	$O(n_1 P_1 + n_3 P_1 P_3^2)$
$J_1 = \varnothing, J_2 \neq \varnothing, J_3 \neq \varnothing$	$O(n_2 P_2 + n_3 P_2 P_3^2)$
$J_1 = \varnothing, J_2 = \varnothing, J_3 \neq \varnothing$	$O(n_3 P_3^2)$
$J_1 = \varnothing, J_2 \neq \varnothing, J_3 = \varnothing$	$O(n_2 P_2)$
$J_1 \neq \varnothing, J_2 = \varnothing, J_3 = \varnothing$	$O(n_1 P_1)$

На всех этапах алгоритма объем памяти, требуемый для поддержания его работы, не превышает $O(nK)$.

Заключение

В статье предложен точный псевдополиномиальный алгоритм динамического программирования для построения оптимального расписания обслуживания требований двумя параллельными приборами при линейно убывающих или постоянных функциях стоимости временных интервалов.

Отметим, что если у всех требований множества N длительности обслуживания единичные, то для непустого множества J_j, $1 \le j \le 3$, имеем $P_j = n_j$. В этом случае алгоритм *DPLinDec2* превращается в полиномиальный алгоритм с общей вычислительной сложностью $O(n_1^2 + n_2^2 + n_1 n_2 n_3^3)$.

Работа выполнена при частичной финансовой поддержке проекта № Ф15СО-043 БРФФИ.

Список литературы

1. Zhao, G. Cost-aware scheduling algorithm based on PSO in Cloud Computing Environment / G. Zhao // Intern. J. of Grid and Distributed Computing. – 2014. – Vol. 7, no. 1. – P. 33–42.

2. Amazon EC2 Pricing Options [Electronic resource]. – 2016. – Mode of access : https://aws.amazon.com/ec2/pricing. – Date of access : 10.04.2016.

3. Wan, G. Scheduling with Variable Time Slot Costs / G. Wan, X. Qi // Naval Research Logistics. – 2010. – Vol. 57, no. 2. – P. 159–171.

4. Zhao, Y. On scheduling with non-increasing time slot cost to minimize total weighted completion time / Y. Zhao, X. Qi, M. Li [Electronic resource]. – 2016. – Mode of access : http://link.springer.com/article/10.1007/s10951-015-0462-9#/page-1. – Date of access : 10.04.2016.

5. Bruno, J. Scheduling independent tasks to reduce mean finishing time/ J. Bruno, E.G. Coffman, Jr., R. Sethi // Communications of the ACM. – 1974. – Vol. 17. – P. 382–387.

6. Sequencing and Scheduling: Algorithms and Complexity / E.L. Lawler [et al.] // Handbooks in Operations Research and Management Science. – North-Holland, Amsterdam, 1993. – Vol. 4. – P. 445–522.

7. Brucker, P. Scheduling Algorithms / P. Brucker. – Springer, 2004. – 367 p.

Поступила 06.06.2016

[1]*Институт математики им. С. Л. Соболева СО РАН,*
Новосибирск, пр. академика Коптюга, 4
e-mail: alvenko@math.nsc.ru

[2]*Белорусский государственный университет*
информатики и радиоэлектроники,
Минск, ул. П. Бровки, 6
e-mail: IrinaLushchakova@yandex.ru

A.V. Kononov, I.N. Lushchakova

SCHEDULING JOBS ON TWO PARALLEL MACHINES WITH LINEAR DECREASING TIME SLOT COSTS

We consider a scheduling problem with two parallel machines to minimize the sum of total weighted completion time and total machine time slot costs. In the case of the constant or linear decreasing sequences of time slotcosts we suggest an exact pseudopolynomial DP algorithm.

УДК 658.512.2;004.4:004.9

В.И. Романов, Ю.Ю. Ланкевич

ОРГАНИЗАЦИЯ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ ДЛЯ ПРОСМОТРА МНОГОСЛОЙНОЙ ТОПОЛОГИИ СБИС

Предлагается один из возможных способов реорганизации графической информации, описывающей множество слоев топологии современных СБИС. Этот способ ориентирован на использование в условиях ограниченной по размеру памяти видеокарты. Дополнительный эффект, обеспечивающий высокое быстродействие формирования демонстрируемого многокадрового изображения многослойной топологии современных СБИС, достигается за счет предварительной загрузки требуемых текстур при помощи вспомогательного фонового процесса.

Введение

Топология СБИС может быть представлена множеством слоев, каждый из которых, в свою очередь, состоит из множества кадров. Каждый кадр описывается отдельным файлом в некотором графическом формате. Учитывая доступную разрешающую способность современных микроскопов, а также размеры исследуемых микросхем, нетрудно показать, что для построения полного видеоизображения могут потребоваться десятки гигабайт памяти. Оказывается, что компьютеры, традиционно используемые при решении задачи анализа топологии СБИС, не в состоянии одновременно разместить в памяти видеокарты всю информацию. В силу последнего утверждения требует решения задача информационного обеспечения просмотра топологии в условиях ограничений по размерам доступной памяти видеокарты с одновременным сохранением дружественности интерфейса, выражающейся в высокой скорости предоставления пользователю необходимой порции видеоинформации. Для обеспечения кроссплатформенности разрабатываемого программного обеспечения предлагается использовать инструментарий Qt 5 [1], поддерживающий кроссплатформенную графическую библиотеку OpenGL [2]. Видеоадаптер компьютера, например NVIDIA GeForce GTX 760, должен обеспечивать поддержку стандартов библиотеки OpenGL 4.0 и иметь не менее 2 Гб видеопамяти.

Рассматривая вопрос обеспечения доступа к графической информации, обычно хранящейся на внешней памяти, можно сделать следующие выводы:

1) время доступа будет тем меньше, чем меньший объем данных потребуется прочитать из внешней памяти. На практике из этого следует, что для повышения скорости необходимо воспользоваться графическими форматами, обеспечивающими внутреннее сжатие информации. Применительно к функциональному набору, реализация которого осуществляется создаваемыми программными средствами, это означает, что в условиях «тяжелых» по объему данных необходимо осуществить однократную конвертацию графики, представленной в формате BMP, в формат DDS, поддерживаемый современными видеокартами в качестве встроенного формата, обеспечивающего высокую скорость обработки [3];

2) время доступа будет тем меньше, чем меньшее число файлов будет задействовано при переносе информации из внешней памяти в оперативную с ее последующим переносом в память видеокарты. Практически это означает, что имеет смысл перераспределить графическую информацию по файлам таким образом, чтобы в отдельном файле собиралось несколько графических объектов, одновременно видимых на экране.

1. Использование формата DDS (DirectDraw Surface)

Для всех слоев топологии отображаемые изображения конвертируются в графический формат DDS [3]. В данном формате обеспечивается возможность хранения текстур – растровых изображений, накладываемых на плоскость рисования топологии и загружаемых в память ви-

деокарты для последующей обработки. При организации формата DDS возможно применение MIP-текстурирования (MIP mapping), использующего несколько копий одной текстуры с разной детализацией.

MIP-текстурирование – это метод улучшения качества текстурных изображений при помощи текстур с разным разрешением для различных объектов одного и того же изображения в зависимости от их размера и глубины. Принцип работы метода заключается в создании так называемой MIP-пирамиды (рис. 1) – последовательности текстур с разрешением от максимального до 1×1, например: 1×1, 2×2, 4×4, 8×8, 16×16, 32×32, 64×64, 128×128, 256×256, 512×512 и 1024×1024. Каждая из этих текстур называется MIP-уровнем [4].

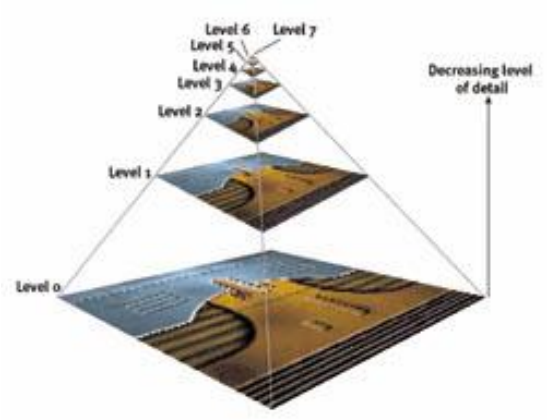


Рис. 1. Пирамидальность MIP-текстуры

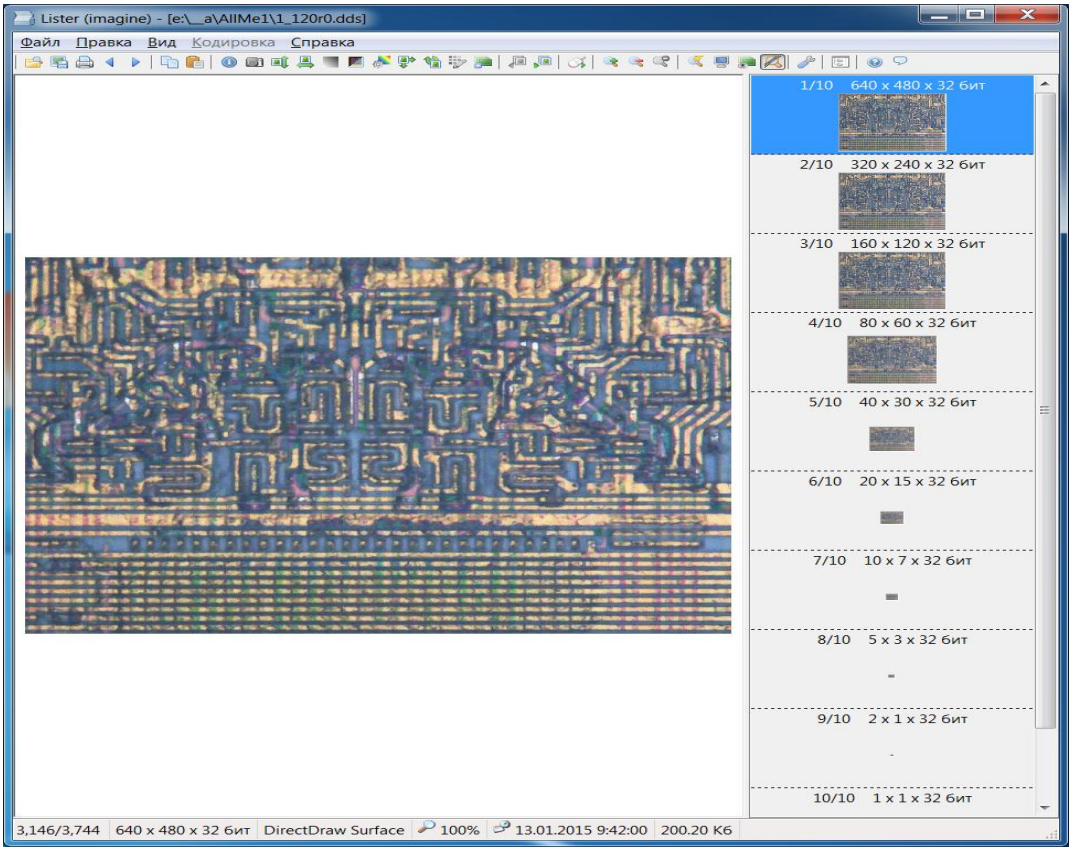


Рис. 2. Демонстрация одного файла топологии формата DDS

На всех текстурах находится одно и то же изображение. В результате расход видеопамяти при MIP-текстурировании увеличивается на треть в соответствии с формулой

$$\sum_{i=0}^{\infty}\left(\frac{1}{4}\right)^{i}=1\frac{1}{3}.$$

При отображении текстур вычисляется расстояние до плоскости рисования и номер (уровень) текстуры находится по формуле

$$\text{miplevel} = \log_2 (d / \text{texsize}\times\text{resolution}) + \text{mipbias},$$

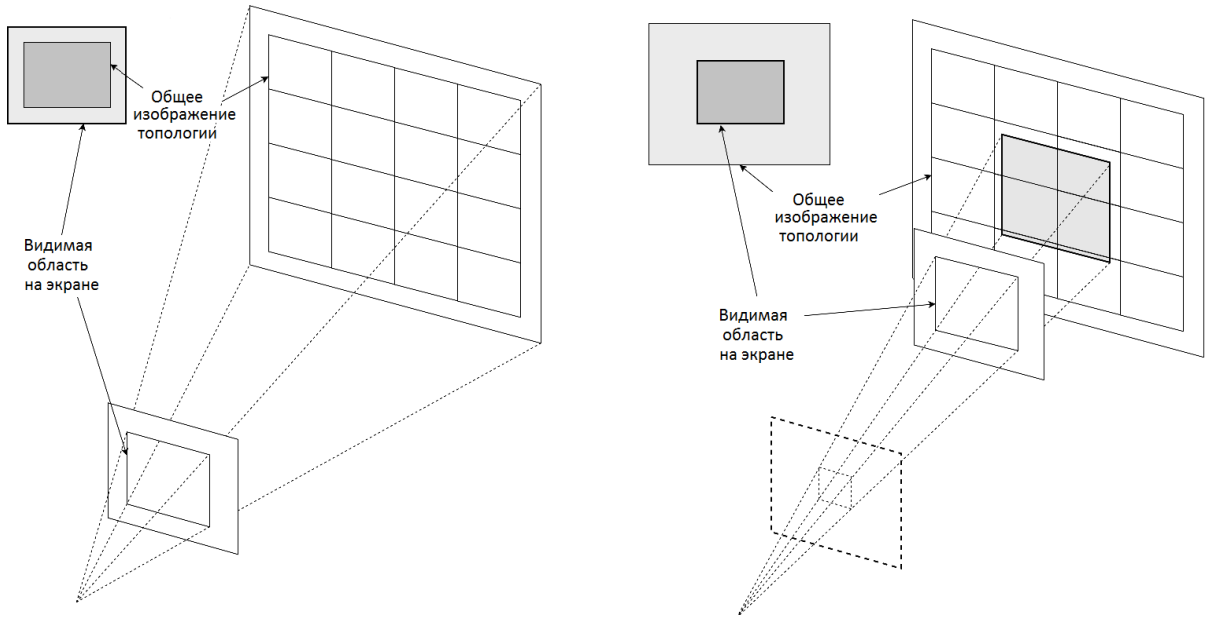
где resolution – разрешение виртуальной камеры (количество пикселов, которое будет в объекте размером в 1 ед., расположенном в 1 ед. от камеры); texsize – размер текселя в единицах трехмерного мира; d – расстояние до объекта в тех же единицах; mipbias – числовая поправка, позволяющая выбирать более или менее детальную текстуру, чем дает формула.

Значение miplevel округляется до целого, и текстура с соответствующим номером (нулевая – самая детальная, первая – вдвое меньшая и т. д.) используется для отображения.

Как уже было сказано, на практике все MIP-уровни текстуры хранятся в одном файле. На рис. 2 показано отображение одного из кадров реальной топологии слоя СБИС. Видно, что правая панель содержит определение всех MIP-уровней демонстрируемого файла формата DDS.

2. Проведение просмотра

При просмотре топологии используется модель, показанная на рис. 3. В ней имеется плоскость изображения, где размещены все кадры отдельного слоя топологии с тем разрешением, которое было использовано при съемках на микроскопе. Между «проектором» и указанной плоскостью изображения размещена параллельная ей дополнительная плоскость, соответствующая экранному окну, которое ограничивает видимый топологический фрагмент. Исходя из позиция «проектора» можно рассчитать масштабирование представленного на экране рисунка.



а) *б)*

Рис. 3. Масштабирование многокадрового изображения: *а)* исходное состояние отображения, при котором коэффициент масштабирования выбирается так, что в окне экрана видна вся схема; *б)* последующая демонстрация топологии, обеспечивающая видимость выбранного пользователем фрагмента

Фактически «сближение» плоскостей экрана и изображения приводит к увеличению масштаба изображения и позволяет пользователю рассмотреть его более детально, но при этом в окне становится видимой только небольшой фрагмент общего изображения топологии.

Многослойность топологии современных СБИС в рамках рассматриваемого подхода учитывается в виде нагрузки на использование ресурсов памяти: чем больше слоев участвует в формировании топологического изображения, тем больше затраты требуемой памяти и времени на ее обслуживание. Именно этот факт делает еще более актуальными предлагаемые средства оптимизации памяти.

