ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

Том 22 № 4(58) 2015

Основан в 1999 году
Выходит 6 раз в год

Адрес редакции: ЯрГУ, ул. Советская, 14, г. Ярославль, 150000, Россия
Website: http://mais.uniyar.ac.ru, e-mail: mais@uniyar.ac.ru; телефон (4852) 79-77-72

Научные статьи в журнал принимаются по электронной почте. Статьи должны содержать УДК, аннотации на русском и английском языках и сопровождаться набором текста в редакторе LaTeX. Плата с аспирантов за публикацию рукописей не взимается.

12+

СОДЕРЖАНИЕ

Моделирование и анализ информационных систем. Т. 22, №4. 2015

Свидетельство о регистрации СМИ ПИ №ФС77-49724 от 11.05.2012 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Учредитель – Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Ярославский государственный университет им. П. Г. Демидова". Подписной индекс – 31907 в Каталоге российской прессы "Почта России". Редактор, корректор А.А. Аладьева. Редактор перевода Э.И. Соколова. Подписано в печать 28.09.2015. Дата выхода в свет 25.10.2015. Формат 60x84$^1/_8$. Усл. печ. л. 16,5. Уч.-изд. л. 15,5. Объем 141 с. Тираж 50 экз. Свободная цена. Заказ 073/015. Адрес типографии: ул. Советская, 14, оф. 109, г. Ярославль, 150000 Россия. Адрес издателя: Ярославский государственный университет им. П. Г. Демидова, ул. Советская, 14, г. Ярославль, 150000 Россия.

ISSN 1818–1015 (Print)
ISSN 2313–5417 (Online)

P.G. Demidov Yaroslavl State University

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

Volume 22 No 4(58) 2015

Founded in 1999
6 issues per year

Editorial Office Address: P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia
Website: http://mais.uniyar.ac.ru, e-mail: mais@uniyar.ac.ru

Contents

Modeling and Analysis of Information Systems. Vol. 22, No 4. 2015

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 453–463
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 453–463

DOI: 10.18255/1818-1015-2015-4-453-463

УДК 519.16 + 514.172.45

Полиэдральные графы задач об остовных деревьях при дополнительных ограничениях

Бондаренко В. А.[1], Николаев А. В.[2], Шовгенов Д. А.[1]

получена 30 июля 2015

Исследуются полиэдральные графы двух задач о минимальном остовном дереве при дополнительных ограничениях. В первой задаче речь идет об отыскании дерева с минимальной суммой весов ребер среди всех остовных деревьев, количество висячих вершин которых не превосходит заданную величину. Во второй задаче дополнительное ограничение заключается в предположении о том, что степени всех вершин искомого дерева не превосходят заданную величину. Обе рассматриваемые задачи в варианте распознавания являются NP-полными.

В работе изучаются многогранники указанных задач и их графы. Устанавливается, что в обоих случаях распознавание смежности вершин этих графов представляет собой NP-полную задачу. Несмотря на это, удается получить сверхполиномиальные нижние оценки плотности (кликового числа) этих графов, которые характеризуют временную трудоемкость в широком классе алгоритмов, использующих линейные сравнения. Приведенные результаты свидетельствуют о принципиальном отличии комбинаторно–геометрических свойств рассматриваемых задач от классической задачи о минимальном остовном дереве.

Ключевые слова: остовное дерево, полиэдральный граф, плотность графа, NP-полная задача, гамильтонова цепь

Для цитирования: Бондаренко В. А., Николаев А. В., Шовгенов Д. А., "Полиэдральные графы задач об остовных деревьях при дополнительных ограничениях", *Моделирование и анализ информационных систем*, **22**:4 (2015), 453–463.

Об авторах: Бондаренко Владимир Александрович, orcid.org/0000-0002-5976-3446, д-р физ.-мат. наук, профессор,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000, Россия, e-mail: bond@bond.edu.yar.ru

Николаев Андрей Валерьевич, orcid.org/0000-0003-4705-2409, канд. физ.-мат. наук,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000, Россия, e-mail: werdan.nik@gmail.com

Шовгенов Джамболет Азаматович, orcid.org/0000-0003-2022-4514, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000, Россия, e-mail: djsh92@mail.ru

Благодарности:

[1]При поддержке гранта РФФИ № 14-01-00333.

[2]При поддержке гранта Президента Российской Федерации МК-5400.2015.13.

1. Введение

Значительное число работ, связанных с вычислительной сложностью комбинаторных задач, направлено на изучение геометрических объектов, ассоциированных с задачами. Чаще всего такими объектами являются многогранники задач и графы этих многогранников. В частности, плотность полиэдрального графа (размер максимальной клики) задачи служит нижней оценкой вычислительной сложности в широком классе алгоритмов, основанных на линейных сравнениях. Более того, выяснилось, что эта характеристика полиномиальна для известных полиномиально разрешимых задач и сверхполиномиальна для труднорешаемых (см., например, [1–3]).

Хорошо известны задачи, которые в общей постановке полиномиально разрешимы, однако при введении дополнительных ограничений становятся NP-полными. Иногда происходит обратное: задача является NP-полной, однако введение дополнительных ограничений даёт возможность сконструировать для нее эффективный алгоритм. В связи с этим возникает вопрос: как введение дополнительных ограничений влияет на характеристики полиэдрального графа задачи?

Ниже рассматриваются задачи комбинаторной оптимизации, являющиеся задачами на графах и допускающие следующую постановку: задан реберно-взвешенный граф $G = (V, E)$ и некоторое множество T его подграфов, требуется найти подграф из T, имеющий минимальный (или максимальный) вес, под которым понимается сумма весов входящих в него рёбер.

Минимальное остовное дерево (minimum spanning tree, MST). В этой классической задаче требуется найти в связном графе G остовное дерево с минимальным весом.

Задача об остовном дереве полиномиально разрешима, например, алгоритмами Прима и Краскала [4].

Минимальное остовное дерево с ограничением на число висячих вершин (leaf constrained minimum spanning tree, LCST). В этой задаче требуется найти в связном графе $G(V, E)$ дерево минимального веса среди всех остовных деревьев, у которых число вершин степени 1 не превосходит заданную величину $k < |V|$.

Минимальное остовное дерево с ограничением на число висячих вершин в подграфе (restricted-leaf-in-subgraph minimum spanning tree, RLSST). Заданы связный граф G, некоторое подмножество U его вершин и положительное целое $k < |U|$, требуется найти в G остовное дерево минимального веса, не более k висячих вершин которого принадлежат множеству U.

Минимальное остовное дерево с ограничением на множество висячих вершин (set version of leaf constrained minimum spanning tree, SVST). Для связного графа $G(V, E)$ и некоторого подмножества U его вершин требуется найти в G остовное дерево минимального веса, все висячие вершины которого принадлежат множеству U.

Минимальное остовное дерево ограниченной степени (degree constrained minimum spanning tree, DCST). В этой задаче требуется найти дерево минимального веса среди всех остовных деревьев, степени вершин которых не превосходят заданную величину k.

В отличие от простой задачи об остовном дереве, для всех приведенных выше

задач уже проверка существования в графе G хотя бы одного остовного дерева, удовлетворяющего дополнительным ограничениям, является NP-полной задачей [5, 6].

Значительное число работ посвящено построению приближенных алгоритмов для задач об остовном дереве с ограничениями на число листьев и степени вершин [6–8]. В частности, можно выделить линейный 2-аппроксимационный алгоритм для двойственной задачи поиска остовного дерева с максимальным числом внутренних узлов [9] и полиномиальный алгоритм построения остовного дерева с максимальной степенью вершин $k+1$ и суммарным весом, не превышающим веса оптимального остовного дерева со степенями вершин не более k [10].

2. Многогранник задачи

Рассмотрим упомянутую выше общую задачу на графе $G = (V, E)$ с множеством T его подграфов. Пусть $|V| = n$, обозначим через d количество ребер полного графа:

$$d = |E| = \frac{n(n-1)}{2}.$$

Рассмотрим пространство $\mathbb{R}^d$, координаты точек в котором ассоциированы с ребрами графа G. Для каждого элемента t из T составим его характеристический вектор $x = x(t) \in \mathbb{R}^d$, положив равными единице значения тех координат, которые соответствуют ребрам, принадлежащим t, а значения остальных координат примем равными нулю. Совокупность характеристических векторов обозначим через X. Рассмотрим вектор $c \in \mathbb{R}^d$, составленный из весов ребер графа G. Тогда поставленная задача является задачей оптимизации линейной функции (c, x) на конечном множестве X.

Обозначим через $M(X)$ многогранник задачи: $M(X) = convX$. Полиэдральным графом задачи называется граф многогранника, множеством вершин которого служит множество геометрических вершин (в данном случае это X), а множеством ребер – совокупность геометрических ребер, то есть множество одномерных граней. Для описания графа многогранника полезно следующее утверждение (см., например, [11]).

Лемма 1. *Две вершины многогранника M смежны тогда и только тогда, когда они строго отделяются от множества остальных его вершин. Или, другими словами, две вершины x и y многогранника M являются несмежными тогда и только тогда, когда их некоторая выпуклая комбинация совпадает с некоторой выпуклой комбинацией остальных вершин, то есть найдутся такие $\alpha \geq 0, \beta \geq 0, \gamma_z \geq 0$, для которых*

$$\alpha x + \beta y = \sum \gamma_z z,$$
$$\alpha + \beta = \sum \gamma_z = 1,$$

и суммирование производится по всем вершинам, отличным от x и y.

3. Многогранник задачи об остовном дереве

Полное внешнее описание многогранника MST_n задачи об остовном дереве для графа $G(V, E)$ на n вершинах известно и имеет вид

$$\sum_{e \in E} x_e = n - 1, \tag{1}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \forall S \subset V, \tag{2}$$

$$x_e \geq 0, \forall e \in E. \tag{3}$$

Если ввести дополнительные переменные, систему (1)–(3) можно переписать в эквивалентном виде с полиномиальным ($O(n^3)$) числом ограничений, что позволяет решать задачу в том числе полиномиальными алгоритмами линейного программирования [12].

Полиэдральный граф многогранника MST_n полностью описан, точное значение плотности приведено в работе [13].

Теорема 1. *Плотность полиэдрального графа многогранника MST_n полиномиальна по n и равна*

$$\omega(MST_n) = \left\lfloor \frac{n^2}{4} \right\rfloor.$$

4. Остовное дерево с ограничением на число висячих вершин

В отличие от общей задачи полное внешнее описание многогранников задач об остовном дереве с ограничениями на число листьев не известно. Формулировка задач в виде целочисленного линейного программирования получается дополнением системы (1)–(3) ограничениями

$$\sum_{e \in \delta_v} x_e + (|\delta_v| - 1) y_v \leq |\delta_v|, \forall v \in V, \tag{4}$$

$$x_e, y_v \in \{0, 1\}, \forall e \in E, v \in V, \tag{5}$$

где δ_v – множество ребер, инцидентных вершине v, и дополнительные переменные y_v соответствуют висячим вершинам [14].

Данная формулировка используется, как правило, для задачи оптимизации числа висячих вершин

$$\sum_{v \in V} y_v \to \max(\min).$$

Вариант с оптимизацией веса остовного дерева можно получить, дополнив систему (1)–(5) неравенствами

$$\sum_{v \in V} y_v \leq k$$

для задачи с простым ограничением на число висячих вершин ($LCST_{n,k}$),

$$\sum_{v \in U} y_v \leq k$$

для задачи с ограничением в подграфе ($RLSST_{n,U,k}$), и

$$\forall v \in V \backslash U : y_v = 0$$

для задачи с ограничением на множество листьев ($SVST_{n,U}$).

Рассмотрим задачу о минимальном остовном дереве с ограничением на число висячих вершин. Пусть $|V| = n$, k – разрешенное количество висячих вершин. Построим остовное дерево t специального вида. Выберем две вершины u, w из V и набор V_{uw} из k вершин, часть $V_u = \{v_1, \cdots, v_s\}$ из которых соединим ребрами с вершиной u, а остальные $\{v_{s+1}, \cdots, v_k\} = V_w$ – с вершиной w. Оставшиеся $n-k-2$ вершины соединяются ребрами только друг с другом либо с вершинами u и w так, чтобы в результате образовалось остовное дерево (Рис. 1).