Прочие аспекты отображения многих слоев в одном изображении в рамках рассматриваемой модели являются несущественными, поскольку в оперативной памяти компьютера или памяти видеоадаптера всегда представлены все слои одного проекта. Одновременное отображение нескольких слоев обеспечивается за счет возможности использования свойства «прозрачности» изображения слоя. Масштабирование, позиционирование и поворот на плоскости изображений осуществляются пользователем при помощи предлагаемых средств проведения просмотра независимо для каждого отдельного слоя.

Вариативность масштабирования показанного на экране изображения хорошо поддерживается на уровне драйвера видеокарты при условии использования MIP-текстурирования – драйвер самостоятельно выбирает наиболее подходящий из представленных в текстурах уровней и освобождает программу просмотра от решения данной задачи.

Анализ действий пользователя в процессе просмотра топологии показал, что можно условно выделить два режима. Один из них, который можно назвать *навигацией*, связан с решением задачи выбора на всем изображении слоя некоторого фрагмента, представляющего интерес для последующего детального анализа. При этом используется достаточно большой набор кадров, каждый из которых рассматривается без углубления в детали рисунка. Альтернативный навигации режим *анализа* связан с детальным рассмотрением совсем небольшого по количеству задействованных кадров фрагмента топологии.

3. Реорганизация данных

Наличие двух режимов просмотра – навигации и анализа – послужило основой предлагаемой реорганизации графической информации.

Для каждого слоя из всего множества образующих его файлов строится единый так называемый MAP-файл, размер которого устанавливается параметрически в зависимости от количественных оценок ширины и высоты слоя в числе кадров, а также размеров используемой памяти компьютера и установленной на нем видеокарты.

При этом в результирующий файл отбираются не все, а только несколько самых верхних (наименьших по объему данных) MIP-уровней от каждой текстуры. На рис. 4 показано графическое представление построения MAP-файла текстур. Использование единого файла для хранения текстур позволяет достаточно существенно сократить время формирования памяти видеокарты.

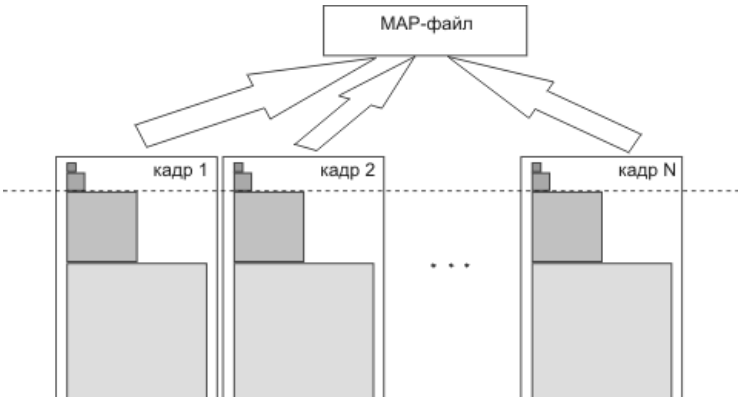


Рис. 4. Принцип отбора в один файл верхних MIP-уровней формата DDS

Отказываясь от нижних (больших по размеру) уровней, разработчик намеренно «огрубляет» изображение: при большом масштабе отображения видеоизображение будет интерполироваться драйвером видеокарты и выглядеть размытым. Однако данный эффект будет заметен только в условиях сильного «приближения» рисунка. Фактические размеры съемки отдельного кадра ограничиваются линейными размерами в несколько сотен точек. Если предположить, что для анализа топологии будут использованы мониторы с высокой разрешающей способностью [5], то и в этом случае общее число MIP-уровней будет не слишком большим: для размера 2560×1440, являющегося статистически заметным (несколько последних лет таким разрешением обладает 1 % от всех используемых дисплеев настольных компьютеров), число уровней равно 12. Учитывая разрешение современных дисплеев, можно заключить, что точное детальное отображение потребуется в условиях видимости на экране только нескольких кадров, во всех остальных случаях представление топологии будет происходить без информационных потерь.

В таблице приведены данные, касающиеся размеров требуемой памяти при использовании MIP-текстурирования. Анализ таблицы показывает, что допустимые размеры MAP-файла текстур (определяемые двумя последними колонками таблицы) ограничиваются восьмым-девятым MIP-уровнями. При небольшом числе кадров, например 30×20, размер MAP-файла даже с включением 10-го уровня составит порядка 2,5 Гб и одновременно видимыми окажутся не более девяти кадров.

Оценки памяти при просмотре топологии слоя с использованием MIP-текстурирования

Номер MIP-уровня	Ширина кадра *w*, пикселы	Высота кадра *h*, пикселы	Представление пиксела, байт	Размер уровня, байт	Размер кадра (текстуры), байт	Объем памяти представления 60×60 кадров, байт	Объем памяти представления 100×100 кадров, байт
11	2048	1536	4	12 582 912	16 777 208	60 397 948 800	167 772 080 000
10	1024	768	4	3 145 728	4 194 300	15 099 480 000	41 943 000 000
9	512	384	4	786 432	1 048 572	**3 774 859 200**	10 485 720 000
8	256	192	4	196 608	262 140	943 704 000	**2 621 400 000**
7	128	96	4	49 152	65 532	235 915 200	655 320 000
6	64	48	4	12 288	16 380	58 968 000	163 800 000
5	32	24	4	3 072	4 092	14 731 200	40 920 000
4	16	12	4	768	1 020	3 672 000	10 200 000
3	8	6	4	192	252	907 200	2 520 000
2	4	3	4	48	60	216 000	600 000
1	2	1	4	8	12	43 200	120 000
0	1	1	4	4	–	–	–

4. Определение набора одновременно видимых кадров

При поддержке просмотра топологии в режиме анализа возникает задача определения контекста ее видимой локальной области и предварительного размещения необходимых текстур в памяти видеокарты.

Вывод кадров изображения осуществляется в некоторой координатной сетке с определенным шагом по горизонтали и по вертикали. Только после привязки кадров к сетке осуществляются такие преобразования выводимого изображения, как поворот, масштабирование и сдвиг, реализуемые при помощи известных функций библиотеки OpenGL [2]. При этом для сопоставления отдельных кадров с глобальными координатами отображения используются:

– величина сдвига по горизонтали (*hShift*) и по вертикали (*vShift*);
– величина шага по горизонтали (*xStep*) и по вертикали (*yStep*);
– масштабные коэффициенты (*hScale* и *vScale*);
– угол поворота (*ang*).

После геометрических преобразований, таких как поворот, масштабирование или сдвиг, глобальные координаты узлов сетки меняются. Поэтому, чтобы получить соответствие между глобальной координатой и конкретным кадром из сетки, следует выполнить обратные координатные преобразования:

– для обратного сдвига

$$x' = x - hShift; \quad (1)$$

$$y' = y + vShift; \quad (2)$$

– для обратного масштабирования

$$x'' = x' / hScale; \quad (3)$$

$$y'' = y' / vScale; \quad (4)$$

– для обратного поворота

$$x''' = x'' * cos(ang) - y'' * sin(ang); \quad (5)$$

$$y''' = x'' * sin(ang) + y'' * cos(ang); \quad (6)$$

– итоговые преобразования, обеспечивающие вычисление позиции кадра по горизонтали и по вертикали,

$$j = \lceil x''' \rceil / xStep; \quad (7)$$

$$i = \lceil y''' \rceil / yStep. \quad (8)$$

Рассмотрим решение следующей задачи. На рис. 5 все текстуры лежат в прямоугольнике *ABCD*, а область отображения (окно видимости) обозначена прямоугольником *A'B'C'D'*. Необходимо выбрать только те текстуры, которые попадают в окно отображения (*A'B'C'D'*) полностью или частично.

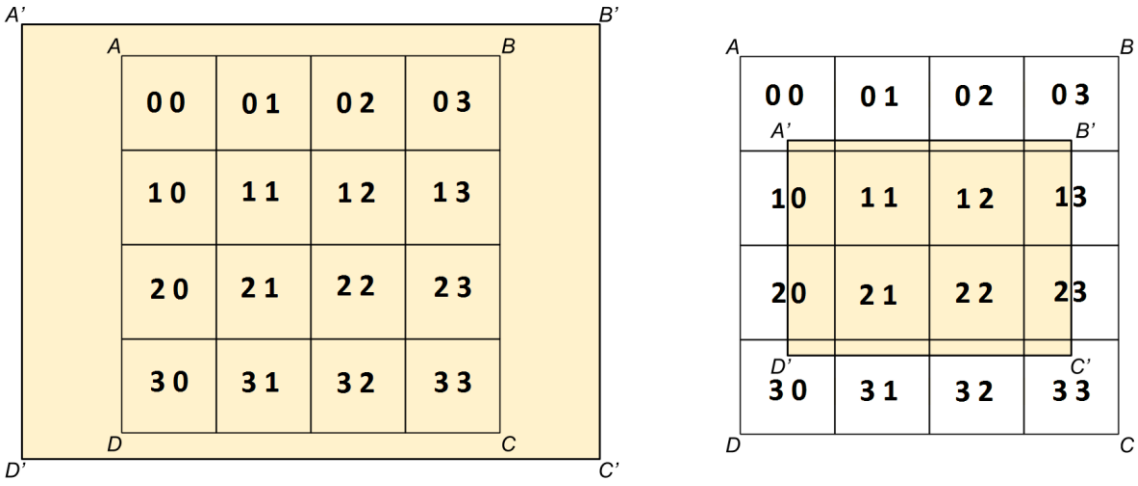


Рис. 5. Попадание всех текстур в область отображения: *а)* полное; *б)* частичное

Алгоритм отбора подмножества текстур, отображаемых в окне видимости, основывается на использовании координат середины окна отображения, его ширины и высоты, масштаба, угла поворота, ширины и высоты текстуры отдельного кадра и реализуется следующим образом:

Шаг 1. Вычисляются координаты вершин *A'*(*x*, *y*), *B'*(*x*, *y*), *C'*(*x*, *y*), *D'*(*x*, *y*) с помощью выражений (1)–(6).

Шаг 2. Вычисляются номера (индексы) текстур, в границах которых лежат угловые точки окна отображения (*A'*, *B'*, *C'*, *D'*), с помощью выражений (7), (8).

Шаг 3. Находятся наименьшие и наибольшие номера строк и столбцов из тех, что были получены на шаге 2.

Предложенный алгоритм подходит и для случая, когда изображение слоя повернуто относительно окна отображения (рис. 6). В данном случае кадры {0 0} и {3 3} не попадают в область отображения *A'B'C'D'*.

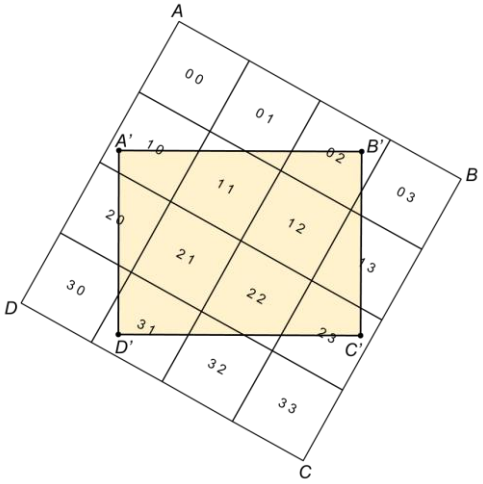


Рис. 6. Частичное попадание текстур в область отображения при их повороте относительно этой области

5. Фоновая загрузка текстур видимой области

На основе предложенного алгоритма можно не только построить подмножество кадров, присутствующих в видимой области, но и организовать их прогнозную загрузку, которая основывается на том факте, что при неизменном масштабе отображения и использовании встроенных в программу средств смещение локального окна видимости без изменения масштаба отображения не приведет более чем к однорядному отклонению.

Уточнение и загрузка набора текстур, участвующих в отображении топологии в текущий момент времени, встраиваются в программу на основе своего рода виртуализации используемых ресурсов, поддерживаемых представленной на рис. 7 архитектурой.

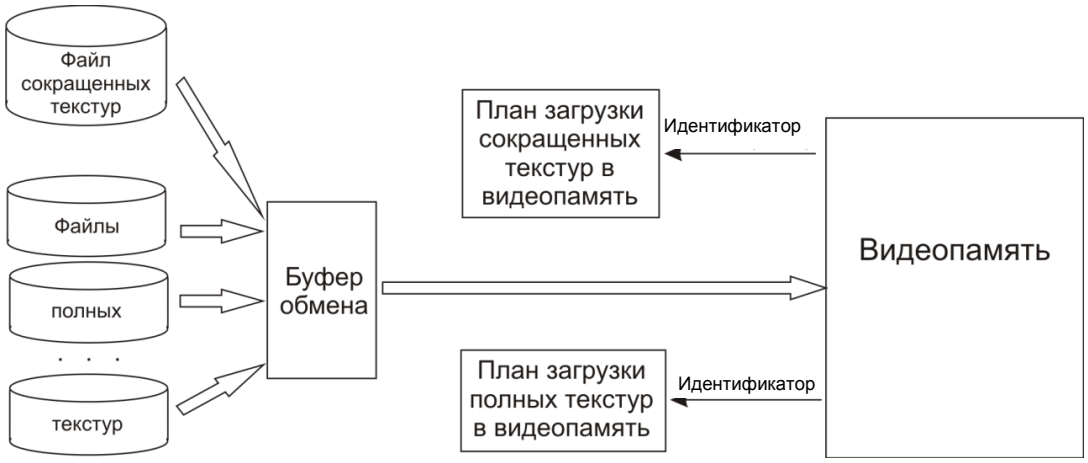


Рис. 7. Архитектура организации памяти

Планы загрузки текстур (как полных, так и сокращенных) представляют собой таблицы кадров слоя, где каждая ячейка предназначена для хранения идентификатора отдельной текстуры, загруженной в память видеокарты. Однако принципы управления планами существенно различаются.

План загрузки сокращенных текстур строится единожды из заранее сформированного MAP-файла при начальной загрузке данных слоя и содержит идентификаторы всех сокращенных текстур слоя. Такие текстуры присутствуют в памяти видеокарты в течение всего сеанса работы над данным проектом.

В отличие от плана сокращенных текстур план полных текстур формируется динамически в процессе сеанса. Полная текстура содержит в себе все уровни MIP-текстурирования и размещается в видеопамяти по специальному запросу управляющей программы просмотра при необходимости построения изображений топологии, требующих использования высокого разрешения.

Загрузчик полных структур представляется фоновым процессом, который реализует виртуализацию памяти на основе обеспеченного в среде Qt [1] сигнально-слотового взаимодействия.

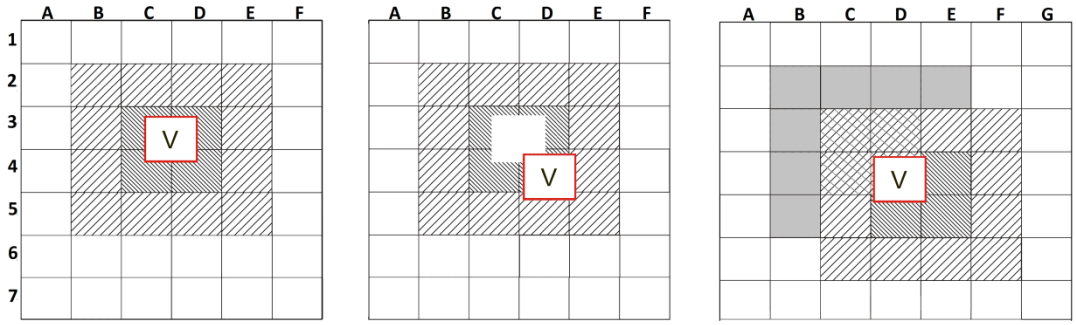
В программе реализации управления просмотром вырабатываются сигналы:

– «запустить управление загрузкой полных текстур»;

– «изменить набор видимых текстур» (при работе с полными текстурами);

– «освободить видеопамять от всех полных текстур» (при выключении режима работы с полными текстурами).