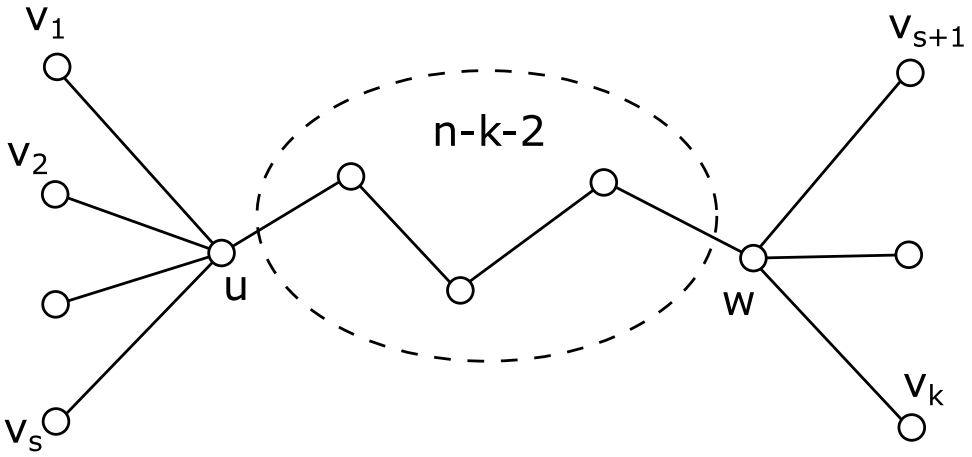


Рис. 1. Конструкция остовного дерева с k листьями
Fig. 1. Spanning tree construction with k leaves

Лемма 2. *Граф t_h, получаемый из дерева t отбрасыванием вершин $v_1, v_2, \cdots, v_k$ (вместе с ребрами (v_i, u) и (v_j, w)), является гамильтоновой цепью на $n-k$ оставшихся вершинах.*

Доказательство. Граф t_h является остовным деревом на множестве вершин $V \backslash V_{uw}$. Поэтому у него по меньшей мере две висячие вершины. Ими могут быть только u и w, так как любая другая вершина из $V \backslash V_{uw}$, оказавшись висячей в дереве t_h, останется таковой и в дереве t. Но лимит висячих вершин исчерпывается множеством V_{uw}. Следовательно, t_h является остовным деревом, у которого ровно две висячие вершины: u и w. Поэтому t_h – цепь, проходящая через все вершины из $V \backslash V_{uw}$, для которой u и w – концевые вершины. □

Зафиксируем множества V_u и V_w и рассмотрим совокупность T_k всех остовных деревьев описанного вида с k висячими вершинами. По лемме 2 каждое такое дерево

содержит цепь t_h с концевыми вершинами u и w, проходящую через все вершины из $V \backslash V_{uw}$. Верно и обратное: каждой цепи указанного вида соответствует дерево из T_k. Обозначим через HC_{uw} выпуклую оболочку характеристических векторов гамильтоновых цепей t_h между вершинами u и w.

Лемма 3. *Вершины x и y многогранника $LCST_{n,k}$, отвечающие деревьям из T_k, несмежны тогда и только тогда, когда несмежны соответствующие им вершины x_h и y_h многогранника HC_{uw}.*

Доказательство. Предположим, что вершины x_h и y_h многогранника HC_{uw} несмежны. Воспользуемся леммой 1, тогда найдутся неотрицательные α, β, γ_z, для которых $\alpha + \beta = \sum \gamma_z = 1$ и выполнено условие:

$$\alpha x_h + \beta y_h = \sum \gamma_z z_h, \;\; z_h \in H_{uw}. \tag{6}$$

Каждой гамильтоновой цепи из H_{uw} однозначно соответствует остовное дерево из T_k. Дополняя (6) равенствами для компонент, соответствующих ребрам (v_i, u) и (v_j, w), получим равенство

$$\alpha x + \beta y = \sum \gamma_z z, \;\; z \in T_k,$$

означающее, что вершины x и y многогранника $LCST_{n,k}$ несмежны.

Пусть теперь вершины x и y несмежны, тогда найдутся неотрицательные α, β, γ_z, для которых $\alpha + \beta = \sum \gamma_z = 1$ и

$$\alpha x + \beta y = \sum \gamma_z z.$$

У точек x и y, все координаты, соответствующие ребрам, инцидентным вершинам $v_1, v_2, \cdots, v_k$, совпадают, так как эти ребра фиксированы для остовных деревьев из T_k, а значит, они совпадают и у точек z, что позволяет перейти к равенству

$$\alpha x + \beta y = \sum \gamma_z z, \;\; z \in T_k.$$

Каждому остовному дереву из T_k однозначно соответствует гамильтонова цепь между вершинами u и w из H_{uw}. Таким образом,

$$\alpha x_h + \beta y_h = \sum \gamma_z z_h, \;\; z_h \in H_{uw},$$

и вершины x_h и y_h многогранника H_{uw} несмежны. □

Доказанная лемма 3 дает возможность воспользоваться свойствами задачи коммивояжера для изучения многогранника $LCST_{n,k}$. Для этого достаточно учесть следующий простой факт: две вершины многогранника HC_{uw} гамильтоновых цепей смежны тогда и только тогда, когда в многограннике задачи коммивояжера смежны вершины, соответствующие гамильтоновым циклам, образованным при отождествлении крайних вершин в одну. Таким образом из леммы 3 и известного результата Х. Пападимитриу [15] следует

Теорема 2. *Задача распознавания несмежности вершин многогранника $LCST_{n,k}$ является NP-полной.*

Несмотря на сложность описания графа многогранника $LCST_{n,k}$, можно получить сверхполиномиальную нижнюю оценку его плотности.

Теорема 3. *Плотность полиэдрального графа многогранника $LCST_{n,k}$ задачи об остовном дереве с ограничением на число висячих вершин сверхполиномиальна по n:*

$$\omega(LCST_{n,k}) \geq 2^{(\sqrt{\lfloor \frac{n-k-1}{2} \rfloor}-9)/2}.$$

Для доказательства теоремы 3 достаточно воспользоваться леммой 3 и нижней оценкой плотности полиэдрального графа многогранника TSP_n задачи коммивояжера для n городов (см. [1, 2]):

$$\omega(TSP_n) \geq 2^{(\sqrt{\lfloor \frac{n}{2} \rfloor}-9)/2)}.$$

Задачи с ограничениями в подграфе и на множество висячих вершин исследуются аналогично. В первом случае для $RLSST_{n,U,k}$ достаточно взять вместо графа G его подграф на вершинах U и построить соответствующую конструкцию остовного дерева. Во втором случае для $SVST_{n,U}$ достаточно в рассматриваемой конструкции отождествить множество листьев V_{uw} с множеством U.

Теорема 4. *Задача распознавания несмежности вершин многогранника $RLSST_{n,U,k}$ является NP-полной.*

Теорема 5. *Плотность полиэдрального графа многогранника $RLSST_{n,U,k}$ задачи об остовном дереве с ограничением на число висячих вершин в подграфе сверхполиномиальна по мощности множества U:*

$$\omega(RLSST_{n,U,k}) \geq 2^{(\sqrt{\lfloor \frac{|U|-k-1}{2} \rfloor}-9)/2}.$$

Теорема 6. *Задача распознавания несмежности вершин многогранника $SVST_{n,U}$ является NP-полной.*

Теорема 7. *Плотность полиэдрального графа многогранника $SVST_{n,U}$ задачи об остовном дереве с ограничением на множество висячих вершин сверхполиномиальна по n:*

$$\omega(SVST_{n,U}) \geq 2^{(\sqrt{\lfloor \frac{n-|U|-1}{2} \rfloor}-9)/2}.$$

5. Задача об остовном дереве ограниченной степени

Теперь обратимся к задаче о построении минимального остовного дерева, степени вершин которого не превосходят некоторого параметра k. Как и для задачи с ограничением на число висячих вершин полное внешнее описание многогранника $DCST_{n,k}$ не известно [8]. В форме целочисленного линейного программирования задача получается дополнением системы (1)–(3) ограничениями

$$\sum_{e \in \delta_v} x_e \leq k,$$

$$x_e \in 0, 1, v \in V.$$

Для $n > 2$ и $k > 1$ обозначим через

$$s = \left\lfloor \frac{n-2}{k-1} \right\rfloor$$

и построим дерево t специального вида. Разобьем множество вершин на s подмножеств вида $V_i = \{v_i, v_{i,1}, \cdots, v_{i,k-2}\}$ по $k-1$ вершине в каждом. Все оставшиеся вершины, которых будет от 2 до $k+1$, разобьем на два подмножества $V_0 = \{v_0, v_{0,1}, ..., v_{0,p}\}$ и $V_{s+1} = \{v_{s+1}, v_{s+1,1} \cdots, v_n\}$. Рассмотрим конструкцию следующего вида: в каждом подмножестве V_i все вершины соединяются ребрами только с вершиной v_i (Рис. 2). Отметим, что степени вершин v_0 и v_{s+1} по построению не могут превосходить k.

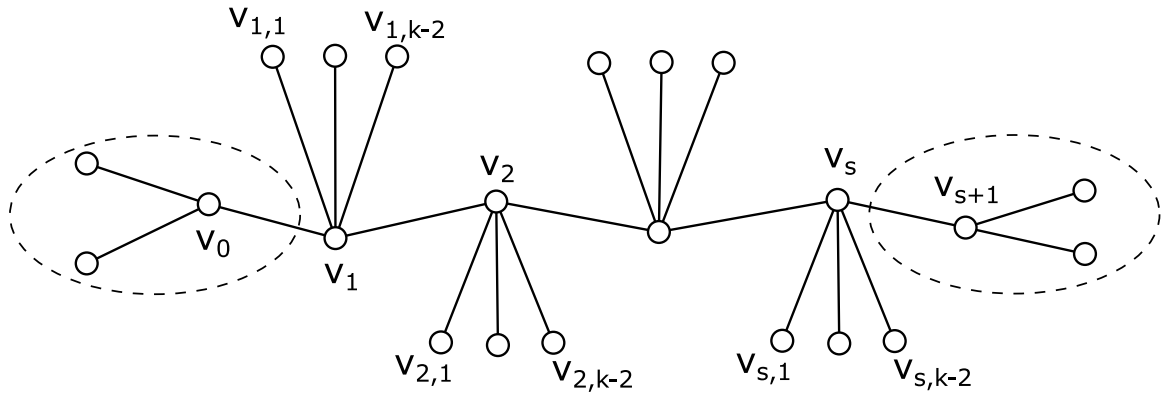


Рис. 2. Конструкция остовного дерева, степени вершин которого не превосходят k
Fig. 2. Spanning tree construction with k-bounded vertex degree

Лемма 4. *Граф t_h, получаемый из дерева t отбрасыванием вершин v_0, v_{s+1} и $v_{i,j}$ вместе с ребрами $(v_{i,j}, v_i)$, является гамильтоновой цепью с концевыми вершинами v_1 и v_s.*

Доказательство. По построению степени вершин v_1 и v_s не могут быть меньше $k-1$, а степени вершин $\{v_2, \cdots, v_{s-1}\}$ меньше $k-2$. Так как степени вершин в дереве t не могут превосходить k, то граф, получаемый после отбрасывания вершин, может быть только гамильтоновой цепью, соединяющей v_1 и v_s. □

Рассмотрим совокупность T_k всех остовных деревьев описанного вида. По лемме 4 каждое такое дерево содержит цепь t_h с концевыми вершинами v_1 и v_s, проходящую через вершины $\{v_2, \cdots, v_{s-1}\}$. Верно и обратное: каждой цепи указанного вида соответствует дерево из T_k. Обозначим через HC_{1s} выпуклую оболочку характеристических векторов гамильтоновых цепей между вершинами v_1 и v_s.

Лемма 5. *Вершины x и y многогранника $DCST_{n,k}$, отвечающие деревьям из T_k, несмежны тогда и только тогда, когда несмежны соответствующие им вершины x_h и y_h многогранника HC_{1s}.*

Доказательство проводится аналогично доказательству леммы 3. Как следствие получаем следующие утверждения.

Теорема 8. *Задача распознавания несмежности вершин многогранника $DCST_{n,k}$ является NP-полной.*

Теорема 9. *Плотность полиэдрального графа многогранника $DCST_{n,k}$ задачи об остовном дереве ограниченной степени сверхполиномиальна по s:*

$$\omega(DCST_{n,k}) \geq 2^{(\sqrt{\lfloor \frac{s-1}{2} \rfloor}-9)/2}.$$

6. Заключение

Таким образом, общая задача о минимальном остовном дереве и задачи с дополнительными ограничениями на число висячих вершин и степени вершин имеют принципиально отличные полиэдральные характеристики. Для классической задачи известны полиномиальные алгоритмы, построено полное внешнее описание многогранника с полиномиальным числом неравенств, полностью описан полиэдральный граф задачи, и установлено, что его плотность полиномиальна по размерности пространства. При этом задачи с дополнительными ограничениями являются труднорешаемыми, для них не найдено полного внешнего описания соответствующих многогранников, полиэдральные графы задач являются крайне сложными: даже проверка смежности вершин является NP-полной задачей, плотности графов сверхполиномиальны по размерности пространства.

Список литературы / References

[1] Бондаренко В. А., "Оценки сложности задач комбинаторной оптимизации в одном классе алгоритмов", *Доклады Академии наук*, **328**:1 (1993), 22–24; [Bondarenko V. A., "Complexity bounds for combinatorial optimization problems in one class of algorithms", *Russian Academy of Sciences Doklady Mathematics*, **328**:1 (1993), 22–24, (in Russian).]

[2] Бондаренко В. А., Максименко А. Н., *Геометрические конструкции и сложность в комбинаторной оптимизации*, ЛКИ, М., 2008, 184 с.; [Bondarenko V. A., Maksimenko A. N., *Geometricheskie konstruktsii i slozhnost' v kombinatornoy optimizatsii*, LKI, Moscow, 2008, (in Russian).]

[3] Бондаренко В. А., Николаев А. В., "Комбинаторно-геометрические свойства задачи о разрезе", *Доклады Академии наук*, **452**:2 (2013), 127–129; [Bondarenko V. A., Nikolaev A. V., "Combinatorial and Geometric Properties of the Max-Cut and Min-Cut Problems", *Doklady Mathematics*, **88**:2 (2013), 516–517]

[4] Пападимитриу Х., Стайглиц К., *Комбинаторная оптимизация: алгоритмы и сложность*, Мир, М., 1985, 512 с.; [Papadimitriou C. H., Steiglitz K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.]

[5] Гэри М., Джонсон Д., *Вычислительные машины и труднорешаемые задачи*, Мир, М., 1982, 416 с.; [Garey M. R., Johnson D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.]

[6] Rahman M. S., Kaykobad M., "Complexities of some interesting problems on spanning trees", *Information Processing Letters*, **94**:2 (2005), 93–97.

[7] Fernandes L. M., Gouveia L., "Minimal spanning trees with a constraint on the number of leaves", *European Journal of Operational Research*, **104**:1 (1998), 250–261.

[8] Goemans M. X., "Minimum Bounded-Degree Spanning Trees", Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, 273–282.

[9] Salamon G., Wiener G., "On finding spanning trees with few leaves", *Information Processing Letters*, **105**:5 (2008), 164–169.

[10] Singh M., Lau L. C., "Approximating minimum bounded degree spanning trees to within one of optimal", Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing., 2007, 661–670.

[11] Бренстед А., *Введение в теорию выпуклых многогранников*, Мир, М., 1988, 240 с.; [Brondsted A., *An Introduction to Convex Polytopes*, Springer-Verlag, 1983.]

[12] Martin R. K., "Using separation algorithms to generate mixed integer model reformulations", *Operations Research Letters*, **10**:3 (1991), 119–128.

[13] Белов Ю. А., "О плотности графа матроида", *Модели исследования операций в вычислительных системах*, Яросл. гос. ун-т., Ярославль, 1985, 95–100; [Belov Y. A., "O plotnosti grafa matroida", *Modeli issledovanija operacij v vychislitelnyh sistemah*, Yaroslavl state university, Yaroslavl, 1985, 95–100, (in Russian).]

[14] Fujie T., "The maximum-leaf spanning tree problem: Formulations and facets", *Networks*, **43**:4 (2004), 212–223.

[15] Papadimitriou C. H., "The Adjacency Relation on the Traveling Salesman Polytope is NP-Complete", *Mathematical Programming*, **14**:1 (1978), 312–324.

DOI: 10.18255/1818-1015-2015-4-453-463

1-Skeletons of the Spanning Tree Problems with Additional Constraints

Bondarenko V. A.[1], Nikolaev A. V.[2], Shovgenov D. A.[1]

Received July 30, 2015

In this paper, we study polyhedral properties of two spanning tree problems with additional constraints. In the first problem, it is required to find a tree with a minimum sum of edge weights among all spanning trees with the number of leaves less than or equal to a given value. In the second problem, an additional constraint is the assumption that the degree of all nodes of the spanning tree does not exceed a given value. The recognition versions of both problems are NP-complete. We consider polytopes of these problems and their 1-skeletons. We prove that in both cases it is a NP-complete problem to determine whether the vertices of 1-skeleton are adjacent. Although it is possible to obtain a superpolynomial lower bounds on the clique numbers of these graphs. These values characterize the time complexity in a broad class of algorithms based on linear comparisons. The results indicate a fundamental difference between combinatorial and geometric properties of the considered problems from the classical minimum spanning tree problem.

Keywords: spanning tree, 1-skeleton, clique number, NP-complete problem, hamiltonian chain

For citation: Bondarenko V. A., Nikolaev A. V., Shovgenov D. A., "1-Skeletons of the Spanning Tree Problems with Additional Constraints", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 453–463.

On the authors: Vladimir Bondarenko, orcid.org/0000-0002-5976-3446, doctor of science, professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: bond@bond.edu.yar.ru,

Andrei Nikolaev, orcid.org/0000-0003-4705-2409, PhD,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: werdan.nik@gmail.com

Dzhambolet Shovgenov, orcid.org/0000-0003-2022-4514, graduate student,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: djsh92@mail.ru

Acknowledgments:

[1]Supported by the Russian Foundation for Basic Research, project 14-01-00333.

[2]Supported by the President's of Russian Federation grant MK-5400.2015.1.

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 464–482
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 464–482

DOI: 10.18255/1818-1015-2015-4-464-482

УДК 517.9

Алгоритмы для мажоритарного декодирования групповых кодов

Деундяк В. М., Косолапов Ю. В.

получена 29 апреля 2015

Решается задача конструктивного описания и обоснования алгоритмов, необходимых при практической реализации мажоритарного декодера для групповых кодов, заданных как левые идеалы групповых алгебр. Кроме алгоритмов, необходимых для реализации классического декодера Дж. Мэсси, построено обобщение классического декодера для кодов с неравной защитой символов, который в ряде случаев может быть лучше классического. Для применения как классического декодера Дж. Мэсси, так и его обобщения к групповым кодам разработан алгоритм построения декодирующих деревьев, которые лежат в основе этих алгоритмов мажоритарного декодирования. В силу того, что групповые коды определяются как левые идеалы групповых алгебр, алгоритм построения декодирующих деревьев позволяет по одному дереву построить все декодирующие деревья. На основе обобщенного алгоритма декодирования разработан алгоритм декодирования групповых кодов, индуцированных кодами на подгруппе. Применение разработанных декодеров проиллюстрировано на примере кодов Рида–Маллера–Бермана и индуцированных ими групповых кодах на неабелевой группе аффинных преобразований. В частности, для кода Рида–Маллера–Бермана приводится описание и обоснование алгоритма построения одного декодирующего дерева, по которому с использованием алгоритма построения всех декодирующих деревьев строится мажоритарный декодер кода Рида–Маллера–Бермана и индуцированных им кодов.

Ключевые слова: мажоритарный декодер, групповые алгебры, групповые коды, коды Рида–Маллера–Бермана

Для цитирования: Деундяк В. М., Косолапов Ю. В., "Алгоритмы для мажоритарного декодирования групповых кодов", *Моделирование и анализ информационных систем*, **22**:4 (2015), 464–482.

Об авторах: Деундяк Владимир Михайлович, orcid.org/0000-0001-8258-2419, канд. физ.-мат. наук, доцент,
ФГНУ НИИ "Спецвузавтоматика",
пер. Газетный, 51, г. Ростов-на-Дону, 344002, Россия, e-mail: vlade@math.rsu.ru,

Косолапов Юрий Владимирович, orcid.org/0000-0002-1491-524X, канд. техн. наук,
Южный Федеральный Университет,
ул. Большая Садовая, 105/42, г. Ростов-на-Дону, 344006, Россия, e-mail: itaim@mail.ru

Введение

Среди комбинаторных методов декодирования линейных кодов, таких как декодирование по синдромам, декодирование по информационным совокупностям [1], мажоритарное декодирование Дж. Мэсси [2], особо выделяется последний метод. Особенность этого метода состоит в том, что скорость декодирования остается высокой при росте длины кода, в то время как первые два метода применимы для небольших длин кодов [3]. Однако для применения мажоритарного декодера необходимо, чтобы линейный код обладал некоторой специальной структурой. К таким кодам относятся введенные Дж. Мэсси MLD-коды и, в частности, классические коды Рида–Маллера, которые применяются как в теории связи, так и в криптографии [4]. Коды Рида–Маллера, как показано С.Д. Берманом в [5], могут быть описаны на основе использования аппарата групповой алгебры. В дальнейшем алгебраический подход Бермана к описанию кодов был развит в ряде работ. В частности, в [6] описываются p-ичные коды Рида–Маллера (p – простое число) и строится соответствующий мажоритарный декодер; в [7] исследуются коды типа Рида–Маллера на конечной абелевой группе, однако при этом декодер не описывается. В [8] показано, что используя алгебраический подход С.Д. Бермана, можно по известным кодам (для которых применим декодер Месси) построить класс других групповых кодов, пригодных для мажоритарного декодирования. Представляется, что новые коды, кроме применения в теории связи, могут быть использованы для усиления теоретико-кодовых методов защиты данных, таких, как кодовое зашумление [9], [10], широковещательное шифрование [11], кодовые криптосистемы [12], [13], [14]. Поэтому актуальной является задача конструктивного описания и обоснования алгоритмов, необходимых для практической реализации процедуры декодирования групповых MLD-кодов.

Настоящая статья состоит из трех разделов. В первом разделе рассматривается мажоритарный декодер Дж. Мэсси; в удобном виде приводится ряд известных фактов и с использованием декодирующих деревьев приводится конструктивное описание этого алгоритма, а также ряда вспомогательных алгоритмов. Далее строится обобщение декодера Дж. Мэсси на случай неравномерной защиты кодовых координат. Неравномерная защита является актуальной в случаях, когда номера координат в кодовых словах наделены разной по важности смысловой нагрузкой, и поэтому представляют интерес коды, позволяющие правильно декодировать важные координаты, даже если остальные координаты при этом декодируются с ошибкой (см., например, [15]).

Во втором разделе статьи рассматривается применение обобщенного мажоритарного алгоритма Дж. Мэсси к групповым кодам. Основными результатами этого раздела являются два алгоритма: алгоритм построения всех декодирующих деревьев для группового кода по одному дереву и алгоритм декодирования индуцированных кодов. Эти результаты позволяют декодировать широкий класс групповых кодов. Полученные результаты иллюстрируются в третьем разделе на группе аффинных преобразований $\mathrm{Aff}(\mathbb{F}_{2^n})$ и ее абелевой 2-подгруппе $\mathcal{H}$ порядка 2^n.