Предположим следующее: после очередного увеличения масштаба изображения управляющая программа обнаружила, что выполняется условие необходимости предъявления пользователю изображения с высоким разрешением и область видимости (отмечена буквой V) размещена так, как показано на рис. 8, *а*. В этом случае вырабатывается сигнал включения режима демонстрации полных текстур. Данный сигнал перехватывается фоновым процессом обработки полных текстур, который, используя доступные ему данные о позиции и размерах окна видимости, определяет множество кадров, чьи текстуры попадают хотя бы частично в область видимости (в примере это текстуры C3, D3, C4, D4, отмеченные густой обратной штриховкой). Если указанных текстур в текущий момент нет в памяти видеокарты, то организуется их загрузка с фиксацией полученных идентификаторов в плане. По завершении подготовки графической информации фоновый процесс информирует управляющую программу о готовности памяти, используемой при построении требуемого изображения. После этого работа управляющей программы и работа фонового процесса протекают параллельно. В то время как управляющая программа занимается прорисовкой топологии, в фоновом процессе осуществляется загрузка в память видеокарты всех текстур, размещенных по соседству с текстурами области видимости и обозначенных прямой штриховкой: B2, C2, D2, E2, B3, E3, B4, E4, B5, C5, D5, E5.



а) *б)* *в)*

Рис. 8. План загрузки полных текстур: *а)* по сигналу «включить режим отображения полных текстур»; *б)* сдвиг видимого фрагмента вправо и вниз; *в)* по сигналу «изменить набор видимых текстур»

Предположим далее, что пользователь сместил область видимости так, как это показано на рис. 8, *б* – вправо и вниз. Возникшая ситуация инициирует посылку сигнала «изменить набор видимых текстур».

Получив указанный сигнал, фоновый процесс оценивает ситуацию на необходимость проведения новой загрузки. На рис. 8, *в* видно, что для реализации отображения в новой позиции не потребуется проведение загрузки новых текстур – вся область видимости будет распо-

лагаться в пределах предварительно загруженных текстур. Следовательно, сигнал о готовности к построению изображения топологии в управляющую программу будет возвращаться практически мгновенно. Вместе с тем фоновый процесс продолжит выполняться параллельно. Как и ранее, реализуется загрузка «прогнозируемых» текстур: F3, F4, F5, C6, D6, E6, F6 – и освобождение занятой памяти видеокарты путем удаления текстур, «вышедших» из области прогноза: B2, C2, D2, E2, B3, B4, B5 (отмеченных на рис. 8, *в* серым цветом).

Заключение

Предлагаемые средства программной организации графической информации, описывающей многослойные и многокадровые изображения топологии СБИС, позволяют увеличивать размерности обрабатываемых схем, обеспечивая при этом дружественность программного интерфейса. Основным результатом их применения является высокое быстродействие отображения на экране фрагментов топологии.

Список литературы

1. Шлее, М. Qt 5.3. Профессиональное программирование на C++ / М. Шлее. – СПб. : БХВ – Петербург, 2015. – 928 с.

2. Райт, Р.С. OpenGL. Суперкнига / Р.С. Райт, Б. Липчак. – М. : Изд. дом «Вильямс», 2006. – 1040 с.

3. Programming Guide for DDS [Электронный ресурс]. – 2015. – Режим доступа : https://msdn.microsoft.com/en-us/enus/library/windows/desktop/ bb943991(v=vs.85).aspx. – Дата доступа : 23.11.2015.

4. MIP-MAP Filtering в процессе выполнения приложения [Электронный ресурс]. – 2015. – Режим доступа : http://www.ixbt.com/video/mip-mapping.html. – Дата доступа : 23.11.2015.

5. Статистика разрешений экрана 2000–2015 гг. [Электронный ресурс]. – 2015. – Режим доступа : http://www.fortress-design.com/display-resolution.– Дата доступа : 23.11.2015.

Поступила 02.06.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: rom@newman.bas-net.by
yurafreedom18@gmail.com

V.I. Romanov, Y.Y. Lankevich

ORGANIZATION OF GRAPHIC INFORMATION FOR VIEWING THE MULTILAYER VLSI TOPOLOGY

One of the possible ways to reorganize of graphical information describing the set of topology layers of modern VLSI. The method is directed on the use in the conditions of the bounded size of video card memory. An additional effect, providing high performance of forming multi-image layout a multi-layer topology of modern VLSI, is achieved by preloading the required texture by means of auxiliary background process.

УДК 519.8

Н.Н. Писарук

БИБЛИОТЕКА MIPCL ДЛЯ РЕШЕНИЯ ЗАДАЧ СМЕШАННО-ЦЕЛОЧИСЛЕННОГО ПРОГРАММИРОВАНИЯ

Рассматривается библиотека MIPCL – свободное программное обеспечение, предназначенное для быстрой и эффективной компьютерной реализации моделей смешанно целочисленного программирования (СЦП). Вычислительные эксперименты доказали, что сегодня MIPCL является наиболее эффективным свободным инструментом для решения задач СЦП. Библиотека MIPCL, документация и многочисленные примеры предоставляются на условиях лицензии GLPL (GNU Lesser Public License), поэтому MIPCL свободно доступна как для некоммерческого, так и коммерческого использования.

Введение

Задача СЦП представляет собой оптимизационную задачу с линейной целевой функцией и линейными ограничениями, в которой некоторые переменные должны принимать целочисленные значения. Многочисленные практические задачи из совершенно разных областей формулируются как задачи СЦП. Более обстоятельно о моделях и методах СЦП можно узнать, например, из источников [1–3].

В течение десятилетий после возникновения СЦП как раздела оптимизационной теории считалось, что СЦП является мощным инструментом для моделирования и почти бесполезно на практике, поскольку имеющиеся тогда компьютеры и программное обеспечение не могли решать практические задачи. За последние 20 лет эти устоявшиеся представления кардинально изменились: произошло впечатляющее улучшение качества программного обеспечения – как коммерческого, так и некоммерческого – для решения задач СЦП. Сегодня, используя более производительные компьютеры и лучшие программные продукты, можно решать многие классы трудных практических задач СЦП.

Некоммерческие программы для решения задач СЦП пока не могут конкурировать в скорости и надежности с лучшими коммерческими программами (CPLEX [4], GUROBY [5], Xpress-MP [6]), но они являются важной альтернативой дорогостоящему коммерческому программному обеспечению. В то же время для практических приложений умеренной размерности (а таких большинство) лучшие некоммерческие программы (SCIP [7], CBC [8], MIPCL [9]) почти так же эффективны, как и упомянутые выше коммерческие программы.

1. Библиотека MIPCL

MIPCL (Mixed Integer Programming Class Library) – это библиотека для решения задач СЦП. Она полностью реализована на C++ и может использоваться самостоятельно как разделяемая (динамическая) библиотека в C++ проектах, вместе с моделирующей оболочкой MIPshell, как модуль в Python программах или как утилита (самостоятельная программа) для решения задач СЦП, записанных в текстовых файлах в MPS-формате.

Подробное описание всех функций MIPCL можно найти в [9].

MIPCL реализована как семейство C++ классов. Два главных класса – это CMIP и CLP. CMIP есть программная система для решения задач СЦП, а CLP – система для решения задач линейного программирования (ЛП).

MIPCL поддерживает следующие функции:

– реализацию алгоритмов ветвей и сечений и алгоритмов ветвей и оценивания [2, 3];

– автоматическую переформулировку решаемых задач с целью их упрощения;

– устранение симметрии;

– реализацию прямых эвристик, которые пытаются «округлять» решения релаксационных задач;

– генерирование отсечений и столбцов;

– «сильное» ветвление.

MIPCL API позволяет разработчикам с легкостью интегрировать оптимизацию в свои приложения. Интерфейсы для разработчиков удобны и просты в использовании. MIPCL – очень гибкая программная система, что обусловлено большим количеством виртуальных функций, которые позволяют разработчикам переопределять (перепрограммировать) многие ее компоненты, чтобы лучше учесть специфику решаемой задачи. Такими переопределяемыми компонентами являются:

– процедуры отделения, в которых должны строиться специфичные для решаемой задачи отсечения;

– процедуры для генерирования столбцов в алгоритмах ветвей и оценивания;

– прямые эвристики для округления решений релаксационных задач;

– селекторы узлов, которые направляют обход узлов дерева поиска;

– процедуры ветвления, которые задают правила для разбиения задач на подзадачи;

– методы для переформулировки решаемых задач с целью их упрощения;

– процедуры и структуры данных для хранения отсечений и генерируемых столбцов.

MIPCL распространяется с понятной и обстоятельной документацией, а также множеством примеров, которые доступны в исходных кодах и должны помочь разработчикам программного обеспечения быстро освоить приемы работы с библиотекой.

1.1. Мультипоточность

Современные многоядерные персональные компьютеры обеспечивают аппаратную и программную среду для эффективных параллельных вычислений. Поэтому дизайны большинства современных библиотек для решения задач СЦП (включая MIPCL) были переработаны с целью распараллелить работу алгоритмов ветвей и сечений и алгоритмов ветвей и оценивания.

Разработка и отладка сложных мультипоточных приложений – далеко не тривиальная задача. При разработке мультипоточной библиотеки MIPCL главной задачей было не добавить будущим пользователям дополнительных сложностей. В результате, если пользователь в своем приложении не перегружает никаких MIPCL-функций, то беспокоиться ему не о чем: приложение будет работать одинаково хорошо как с однопоточной, так и с многопоточной библиотеками. Однако если приложение генерирует отсечения или столбцы, реализует более подходящую для решаемой задачи стратегию ветвления или просто переопределяет любую виртуальную функцию базового класса CMIP, то в своем производном классе пользователю необходимо реализовать конструктор клонирования вместе с интерфейсом, который MIPCL использует для создания объектов производного класса. Конструктор клонирования вызывается при создании нового потока. Он очень похож на конструктор копирования за исключением того, что не должен выделять память для хранения разделяемых (общих для всех потоков) данных.

1.2. Эффективность

Для тестирования MIPCL утилита *mps_mipcl* с лимитом времени 2 ч на тест решала задачи СЦП из библиотеки тестов MIPLIB2010 [10] (основное семейство тестов). Результаты тестов фиксировались с помощью MIPLIB2010-скриптов. Характеристики компьютера, на котором проводились тесты, следующие: AMD FX8120, 8 ядер, 8 GB, 3,1 GHz. В сжатой форме эти результаты представлены в последнем столбце таблицы.

Результаты тестов

	CBC	CPLEX	SCIPC	SCIPS	GUROBI	XPRESS	MIPCL
Решены	61	86	72	67	86	86	82
Прерваны	24	1	10	17	1	1	5
Не решены	2	0	5	3	0	0	0
GMRT	17,4	1,05	8,19	12,7	1	1,3	4,1

Для сравнения MIPCL с лучшими конкурирующими программами (коммерческими и некоммерческими) были использованы результаты тестов, проведенных Х. Миттельманом (http://mittelmann@asu.edu). В этих тестах с лимитом времени в 2 ч решались задачи из той же библиотеки MIPLIB2010 с помощью следующих программ:

CPLEX: CPLEX-12.6.1 [4];

GUROBY: GUROBY-6.0.0 [5];

SCIPC: ug[SCIP/cpx]-3.1.1 (разрабатываемая параллельная версия SCIP+CPLEX) (http://www.zib.de);

SCIPS: ug[SCIP/spx]-3.1.1 (разрабатываемая параллельная версия SCIP+SOPLEX) (http://www.zib.de);

CBC: CBC-2.9.4 [11];

XPRESS: XPRESS-7.8.0 [6].

Отметим, что среди этих программ некоммерческими являются только SCIPS и CBC.

Результаты MIPCL сравнивались с результатами конкурентов, полученными на компьютере со следующими характеристикам: Intel i7-2600, 4 ядра, 16 GB, 3,4 GHz. Заметим, что производительность такого компьютера даже несколько выше производительности компьютера, на котором осуществлялось тестирование MIPCL.

В компактном виде результаты всех тестов представлены в таблице, где GMRT означает отношение геометрического среднего времен решения всех тестов к такому же показателю самой быстрой программы.

2. Моделирующая оболочка MIPshell

Каждый, кто когда-либо пробовал применять математическое программирование на практике, знает, что в большинстве случаев это непростое упражнение. Для упрощения моделирования и решения оптимизационных задач были разработаны языки оптимизационного моделирования OPL [12], AMPL [13] и др. MIPshell не является таким языком. Это просто программная оболочка для облегчения моделирования и решения задач ЛП и СЦП с использованием библиотеки MIPCL. MIPshell включает в себя:

– семейство C++ классов, разработанных с использованием библиотек STL, для представления переменных, ограничений, множеств, векторов, ассоциативных массивов (словарей) и других структур данных;

– набор функций для упрощения записи оптимизационных задач;

– препроцессор, который транслирует новые (не из C++) операторы моделирования в фрагменты кода на языке C++.

Существует несколько причин, по которым оболочка MIPshell не трансформировалась в полноценный язык оптимизационного моделирования. При разработке сложных практических приложений требуется программная среда для реализации графических интерфейсов пользователя, интерфейсов для работы с базами данных, сетевыми протоколами (для связи с удаленными серверами) и еще много чего другого. Ни один из существующих языков оптимизационного моделирования не реализует даже минимум из необходимого. Между тем современные средства разработки программного обеспечения, такие как Qt или Microsoft Visual Studio, имеют в своем составе все необходимое. Разработчики могут легко интегрировать MIPCL вместе с MIPshell в любую из таких систем разработки.

Демонстрационный пример. Рассмотрим следующую задачу СЦП:

$$\sum_{e \in E} (f_e y_e + c_e x_e) \to \min;$$

$$\sum_{e \in E: t_e = v} x_e - \sum_{e \in E: h_e = v} x_e = d_v, v \in V,$$

$$0 \le x_e \le u_e y_e,\ e \in E,$$

$$y_e \in \{0, 1\},\ e \in E,$$

которая является формулировкой сетевой транспортной задачи с постоянными издержками (см. [2]). Здесь $G=(V, E)$ – ориентированный граф, d_v – спрос в узле (вершине) $v \in V$ на некоторый продукт, а каждой дуге $e=(v, w) \in E$, соединяющей начальную вершину $t_e=v$ с конечной вершиной $h_e=w$, приписаны пропускная способность u_e, а также постоянные издержки f_e и переменные издержки c_e: $f_e + c_e x_e$ есть стоимость транспортировки $x_e > 0$ единиц продукта по дуге e (если $x_e=0$, то ничего платить не нужно).

В MIPshell задача СЦП записывается следующим образом:

```
int e;
VAR_VECTOR x("x",REAL_GE,E);
VAR_VECTOR y("y",BIN,E);

minimize(sum(e in E) (f(e)*y(e) + c(e)*x(e)));
forall(v in V)
    sum(e in E: h(e)==v) x(e) – sum(e in E: t(e)==v) x(e) == d(v);
forall(e in E)
    x(e) <= u(e)*y(e);
```

Чтобы облегчить изучение оболочки, MIPshell поставляется с ясной и обстоятельной документацией, а также большим количеством примеров. Поскольку все примеры доступны в исходных кодах, они будут полезны и при изучении СЦП.

3. Python-MIPCL

Python-MIPCL представляет собой модуль на языке программирования Python. По своему назначению и функциональности он аналогичен оболочке MIPshell. Python-MIPCL включает в себя следующие компоненты:

– разделяемую библиотеку mipcl.so (на windows компьтерах mipcl.pyd), которая является библиотекой MIPCL с дополнительным интерфейсом для вызовов из программ на языке Python;

– набор классов на языке Python, которые спроектированы для представления переменных, ограничений и оптимизационных задач;

– семейства функций для упрощения записи формулировок задач СЦП.

Python-MIPCL реализует только часть функциональности базовой оболочки MIPshell. В частности, Python-MIPCL не позволяет реализовывать приложения, которые должны генерировать отсечения. Причина заключается в том, что реализовывать такие приложения на скриптовом языке неэффективно, а подавляющее большинство практических приложений этого и не требует. Далеко не каждый, кто использует оптимизацию на практике, знает C++. Python прост для начинающих и поэтому Python-MIPCL поможет расширить круг пользователей MIPCL.