1. Мажоритарный декодер Дж. Мэсси для MLD–кодов и его обобщение на случай неравномерной защиты кодовых символов

1.1. Мажоритарный декодер Дж. Мэсси

В работе Дж. Мэсси [2] разработан мажоритарный декодер для линейных кодов на основе M–ортогональных множеств. Следуя [8], приведем необходимые сведения о конструкции соответствующего декодера. Для натурального n символом $\underline{n}$ будем обозначать множество $\{1;...;n\}$. Пусть V – векторное пространство над конечным полем $\mathbb{F}$. Зафиксируем в V базис B и символом (V, d_B) обозначим метрическое пространство V с метрикой Хэмминга d_B, построенной относительно базиса B. Для вектора $\mathbf{x}(\in V)$ множество базисных векторов, коэффициенты при которых в разложении $\mathbf{x} = \sum_{\mathbf{b}\in B} x_{\mathbf{b}}\mathbf{b}$ ненулевые, называется носителем вектора x относительно базиса B и обозначается $\mathrm{supp}_B(\mathbf{x})$. Вес $w_B(\mathbf{x})$ вектора $\mathbf{x}$ определяется как $|\mathrm{supp}_B(\mathbf{x})|$. (Здесь и далее символом $|A|$ обозначается мощность множества A.) Всякое линейное подпространство C метрического пространства (V, d_B) называется линейным кодом. Размерность и длину кода будем обозначать соответственно $k(C)$ и $n(C)$, а минимальное кодовое расстояние кода C обозначим $\mathrm{dist}_B(C)$. Двойственный код к коду C обозначим $C^{\perp}$. Множество векторов $\mathcal{M}_{\mathbf{v}} = \{\mathbf{v}^{(1)};...;\mathbf{v}^{(r)}\}(\subset V)$ называется M-ортогональным вектору $\mathbf{v}(\in V)$, если для любого $i \in \underline{r}$ найдется $\mathbf{w}^{(i)}(\in V\backslash\{(0,...,0)\})$ такой, что:

1)$\mathbf{v}^{(i)} = \mathbf{v} + \mathbf{w}^{(i)}$;

2)$\mathrm{supp}_B(\mathbf{v}) \cap \mathrm{supp}_B(\mathbf{w}^{(i)}) = \emptyset$;

3)$\mathrm{supp}_B(\mathbf{w}^{(j')}) \cap \mathrm{supp}_B(\mathbf{w}^{(j)}) = \emptyset$ для всех $j, j' \in \underline{r}$ таких, что $j \neq j'$.

Свойство M-ортогональности множества $\mathcal{M}_{\mathbf{v}}$ вектору $\mathbf{v}$, в частности, означает, что $|\mathrm{supp}_B(\mathbf{v}^{(i)})| > |\mathrm{supp}_B(\mathbf{v})|$ для всех $i = 1, ..., r$ и

$$\forall\ i \neq j: \quad \mathrm{supp}_B(\mathbf{v}^{(i)}) \cap \mathrm{supp}_B(\mathbf{v}^{(j)}) = \mathrm{supp}_B(\mathbf{v}). \tag{1}$$

Исходные параметры: $\mathbf{b}(\in B)$, $\mathcal{M}_{\mathbf{b}} = \{\mathbf{v}^{1,\mathbf{b}};...;\mathbf{v}^{r_{\mathbf{b}},\mathbf{b}}\}(\subseteq C^{\perp})$, $\mathbf{v} = \mathbf{c} + \mathbf{e}$ — принятый из канала вектор, $w_B(\mathbf{e}) \leq \lfloor r_{\mathbf{b}}/2 \rfloor$, $\mathbf{c} \in C$

Результат: значение координаты $c_{\mathbf{b}}$ кодового вектора

Вычислить последовательность $l[\mathcal{M}_{\mathbf{b}}, \mathbf{v}]$ вида

$$l[\mathcal{M}_{\mathbf{b}}, \mathbf{v}] := \left(l(\mathbf{v}^{1,\mathbf{b}}), ..., l(\mathbf{v}^{r_{\mathbf{b}},\mathbf{b}})\right); \tag{2}$$

// $l(\mathbf{v}^{i,\mathbf{b}}) = \left\langle \mathbf{v}, \mathbf{v}^{(i,\mathbf{b})} \right\rangle_B = \left\langle \mathbf{e}, \mathbf{v}^{(i,\mathbf{b})} \right\rangle_B$ – скалярное произведение

// векторов $\mathbf{v}$ и $\mathbf{v}^{(i,\mathbf{b})}$ в базисе B, $i = 1, ..., r_{\mathbf{b}}$;

$c_{\mathbf{b}} := v_{\mathbf{b}} - \mathrm{MajorVote}(r_{\mathbf{b}}, l[\mathcal{M}_{\mathbf{b}}, \mathbf{v}])$;

возвратить $c_{\mathbf{b}}$

Алгоритм 1: Decoder1

В [2] описана схема декодирования значения $\mathbf{b}$-координаты кодового слова с использованием имеющегося M-ортогонального множества $\mathcal{M}_{\mathbf{b}}$ $(\subseteq C^{\perp})$, где $\mathbf{b} \in B$. В соответствии с этой схемой в удобном для дальнейшего виде построим соответ-

ствующий алгоритм декодирования Decoder1, обращающийся к вспомогательному алгоритму MajorVote.

Лемма 1. *Пусть C – линейный код в V, B – базис в V, $\mathbf{b} \in B$, $\mathbf{v}$ – вектор на выходе канала связи,*

$$\mathbf{v} = \mathbf{c} + \mathbf{e}, \quad w_B(\mathbf{e}) = t \le \lfloor r_{\mathbf{b}}/2 \rfloor, \tag{3}$$

где $r_{\mathbf{b}} \in \mathbb{N}$, $\mathbf{c} \in C$. Если в коде $C^{\perp}$ найдется $r_{\mathbf{b}}$-элементное множество векторов $\mathcal{M}_{\mathbf{b}} = \{\mathbf{v}^{1,\mathbf{b}};\ldots;\mathbf{v}^{r_{\mathbf{b}},\mathbf{b}}\}$, которое M-ортогонально базисному вектору $\mathbf{b}$, то значение координаты $_{\mathbf{b}}$ в кодовом векторе может быть восстановлено с помощью алгоритма Decoder1. *Кроме того,*

$$\operatorname{dist}_B(C) \ge r_{\mathbf{b}} + 1. \tag{4}$$

Доказательство леммы 1 вытекает из [2], с. 24. Если для какого-то $\mathbf{b}$ в (4) выполняется строгое неравенство, то это означает, что декодер Decoder1 исправляет меньше ошибок, нежели теоретически возможно для кода C. Ниже строится модификация алгоритма Decoder1, которая может исправлять большее количество ошибок. Для описания обобщенного алгоритма Decoder2 нам понадобится понятие помеченного дерева, использованное в [8]. Символом $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}} = \mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$ будем обозначать помеченное дерево с корнем $\mathbf{b}$, обладающее следующими свойствами:

- множество вершин этого дерева состоит из $L_{\mathbf{b}} + 1$ уровня; корень $\mathbf{b}(\in B)$ находится на уровне 0, а листья – на уровне $L_{\mathbf{b}}$;
- каждая вершина, не являющаяся листом, имеет не менее $r_{\mathbf{b}}(\in \mathbb{N})$ непосредственно следующих за ней вершин;
- листья дерева помечены элементами из $C^{\perp}$;
- метки вершин, непосредственно следующих из произвольной вершины $\mathbf{v}$, находящейся на уровне $i(0 \le i < L_{\mathbf{b}})$, образуют в совокупности множество, M-ортогональное $\mathbf{v}$.

Замечание 1. *В силу определения дерева $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}$ и определения M-ортогональных систем, из существования для базисного вектора $\mathbf{b}$ дерева $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}$ следует существование дерева $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}}-1,L_{\mathbf{b}}}$ для $r_{\mathbf{b}} > 1$.*

Исходные параметры: $r(\in \mathbb{N})$, $\mathcal{A}$ – последовательность чисел из $\mathbb{F}$, $|\mathcal{A}| = r$
Результат: элемент $v(\in \mathbb{F})$, который в последовательности $\mathcal{A}$ встречается наибольшее число раз
для каждого $a \in \mathbb{F}$ **выполнять**
  вычислить величину $\mathrm{count}(a)$, равную числу появления элемента a в последовательности $\mathcal{A}$
конец цикла
если *найдется только один* $a'(\in \mathbb{F})$, *что* $\mathrm{count}(a') \ge \lceil r/2 \rceil$ **тогда**
  $v := a'$
иначе
  $v := 0$
конец условия
возвратить v

Алгоритм 2: MajorVote

С каждой меткой $\mathbf{p}$ в дереве $\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}$ будет связано числовое значение $l(\mathbf{p})(\in \mathbb{F})$ метки. Алгоритм вычисления этих меток, по сути, и есть алгоритм вычисления значения вектора ошибок в координате $\mathbf{b}$. Поэтому в дальнейшем дерево $\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}$ будем называть *декодирующим* деревом. Для удобства далее символом $\mathcal{M}_\mathbf{p}$ будем обозначать M-ортогональное множество, состоящее из векторов, которыми помечены вершины, непосредственно следующие из вершины с меткой $\mathbf{p}$, а символом V_i обозначим множество всех вершин на уровне i дерева $\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}$. Пусть также $l[\mathcal{M}_\mathbf{p}] := (l(\mathbf{q}))_{\mathbf{q}\in\mathcal{M}_\mathbf{p}}$.

Исходные параметры: $\mathbf{b}(\in B)$, принятый из канала вектор $\mathbf{v}$ вида (3),
декодирующее дерево $\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}$
Результат: значение координаты $c_\mathbf{b}$ кодового вектора
для каждого $\mathbf{p} \in V_{L_\mathbf{b}-1}$ **выполнять**
  $l(\mathbf{p}) := \mathrm{Decoder1}(\mathbf{p}, \mathcal{M}_\mathbf{p}, \mathbf{v})$
конец цикла
$\lambda := L_\mathbf{b} - 2$;
повторять
  для каждого $\mathbf{p} \in V_\lambda$ **выполнять**
    $l(\mathbf{p}) := \mathrm{MajorVote}(r_\mathbf{b}, l[\mathcal{M}_\mathbf{p}])$
  конец цикла
  $\lambda := \lambda - 1$;
до тех пор, пока $\lambda \geq 1$;
$c_\mathbf{b} := v_\mathbf{b} - l(\mathbf{b})$;
возвратить $c_\mathbf{b}$

Алгоритм 3: Decoder2

Лемма 2. *Пусть C – линейный код в V, B – базис в V, $\mathbf{b} \in B$, $\mathbf{v}$ – вектор на выходе канала связи вида (3). Если найдется декодирующее дерево $\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}[C]$, то значение координаты $c_\mathbf{b}$ в кодовом векторе может быть восстановлено с помощью алгоритма* Decoder2.

Доказательство. В алгоритме Decoder2 с помощью алгоритмов Decoder1 и MajorVote вычисляются значения линейных комбинаций координат вектора ошибок. Корректность этих значений следует из леммы 1, так как по условию $w_B(\mathbf{e}) \leq \lfloor r_\mathbf{b}/2 \rfloor$. При $\lambda = 1$ множество V_λ состоит из одной вершины с меткой $\mathbf{b}$, поэтому значение $l(\mathbf{b})$ также будет вычислено корректно и $l(\mathbf{b}) = e_\mathbf{b}$. Отсюда $c_\mathbf{b} = v_\mathbf{b} - l(\mathbf{b})$. □

1.2. Обобщение мажоритарного декодера Дж. Мэсси

Рассмотрим код $C(\subseteq V)$, B – базис в V,

$$\mathcal{WB}(C) = \{\mathrm{WB}_{\mathbf{b},r_\mathbf{b},L_\mathbf{b}}\}_{\mathbf{b}\in B} \tag{5}$$

– набор всех декодирующих деревьев для кода C. Если для всех $\mathbf{b}(\in B)$ выполняется равенство $\mathrm{dist}_B(C) = r_\mathbf{b} + 1$, то такой код называется MLD-кодом, при этом если $L_\mathbf{b} = L$ для каждого $\mathbf{b}(\in B)$, то такой код называется L-MLD-кодом [8]. Примеры

L-MLD кодов приведены в разделах 2.2 и 2.3. Код C, такой что для каждого $\mathbf{b}(\in B)$ можно построить декодирующее дерево $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$ с $r_{\mathbf{b}} \geq 2$, будем называть *обобщенным* MLD-кодом. Очевидно, что MLD-коды и L-MLD-коды входят в более широкий класс обобщенных MLD-кодов. Далее будем полагать, что $r_{\mathbf{b}}$ принимает максимальное из возможных значений для данного $\mathbf{b}$. Величина $\mathrm{dmaj}_B(C) := \min_{\mathbf{b}\in B}\{r_{\mathbf{b}}\}$ называется MLD-расстоянием кода C [8]; из (4) следует, что $\mathrm{dist}_B(C) \geq \mathrm{dmaj}_B(C)+1$.