Для сравнения функциональности Python-MIPCL с MIPshell ниже представлен фрагмент кода на языке Python, в котором записана модель рассмотренной задачи СЦП:

```
class Fcnf(Problem):
    def model(self,t,h,u,d,f,c):
        m = len(t)
        n = len(d)
        x = VarVector([m],"x",REAL|GE)
        y = VarVector([m],"y",BIN)

        minimize(sum_(f[e]*y[e] + c[e]*x[e] for e in range(0,m)))
        for v in range(0,n):
            sum_(x[e] for e in range(0,m) if  h[e]==v) -\
```

```
        sum_(x[e] for e in range(0,m) if  t[e]==v) == d[v]
    for e in range(0,m):
        x[e] <= u[e]*y[e]
```

Список литературы

1. 50 Years of Integer Programming 1958–2008 / M. Jürgen [et al.]. – Berlin : Springer, 2009.

2. Wolsey, L.A. Integer Programming / L.A. Wolsey. – Wiley, 1998. – 288 p.

3. Pisaruk, N.N. Models and methods of mixed-integer programming / N.N. Pisaruk. – Minsk : Belarus State University, 2010.

4. IBM. IBM ILOG CPLEX Optimization Studio [Electronic resource]. – 2015. – Mode of access : http://www-01.ibm.com/software/integration/optimisation/cplex-optimization-studio. – Date of access : 06.03.2015.

5. Guroby Optimization Inc. Guroby Optimizer [Electronic resource]. – 2012. – Mode of access : http://www.guroby.com/welcome.html. – Date of access : 06.03.2015.

6. Dash Optimization. Xpress-MP [Electronic resource]. – 2015. – Mode of access : http://www.dashoptimization.com. – Date of access : 06.03.2015.

7. Achterberg, T. SCIP: solving constraint integer programs / T. Achterberg // Math. Prog. Comp. – 2009. – Vol. 1. – P. 1–41.

8. COIN-OR. Computational Infrastructure for Operations Research [Electronic resource]. – 2015. – Mode of access : http://www.coin-or.org. – Date of access : 06.03.2015.

9. MIPCL Reference Manual [Electronic resource]. – 2015. – Mode of access : http://mipcl-cpp. appspot.com/static/docs/mipcl/html/index.html. – Date of access : 06.03.2015.

10. MIPLIB2010 [Electronic resource]. – 2015. – Mode of access : http://miplib.zib.de. – Date of access : 06.03.2015.

11. Forrest, J.J.H. COIN branch and cut / J.J.H. Forrest [Electronic resource]. – 2015. – Mode of access : http://www.coin-or.org. – Date of access : 06.03.2015.

12. Modeling with OPL [Electronic resource]. – 2015. – Mode of access : http://www-01. ibm.com/software/commerce/optimization/modeling/index.html. – Date of access : 06.03.2015.

13. Fourer, R. AMPL: A Modeling Language for Mathematical Programming / R. Fourer, B.W. Kernigan. – Duxbury Press, 2002.

Поступила 20.05.2016

Белорусский государственный университет,
Минск, ул. К. Маркса, 31
e-mail: pisaruk@yandex.by

N.N. Pisaruk

MIPCL LIBRARY FOR SOLVING MIXED INTEGER PROGRAMMING PROBLEMS

The Mixed Integer Programming Class Library (MIPCL) is a free software designed for implementing mixed integer programming models quickly, easily, and efficiently. Computational experiments show that currently MIPCL is one of the best noncommercial mixed-integer programming solvers. MIPCL libraries, documentation and examples are provided under the terms of the GNU Lesser Public License. Therefore, MIPCL is equally freely available for noncommercial and commercial use.

УДК 510.5

В.Г. Найденко

АЛГОРИТМИЧЕСКОЕ ПЕРЕЧИСЛЕНИЕ ЗАДАЧ В КЛАССЕ NP∩coNP

Рассматривается проблема рекурсивного (алгоритмического) представления класса сложности NP∩coNP. *Предлагается новый метод алгоритмического перечисления всех задач в классе сложности* NP∩coNP *с использованием полиномиальных недетерминированных машин Тьюринга.*

Введение

Класс сложности NP∩coNP занимает важное положение в теории вычислительной сложности и играет исключительную роль в криптографии с открытым ключом [1], поскольку последняя во многом основана на задаче факторизации, лежащей в NP∩coNP (под задачей факторизации подразумевается следующая проблема: для заданных натуральных чисел n и k нужно ответить на вопрос, имеет ли число n простой делитель, меньший, чем k). Напомним, что язык находится в NP∩coNP, если существуют две недетерминированные полиномиальные машины Тьюринга: одна машина для распознавания языка, а вторая – для распознавания дополнения этого языка. Такие машины называются комплементарными друг другу. Однако требование комплементарности препятствует эффективной характеризации класса NP∩coNP. Так, Wojciech Kowalczyk [2] показал, что перечисление языков из NP∩coNP с помощью пар комплементарных машин весьма затруднительно. Известны оракулы, относительно которых не существует полной проблемы в релятивизированном классе NP∩coNP. Поэтому вполне вероятно, что сам класс NP∩coNP не содержит полной проблемы. В таком случае невозможно рекурсивно перечислить все языки из NP∩coNP с помощью пар комплементарных машин. Из этого вытекает следующее утверждение: независимо от мощи формальной математической теории (типа арифметики Пеано, теории множеств Цермело – Френкеля и т. д.) всегда найдется такая пара (T, M) полиномиальных недетерминированных машин Тьюринга, что невозможно доказать их комплементарность в рамках данной теории, а следовательно, и принадлежность распознаваемого машиной T языка классу NP∩coNP. В связи с этим ведущими специалистами в логике и теории вычислительной сложности предполагалось крайне маловероятным нахождение какого-либо алгоритмического представления класса сложности NP∩coNP [3, 4]. Так, президент Европейской ассоциации по логике в информатике, профессор Anuj Dawar [3] предполагал, что классы сложности, определяемые семантическими ограничениями на удостоверяющие машины, например, такие, как классы NP∩coNP и RP, не допускают очевидных рекурсивных представлений. Кроме того, он считал [3], что нахождение рекурсивного представления для класса NP∩coNP потребует фундаментально новой характеризации данного класса и явится главным прорывом в теории сложности.

Цель настоящей работы – найти такое алгоритмическое перечисление задач из класса сложности NP∩coNP, которое не требует соблюдения условия комплементарности.

1. Основные результаты

Дадим необходимые определения. Пусть Σ – конечный алфавит. Как обычно, через Σ^* обозначим множество всех слов (или конечных цепочек) в алфавите Σ. Языком называется любое подмножество множества Σ^*. Через $|w|$ обозначим длину слова w. Язык L распознается машиной Тьюринга T, когда выполняются следующие условия: если $w \in L$, то T останавливается на слове w в специальном допускающем состоянии (T допускает слово w); если $w \notin L$, то T останавливается на слове w в специальном отвергающем состоянии (T отвергает слово w).

Через $T(w)$ обозначим предикат, который принимает значение ИСТИНА, если T допускает слово w; в противном случае значение $T(w)$ – ЛОЖЬ.

Множество языков $\{L_i \mid i = 1,\ 2, ...\}$ называется рекурсивно представимым, если существует рекурсивное перечисление машин Тьюринга $\{T_i \mid i = 1,\ 2, ...\}$, такое, что выполняются два условия:

– для каждого языка из $\{L_i \mid i = 1,\ 2, ...\}$ существует машина Тьюринга из $\{T_i \mid i = 1,\ 2, ...\}$, распознающая этот язык;

– для каждой машины Тьюринга из $\{T_i \mid i = 1,\ 2, ...\}$ существует распознаваемый ею язык из $\{L_i \mid i = 1,\ 2, ...\}$.

Перейдем к рассмотрению класса сложности NP∩coNP. Каждой паре полиномиальных недетерминированных машин Тьюринга (T, M) сопоставим следущий язык $L_{T,M} \subseteq \Sigma^*$:

$$L_{T,M} = \{x \mid T(x) \wedge \forall y[\,|y| > \log_2(\log_2(|x|+1)+1) \vee (T(y) \Leftrightarrow \neg M(y))]\}. \quad (1)$$

Покажем, что язык $L_{T,M}$ принадлежит классу NP. Сначала оценим количество времени, достаточное для проверки условия, входящего в определение (1):

$$\forall y[\,|y| > \log_2(\log_2(|x|+1)+1) \vee (T(y) \Leftrightarrow \neg M(y))]. \quad (2)$$

Необходимо проверить соотношение $T(y) \Leftrightarrow \neg M(y)$ для всех цепочек y достаточно малой длины, т. е. для $|y| \le \log_2(\log_2(|x|+1)+1)$. Отметим, что проверка отдельного условия $T(y) \Leftrightarrow \neg M(y)$ занимает экспоненциальное время по длине |y|, но поскольку длина |y| достаточно мала, общее время проверки условия (2) будет полиномиально по длине входа |x|. Следовательно, язык $L_{T,M}$ принадлежит классу NP и можно представить некоторую полиномиальную недетерминированную машину Тьюринга $D_{T,M}$ для распознавания языка $L_{T,M}$. Работа машины $D_{T,M}$ на входной цепочке $x \in \Sigma^*$ описывается следующим образом.

Для всех цепочек $y \in \Sigma^*$, таких, что $|y| \le \log_2(\log_2(|x|+1)+1)$, машина $D_{T,M}$ моделирует работу машин T и M на входе y. Если для некоторой цепочки y окажется, что либо обе машины T и M допускают y, либо обе отвергают y, то $D_{T,M}$ отвергает входную цепочку x и завершает работу. Иначе после проверки всех цепочек y машина $D_{T,M}$ начинает работать так же, как машина T на входе x.

Справедлива следующая

Теорема. *Пусть* $\{(T_i, M_i) \mid i = 1,\ 2, ...\}$ *– рекурсивное перечисление всех пар полиномиальных недетерминированных машин Тьюринга. Тогда, взяв рекурсивное перечисление* $\{D_{T_i,M_i} \mid i = 1,\ 2, ...\}$*, получим рекурсивное представление класса сложности* NP∩coNP.

Доказательство. Сначала покажем, что любой язык $L_{T,M}$ из перечисления $\{L_{T_i,M_i} \mid i = 1,\ 2, ...\}$ находится в классе NP∩coNP. Заметим, что если соотношение $T(y) \Leftrightarrow \neg M(y)$ выполняется вообще для всех цепочек y (независимо от их длины), то язык $L_{T,M}$ по определению будет принадлежать классу NP∩coNP (поскольку в этом случае $L_{T,M}$ будет распознаваться машиной T, а его дополнение – машиной M). В противном случае язык $L_{T,M}$ будет конечным множеством и, следовательно, опять $L_{T,M} \in \mathrm{NP} \cap \mathrm{coNP}$. Таким образом, в любом случае доказывается принадлежность $L_{T,M}$ классу NP∩coNP.

Осталось показать, что любой язык L из NP∩coNP находится в перечислении $\{L_{T_i,M_i} \mid i = 1,\ 2,\ ...\}$. Поскольку $L \in \mathrm{NP} \cap \mathrm{coNP}$, то существуют полиномиальные

недетерминированные машины Тьюринга *T* и *M*, распознающие язык *L* и его дополнение $\Sigma^* \setminus L$ соответственно. Так как $\{(T_i, M_i) \mid i = 1,\ 2, ...\}$ – рекурсивное перечисление всевозможных пар полиномиальных недетерминированных машин Тьюринга, из этого перечисления найдется такая пара (T_i, M_i), что $T_i = T$ и $M_i = M$. Поскольку условие $T_i(y) \Leftrightarrow \neg M_i(y)$ выполняется для всех цепочек $y \in \Sigma^*$ (независимо от их длины), условие (2) будет выполняться для всех входных цепочек $x \in \Sigma^*$. Поэтому язык L_{T_i,M_i} распознается не только машиной D_{T_i,M_i}, но и машиной T_i. Следовательно, $L = L_{T_i,M_i}$ в связи с тем, что $T_i = T$. Таким образом, $L \in \{L_{T_i,M_i} \mid i = 1,\ 2, ...\}$.

Итак, показано, что $\{L_{T_i,M_i} \mid i = 1,\ 2, ...\} = \text{NP} \cap \text{coNP}$. Следовательно, $\{D_{T_i,M_i} \mid i = 1,\ 2, ...\}$ – рекурсивное представление класса сложности NP∩coNP. ■

2. Методологическое значение

Отметим, что алгоритмическая неразрешимость проблемы установления комплементарности пары машин Тьюринга крайне затрудняет (если не делает невозможным) доказательство принадлежности многих задач классу NP∩coNP. Разработанная характеризация задач из класса NP∩coNP не требует использования алгоритмически неразрешимого условия комплементарности машин Тьюринга, что дает возможность нахождения множества новых задач в NP∩coNP. Действительно, общепринятый конструктивный способ доказательства принадлежности какого-либо языка *L* классу NP∩coNP предусматривает построение пары комплементарных полиномиальных недетерминированных машин Тьюринга. Однако Wojciech Kowalczyk [2] показал, что следующее перечисление «доказуемых» комплементарных пар:

$$\{(T_i, M_i, \text{«доказательство комплементарности машин } T_i \text{ и } M_i\text{»}) \mid i = 1,\ 2, ...\} \qquad (3)$$

охватывает не все комплементарные пары, если класс NP∩coNP не содержит полной проблемы. Однако тогда в NP∩coNP существуют языки, для которых невозможно построить комплементарные пары машин Тьюринга в рамках формальной математической теории множеств Цермело – Френкеля [2].

Рассмотрим теперь другое перечисление:

$$\{(T_i, \text{«доказательство существования машины } M_i, \text{ комплементарной к машине } T_i\text{»}) \mid i = 1, 2, ...\}. \qquad (4)$$

Поскольку перечисление (4) включает в себя все машины из перечисления $\{D_{T_i,M_i} \mid i = 1,\ 2, ...\}$, то оно является рекурсивным представлением класса сложности NP∩coNP. Это подсказывает следующий неконструктивный способ установления принадлежности языка классу NP∩coNP: вместо построения пары комплементарных машин (что может оказаться невозможным в рамках формальной теории) следует доказать существование пары комплементарных машин. Вполне возможно, что для ряда задач (например, для задачи изоморфизма графов), лежащих предположительно в классе NP∩coNP, действительно нельзя построить пары комплементарных машин в рамках формальной математической теории множеств Цермело – Френкеля. Однако отыщется неконструктивное доказательство, если рассматриваемый язык действительно принадлежит классу NP∩coNP.

Заключение

В работе впервые получено фундаментально новое описание задач из класса NP∩coNP, при котором не требуется использование комплементарных пар машин Тьюринга.

С учетом центральной роли класса NP∩coNP в криптографии с открытым ключом новая характеризация имеет не только фундаментальное, но и важное прикладное значение. Кроме

того, алгоритмическое представление может быть использовано для логической характеризации данного класса сложности [5].

Работа профинансирована Институтом математики НАН Беларуси в рамках государственной программы фундаментальных исследований «Конвергенция – 2020».

Список литературы

1. Brassard, G. A Note on Cryptography and NP∩CoNP–P / G. Brassard, S. Fortune, J. Hopcroft // Technical Report TR-338, Department of Computer Science. – Ithaca, N.Y. : Cornell University, 1978.

2. Kowalczyk, W. Some Connections between Representability of Complexity Classes and the Power of Formal Systems of Reasoning / W. Kowalczyk // Proc. of the Mathematical Foundations of Computer Science. – Heidelberg : Springer, 1984. – Vol. 176. – P. 364–369.

3. Dawar, A. On Complete Problems, Relativizations and Logics for Complexity Classes / A. Dawar // Lecture Notes in Computer Science. – 2010. – Vol. 6300. – P. 201–207.

4. Papadimitriou, Ch. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence / Ch. Papadimitriou // J. of Computer and System Sciences. – 1994. – Vol. 48, no. 3. – P. 498–532.

5. Naidenko, V. Logics for complexity classes / V. Naidenko // Logic J. of the IGPL. – 2014. – Vol. 22, no. 6. – P. 1075–1093.

Поступила 31.05.2016

Институт математики НАН Беларуси,
Минск, ул. Сурганова, 11
e-mail: vladimir.naidenko@gmail.com

V.G. Naidenko

ALGORITHMIC ENUMERATION OF PROBLEMS IN THE CLASS NP∩coNP

The problem of recursive (algorithmic) representation is considered for the complexity class NP∩coNP. A new method is proposed for algorithmically enumerating all problems in NP∩coNP, using polynomial-time nondeterministic Turing machines.

УДК 681.511.2; 621: 658.011.56

А.А. Несенчук

МОДЕЛИРОВАНИЕ ДИНАМИКИ ЭЛЕКТРОПРИВОДА НА ОСНОВЕ КОРНЕВОЙ МОДЕЛИ

Рассматривается задача моделирования управления в системе векторного управления асинхронного электродвигателя с использованием математической модели в форме расширенного корневого годографа. Выполняется расчет параметров системы, обеспечивающих устойчивость при функционировании в условиях существенной параметрической неопределенности объекта.

Введение

Среди современных методов синтеза систем управления сложными техническими объектами наиболее распространенными являются частотные [1], корневые [1–5], модального управления [2], позволяющие задавать желаемое расположение корней, пространства состояний [1], хорошо подходящие для структурного синтеза, и ряд других. Интересны задачи об устойчивости в условиях неопределенности, решаемые в современных постановках в робастном варианте [3–6], для которых может быть использован корневой подход. Преимущество данного подхода к проблеме состоит в том, что само его использование уже предполагает параметрические вариации (неопределенность). Он идеально подходит для синтеза систем и отличается большой наглядностью, позволяя не только рассчитывать требуемые значения параметров системы, но и в деталях наблюдать характер изменения динамических свойств, реакцию системы в ответ на параметрические вариации, что особенно важно при исследовании систем с неопределенными параметрами.

Качество управления потокосцеплением в системах векторного управления электроприводом в значительной степени определяет качество управления электромагнитным моментом и скоростью, энергетическую эффективность привода. Актуальной является проблема управления потокосцеплением в условиях параметрической неопределенности [7]. Поэтому важное значение имеет вопрос выбора параметров характеристического полинома синтезируемой системы по критерию принадлежности корней заданной области при условии неопределенности параметров объекта.

1. Структурная схема канала управления потокосцеплением

На структурной схеме системы управления потокосцеплением электропривода (рис. 1) управляемым объектом является асинхронный электродвигатель.

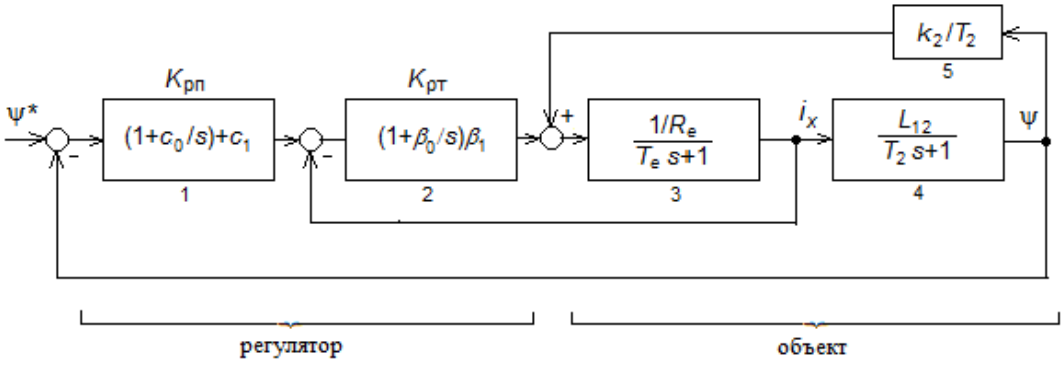


Рис. 1. Структурная схема системы управления потокосцеплением электропривода

Уравнение динамики системы управления имеет следующий вид:

$$(((\psi^* - \psi)K_{pn} - i_x)K_{pm} + \frac{k_2}{T_2}\psi)\frac{1/R_e}{T_e s + 1} = i_x = \frac{\psi(T_2 s + 1)}{L_{12}}, \quad (1)$$

где ψ – потокосцепление; $K_{\text{рп}}$, $K_{\text{рт}}$ – передаточные функции регулятора потокосцепления и тока соответственно; c_0, c_1, β_0, β_1 – коэффициенты регуляторов потокосцепления и тока (c_0 = const); R_e – сопротивление эквивалентное, Ом; T_e – постоянная времени эквивалентная, с; L_{12} – взаимоиндуктивность статора и ротора, Гн (L_{12} = const); T_2 – постоянная времени ротора, с; k_2 = const; i_x – сила тока, А.

На основе (1) запишем характеристический полином системы в виде

$$N = (s + \beta_0)\beta_1(s + c_0)c_1 + \beta_1(s + \beta_0)(T_2 s + 1)s - \frac{k_2}{T_2}s^2 + (T_2 s + 1)(T_e s + 1)\frac{R_e}{L_{12}}s^2, \quad (2)$$

где $\beta_0 = \frac{1}{T_e}$; $\beta_1 = \frac{T_e R_e}{2T_\mu}$; $\beta_{pn} = \frac{T_2}{4T_\mu L_{12}}$; $c_0 = \frac{1}{T_2}$; $c_1 = \beta_{\text{рп}}$; T_μ – постоянная времени.

Параметры β_0, β_1, c_0, c_1 представляют собой параметры регулятора; параметры R_e, L_{12}, T_e, T_2, k_2 являются параметрами объекта. На основе (2) формируются следующие выражения для определения коэффициентов полинома (2), которые связывают эти коэффициенты с физическими параметрами системы:

$$a_0 = T_2 T_e \frac{R_e}{L_{12}}; \quad (3)$$

$$a_1 = T_e \frac{R_e}{L_{12}} + T_2 \frac{R_e}{L_{12}} + \beta_1 T_2; \quad (4)$$

$$a_2 = \beta_1 c_1 + \beta_1\beta_0 T_2 + \beta_1 - \frac{k_2}{T_3} + \frac{R_e}{L_{12}}; \quad (5)$$

$$a_3 = \beta_0\beta_1 c_1 + \beta_1 c_0 c_1 + \beta_0\beta_1; \quad (6)$$

$$a_4 = \beta_0\beta_1 c_0 c_1. \quad (7)$$

2. Характеристический полином системы и его расширение. Постановка задачи

Ввиду воздействия различных внутренних и внешних факторов параметры объекта (электродвигателя) при его работе могут изменяться в широких пределах. Поэтому будем рассматривать систему управления электродвигателем как систему с параметрической, интервальной, неопределенностью. Преобразуем характеристический полином (2) системы управления приводом к следующему общему виду:

$$s^4 + a_1 s^3 + a_2 s + a_3 = p(s), \quad (8)$$

где коэффициенты a_j являются действительными и определяются выражением

$$\underline{a}_j \le a_j \le \overline{a}_j, \ \ j = \overline{0, 4}, \ \ a_0 \ne 0; \quad (9)$$

$\underline{a}_j$ и $\overline{a}_j$ – граничные значения интервалов изменения коэффициентов a_j.

Расширение [4] от полинома (3) имеет следующий вид:

$$E_4(p(s)) = \begin{cases} s + a_1 = 0, & (a) \\ s^2 + a_1 s + a_2 = 0, & (b) \\ s^3 + a_1 s^2 + a_2 s = 0, & (c) \\ s^4 + a_1 s^3 + a_2 s^2 + a_3 s + a_4 = 0. & (d) \end{cases} \quad (10)$$

Выражение (10) представляет собой систему полиномов, где снизу записывается полином (8), а каждый следующий перед ним полином получается из предыдущего посредством вычитания свободного члена и деления на оператор s. Каждый предыдущий сверху полином системы (10) называется *порождающим*, а последующий – *порождаемым* полиномом системы.

Устанавливаются границы

$$\sigma_{max} = \sigma_1,\ \sigma_{min} = \sigma_2 \tag{11}$$

области качества Q, ограниченной двумя линиями β_1 и β_2 (рис. 2) равной степени устойчивости, внутри которой требуется разместить семейство корней интервального полинома (8) с целью обеспечения заданного качества работы привода. С целью решения поставленной задачи используется расширение (10) полинома (3) и, соответственно, его расширенный корневой годограф [4]. Примем значения $\sigma_{max} = -4{,}5$, $\sigma_{min} = -105$.

Интервалы значений параметров коэффициентов (8), в пределах которых сохраняется робастная устойчивость и робастное качество [3, 6] системы, определяются на основании заданных границ (11) области качества Q и с использованием расширения (10) характеристического полинома (8) системы.

3. Вычисление значений интервальных параметров

С целью вычисления значений границ интервалов параметров a_2, a_3 и a_4 используем соответственно уравнения (10.*b*), (10.*c*) и (10.*d*) расширенного полинома (10) в соответствии с разработанной методикой [4]. Чтобы применить данную методику к интервальному характеристическому полиному для обеспечения заданных требований качества переходного процесса, необходимо разделить область Q линиями равной степени устойчивости b_1 и b_2 (рис. 2), которые соответственно будут являться границами областей размещения корней (области качества) полиномов (10.*b*) и (10.*c*). В результате область качества Q, расположенная между заданными границами β_1 и β_2 локализации корней системы, оказалась разделенной на три подобласти – Q_1, Q_2 и Q_3, две первые из которых ограничивают размещение корней полиномов (10.*b*) и (10.*c*) расширения соответственно. Границы интервалов a_1, a_2 и a_3 вычисляются последовательно.

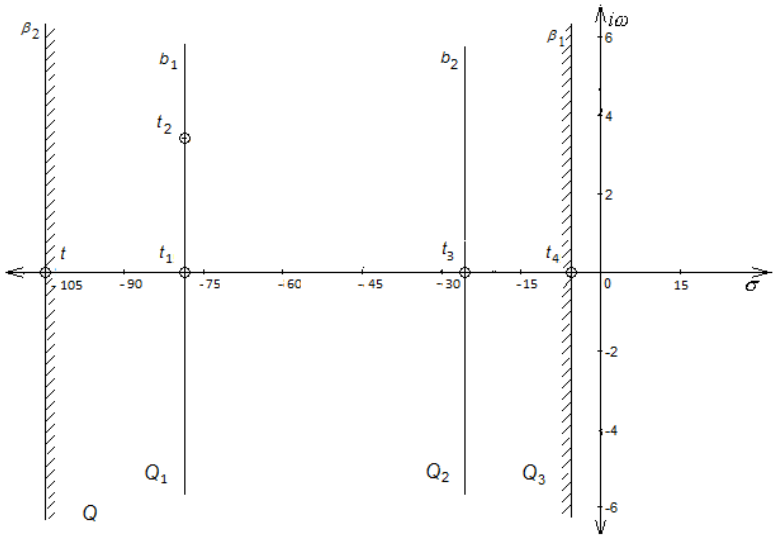


Рис. 2. Расположение границ β_1 и β_2 области качества Q системы

С целью синтеза требуемого характеристического полинома (8) рассмотрим вначале полином второй степени (10.*b*). Подставив в (10.*b*) выражение $s = \sigma + i\omega$, запишем

$$(\sigma + i\omega)^2 + a_1(\sigma + i\omega) + a_2 = 0$$

и после преобразования получим

$$\sigma^2 + i(2\sigma\omega + a_1\omega) - \omega^2 + a_1\sigma + a_2 = 0.$$

На основе последнего выражения запишем уравнение корневого годографа (УКГ)

$$2\sigma\omega + a_1\omega = 0 \qquad (12)$$

и уравнение параметра (УП) годографа

$$\sigma^2 - \omega^2 + a_1\sigma + a_2 = 0. \qquad (13)$$

УКГ (12) можно представить в виде двух уравнений, первое из которых представляет собой уравнение Бендрикова – Теодорчика, т. е. УКГ в комплексной плоскости:

$$\sigma = -a_1/2, \qquad (14)$$

а второе является УКГ на действительной оси:

$$\omega = 0.$$

Используя УП (13), получим следующее выражение для определения параметра a_2:

$$a_2 = -\sigma^2 + \omega^2 - a_1\sigma = \sigma^2 + \omega^2. \qquad (15)$$

Установим значение параметра a_1 таким образом, чтобы корневой годограф (14) располагался внутри области качества Q ближе к левой границе β_1 этой области. Определим значение $a_1 = 155 = \text{const}$, поскольку, исходя из установленных особенностей конфигурации, динамических, асимптотических свойств семейств годографов систем второго порядка [3], при любых значениях a_1, изменяющихся в пределах $9 < a_1 < 155$, семейство комплексных ветвей годографов полинома второй степени будет располагаться в пределах области Q, ограниченной параллельными линиями β_1 и β_2. Уравнение корневого годографа будет иметь следующий вид:

$$\sigma = -77{,}5.$$

Комплексные ветви данного корневого годографа совпадают с границей b_1, для которой координата σ_{b1} пересечения с осью σ определяется выражением

$$\sigma_{b1} = -a_1/2 = -77{,}5.$$

Интервал изменения параметра a_2 определим между точками $t_1(-77{,}5;0)$ и $t_2(-77{,}5;40)$ как

$$[\underline{a}_2, \bar{a}_2] = [a_2(t_1), a_2(t_2)]$$

и, используя (15), вычислим значения a_2 в точках t_1 и t_2:

$$a_2(t_1) = \underline{a}_2 = (-77{,}5)^2 + 0 = 6006{,}25; \; a_2(t_2) = \bar{a}_2 = (-77{,}5)^2 + 40^2 = 7606{,}25.$$

Вычисленный на основании (10.*b*) интервал изменения параметра a_2 годографа имеет следующий вид: $[\underline{a}_2, \bar{a}_2] = [6006{,}25,\ 7606{,}25]$.

С целью вычисления значений границ интервалов параметра a_3 используем соответственно уравнение (10.*c*) расширенного полинома (10). Осуществив преобразования, аналогич-

ные выполненным при вычислении допустимых границ вариации параметра a_2, запишем УКГ и УП для уравнения (10.*c*):

$$3\sigma^2\,\omega - \omega^3 + 2a_1\sigma\omega + a_2\omega = v(\sigma, \omega) = 0; \tag{16}$$

$$\sigma^3 - 3\sigma\omega^2 + a_1\sigma^2 - a_1\omega^2 + a_2\sigma = u(\sigma, \omega) = -a_3. \tag{17}$$

На основе (16) и (17) запишем в комплексной плоскости УКГ

$$3\sigma^2 - \omega^2 + 2a_1\sigma + a_2 = 0 \tag{18}$$

и УП

$$a_3 = -\sigma^3 + 3\sigma\omega^2 - a_1\sigma^2 + a_1\omega^2 - a_2\sigma. \tag{19}$$

Границу области качества Q_2 (линию b_2) для семейства полиномов третьей степени определим уравнением

$$\sigma_{b2} = -25. \tag{20}$$

Искомые границы интервалов a_3 вычисляются на основании УКГ (18) и УП (19) после подстановки в них соответствующих значений координат σ и ω точек пересечения ветвями годографов границы b_2. Поскольку функция параметра $a_3(\omega)$ (19) является непрерывной дифференцируемой функцией, то на границе b_2 в общем случае будет иметь место чередование участков возрастания и убывания этой функции. Следовательно, для вычисления границ интервала a_3 вначале определим минимальное и максимальное значения координат ω в области пересечения семейством годографов (10.*c*) границы b_2, т. е. координат $\omega_{\min}$ и $\omega_{\max}$ соответственно по формулам

$$\omega_{\min} = \pm\sqrt{3\sigma^2 + 2a_1\sigma + \underline{a}_2}; \tag{21}$$

$$\omega_{\max} = \pm\sqrt{3\sigma^2 + 2a_1\sigma + \overline{a}_2}. \tag{22}$$

На основе полученных значений $\omega_{\min}$ и $\omega_{\max}$ вычислим значения $a_{3\min}$ и $a_{3\max}$ параметра a_3 соответственно в точках (21) и (22) на границе b_2:

$$a_{3\min} = -\sigma^3 + 3\sigma\omega_{\min}^2 - a_1\sigma^2 + a_1\omega_{\min}^2 - \underline{a}_2\sigma; \tag{23}$$

$$a_{3\max} = -\sigma^3 + 3\sigma\omega_{\max}^2 - a_1\sigma^2 + a_1\omega_{\max}^2 - \overline{a}_2\sigma, \tag{24}$$

тогда $a_{3\min} = 75406{,}1$; $a_{3\max} = 247351{,}1$.