В силу леммы 2 алгоритм Decoder2 позволяет гарантированно исправить любые векторы ошибок веса не более $\lfloor \mathrm{dmaj}_B(C)/2 \rfloor$. И в этом случае декодирующее дерево для каждого базисного вектора $\mathbf{b}$ можно минимизировать так, чтобы каждый узел на уровне $i (0 \leq i \leq L_{\mathbf{b}} - 1)$ был соединен с $\mathrm{dmaj}_B(C)$ узлами на уровне $i+1$. Такая минимизация позволит, во-первых, сократить количество вычислений скалярных произведений в алгоритме Decoder1 (который также используется в алгоритме Decoder2) и, во-вторых, сократить длину последовательности (2), которая используется в обоих алгоритмах декодирования посредством обращения их к алгоритму MajorVote.

Построим алгоритм Decoder3, который в ряде случаев может исправить ошибки веса больше, чем $\lfloor \mathrm{dmaj}_B(C)/2 \rfloor$. Упорядочим множество (5):

$$\mathcal{WB}(C) := \{\mathrm{WB}_{\mathbf{b}_1,r_{\mathbf{b}_1},L_{\mathbf{b}_1}}; ...; \mathrm{WB}_{\mathbf{b}_n,r_{\mathbf{b}_n},L_{\mathbf{b}_n}}\}, \quad r_{\mathbf{b}_1} \geq ... \geq r_{\mathbf{b}_n} (\geq 2). \tag{6}$$

Исходные параметры: принятый вектор $\mathbf{v} = (v_1, ..., v_n) = \mathbf{c} + \mathbf{e}$, набор декодирующих деревьев $\mathcal{WB}(C)$ вида (6)
Результат: вектор $\mathbf{c}'$ – результат декодирования
$w := \lfloor r_{\mathbf{b}_1}/2 \rfloor$, $i := 1$, $exit := 0$;
до тех пор, пока $i \leq n$ *и* $exit = 0$ **выполнять**
  $a := \mathrm{Decoder2}(\mathbf{v}, \mathbf{b}_i, \mathrm{WB}_{\mathbf{b}_i,r_{\mathbf{b}_i},L_{\mathbf{b}_i}})$;
  если $a \neq v_i$ **тогда**
    $v_i := a$;
    $w := w - 1$;
  конец условия
  если $w = 0$ **тогда**
    $exit := 1$;
  иначе
    если $\lfloor r_{\mathbf{b}_{i+1}}/2 \rfloor < w$ **тогда**
      $w := \lfloor r_{\mathbf{b}_{i+1}}/2 \rfloor$;
    конец условия
  конец условия
  i:=i+1;
конец цикла
$\mathbf{c}' := \mathbf{v}$;
возвратить $\mathbf{c}'$

Алгоритм 4: Decoder3

Поясним алгоритм. При упорядочении (6) $\mathbf{b}_i$-координаты кодовых слов являются не менее значащими, чем $\mathbf{b}_j$-координаты для всех $1 \leq i < j \leq n$, так как

в этом случае декодирующее дерево $\mathrm{WB}_{\mathbf{b}_i, r_{\mathbf{b}_i}, L_{\mathbf{b}_i}}$ позволяет корректно восстановить значение $\mathbf{b}_i$-координаты при весе ошибки, не меньшем чем при корректном восстановлении $\mathbf{b}_j$-координаты. Переменная w хранит максимальный вес ошибки, при котором может быть правильно декодировано значение хотя бы одной координаты. Если декодированное значение координаты отличается от принятого из канала значения, то считается, что исправлена ошибка и значение переменной w уменьшается. Если исправлены все возможные ошибки ($w = 0$), то алгоритм завершает работу и возвращается результат декодирования. Если же $w > 0$, то далее проверяется, что следующее дерево в упорядоченном наборе (5) позволяет исправить w и менее ошибок. Если $\lfloor r_{\mathbf{b}_{i+1}}/2 \rfloor < w$, то это означает, что следующее дерево (вообще говоря, все последующие деревья) не позволяет исправить w и менее ошибок. В этом случае значение переменной w корректируется так, чтобы следующее дерево могло исправить w и менее ошибок. Такая коррекция гарантирует, что алгоритм всегда будет корректно работать, если ошибок произошло не более чем $\lfloor \mathrm{dmaj}_B(C)/2 \rfloor$.

Лемма 3. *Пусть C – линейный код в V, B – базис в V, $\mathbf{v} = \mathbf{c} + \mathbf{e}$, (6) – упорядоченный набор декодирующих деревьев. Если $w_B(\mathbf{e}) \le \lfloor \mathrm{dmaj}_B(C)/2 \rfloor$, то кодовый вектор $\mathbf{c}$ восстанавливается алгоритмом* Decoder3.

Доказательство. В этом случае кодовый вектор восстанавливается путем применения n раз алгоритма Decoder2, корректность результата работы которого при $w_B(\mathbf{e}) \le \lfloor \mathrm{dmaj}_B(C)/2 \rfloor$ следует из леммы 2. □

Заметим, что у каждого базисного вектора $\mathbf{b}(\in B)$ в общем случае число $r_{\mathbf{b}}$ для обобщенного MLD-кода свое. Это означает, что символы кодового слова имеют неравную защиту при использовании декодера Decoder2. Это свойство, например, потенциально позволяет с помощью декодера Decoder2 исправлять наиболее значащие символы кодового слова даже при большом уровне помех. Методам кодирования и декодирования кодов с неравной защитой символов посвящена, например, работа [15]. Применение алгоритма Decoder3 представляется целесообразным в случае, когда почти всегда ошибки имеют вес не более $\lfloor \mathrm{dmaj}_B(C)/2 \rfloor$ и в редких случаях появляются ошибки большего веса.

В разделах 2 и 3 рассматриваются коды, для которых применим алгоритм декодирования Decoder3.

2. Групповые MLD-коды

2.1. Групповые алгебры и групповые коды

Пусть $\mathcal{G}$ — конечная группа, $\mathbb{F}$ — поле Галуа. Рассмотрим групповую алгебру $\mathbb{F}\mathcal{G}$, элементами которой являются формальные суммы (функции):

$$\sum_{g \in \mathcal{G}} a_g g, a_g \in \mathbb{F}. \tag{7}$$

Сложение в $\mathbb{F}\mathcal{G}$ выполняется покомпонентно, а умножение $\sum_{g\in\mathcal{G}} a_g g$ и $\sum_{g\in\mathcal{G}} b_g g$ выполняется по правилу:

$$\left(\sum_{g\in\mathcal{G}} a_g g\right)\left(\sum_{g\in\mathcal{G}} b_g g\right) := \sum_g \left[\sum_w a_{gw^{-1}} b_w\right] g, \tag{8}$$

при этом внешняя сумма в правой части (8) является формальной, а внутренняя сумма – это сумма над полем $\mathbb{F}$.

Функцию, принимающую значение 1 ($\in \mathbb{F}$) на нейтральном элементе $\hat{1}$ группы $\mathcal{G}$, а на всех остальных элементах группы принимающая значение 0 ($\in \mathbb{F}$), будем называть единицей групповой алгебры $\mathbb{F}\mathcal{G}$ и обозначать $\mathbf{1}$; функцию, принимающую значение 0 на всех элементах группы $\mathcal{G}$, назовем нулем групповой алгебры и обозначим $\mathbf{0}$. В формальных суммах $\sum_{g\in\mathcal{G}} a_g g$, где не все коэффициенты a_g равны нулю, слагаемые с нулевыми коэффициентами обычно будут опускаться. Далее функцию Дирака в точке g, как принято, будем обозначать $\delta_g = 1g$. Отметим, что функция $1\hat{1}$ совпадает с $\delta_{\hat{1}} = \mathbf{1}$, а произведение функций $1x = \delta_x$ и $1y = \delta_y$ в $\mathbb{F}\mathcal{G}$ равно $\delta_x\delta_y = \delta_{xy}$. Элементы групповой алгебры можно записывать в виде: $\sum_{g\in\mathcal{G}} a_g\delta_g, a_g \in \mathbb{F}$.

В конечномерной групповой алгебре $\mathbb{F}\mathcal{G}$ зафиксируем базис $B = B^{\mathcal{G}} := \{\mathbf{g} = \delta_g\}_{g\in\mathcal{G}}$. Это позволяет рассматривать в $\mathbb{F}\mathcal{G}$ метрику Хэмминга d_B. Отметим, что в категории конечномерных линейных пространств $\mathbb{F}\mathcal{G}$ и $\mathbb{F}^{|\mathcal{G}|}$ изоморфны. Группа $\mathcal{G}$ действует слева на групповой алгебре $\mathbb{F}\mathcal{G}$ следующим естественным образом (см. [8], с. 32):

$$\mathcal{G}\times\mathbb{F}\mathcal{G} \ni (g, \phi = \sum_{h\in\mathcal{G}} \phi_h h) \mapsto \phi g^{-1} := \sum_{h\in\mathcal{G}} \phi_{hg^{-1}} h \in \mathbb{F}\mathcal{G}. \tag{9}$$

Пусть A – алгебра, $A_0 \subset A$. Через $< A_0 >$ обозначим наименьшую подалгебру алгебры A, содержащую A_0. Если $A =< A_0 >$, то будем говорить, что A_0 порождает A. Если $\mathcal{G}_0$ — подгруппа группы $\mathcal{G}$, то, как легко видеть, $\mathbb{F}\mathcal{G}_0$ — подалгебра алгебры $\mathbb{F}\mathcal{G}$. Если $\mathcal{I}$ — левый (правый, двусторонний) идеал алгебры $\mathbb{F}\mathcal{G}$, то через $\mathcal{I}^t$ обозначим подалгебру алгебры $\mathbb{F}\mathcal{G}$, порожденную элементами вида $\mathbf{x}_1 ... \mathbf{x}_t$, где $\mathbf{x}_i \in \mathcal{I}$; известно, что $\mathcal{I}^t$ – это левый (правый, двусторонний) идеал алгебры $\mathbb{F}\mathcal{G}$.

В соответствии с [8], с. 39, всякий отличный от $\{\mathbf{0}\}$ *левый* идеал C в групповой алгебре $\mathbb{F}\mathcal{G}$ называется *групповым кодом* ($\mathbb{F}\mathcal{G}$–кодом) длины $n(C) = |\mathcal{G}|$. Отметим, что как левый идеал код является также левым $\mathbb{F}\mathcal{G}$–модулем. Пусть $< \cdot, \cdot >$ — невырожденная симметрическая $\mathbb{F}$-билинейная форма на $\mathbb{F}\mathcal{G}$, которая однозначно определяется условием: для любых $g(\in \mathcal{G})$ и $h(\in \mathcal{G})$

$$< \delta_g, \delta_h >= \begin{cases} 1, & \text{если } g = h, \\ 0, & \text{если } g \neq h. \end{cases}$$

Если C — произвольный $\mathbb{F}\mathcal{G}$-код, то множество

$$C^{\perp} = \{\mathbf{x} \in \mathbb{F}\mathcal{G} \mid \forall \mathbf{z} \in C \ < \mathbf{x}, \mathbf{z} >= 0\}$$

является $\mathbb{F}\mathcal{G}$-кодом (см. [8], теорема 1.6.5c) и называется дуальным кодом к коду C.