Исходя из рассмотренных в [3] особенностей динамики корневых портретов систем третьего порядка установлено, что ветви годографов подобных портретов мигрируют в правую полуплоскость только в двух точках: через границу b_2 и точку t_3 (см. рис. 2) ее пересечения с действительной осью σ. Значения a_3 в верхней и нижней точках границы области миграции корней портрета через границу b_2 определены выражениями (23) и (24). Очевидно, что в данном случае верхняя граница интервала параметра a_3 определяется выражением $\overline{a}_3 = \min(a_{3\min}, a_{3\max}) = 75406{,}1$.

Аналогичным образом, используя формулы (23) и (24), вычисляются минимальное и максимальное значения a_3, $a'_{3\min}$ и $a'_{3\max}$ при миграции корней портрета через точку t_4 (см. рис. 2). Очевидно, что на основе этих значений определяется нижняя граница интервала параметра a_3, $\underline{a}_3 = \max(a'_{3\min}, a'_{3\max}) = 68960{,}6$. Тогда вычисленный на основании (10.*c*) интервал изменения параметра a_3 годографа имеет следующий вид: $a_3 \in [\underline{a}_3, \overline{a}_3] = [68960{,}6,\ 75406{,}1]$.

Исследование миграции корней через границу β_2 показало, что предельные значения параметра в точках пересечения ветвей портрета с этой границей в данном случае на предельные значения интервалов вариации параметра a_3 не влияют.

С целью определения интервала вариации a_4 рассмотрим полином (10.*d*) четвертой степени расширения (10). Подставив в (10.*d*) значение комплексного переменного $s = \sigma + i\omega$ и сделав соответствующие преобразования аналогично выполненным выше для $n = 2$ и $n = 3$, запишем соответственно УКГ и УП годографа полинома (10.*d*):

$$4\sigma^3\omega - 4\sigma\omega^3 + 3a_1\sigma^2\omega - a_1\omega^3 + 2a_2\sigma\omega + a_3\omega = v(\sigma,\omega) = 0; \tag{25}$$

$$\sigma^4 - 6\sigma^2\omega^2 + \omega^4 + a_1\sigma^3 - 3a_1\sigma\omega^2 + a_2\sigma^2 - a_2\omega^2 + a_3\sigma = u(\sigma,\omega) = -a_4. \tag{26}$$

На основе (25) и (26) запишем в комплексной плоскости УКГ

$$4\sigma^3 - 4\sigma\omega^2 + 3a_1\sigma^2 - a_1\omega^2 + 2a_2\sigma + a_3 = 0 \tag{27}$$

и УП

$$a_4 = -\sigma^4 + 6\sigma^2\omega^2 - \omega^4 - a_1\sigma^3 + 3a_1\sigma\omega^2 - a_2\sigma^2 + a_2\omega^2 - a_3\sigma. \tag{28}$$

Граница области качества Q (линия β_1 на рис. 2) для семейства полиномов четвертой степени определена выражением (11). Искомые границы интервалов a_4 вычисляются на основании УКГ (27) и УП (28) после подстановки в них соответствующих значений координат σ и ω точек пересечения ветвями годографов границы β_1. Поскольку функция параметра $a_4(\omega)$ (28) является непрерывной дифференцируемой функцией, то на границе β_1 в общем случае будет иметь место чередование участков возрастания и убывания этой функции. Следовательно, для вычисления границ интервала a_4 вначале определим минимальное и максимальное значения координат ω в области пересечения семейством годографов (10.*c*) границы b_2, т. е. координат $\omega_{\min}$ и $\omega_{\max}$ соответственно по формулам

$$\omega_{\min} = \pm\sqrt{\frac{4\sigma^3 + 3a_1\sigma^2 + 2\bar{a}_2\sigma + \underline{a}_3}{4\sigma + a_1}}; \tag{29}$$

$$\omega_{\max} = \pm\sqrt{\frac{4\sigma^3 + 3a_1\sigma^2 + 2\underline{a}_2\sigma + \bar{a}_3}{4\sigma + a_1}}. \tag{30}$$

На основе полученных значений $\omega_{\min}$ и $\omega_{\max}$ вычисляются значения $a_{4\min}$ и $a_{4\max}$ параметра a_4 соответственно в точках t_5 (29) и t_6 (30) на границе β_1 (рис. 3):

$$a_{4\min} = -\sigma^4 + 6\sigma^2\omega_{\min}^2 - \omega_{\min}^4 - a_1\sigma^3 + 3a_1\sigma\omega_{\min}^2 - \bar{a}_2\sigma^2 + \bar{a}_2\omega_{\min}^2 - \underline{a}_3\sigma; \tag{31}$$

$$a_{4\max} = -\sigma^4 + 6\sigma^2\omega_{\max}^2 - \omega_{\max}^4 - a_1\sigma^3 + 3a_1\sigma\omega_{\max}^2 - \underline{a}_2\sigma^2 + \underline{a}_2\omega_{\max}^2 - \bar{a}_3\sigma. \tag{32}$$

Тогда $a_{4\min} = 558052{,}975$; $a_{4\max} = = 1077992{,}615$.

Корневой портрет, иллюстрирующий полученные результаты, показан на рис. 3. Он представлен полем корневых траекторий в виде линий уровня двух годографов. Ограничивающие поле линии пересекают границу β_1 в точках t_5 и t_6. На рис. 3 положительные ветви и асимптоты годографов отмечены знаком «+», отрицательные – знаком «–». Ветви первого годографа, одна из ветвей которого пересекает границу β_1 в точке t_5, обозначены цифрой 1; ветви второго годографа, одна из ветвей которого пересекает границу β_1 в точке t_6, обозначены цифрой 2. Область пересечений корневого портрета с границей β_1 представляет собой отрезок t_5t_6 на этой границе (показан утолщенной линией на рис. 3).

На основании рассмотренных в [3] особенностей динамики корневых портретов систем четвертого порядка установлено, что ветви годографов подобных портретов мигрируют в правую полуплоскость только в двух точках: через границу β_1 (точки t_5 и t_6) и точку t_7 ее пересечения с действительной осью σ (см. рис. 3). Значения a_4 в верхней и нижней точках границы области миграции корней портрета через границу β_1 находятся из выражений (31) и (32). Очевидно, что в данном случае из выражений (31) и (32) определяется верхняя граница интервала параметра a_4: $\bar{a}_3 = \min(a_{4\min},\ a_{4\max}) = 558053{,}0$.

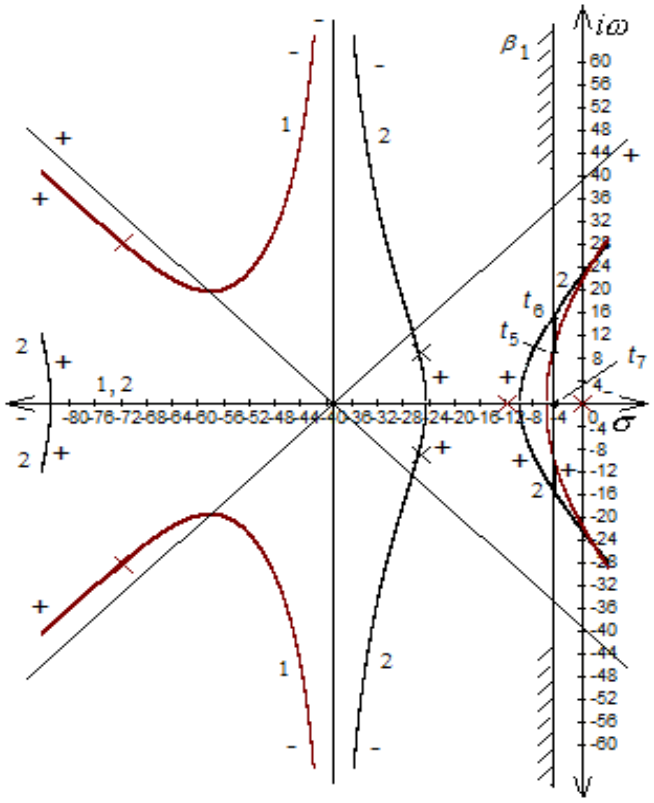


Рис. 3. Определение коэффициента a_4

Минимальное и максимальное значения a_4, $a'_{4\min}$ и $a'_{4\max}$ при миграции корней портрета через точку t_7 (см. рис. 3) вычисляются аналогично с помощью формул (31) и (32). Очевидно, что на основе этих значений определяется нижняя граница интервала изменения параметра a_4 по формуле $\underline{a}_4 = \max(a'_{4\min}, a'_{4\max}) =$ 231416,2. Тогда вычисленный на основании (10.*d*) интервал изменения параметра a_4 годографа имеет следующий вид: $a_4 \in [\underline{a}_4, \bar{a}_4]$=[231416,2, 558053,0].

Исследование миграции корней через границу β_2 показало, что предельные значения параметра a_4 в точках пересечения ветвей портрета с этой границей в данном случае не влияют на предельные значения интервалов вариации параметра a_4.

На рис. 3 показано доминирующее поле корневых траекторий для портрета системы четвертого порядка, используемое для определения параметра a_4. Поле представлено своими граничными годографами (линиями уровня поля). Очевидно, что миграция ветвей в комплексной плоскости в данном случае происходит в области пересечения границы β_1 годографами поля, расположенной между точками t_5 и t_6, а миграция действительных ветвей – через точку t_7, расположенную на оси σ, т. е. точку пересечения данной оси с границей области качества *Q*.

Заключение

Выполнено моделирование динамической системы четвертого порядка с учетом параметрической неопределенности интервального характера в применении к системе автоматического управления электропривода на основе математических моделей в форме корневого портрета и расширенного корневого годографа системы. На основе установленных закономерностей динамики корневых портретов систем четвертого порядка и результатов моделирования определена последовательность (алгоритм) расчета граничных значений интервалов параметров подобных систем, обеспечивающих размещение семейства корней в заданной области плоскости собственных частот, а именно робастную устойчивость и робастное качество систем. Проведенное исследование также демонстрирует характер изменения динамики системы в ответ на параметрические вариации.

Список литературы

1. Дорф, Р. Современные системы управления / Р. Дорф, Р. Бишоп. – М. : Лаборатория базовых знаний, 2004. – 632 с.

2. Кузовков, Н.Т. Модальное управление и наблюдающие устройства / Н.Т. Кузовков. – М. : Машиностроение, 1976. – 125 с.

3. Несенчук, А.А. Анализ и синтез робастных динамических систем на основе корневого подхода / А.А. Несенчук. – Минск : ОИПИ НАН Беларуси, 2005. – 234 с.

4. Несенчук, А.А. Корневой метод синтеза устойчивых полиномов путем настройки всех коэффициентов / А.А. Несенчук // Автоматика и телемеханика. – 2010. – № 8. – С. 13–24.

5. Barmish, B.R. The robust root locus / B.R. Barmish, R. Tempo // Automatica. – 1993. – Vol. 26. – P. 183–192.

6. Поляк, Б.Т. Робастная устойчивость и управление / Б.Т. Поляк, П.С. Щербаков. – М. : Наука, 2002. – 303 с.

7. Опейко, О.Ф. Синтез регулятора тока системы векторного управления электродвигателем / О.Ф. Опейко // Вісник КДУ ім. Михайла Остроградського. – 2014. – Вип. 1. – С. 9–14.

Поступила 19.05.2016

Объединенный институт проблем информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: anes@newman.bas-net.by

A.A. Nesenchuk

SIMULATION OF THE ELECTRIC DRIVE DYNAMICS ON THE BASIS OF ROOT LOCUS MODEL

The paper represents the task of simulation of the control system dynamics in the rotor flux oriented vector control system of the induction motor with application of the mathematical model in the form of the extended root locus. An algorithm for control system parameters calculation has been proposed, ensuring stability and required quality when operating in conditions of the plant parametric uncertainty.

УДК 519.7

Ю.В. Поттосин

ЭВРИСТИЧЕСКИЙ МЕТОД ЭНЕРГОСБЕРЕГАЮЩЕГО ПРОТИВОГОНОЧНОГО КОДИРОВАНИЯ СОСТОЯНИЙ АСИНХРОННОГО АВТОМАТА

Рассматривается задача кодирования состояний дискретного автомата с минимизацией переключательной активности элементов памяти в реализующей логической схеме. Предлагается эвристический метод решения этой задачи для асинхронного автомата, где при кодировании состояний устраняются опасные состязания между элементами памяти реализующей схемы.

Введение

Одним из важных критериев оптимизации при проектировании дискретных устройств является величина потребляемой энергии. Это обусловлено, с одной стороны, стремлением увеличить время действия источника энергии в портативных приборах, а с другой – снизить остроту проблемы отвода тепла при проектировании сверхбольших интегральных схем.

Как отмечено в работах [1, 2], потребляемая мощность схемы, построенной на основе КМОП-технологии, пропорциональна интенсивности переключений логических элементов и элементов памяти. В частности, снижения энергопотребления можно добиваться на этапе кодирования состояний автомата, т. е. когда абстрактным символам состояний приписываются булевы векторы. Очевидно, кодировать состояния при этом надо таким образом, чтобы при переходе автомата из одного состояния в другое меняли свое состояние как можно меньше элементов памяти. Этой задаче посвящены, например, работы [3–5], где она решалась для синхронной реализации автомата. Асинхронным автоматам давно уделяется большое внимание [6–8]. Задача энергосбережения для асинхронных автоматов также может быть частично сведена к уменьшению переключательной активности элементов схемы. Энергосберегающее противогоночное кодирование состояний асинхронного автомата рассматривалось в статье [9], где эта задача сводилась к нахождению минимального взвешенного покрытия, и тем самым обеспечивалось получение кода минимальной длины. Поскольку задача нахождения минимального покрытия имеет неполиномиальную сложность, данный метод не всегда позволяет получить решение за практически приемлемое время. Представляемый здесь эвристический метод решения данной задачи не гарантирует получения минимума длины кода состояния и минимума интенсивности переключений элементов памяти, но в ряде случаев позволяет получать решение, близкое к оптимальному по данным критериям, за приемлемое время. Случаев получения наихудших решений по указанным критериям на практических примерах применения предлагаемого метода не выявлено.

1. Модель поведения асинхронного автомата

Моделью поведения логической схемы с памятью является конечный автомат, представляющий собой пятерку (A, B, Q, Ψ, Φ), где A, B и Q – соответственно множества входных сигналов, выходных сигналов и состояний автомата, а Ψ и Φ – функции $\Psi : A \times Q \to Q$ и $\Phi : A \times Q \to B$, называемые соответственно *функцией переходов* и *функцией выходов*. Для состояний $q_i, q_j \in Q$ и входного сигнала $a \in A$ состояние $q_j = \Psi(a, q_i)$ является тем состоянием, в которое автомат переходит из состояния q_i под воздействием входного сигнала a. Конечный автомат функционирует в дискретном времени, т. е. время разбивается на конечные промежутки, называемые *тактами*, в течение каждого из которых автомат переходит из одного состояния в другое и выдает соответствующий выходной сигнал. Рассматриваемая задача позволяет игнорировать функцию выходов Ф, поэтому в дальнейшем она не будет упоминаться.

Здесь рассматривается асинхронная реализация конечного автомата, называемая *асинхронным автоматом*, которая в отличие от синхронной реализации не имеет внешнего источника тактирующих сигналов. Переход от такта к такту происходит в момент изменения входного сигнала. При действии любого входного сигнала асинхронный автомат приходит в некоторое устойчивое состояние, из которого он не выходит до конца действия данного сигнала. При этом должно выполняться требование прямого перехода, которое формально выражается следующим образом: если $\Psi(a, q_i) = q_j$ для фиксированного входного сигнала a и некоторых состояний q_i и q_j, то $\Psi(a, q_j) = q_j$.