Для групповых кодов с целью их применения в борьбе с помехами в системах передачи данных необходимо, разумеется, задать правило кодирования. Пусть C —

групповой код размерности $k = k(C)$, $C \subseteq \mathbb{F}\mathcal{G}$, $B(C) := \{\epsilon_1; ...; \epsilon_k\}$ – базис кода C, где для $i = 1, ..., k$ имеет место представление:

$$\epsilon_i = \sum_{g \in \mathcal{G}} \alpha_{i,g}\delta_g, \quad \alpha_{i,g} \in \mathbb{F}.$$

Тогда при кодировании произвольному информационному вектору $\mathbf{s} = (s_1, ..., s_k)$ ставится в соответствие кодовый вектор $\mathbf{c}(\in C)$ вида:

$$\mathbf{c} = \sum_{i=1}^{k} s_i\epsilon_i = \sum_{i=1}^{k} s_i \sum_{g \in \mathcal{G}} \alpha_{i,g}\delta_g = \sum_{g \in \mathcal{G}} (\sum_{i=1}^{k} s_i\alpha_{i,g})\delta_g.$$

Для определения порождающей матрицы $G(C)$ группового кода $C(\subseteq \mathbb{F}\mathcal{G})$ необходимо задать линейный порядок на множестве элементов группы $\mathcal{G}$ и, таким образом, на базисе B. Тогда строками порождающей матрицы $G(C)$ будут являться элементы базиса $B(C)$, представленные в виде упорядоченных наборов коэффициентов в формальных суммах вида (7), а столбцами этой матрицы будут векторы вида $(\alpha_{1,g}, ..., \alpha_{k,g})$, $g \in \mathcal{G}$. Другими словами, порождающая матрица $G(C)$ – это матричное представление базиса $B(C)$ через базис B групповой алгебры $\mathbb{F}\mathcal{G}$. Аналогично можно определить и проверочную матрицу $H(C)$, совпадающую с $G(C^{\perp})$.

Напомним, что пересечение всех максимальных левых идеалов $\mathbb{F}$–алгебры $\mathcal{A}$ называется радикалом $\mathrm{Rad}\mathcal{A}$ (см. [8], с.69); $\mathrm{Rad}\mathcal{A}$ — пример группового кода. Так как степень левого идеала — левый идеал, то и $(\mathrm{Rad}\mathcal{A})^t$ – групповой код.

Для произвольного подмножества X алгебры $\mathcal{A}$ в [8], с. 69, определен левый идеал

$$\mathcal{A} \cdot X := \sum_{x \in X} \mathcal{A}x(\subseteq \mathcal{A}), \tag{10}$$

порожденный множеством X. Следовательно, $\mathcal{A} \cdot X$ — тоже групповой код.

В заключение рассмотрим важное для дальнейшего семейство групповых кодов, открытое С.Д. Берманом [5]. Пусть $\mathbb{F} = \mathbb{F}_2$, $\Delta_{2,n}$ – 2-группа порядка 2^n, имеющая представление в виде внутреннего произведения n подгрупп, каждая из которых имеет порядок 2:

$$\Delta_{2,n} =< g_1 > \times ... \times < g_n >, \tag{11}$$

где для $i = 1, ..., n$, элемент g_i – образующий элемент порядка 2, $g_i^2 = \hat{1}$. Отметим, что $\Delta_{2,n}$ – это аддитивная группа поля $\mathbb{F}_{2^n}$. Множество $\{h_1; ...; h_n\}$ называется минимальной системой порождающих элементов группы $\Delta_{2,n}$, если любой элемент $g(\in \Delta_{2,n})$ может быть представлен в виде $g = h_1^{a_1}...h_n^{a_n}$, $(a_1, ..., a_n \in \{0; 1\})$, и никакое подмножество множества $\{h_1; ...; h_n\}$ этим свойством не обладает [8]. В групповой алгебре $\mathbb{F}_2\Delta_{2,n}$ Берманом построено семейство кодов, изоморфное семейству классических кодов Рида–Маллера $\mathcal{RM}_2(r, n)$. Приведем соответствующие конструкции. В коммутативной групповой алгебре $\mathbb{F}_2\Delta_{2,n}$, которая как $\mathbb{F}_2$–линейное пространство изоморфна пространству $\mathbb{F}_2^{2^n}$, кроме стандартного упорядоченного базиса

$$B = \{\mathbf{1}; \delta_{g_1}; ...; \delta_{g_n}; \delta_{g_1}\delta_{g_2}; ...; \delta_{g_1}\delta_{g_n}; ...; \delta_{g_{n-1}}\delta_{g_n}; ...; \delta_{g_1}...\delta_{g_n}\}$$

рассмотрим базис

$$B_0 = \left\{ (\delta_{g_1} - \mathbf{1})^{a_1}...(\delta_{g_n} - \mathbf{1})^{a_n} | a_1, ..., a_n \in \{0; 1\}, \sum_{s=1}^{n} a_s \geq 0 \right\}$$

(см. [8], с. 49) и зафиксируем *лексикографическое* упорядочение по степеням $(\delta_{g_i}-\mathbf{1})$ в этом базисе:

$$B_0 = \{\mathbf{1}; (\delta_{g_1}-\mathbf{1}); ...; (\delta_{g_n}-\mathbf{1}); (\delta_{g_1}-\mathbf{1})(\delta_{g_2}-\mathbf{1}); ...; (\delta_{g_1}-\mathbf{1})(\delta_{g_n}-\mathbf{1}); ...; (\delta_{g_1}-\mathbf{1})...(\delta_{g_n}-\mathbf{1})\}.$$

В $\mathbb{F}_2\Delta_{2,n}$ рассмотрим $\mathbb{F}_2$–подпространство RM_t, порожденное множеством:

$$B_t := \left\{ (\delta_{g_1} - \mathbf{1})^{a_1}...(\delta_{g_n} - \mathbf{1})^{a_n} | a_1, ..., a_n \in \{0; 1\}, \sum_{s=1}^{n} a_s \geq t \right\}, \tag{12}$$

где $t \in \mathbb{N}$. Из определения RM_t вытекает, что

$$\mathrm{RM}_n = \{\mathbf{0}; (\delta_{g_1} - \mathbf{1})...(\delta_{g_n} - \mathbf{1})\}. \tag{13}$$

Из вложения $B_0 \supset B_1 \supset B_2 \supset ... \supset B_n \supset \{\mathbf{0}\}$ следует вложение $\mathbb{F}_2\Delta_{2,n} = \mathrm{RM}_0 \supset \mathrm{RM}_1 \supset \mathrm{RM}_2 \supset ... \supset \mathrm{RM}_n \supset \{\mathbf{0}\}$.

В [8] показано, что для любого t пространство $\mathrm{RM}_t = (\mathrm{RM}_1)^t$ и является идеалом в $\mathbb{F}\Delta_{2,n}$, причем $\mathrm{RM}_1 = \mathrm{Rad}\mathbb{F}_2\Delta_{2,n}$. Таким образом, для $t = 1, ..., n$ пространство RM_t является групповым кодом размерности $k(\mathrm{RM}_t) = \sum_{i=t}^{n} C_n^t$ и длины $n(\mathrm{RM}_t) = 2^n$. Так как код RM_t является подпространством $\mathbb{F}_2$-пространства $\mathbb{F}_2^{2^n}$, то для RM_t можно предъявить порождающую матрицу $G(\mathrm{RM}_t)$, в которой $k(\mathrm{RM}_t)$ строк и $n(\mathrm{RM}_t)$ столбцов. Для явного выписывания матрицы $G(\mathrm{RM}_t)$ достаточно воспользоваться упорядоченным базисом B пространства $\mathbb{F}_2^{2^n}$ и записать матричное представление базиса B_t в базисе B. В работе [5] доказано, что для любого t код RM_t естественно изоморфен классическому коду Рида–Маллера $\mathcal{RM}_2(n-t, n)$. В частности, отсюда вытекает, что согласно [17], с. 203, $\mathrm{RM}_t^{\perp} = \mathrm{RM}_{n-t+1}$. Далее эти коды будем называть кодами Рида–Маллера–Бермана.

2.2. Декодирование групповых MLD–кодов

В классе MLD–кодов групповые MLD–коды играют особую роль. Именно, группа $\mathcal{G}$ действует транзитивно по правилу (9) на базисных элементах из $B^{\mathcal{G}}$, поэтому по декодирующему дереву для какого-либо одного базисного элемента $B^{\mathcal{G}}$ можно найти декодирующие деревья для всех остальных элементов из базиса. Построим соответствующий алгоритм CloneTree.

Исходные параметры: $\mathcal{G}$, $\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$, $\mathbf{b}'$
Результат: $\mathrm{WB}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$
$\mathrm{WB}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C] := \mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[C]$;
Найти $g(\in \mathcal{G})$ такой, что $(g, \mathbf{b}) = \mathbf{b}'$;
для каждой *метки* $\mathbf{p}$ *дерева* $\mathrm{WB}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$ **выполнять**
  // Действие элементом g на $\mathbf{p}$ по правилу (9);
  $\mathbf{p} := (g, \mathbf{p})$;
конец цикла
возвратить $\mathrm{WB}_{\mathbf{b}',r'_{\mathbf{b}},L_{\mathbf{b}'}}[C]$

Алгоритм 5: CloneTree

Так как $C^{\perp}$ – идеал, то действие группы $\mathcal{G}$ по правилу (9) на элементах кода $C^{\perp}$ не выводит за границы этого кода. Поэтому листья нового декодирующего дерева на выходе алгоритма CloneTree будут принадлежать коду $C^{\perp}$, при этом свойство M-ортогональности в этом дереве не нарушается.

Все декодирующие деревья кода C можно построить по алгоритму MakeAllTrees.

Исходные параметры: $\mathcal{G}$, $\mathrm{WB}_{\mathbf{1}_{\mathcal{G}},r_{\mathbf{1}_{\mathcal{G}}},L_{\mathbf{1}_{\mathcal{G}}}}[C]$
Результат: $\mathcal{WB}(C)$
$\mathcal{WB}(C) := \emptyset$;
для каждого $\mathbf{b} \in B^{\mathcal{G}}$ **выполнять**
  $\mathcal{WB}(C) := \mathcal{WB}(C) \cup \mathrm{CloneTree}(\mathcal{G}, \mathrm{WB}_{\mathbf{1}_{\mathcal{G}},r_{\mathbf{1}_{\mathcal{G}}},L_{\mathbf{1}_{\mathcal{G}}}}[C], \mathbf{b})$
конец цикла
возвратить $\mathcal{WB}(C)$

Алгоритм 6: MakeAllTrees

В силу алгоритма MakeAllTrees, все декодирующие деревья группового MLD-кода C имеют одинаковую структуру, а отличаются только метками узлов. В этом случае представления вида (5) и (6) совпадают. Таким образом, для применения алгоритма декодирования Decoder3 для группового MLD-кода C достаточно иметь одно декодирующее дерево $\mathrm{WB}_{\mathbf{1}_{\mathcal{G}},r_{\mathbf{1}_{\mathcal{G}}},L_{\mathbf{1}_{\mathcal{G}}}}[C]$, так как все остальные деревья строятся алгоритмом MakeAllTrees.

Отметим, что коды Рида–Маллера–Бермана, рассмотренные в разделе 2.1., являются групповыми MLD-кодами.