Задача кодирования состояний автомата заключается в присвоении каждому состоянию определенного булева вектора $(z_1, z_2, \ldots, z_k)$, называемого *кодом состояния*, который соответствует набору состояний двоичных элементов памяти (триггеров) в логической схеме, где каждый переход из состояния в состояние представляется переключением одного или нескольких триггеров. Естественно, что в реальной электронной схеме такое переключение не может происходить мгновенно и одновременно. Явление неодновременного переключения элементов памяти называется *состязаниями* или *гонками* элементов памяти [10]. Принято называть состязания *неопасными*, если все промежуточные состояния, в которых автомат может оказаться при переходе из одного состояния в другое под воздействием некоторого входного сигнала a, являются неустойчивыми для сигнала a, т. е. при любом порядке переключений элементов памяти автомат из некоторого состояния q_i под воздействием входного сигнала a переходит всегда в состояние $q_j = \Psi(a, q_i)$. Если же при этом автомат может оказаться в некотором устойчивом состоянии q_k, отличном от q_j, то состязания называются *опасными*.

Кодирование состояний, обеспечивающее отсутствие опасных состязаний (гонок), называется *противогоночным*. Естественно, здесь возникает задача минимизации длины кода состояния, приводящая к наименьшему числу элементов памяти в реальной схеме.

Другим критерием оптимизации схемы, как указано выше, является величина потребляемой энергии.

2. Условия отсутствия опасных состязаний

Прежде всего в решении задачи противогоночного кодирования состояний асинхронного автомата необходимо установить условия отсутствия опасных состязаний в реализуемой схеме. Существование опасных состязаний для пары переходов $q_i \to q_j$, $q_k \to q_l$ $(q_j \neq q_l)$ при одном и том же входном сигнале a может привести к тому, что автомат вместо перехода в состояние q_j из состояния q_i окажется в состоянии q_l, которое также является устойчивым при входном сигнале a. Условие отсутствия опасных состязаний для этой пары можно выразить троичным вектором, в котором компоненты соответствуют состояниям автомата и компоненты i и j имеют одно значение, 0 или 1, а компоненты k и l – противоположное ему значение [11]. Остальным компонентам приписывается значение «–». В схеме, реализующей заданный автомат, это условие выполняется триггером, который в процессе одного из переходов рассматриваемой пары хранит состояние 0, а в процессе другого перехода – состояние 1.

Пусть, например, табл. 1 представляет функцию переходов $\Psi(a, q)$ заданного автомата, т. е. является его таблицей переходов. Строкам ее соответствуют состояния автомата, а столбцам – входные сигналы. В клетках таблицы показаны значения функции $\Psi(a, q)$ при соответствующих значениях аргументов a и q. Устойчивые состояния для каждого входного сигнала выделены.

Условие отсутствия опасных состязаний для пары переходов $q_3 \to q_1$, $q_4 \to q_2$ при входном сигнале a_1 выражается вектором (0 1 0 1 – –) либо покомпонентной инверсией этого вектора (1 0 1 0 – –).

На множестве векторов, представляющих условия отсутствия опасных состязаний, устанавливается отношение импликации: троичный вектор **a** имплицирует троичный вектор **b**, если **b** получается из **a** заменой некоторых нулей или единиц значением «–» и, возможно, инвертированием полученного результата. Например, вектор (1 0 – – 1 0 1) имплицирует вектор (1 0 – – – 0 1), а также вектор (0 1 – – – 1 –). Смысл этого отношения в том, что условие, представленное вектором **b**, автоматически выполняется при соблюдении условия, представленного вектором **a**.

Таблица 1

Переходы автомата

	a_1	a_2	a_3	a_4
q_1	$\boldsymbol{q}_1$	q_5	$\boldsymbol{q}_1$	q_5
q_2	$\boldsymbol{q}_2$	q_6	$\boldsymbol{q}_2$	$\boldsymbol{q}_2$
q_3	q_1	$\boldsymbol{q}_3$	q_4	$\boldsymbol{q}_3$
q_4	q_2	q_5	$\boldsymbol{q}_4$	$\boldsymbol{q}_4$
q_5	q_2	$\boldsymbol{q}_5$	q_1	$\boldsymbol{q}_5$
q_6	q_7	$\boldsymbol{q}_6$	q_2	q_3
q_7	$\boldsymbol{q}_7$	q_5	$\boldsymbol{q}_7$	q_3

Все условия отсутствия опасных состязаний в виде описанных векторов составляют троичную матрицу, в которой отсутствуют имплицируемые строки. Эта матрица называется *матрицей условий* [11]. Для автомата, поведение которого описывает табл. 1, матрица условий имеет следующий вид:

$$\boldsymbol{C} = \begin{array}{c} \begin{matrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & – & – & – \\ 0 & 1 & 0 & – & 1 & – & – \\ 0 & – & 0 & – & – & 1 & 1 \\ – & 0 & – & 0 & – & 1 & 1 \\ – & 0 & – & – & 0 & 1 & 1 \\ 0 & 1 & – & – & 0 & 1 & – \\ – & 0 & – & 1 & 1 & 0 & – \\ – & 0 & – & – & 1 & 0 & 1 \\ – & – & 0 & 1 & 1 & – & – \\ – & – & 0 & – & 1 & – & 1 \\ 0 & – & 1 & 1 & 0 & – & – \\ – & 0 & 1 & 1 & – & 0 & – \\ – & – & 0 & 0 & – & – & 1 \\ 0 & – & 1 & – & 0 & 1 & – \\ 0 & – & 1 & – & 0 & – & 1 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix} .$$

Задача противогоночного кодирования состояний асинхронного автомата с минимизацией числа внутренних переменных $z_1, z_2, \ldots, z_k$ сводится к получению минимального числа векторов, таких, что для каждой строки матрицы $\boldsymbol{C}$ среди полученных векторов найдется вектор, имплицирующий данную строку [11]. Результатом кодирования состояний автомата является *матрица кодирования*. Ее строкам соответствуют состояния автомата, а столбцам – внутренние переменные, строки этой матрицы представляют коды соответствующих состояний, а столбцы – указанные выше векторы.

3. Вычисление весов строк матрицы условий

Для снижения интенсивности переключений элементов памяти можно воспользоваться следующими соображениями. Каждому i-му столбцу матрицы кодирования можно поставить в соответствие множество переходов. Данное множество составляют те переходы, которыми связаны состояния автомата с кодами, где переменная z_i имеет различные значения. При таких переходах i-й триггер в реальной схеме, реализующей заданный автомат, меняет свое состояние.

Если удается вычислить вероятности переходов между состояниями, то i-му столбцу матрицы кодирования состояний ставится в соответствие в виде веса w_i вероятность события, заключающегося в том, что происходит некоторый переход, при котором меняется значение перемен-

ной z_i. Поскольку переходы между состояниями автомата являются несовместимыми событиями, эта вероятность равна сумме вероятностей отдельных переходов из множества, соответствующего *i*-му столбцу. Для подсчета вероятностей переходов между состояниями в статье [5] используется метод Чэпмена – Колмогорова, где такие вероятности получаются в результате решения системы линейных уравнений с этими вероятностями в качестве неизвестных. Однако данный метод можно применять только тогда, когда автомат является полностью определенным, а его граф поведения представляет собой сильно связный ориентированный граф. В противном случае столбцу матрицы кодирования состояний автомата можно, например, приписывать мощность связанного с ним множества переходов.

Каждый столбец матрицы кодирования рассматривается как вектор, имплицирующий некоторые строки матрицы условий ***C***. Таким образом, каждой строке матрицы ***C*** можно приписать вес в виде суммы вероятностей переходов между состояниями, которым соответствуют компоненты с различными значениями, отличными от «–». Для упрощения вычислений вместо таких сумм можно использовать пропорциональные им величины, например числители при общем знаменателе вероятностей.

Искомое решение рассматриваемой задачи представляется матрицей кодирования с минимальной суммой весов ее столбцов.

Вероятность перехода автомата в состояние q_j, вызываемого входным сигналом a, когда он находится в состоянии q_i, равна вероятности прихода входного сигнала a. Если имеется несколько входных сигналов, переводящих автомат из состояния q_i в состояние q_j, условная вероятность p'_{ij} такого перехода равна сумме вероятностей этих сигналов, поскольку поступления на вход автомата различных входных сигналов – несовместимые события. Условием является то, что автомат находится в состоянии q_i. Пребывание автомата в состоянии q_i и приход сигнала, переводящего его в состояние q_j, – независимые события. Поэтому абсолютная вероятность p_{ij} перехода из состояния q_i в состояние q_j в течение всего времени работы автомата равна $P_i p'_{ij}$, где P_i – вероятность того, что автомат находится в состоянии q_i.

Вероятности P_i ($i = 1, 2, \ldots, |Q|$) находятся путем решения системы уравнений Чэпмена – Колмогорова, которая имеет следующий вид:

$$\sum_{i=1}^{|Q|} P_i p'_{ij} = P_j, \quad j = 1, 2, \ldots, |Q|,$$

$$\sum_{i=1}^{|Q|} P_i = 1.$$

Вероятности p'_{ij} должны быть известны. Таким образом, решив эту систему уравнений, получим вероятности P_i. Как было сказано раньше, абсолютная вероятность p_{ij} определяется как $p_{ij} = P_i p'_{ij}$.

Асинхронный автомат, поведение которого описывает табл. 1, является полностью определенным, а граф его поведения представляет собой сильно связный ориентированный граф. Следовательно, вероятности переходов между состояниями можно определять по методу Чэпмена – Колмогорова. Допустим, что вероятности входных сигналов данного автомата имеют равномерное распределение. Тогда условная вероятность p'_{ij} перехода (перехода в состояние q_j, когда автомат находится в состоянии q_i) представлена в табл. 2, где строки и столбцы соответствуют состояниям автомата и на пересечении строки q_i и столбца q_j расположена вероятность p'_{ij}. Пустые клетки означают отсутствие перехода между соответствующими состояниями (вероятность такого перехода равна нулю).

Для нахождения вероятностей состояний (вероятностей попадания автомата в те или иные состояния) надо решить следующую систему линейных уравнений (для упрощения вычислений используем величины, пропорциональные условным вероятностям):

$$2P_1 + P_3 + P_5 = 4P_1;$$

$$3P_2 + P_4 + P_5 + P_6 = 4P_2;$$

$$2P_3 + P_6 + P_7 = 4P_3;$$

$$P_3 + 2\,P_4 = 4\,P_4;$$

$$2\,P_1 + P_4 + 2\,P_5 + P_7 = 4\,P_5;$$

$$P_2 + P_6 = 4\,P_6;$$

$$P_6 + 2\,P_7 = 4\,P_7;$$

$$P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 = 1.$$

В результате решения этой системы уравнений получаем $P_1 = 19/135$, $P_2 = 48/135$, $P_3 = 12/135$, $P_4 = 6/135$, $P_5 = 26/135$, $P_6 = 16/135$, $P_7 = 8/135$. Абсолютные вероятности переходов представлены в табл. 3, где строки и столбцы соответствуют состояниям автомата и на пересечении строки q_i и столбца q_j расположена вероятность p_{ij}. Пустые клетки так же, как в табл. 2, соответствуют нулевым вероятностям.

Таблица 2

Условные вероятности переходов

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_1	1/2				1/2		
q_2		3/4				1/4	
q_3	1/4		1/2	1/4			
q_4		1/4		1/2	1/4		
q_5	1/4	1/4			1/2		
q_6		1/4	1/4			1/4	1/4
q_7			1/4		1/4		1/2

Таблица 3

Абсолютные вероятности переходов

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
q_1	19/270				19/270		
q_2		72/270				24/270	
q_3	6/270		12/270	6/270			
q_4		3/270		6/270	3/270		
q_5	13/270	13/270			26/270		
q_6		8/270	8/270			8/270	8/270
q_7			4/270		4/270		8/270

В табл. 4 представлены веса строк матрицы условий $\boldsymbol{C}$ для рассматриваемого примера, где верхняя строка содержит номера строк, а нижняя – веса соответствующих строк. В качестве веса строки взята величина, пропорциональная вероятности любого из переходов, связывающих состояния, которым соответствуют элементы строки с несовпадающими значениями, отличными от «–». В данном случае взяты числители при общем знаменателе вероятностей.

Таблица 4

Веса строк матрицы условий

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	32	12	40	36	13	20	21	6	4	9	11	4	6	10

4. Получение матрицы кодирования состояний

После построения матрицы условий $\boldsymbol{C}$ требуется получить по возможности минимальное количество совместимых множеств строк этой матрицы, покрывающих все ее строки. Множество строк матрицы $\boldsymbol{C}$ называется совместимым, если существует вектор, имплицирующий все

строки из этого множества [11]. Естественно, две строки матрицы ***C*** следует назвать несовместимыми, если не существует вектора, имплицирующего обе эти строки. Матрица совместимости ***T***, где строки и столбцы соответствуют строкам матрицы ***C*** и элемент на пересечении *i*-й строки и *j*-го столбца имеет значение 0, если *i*-я и *j*-я строки матрицы ***C*** несовместимы, и значение 1 – в противном случае, имеет следующий вид:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix}.$$

Процесс получения искомого семейства совместимых множеств состоит из двух этапов. На первом этапе предлагаемого процесса находится как можно большее множество попарно несовместимых строк матрицы ***C***. При этом достаточно использовать «жадный» алгоритм, выбирающий для внесения в данное множество каждый раз строку, имеющую наибольшее число несовместимых строк. Если таких строк несколько, выбираем ту, которая обладает минимальным весом. Назовем *степенью совместимости* строки матрицы условий ***C*** число строк, совместимых с этой строкой. В табл. 5 представлены степени совместимости строк матрицы ***C***, которые легко определяются по матрице ***T***.

Таблица 5

Степени совместимости строк матрицы условий

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

Из табл. 5 видно, что самой низкой степенью совместимости обладают строки 1, 2 и 15. Из них выбираем строку 1, обладающую наименьшим весом. После удаления из рассмотрения строки 1 и совместимых с ней строк получаем матрицу совместимости оставшихся строк:

$$\boldsymbol{T} = \begin{array}{c} \begin{matrix} 7 & 11 & 12 & 13 & 14 & 15 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} 7 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{matrix},$$

из которой видно, что все оставшиеся строки обладают одной и той же степенью совместимости, равной 4. Из них выбираем строку 15 с наименьшим весом. Строка 13 осталась единственной строкой, не совместимой ни со строкой 1, ни со строкой 15. Упомянутые строки образуют три одноэлементных множества {1}, {13} и {15}, на основе которых строится искомая совокупность совместимых множеств.

Текущая ситуация в процессе, выполняемом на втором этапе, характеризуется совокупностью совместимых множеств $C_1, C_2, \ldots, C_k$. Каждому множеству C_i ($i = 1, 2, \ldots, k$) соответствуют:

– вектор $\boldsymbol{c}_i$, имплицирующий все строки из C_i, с весом w_i, определяемым, как описано в разд. 3;

– множество строк B_i матрицы условий, совместимых со всеми строками из C_i и не принадлежащих ни одному из множеств $C_1, C_2, \ldots, C_k$;

– множество строк D_i матрицы условий, каждая из которых не совместима хотя бы с одной строкой из C_i и не принадлежит ни одному из множеств $C_1, C_2, \ldots, C_k$.

В начале второго этапа все совместимые множества являются одноэлементными, состоящими из строк матрицы $\boldsymbol{C}$, выделенных на первом этапе. Очередной шаг процесса представляет собой выбор пары $(C_i, \mathbf{b}_i)$, где $\mathbf{b}_i \in B_i$, включение вектора $\mathbf{b}_i$ в множество C_i и коррекцию вектора $\boldsymbol{c}_i$, связанную с данным включением. После этого $\mathbf{b}_i$ удаляется из всех $B_1, B_2, \ldots, B_k$ и $D_1, D_2, \ldots, D_k$. Из тех же множеств удаляются все строки, которые оказались имплицируемыми вектором $\boldsymbol{c}_i$. Они также вносятся в C_i. Если $B_i = \varnothing$, то C_i вносится в решение и исключается из дальнейшего рассмотрения. Этот шаг повторяется, и процесс заканчивается, когда все множества $B_1, B_2, \ldots, B_k$ окажутся пустыми.

При коррекции вектора $\boldsymbol{c}_i$ в множество D_i могут переноситься строки из множества B_i, оказавшиеся не имплицируемыми вектором $\boldsymbol{c}_i$. Если при этом некоторая строка будет присутствовать во всех множествах $D_1, D_2, \ldots, D_k$, то из нее образуется одноэлементное совместимое множество C_{k+1} и соответственно множества B_{k+1} и D_{k+1}, т. е. k увеличивается на единицу.