2.3. Индуцированные групповые MLD-коды

Пусть $\mathcal{G}$ — конечная группа, $\mathcal{H}$ — ее нормальная подгруппа, $\lambda = [\mathcal{G} : \mathcal{H}]$ — индекс подгруппы $\mathcal{H}$ в группе $\mathcal{G}$. Группа $\mathcal{G}$ разбивается на множество непересекающихся правых классов смежности $\{\mathcal{H}y\}_{y\in Y}$, где $Y = \{y_1 = \hat{1}; y_2; ...; y_\lambda\}(\subset \mathcal{G})$ — линейно упорядоченная правая трансверсаль $\mathcal{G}$ по $\mathcal{H}$. Групповая алгебра $\mathbb{F}\mathcal{G}$ является свободным левым $\mathbb{F}\mathcal{H}$-модулем с базисом Y, т.е.

$$\mathbb{F}\mathcal{G} = \bigoplus_{i=1}^{\lambda} (\mathbb{F}\mathcal{H})\delta_{y_i}. \tag{14}$$

Если $B^{\mathcal{H}} = \{\delta_h\}_{h\in\mathcal{H}}$ — базис в линейном векторном пространстве $\mathbb{F}\mathcal{H}$, то базис $B^{\mathcal{G}} = \{\delta_g\}_{g\in\mathcal{G}}$ в $\mathbb{F}\mathcal{G}$ представим в виде: $B^{\mathcal{G}} = \cup_{i=1}^{\lambda} B^{\mathcal{H}}\delta_{y_i}$ (см. (14)).

Рассмотрим $\mathbb{F}\mathcal{H}$–код N размерности $k(N) = |B(N)|$ и длины $n(N) = |\mathcal{H}|$ с $\mathbb{F}$-базисом $B(N) = \{\eta_1; ...; \eta_{k(N)}\}$, $\eta_i = \sum_{h\in\mathcal{H}} \beta_{i,h}\delta_h$; пусть $\mathcal{WC}(N)$ – множество декодирующих деревьев кода N, например, построенное с помощью алгоритма MakeAllTrees. В силу [8], с. 84, код $\mathbb{F}\mathcal{G}\cdot N$ как левый $\mathbb{F}\mathcal{G}$–модуль изоморфен тензорному произведению $\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N(\subseteq \mathbb{F}\mathcal{G})$, определенному в [16], с. 80, при этом размерность $k(\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N) = |Y|k(N)$, а длина $n(\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N) = |\mathcal{G}|$ и совпадает с $|Y|n(N)$. Этот изоморфизм является также $\mathbb{F}$–изометрией относительно метрики Хемминга. Говорят, что код $\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N$ индуцирован кодом N. Из определений вытекает, что базис индуцированного кода состоит из векторов (функций) вида:

$$\kappa_{i,j} = \eta_i\delta_{y_j}, \quad i = 1, ..., k(N), \quad j = 1, ..., \lambda.$$

Введем линейное упорядочение на базисных векторах индуцированного кода: $\kappa_l := \kappa_{i,j}$, где $l = i + (j-1)k(N)$. Рассмотрим две такие вспомогательные функции $i : \{1;...;k(N)\lambda\} \to \{1;...;k(N)\}$ и $j : \{1;...;k(N)\lambda\} \to \{1;...;\lambda\}$, что $i(l) + (j(l) - 1)k(N) = l$. Эти две функции позволяют переходить от одинарной нумерации базисных векторов к двойной. Тогда информационному вектору $\mathbf{s} = (s_1, ..., s_{k(N)\lambda})(\in \mathbb{F}^{k(N)\lambda})$ соответствует кодовое слово вида:

$$\mathbf{c} = \sum_{l=1}^{k(N)\lambda} s_l\kappa_l = \sum_{l=1}^{k(N)\lambda} s_l\kappa_{i(l),j(l)} = \sum_{l=1}^{k(N)\lambda} s_l\eta_{i(l)}\delta_{y_{j(l)}}$$
$$= \sum_{l=1}^{k(N)\lambda} s_l \sum_{h\in\mathcal{H}} \beta_{i(l),h}\delta_h\delta_{y_{j(l)}} = \sum_{h\in\mathcal{H}} \sum_{l=1}^{k(N)\lambda} s_l\beta_{i(l),h}\delta_{hy_{j(l)}}.$$

Таким образом, при фиксированном $h(\in \mathcal{H})$ вычисляются λ коэффициентов кодового вектора $\mathbf{c} = (c_1, ..., c_{n(N)\lambda})$ при базисных элементах $\delta_{hy_1}, ..., \delta_{hy_\lambda}$:

$$c_{\delta_{hy_v}} = \left\langle (s_{1+(v-1)k(N)}, ..., s_{vk(N)}), (\beta_{1,h}, ..., \beta_{k(N),h}) \right\rangle \quad v = 1, ..., \lambda,$$

где $\langle \mathbf{a}, \mathbf{b} \rangle$ — скалярное произведение векторов $\mathbf{a}$ и $\mathbf{b}$. Пусть $\chi = |\mathcal{H}|$. Занумеруем элементы подгруппы $\mathcal{H}$: $\mathcal{H} = \{h_1; ...; h_\chi\}$. Тогда можно упорядочить базис алгебры $\mathbb{F}\mathcal{G}$ так, что кодовый вектор $\mathbf{c}$ можно представить в виде:

$$\mathbf{c} = (\mathbf{c}^1||...||\mathbf{c}^\lambda), \mathbf{c}^i = (c_{\delta_{h_1 y_i}}, ..., c_{\delta_{h_\chi y_i}}) \in N, i = 1, ..., \lambda, \tag{15}$$

где $(\mathbf{a}||\mathbf{b})$ — приписывание вектора $\mathbf{b}$ справа к вектору $\mathbf{a}$. Итак, если $G(N)$ – порождающая $(k(N) \times n(N))$-матрица кода N, то кодирование для индуцированного кода представимо в виде умножения вектора длины $k(N)\lambda$ на матрицу вида:

$$G(\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N) = \left(\begin{array}{c|c|c|c} G(N) & O & ... & O \\ \hline O & G(N) & ... & O \\ \hline ... & ... & ... & ... \\ \hline O & ... & ... & G(N) \end{array}\right), \tag{16}$$

где O – нулевая $(k(N) \times n(N))$-матрица. Из представления (16) порождающей матрицы индуцированного кода вытекают доказанные в [8] равенства: $\mathrm{dmaj}_{B^\mathcal{G}}(\mathbb{F}\mathcal{G}\cdot N) = \mathrm{dmaj}_{B^\mathcal{H}}(N)$ и $\mathrm{dist}_{B^\mathcal{G}}(\mathbb{F}\mathcal{G}\cdot N) = \mathrm{dist}_{B^\mathcal{H}}(N)$. Вид (16) порождающей матрицы позволяет построить алгоритм мажоритарного декодирования Decoder4.

Теорема 1. *Пусть $\mathcal{H}$ – нормальный делитель группы $\mathcal{G}$, $N(\subseteq \mathbb{F}\mathcal{H})$ – групповой* MLD*-код, $\mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N(\subseteq \mathbb{F}\mathcal{G})$ – индуцированный код. Тогда алгоритм* Decoder4 *гарантированно исправляет любые ошибки веса не более $\lfloor \mathrm{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$.*

Доказательство. Так как, по условию, вес ошибки не более $\lfloor \mathrm{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$, то в представлении (17) каждый вектор $\mathbf{v}^i$ имеет вид: $\mathbf{v}^i = \mathbf{c}^i + \mathbf{e}^i$, где $w_{B^\mathcal{H}}(\mathbf{e}^i) \leq \lfloor \mathrm{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$, $\mathbf{c}^i \in N$ (см. (15)). Тогда, по лемме 3, алгоритм Decoder3 гарантированно восстановит вектор $\mathbf{c}^i$ для всех $i = 1, ..., \lambda$. □

Исходные параметры: $\mathcal{H}$ – нормальный делитель группы $\mathcal{G}$, $\mathcal{WC}(N)$ – множество декодирующих деревьев группового MLD-кода $N(\subseteq \mathbb{F}\mathcal{H})$, $H(N)$ – проверочная матрица кода N, $\mathbf{v} = (v_1, ..., v_{k(N)\lambda})$ – принятый вектор, W – максимальный вес ошибки в векторе $\mathbf{v}$

Результат: $\mathbf{c}'(\in \mathbb{F}\mathcal{G} \otimes_{\mathbb{F}\mathcal{H}} N)$ – результат декодирования

Представить вектор $\mathbf{v}$ в виде конкатенации λ векторов:

$$\mathbf{v} = (\mathbf{v}^1||...||\mathbf{v}^\lambda), \quad \mathbf{v}^i \in \mathbb{F}^{n(N)}, \quad i = 1, ..., \lambda; \tag{17}$$

$j := 1$, $\omega := 0$, exit $:= 0$;
до тех пор, пока $j \leq \lambda$ *и* exit $= 0$ **выполнять**
  если $\omega \geq W$ **тогда**
    exit $:= 1$;
    $i := j$ **до тех пор, пока** $i \leq \lambda$ **выполнять**
      $\mathbf{c}'^i := \mathbf{v}^i$;
      $i := i + 1$;
    конец цикла
  иначе
    если $\mathbf{v}^i H(N)^\top = \mathbf{0}(\in \mathbb{F}^{n(N)})$ **тогда**
      // Блок не содержит ошибок;
      $\mathbf{c}'^i := \mathbf{v}^i$;
    иначе
      $\mathbf{c}'^i = \text{Decoder3}(\mathbf{v}^i, \mathcal{WB}(N))$;
      $\omega := \omega + w_{B^\mathcal{H}}(\mathbf{c}'^i - \mathbf{v}^i)$;
    конец условия
  конец условия
  $j := j + 1$;
конец цикла
возвратить $\mathbf{c}' := (\mathbf{c}'^1||...||\mathbf{c}'^\lambda)$

Алгоритм 7: Decoder4

Отметим, что у индуцированного кода $\mathbb{F}\mathcal{G} \cdot N$ отношение размерность/длина равно значению аналогичного отношения для кода N, при этом отношение количества исправляемых ошибок кодом $\mathbb{F}\mathcal{G} \cdot N$ к длине кодового слова уменьшается в λ раз по отношению к аналогичной характеристике кода N. Это, в частности, означает, что если в канале передачи данных ошибки группирующиеся, то декодер Decoder4 с большой вероятностью ошибется при весе ошибки, большем чем $\lfloor \text{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$. Если же заранее известно, что вес группирующейся ошибки не более $\lfloor \text{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$, то для ее исправления в алгоритме Decoder4 (с входным параметром $W = \lfloor \text{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$) потребуется применить алгоритм Decoder3 не более двух раз, что может заметно увеличить скорость декодирования блоков длины $n(N)\lambda$. С другой стороны, если ошибки равномерно распределены по всему кодовому вектору длины $n(N)\lambda$, то алгоритм Decoder4 может исправить помехи веса до $\lambda \lfloor \text{dmaj}_{B^\mathcal{H}}(N)/2 \rfloor$. Поэтому алгоритм Decoder4 может быть применим для борьбы

с группирующимися ошибками веса до $\lambda\lfloor \mathrm{dmaj}_{B^{\mathcal{H}}}(N)/2\rfloor$, если кодовые слова перед поступлением в канал подвергаются перемежению (скремблированию). Таким образом, на основе кода N и декодера Decoder4 можно строить наборы кодов для подстраивания под помеховую обстановку в канале передачи данных. Далее приводится пример кодов, индуцированных кодами Рида–Маллера–Бермана RM_t.