Выбор пары $(C_i, \mathbf{b}_i)$ осуществляется по следующим правилам:

1. Выбирается вариант с наименьшим числом строк, вносимых в D_i.

2. Выбирается вариант с наименьшим увеличением веса w_i вектора $\boldsymbol{c}_i$.

При этом второе правило применяется только тогда, когда согласно первому правилу имеется несколько вариантов выбора.

Как отмечено в работе [6], число k, которое становится длиной кода состояния, растет пропорционально величине $\log_2|Q|$. Выбор пары $(C_i, \mathbf{b}_i)$ происходит за время, пропорциональное произведению $k\,l$, где l – число строк матрицы условий, не вошедших в множества $C_1, C_2, \ldots, C_k$. С каждым шагом число l уменьшается, по крайней мере, на единицу. Если m – число строк и n – число столбцов матрицы условий, т. е. $l < m$ и $n = |Q|$, то, как нетрудно подсчитать, время выполнения второго этапа ограничено величиной, пропорциональной произведению $m^2 \log_2 n$.

В рассматриваемом примере начальная ситуация представляется следующими множествами и векторами:

$C_1 = \{1\};$	$\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$;	$B_1 = \{2,3,4,5,6,8,9,10\};$	$D_1 = \{7,11,12,14\};$	$w_1 = 6;$
$C_2 = \{13\};$	$\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$;	$B_2 = \{2,3,4,5,6,7,8,10,11,12,14\};$	$D_2 = \{9\};$	$w_2 = 4;$
$C_3 = \{15\};$	$\boldsymbol{c}_3 = (0 - 1 - 0 - 1)$;	$B_3 = \{4,5,6,7,9,11,12,14\};$	$D_3 = \{2,3,8,10\};$	$w_3 = 10.$

По первому правилу выбирается пара $(C_3, 14)$. Этот вариант является единственным, когда из B_i в D_i переносится всего одна строка. Вес вектора $\boldsymbol{c}_3$ при этом не меняется. В результате соответствующего шага ситуация представится следующим образом:

$C_1 = \{1\};$	$\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$;	$B_1 = \{2,3,4,5,6,8,9,10\};$	$D_1 = \{7,11,12\};$	$w_1 = 6;$
$C_2 = \{13\};$	$\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$;	$B_2 = \{2,3,4,5,6,7,8,10,11,12\};$	$D_2 = \{9\};$	$w_2 = 4;$
$C_3 = \{14,15\};$	$\boldsymbol{c}_3 = (0 - 1 - 0\ 1\ 1)$;	$B_3 = \{4,5,6,7,9,11\};$	$D_3 = \{2,3,8,10,12\};$	$w_3 = 10.$

На следующем шаге тоже удается воспользоваться только первым правилом, поэтому приходим к ситуации

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8,10\}$; $D_1 = \{7,11,12\}$; $w_1 = 6$;
$C_2 = \{13\}$; $\boldsymbol{c}_2 = (- - 0\ 0 - - 1)$; $B_2 = \{2,3,4,5,6,7,8,10,11,12\}$; $D_2 = \varnothing$; $w_2 = 4$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,10,11,12\}$; $w_3 = 16$.

Далее по первому правилу выбираются три варианта $(C_1, 2)$, $(C_2, 3)$ и $(C_2, 10)$, когда должны быть перенесены из множества совместимых в множество несовместимых строк две строки. Последний вариант обладает минимальным приращением веса имплицирующего вектора. Применяется второе правило, и очередная ситуация выглядит так:

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8\}$; $D_1 = \{7,11,12\}$; $w_1 = 6$;
$C_2 = \{10,13\}$; $\boldsymbol{c}_2 = (- - 0\ 0\ 1 - 1)$; $B_2 = \{2,3,4,6,8,11,12\}$; $D_2 = \{5,7\}$; $w_2 = 7$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,11,12\}$; $w_3 = 16$.

На следующем шаге после применения первых двух правил приходим к ситуации вида

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5,6,8\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_2 = \{10,11,13\}$; $\boldsymbol{c}_2 = (0 - 1\ 1\ 0 - 0)$; $B_2 = \{4,6,8,12\}$; $D_2 = \{2,3,5,7\}$; $w_2 = 7$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,6,7\}$; $D_3 = \{2,3,8,12\}$; $w_3 = 16$.

Далее, применяя первые два правила, выбираем вариант $(C_2, 6)$, при котором получается вектор $\boldsymbol{c}_2$, имплицирующий, кроме строк, присутствующих в C_2, и строки 6, еще строку 8. Рассматриваем следующую ситуацию:

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_2 = \{6,8,10,11,13\}$; $\boldsymbol{c}_2 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$; $B_2 = \varnothing$; $D_2 = \{2,3,4,5,7,12\}$; $w_2 = 34$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,7\}$; $D_3 = \{2,3,12\}$; $w_3 = 16$.

Данная ситуация отличается от предыдущих ситуаций тем, что $B_2 = \varnothing$ (вектор $\boldsymbol{c}_2$ не имплицирует ни одну из строк, не вошедших в множества C_1, C_2, C_3), а строка 12 вошла во все множества D_1, D_2, D_3. Поэтому множество C_2 вносим в решение, исключаем его из дальнейшего рассмотрения и вводим $C_4 = \{12\}$ вместе с соответствующими B_4 и D_4. Текущая ситуация примет тогда вид

$C_1 = \{1\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1 - - -)$; $B_1 = \{2,3,4,5\}$; $D_1 = \{7,12\}$; $w_1 = 6$;
$C_3 = \{9,14,15\}$; $\boldsymbol{c}_3 = (0 - 1\ 0\ 0\ 1\ 1)$; $B_3 = \{4,5,7\}$; $D_3 = \{2,3,12\}$; $w_3 = 16$;
$C_4 = \{12\}$; $\boldsymbol{c}_4 = (- 0\ 1\ 1 - 0 -)$; $B_4 = \{2,3,7\}$; $D_4 = \{4,5\}$; $w_2 = 11$.

Применяя первые два правила, получаем приращения множеств в следующем порядке: $C_4 = \{3,12\}$, $C_1 = \{1,2\}$, $C_4 = \{3,7,12\}$, $C_3 = \{4,5,9,14,15\}$. Результатом описанного процесса являются следующие множества и соответствующие им векторы с весами:

$C_1 = \{1,2\}$; $\boldsymbol{c}_1 = (0\ 1\ 0\ 1\ 1 - -)$; $w_1 = 38$;
$C_2 = \{6,8,10,11,13\}$; $\boldsymbol{c}_2 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$; $w_2 = 34$;
$C_3 = \{4,5,9,14,15\}$; $\boldsymbol{c}_3 = (0\ 0\ 1\ 0\ 0\ 1\ 1)$; $w_3 = 48$;
$C_4 = \{3,7,12\}$; $\boldsymbol{c}_4 = (0\ 1\ 0\ 0\ 0\ 1\ 1)$; $w_2 = 32$.

Из четырех вариантов замены символов «–» в векторе $\boldsymbol{c}_1$ выбираем комбинацию 1 1, дающую минимум веса w_1. Окончательным решением рассматриваемого примера является матрица кодирования с суммарным весом 164:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

В результате применения описанного в работе [11] точного метода противогоночного кодирования состояний с минимизацией длины кода без учета интенсивности переключений элементов памяти могут быть получены следующие матрицы кодирования:

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array},$$

$$\begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array},\quad \begin{array}{c} \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{array}\begin{array}{c} \begin{array}{cccc} z_1 & z_2 & z_3 & z_4 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

Первая из этих шести матриц совпадает с точностью до перестановки столбцов с матрицей, полученной с помощью описанного эвристического метода. Остальные пять матриц имеют соответственно веса 154, 178, 156, 196 и 171. Вес матрицы кодирования можно рассматривать как оценку качества решения. Чем меньше вес, тем лучше решение. В работе [5] применена оценка качества решения рассматриваемой задачи по критерию, который выражается формулой $\sum w_{ij}(d_{ij} - 1)$. Здесь d_{ij} – расстояние по Хэммингу между кодами состояний q_i и q_j; w_{ij} – вес перехода между состояниями q_i и q_j, суммирование ведется по всем парам состояний. Чем меньше эта оценка, тем лучше результат. Для приведенных матриц она равна соответственно 39, 35, 91, 37, 120 и 43. Обе эти оценки не противоречат друг другу, и отсюда следует, что описанный метод дает не худший результат.

Для оценки времени выполнения метода можно использовать следующие соображения. В работе [6] в качестве верхней границы длины кода состояния при отсутствии опасных состязаний приведена величина $2\log_2\lceil |Q| \rceil$, где $|Q|$ – число состояний автомата, а $\lceil a \rceil$ – ближайшее сверху целое число, не меньшее a. Эту величину можно принять за верхнюю границу числа множеств $C_1, C_2, \ldots, C_k$. Число вариантов, перебираемых на каждом шаге, меньше чем

$m \times 2 \log_2\lceil |Q| \rceil$, где m – число строк матрицы условий, уменьшающееся на каждом шаге не менее чем на единицу. Таким образом, по грубой оценке, время выполнения метода не достигает величины, пропорциональной $m^2 \times 2 \log_2\lceil |Q| \rceil$. Время выполнения точного метода растет экспоненциально относительно роста числа m. Такой трудностью обладает задача о покрытии, решение которой используется в рассматриваемом методе.

Заключение

Предлагаемый эвристический метод энергосберегающего противогоночного кодирования состояний асинхронного автомата рассчитан на использование его в автоматизированной системе логического проектирования. На примере показано, что описанный метод может дать решение, сравнимое по качеству с решением, получаемым точным методом. При этом время, затрачиваемое на получение решения, значительно меньше времени, за которое выполняется точный метод, поскольку точный метод сводит данную задачу к задаче о покрытии, имеющей неполиномиальную сложность.

Список литературы

1. Мурога, С. Системное проектирование сверхбольших интегральных схем : в 2 кн. Кн. 1 / С. Мурога. – М. : Мир, 1985. – 288 с.

2. Pedram, M. Power minimization in IC design: Principles and applications / M. Pedram // ACM Trans. Design Automat. Electron. Syst. – 1996. – Vol. 1. – P. 3–56.

3. Kashirova, L. State assignment of finite state machine for decrease of power dissipation / L. Kashirova, A. Keevallik, M. Meshkov // Second Intern. Conf. Computer-Aided Design of Discrete Devices, CAD DD'97, Minsk, Republic of Belarus, November 12–14, 1997. – Minsk : National Academy of Sciences of Belarus, Institute of Engineering Cybernetics, 1997. – Vol. 1. – P. 60–67.

4. Sudnitson, A. Partition search for FSM low power synthesis / A. Sudnitson // Fourth Intern. Conf. Computer-Aided Design of Discrete Devices, CAD DD'2001, Minsk, November 14–16, 2001. – Minsk : National Academy of Sciences of Belarus, Institute of Engineering Cybernetics, 2001. – Vol. 1. – P. 44–49.

5. Закревский, А.Д. Алгоритмы энергосберегающего кодирования состояний автомата / А.Д. Закревский // Информатика. – 2011. – № 1(29). – С. 68–78.

6. Ангер, С. Асинхронные последовательностные схемы / С. Ангер. – М. : Наука, 1977. – 400 с.

7. Синтез асинхронных автоматов на ЭВМ ; под ред. А.Д. Закревского. – Минск : Наука и техника, 1975. – 184 с.

8. Автоматизированное проектирование цифровых устройств ; под ред. С.С. Бадулина. – М. : Радио и связь, 1981. – 240 с.

9. Поттосин, Ю.В. Энергосберегающее противогоночное кодирование состояний асинхронного автомата / Ю.В. Поттосин // Информатика. – 2015. – № 2(46). – С. 94–101.

10. Закревский, А.Д. Алгоритмы синтеза дискретных автоматов / А.Д. Закревский. – М. : Наука, 1971. – 512 с.

11. Закревский, А.Д. Логические основы проектирования дискретных устройств / А.Д. Закревский, Ю.В. Поттосин, Л.Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.

Поступила 13.06.2016

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: pott@newman.bas-net.by

Yu.V. Pottosin

A HEURISTIC METHOD FOR LOW POWER RACE-FREE

STATE-ASSIGNMENT OF AN ASYNGHRONOUS AUTOMATON

The problem of race free state-assignment of an asynchronous automaton minimizing the switching activity of memory elements in an implementing circuit is considered. A heuristic method to solve this problem for an asynchronous automaton is suggested where the critical races between memory elements of the implementing circuit are eliminated.

ПРАВИЛА ДЛЯ АВТОРОВ

1. Статьи принимаются в редакцию через электронную систему подачи по адресу **http://jinfo.bas-net.by** в формате файлов текстовых редакторов Microsoft Word 97 и Word 2000 для Windows. Основной текст статьи набирается с переносами шрифтом Times New Roman 11 пт, интервал между строками – одинарный, абзацный отступ 1 см, поля по 2,5 см со всех сторон.

2. Статья должна иметь индекс УДК (универсальная десятичная классификация).

3. Название статьи, фамилии всех авторов и аннотация должны быть переведены на английский язык. Для каждого из авторов приводится развернутое название учреждения с полным почтовым адресом, а также номер телефона и электронный адрес (e-mail) для связи с редакцией.

4. Формулы, иллюстрации, таблицы, встречающиеся в статье, должны быть пронумерованы в соответствии с порядком цитирования в тексте. Ссылки на рисунки и таблицы в тексте обязательны. Необходимо избегать повторения одних и тех же данных в таблицах, графиках и тексте статьи.

Рисунки должны быть выполнены с хорошим разрешением в масштабе, позволяющем четко различать надписи и обозначения. Подрисуночные подписи с расшифровкой всех позиций, представленных на рисунке, набираются шрифтом гарнитуры основного текста, размер символов 9 пт. Цветные иллюстрации печатаются только в том случае, когда это необходимо для понимания излагаемого материала.

5. Набор формул выполняется в формульных редакторах Microsoft Equation или Math Type и должен быть единообразным по применению шрифтов и знаков по всей статье.

Прямо (_) набираются: греческие и русские буквы; математические символы (sin, lg, ∞); символы химических элементов (C, Cl, $CHCl_3$); цифры (римские и арабские); векторы; индексы (верхние и нижние), являющиеся сокращениями слов.

Курсивом (~) набираются: латинские буквы – переменные, символы физических величин (в том числе и в индексе).

6. Сокращения в тексте статьи (за исключением единиц измерения) могут быть использованы только после упоминания полного термина. Единицы измерения физических величин следует приводить в Международной системе СИ.

7. Литература приводится автором общим списком в конце статьи. Ссылки на литературу в тексте идут по порядку и обозначаются цифрой в квадратных скобках. Ссылаться на неопубликованные работы не допускается. С примерами оформления библиографического описания в списке литературы можно ознакомиться в приложении 2 к *Инструкции по оформлению диссертации, автореферата и публикаций по теме диссертации* на сайте Высшей аттестационной комиссии Республики Беларусь **http://vak.org.by**.

8. Поступившие в редакцию статьи направляются на рецензирование специалистам. Основным критерием целесообразности публикации является новизна и информативность статьи. Если по рекомендациям рецензента статья возвращается автору на доработку, а переработанная рукопись вновь рассматривается редколлегией, датой поступления считается день получения редакцией ее окончательного варианта. Статьи не по профилю журнала возвращаются авторам после заключения редколлегии.

9. Статьи, направляемые на доработку, должны быть возвращены в исправленном виде с ответами на все вопросы.

10. Редакция журнала предоставляет возможность первоочередного опубликования статей, представленных лицами, которые осуществляют послевузовское обучение (аспирантура, докторантура, соискательство) в год завершения обучения.

11. Авторы несут ответственность за направление в редакцию статей, уже опубликованных ранее, или статей, принятых к публикации другими изданиями.

12. Редакция оставляет за собой право на редакционные изменения, не искажающие основное содержание статьи.

Журнал «Информатика» включен Высшей аттестационной комиссией Республики Беларусь в список научных изданий для опубликования результатов диссертационных исследований.

Индексы

00827

для индивидуальных подписчиков

008272

для предприятий и организаций

ИНФОРМАТИКА 3 (51) июль–сентябрь 2016