3. Групповые MLD-коды типа Рида–Маллера–Бермана

3.1. Групповые коды на группе $\mathrm{Aff}(\mathbb{F}_{2^n})$, индуцированные кодами Рида–Маллера–Бермана

Рассмотрим группу аффинных преобразований $\mathcal{G} = \mathrm{Aff}(\mathbb{F}_{2^n})$ одномерного линейного пространства $\mathbb{F}_{2^n}$, элементами которой являются преобразования вида:

$$\varphi_{\alpha,\beta} : \mathbb{F}_{2^n} \ni x \to \alpha x + \beta \in \mathbb{F}_{2^n}, (\alpha, \beta \in \mathbb{F}_{2^n}, \alpha \neq 0, n > 1).$$

Эта группа неабелева и соответствие

$$\varphi_{\alpha,\beta} \mapsto \begin{pmatrix} \alpha & \beta \\ 0 & 1 \end{pmatrix}$$

определяет естественный изоморфизм $\mathcal{G}$ на группу невырожденных матриц второго порядка вида:

$$\begin{pmatrix} \alpha & \beta \\ 0 & 1 \end{pmatrix}.$$

Далее группу $\mathcal{G}$ и группу матриц будем отождествлять. Единичный элемент этой группы, как и раньше, будем обозначать

$$\hat{1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Отметим, что

$$\begin{pmatrix} \alpha & \beta \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} \alpha^{-1} & \alpha^{-1} - \alpha^{-1}\beta \\ 0 & 1 \end{pmatrix}.$$

Несложно проверить, что для любых $g_1, g_2 \in \mathcal{G}$ найдется элемент $g_3 (\in \mathcal{G})$ такой, что при его действии на функцию δ_{g_1} получается функция δ_{g_2} (см. (9)). Таким образом, группа $\mathcal{G}$ действует транзитивно на базисе $B^{\mathcal{G}}$ групповой алгебры $\mathbb{F}\mathcal{G}$ по правилу (9). Рассмотрим подгруппу

$$\mathcal{H} = \{\varphi_{1,\beta} | \beta \in \mathbb{F}_{2^n}\} \tag{18}$$

группы $\mathcal{G}$. Легко видеть, что $\mathcal{H}$ – нормальная подгруппа группы $\mathcal{G}$, изоморфная группе $\Delta_{2,n}$. Пусть

$$\varsigma : \mathbb{F}_2\Delta_{2,n} \to \mathbb{F}_2\mathcal{H} \tag{19}$$

— изоморфизм групповых алгебр, индуцированный изоморфизмом групп $\varsigma' : \mathcal{H} \to \Delta_{2,n}$, определенным равенством $\varsigma'(\varphi_{1,\beta}) = \beta$. Так как $\mathcal{H}$ — нормальная подгруппа

группы $\mathrm{Aff}(\mathbb{F}_{2^n})$ и $|\mathrm{Aff}(\mathbb{F}_{2^n})/\mathcal{H}| = 2^n - 1$ не делится на $\mathrm{char}(\mathbb{F}_2) = 2$, то, согласно [8], с. 74,

$$(\mathrm{Rad}\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}))^t = \mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) \cdot (\mathrm{Rad}\mathbb{F}_2\mathcal{H})^t.$$

Таким образом, в силу изоморфизма (19) получаем:

$$\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) \otimes_{\mathbb{F}_2\mathcal{H}} \varsigma(\mathrm{RM}_t) \cong (\mathrm{Rad}\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}))^t = \mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) \cdot \varsigma(\mathrm{RM}_t).$$

Отметим, что размерность индуцированного кода $\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) \otimes_{\mathbb{F}_2\mathcal{H}} \varsigma(\mathrm{RM}_t)$ равна $(2^n - 1)\sum_{i=t}^{n} C_n^i$, длина равна $2^n(2^n - 1)$, а порождающая матрица имеет вид (16), где $N = \mathrm{RM}_t$.

3.2. Декодирование кодов RM_t и $\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) \otimes_{\mathbb{F}_2\mathcal{H}} \varsigma(\mathrm{RM}_t)$

Приведем сначала алгоритм построения декодирующих деревьев для кодов Рида–Маллера. Отметим, что код RM_n имеет вид (13), $\mathrm{RM}_1 = \mathrm{RM}_n^{\perp}$ и

$$\mathcal{M} = \left\{\delta_g - \mathbf{1} : g \in \Delta_{2,n}, g \neq \hat{1}\right\} \subseteq \mathrm{RM}_1.$$

Множество $\mathcal{M}$ является M-ортогональным относительно базисного элемента $\mathbf{1}(\in B^{\Delta_{2,n}})$, поэтому для $\mathbf{1}$ можно построить декодирующее дерево $\mathrm{WB}_{\mathbf{1},2^n-1,1}$, состоящее из двух уровней. Несложно видеть, что группа $\Delta_{2,n}$ по правилу (9) действует транзитивно на каноническом базисе $B^{\Delta_{2,n}}$ групповой алгебры $\mathbb{F}_2\Delta_{2,n}$. Код RM_t является идеалом в $\mathbb{F}_2\Delta_{2,n}$, поэтому он также является и $\mathbb{F}_2\Delta_{2,n}$-модулем, то есть действие любого элемента $\alpha(\in \mathbb{F}_2\Delta_{2,n})$ на функцию из RM_t не выводит за границы RM_t. Тогда для любого базисного элемента $\mathbf{b} \in B^{\Delta_{2,n}}$ декодирующее дерево кода RM_n можно построить по дереву $\mathrm{WB}_{\mathbf{1},2^n-1,1}$, используя алгоритм CloneTree:

$$\mathrm{WB}_{\mathbf{b},r_{\mathbf{b}},L_{\mathbf{b}}}[\mathrm{RM}_n] := \mathrm{CloneTree}(\Delta_{2,n}, \mathrm{WB}_{\mathbf{1},r_{\mathbf{1}},L_{\mathbf{1}}}[\mathrm{RM}_n], \mathbf{b}).$$

Таким образом, $\mathrm{dmaj}_{B^{\Delta_{2,n}}}(\mathrm{RM}_n) = \mathrm{dist}_{B^{\Delta_{2,n}}}(\mathrm{RM}_n) - 1$.

В [8] без обоснования показано, как для $1 \leq t < n$ и $\mathbf{1} \in B^{\Delta_{2,n}}$ декодирующее дерево RM_t может быть построено по декодирующему дереву для кода RM_{t+1}. Ниже построим соответствующий алгоритм и приведем обоснование. Нам понадобятся два простых естественных алгоритма. Первый алгоритм, MakeGenSystem, по элементу $\mathbf{p}(\in \mathbb{F}_2\Delta_{2,n})$ вида

$$\mathbf{p} = (\delta_{h_1} - \mathbf{1})...(\delta_{h_s} - \mathbf{1}), \tag{20}$$

где $S_{\mathbf{p}} = \{h_1; ...; h_s\}$ — подмножество некоторой минимальной системы порождающих элементов, достраивает $S_{\mathbf{p}}$ до, вообще говоря, другой минимальной системы порождающих $\mathcal{M}_{\mathbf{p}} = \{h_1; ...; h_s; f_1; ...; f_{n-s}\}$ и возвращает $\mathcal{M}_{\mathbf{p}} \backslash \mathcal{S}_{\mathbf{p}}$. Отметим, что для $\mathbf{p} \neq \mathbf{p}'$ в общем случае $\mathcal{S}_{\mathbf{p}} \neq \mathcal{S}'_{\mathbf{p}}$ и $\mathcal{M}_{\mathbf{p}} \neq \mathcal{M}'_{\mathbf{p}}$. Второй алгоритм, MakeGroup, по произвольному подмножеству S группы $\Delta_{2,n}$ строит подгруппу $< S >$ в виде списка.

Теперь приведем алгоритм MakeRMWBTree для построения декодирующего дерева кода RM_t с корнем $\mathbf{1}$ по декодирующему дереву кода RM_{t+1} с тем же корнем.

Теорема 2. *Алгоритм* MakeRMWBTree *по декодирующему дереву* $\mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathrm{RM}_{t+1}]$ *строит декодирующее дерево* $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$.

Исходные параметры: $t(\in\{1;...;n-1\})$, дерево $\mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathrm{RM}_{t+1}]$, каждая метка $\mathbf{p}$ которого на уровне $L_{\mathbf{1}}$ имеет вид (20), где $s=n-(t+1)+1$
Результат: дерево $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$
$\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t] := \mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathrm{RM}_{t+1}]$;
для каждой *метки* $\mathbf{p}$ *на уровне* $L_{\mathbf{1}}$ *в дереве* $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$ **выполнять**
на уровень $L_{\mathbf{1}}+1$ добавить 2^t-1 вершину и соединить их с вершиной, имеющей метку $\mathbf{p}$;
добавленным вершинам однозначно присвоить метки

$$\mathbf{q}_h := \mathbf{p}(\delta_h-\mathbf{1}),\ h\in \mathrm{MakeGroup}(\mathrm{MakeGenSystem}(\mathbf{p})),\ h\neq\hat{1}; \tag{21}$$

конец цикла
возвратить $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$

Алгоритм 8: MakeRMWBTree

Доказательство. Для доказательства теоремы достаточно показать, во-первых, что в дереве $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$ на один уровень больше, чем в $\mathrm{WB}_{\mathbf{1},2^{t+1}-1,L_{\mathbf{1}}}[\mathrm{RM}_{t+1}]$, во-вторых, что листья дерева $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$ принадлежат коду $\mathrm{RM}_t^{\perp}=\mathrm{RM}_{n-t+1}$ и, в-третьих, что в алгоритме для каждой вершины на уровне $L_{\mathbf{1}}$ строится M-ортогональное множество.

Первый факт очевиден. Вновь добавленные узлы на уровне $L_{\mathbf{1}}+1$ имеют метки, представленные в виде произведения $n-(t+1)+1+1=n-t+1$ различных функций вида $(\delta_h-\mathbf{1})$, $h\in\Delta_{2,n}$. В [8], с. 73, доказано, что код RM_t содержит все элементы вида $(\delta_{h_1}-\mathbf{1})...(\delta_{h_s}-\mathbf{1})$, где $s\geq t$ и $h_1,...,h_s\in\Delta_{2,n}$. Таким образом, метки добавленных узлов принадлежат RM_{n-t+1}. Покажем, что строятся M-ортогональные множества для вершин на уровне $L_{\mathbf{1}}$. Пусть $\mathbf{p}$ имеет вид (20), $h\in\mathcal{M}_{\mathbf{p}}\backslash\mathcal{S}_{\mathbf{p}}$. Так как поле $\mathbb{F}_2$, то $-1=1$. Следовательно,

$$\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p}(\delta_h-\mathbf{1}))=\mathrm{supp}_{B^{\Delta_{2,n}}}\mathbf{p}\delta_h+\mathbf{p}.$$

Учитывая, что $\mathcal{M}_{\mathbf{p}}$ – система порождающих группы $\Delta_{2,n}$, то

$$\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p}\delta_h)\cap\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p})=\emptyset. \tag{22}$$

Отметим, что из свойств системы порождающих следует, что равенство (22) выполняется и в случае, когда $h\in<\mathcal{M}_{\mathbf{p}}\backslash\mathcal{S}_{\mathbf{p}}>$. Поэтому для различных $h,g\in<\mathcal{M}_{\mathbf{p}}\backslash\mathcal{S}_{\mathbf{p}}>$ получим:

$$\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p}(\delta_h-\mathbf{1}))\cap\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p}(\delta_g-\mathbf{1}))=\mathrm{supp}_{B^{\Delta_{2,n}}}(\mathbf{p}).$$

Из (1) следует, что для каждой вершины на уровне $L_{\mathbf{1}}$ строится M-ортогональное множество. □

Так как RM_t — групповой MLD-код, то множество всех декодирующих деревьев $\mathcal{WB}(\mathrm{RM}_t)$ этого кода может быть построено по дереву $\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]$ с помощью алгоритма MakeAllTrees:

$$\mathcal{WB}(\mathrm{RM}_t)=\mathrm{MakeAllTrees}(\Delta_{2,n},\mathrm{WB}_{\mathbf{1},2^t-1,L_{\mathbf{1}}+1}[\mathrm{RM}_t]).$$

Рассмотрим групповой код $\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n})\otimes_{\mathbb{F}_2\mathcal{H}}\varsigma(\mathrm{RM}_t)$. Предъявим конструкцию правой трансверсали группы $\mathrm{Aff}(\mathbb{F}_{2^n})$ по подгруппе $\mathcal{H}$ вида (18). Пусть $\{\mathcal{H}g\}$ – множество классов смежности фактор–группы $\mathrm{Aff}(\mathbb{F}_{2^n})/\mathcal{H}$. Рассмотрим подгруппу

$$Y = \{m(\alpha) = \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} : \alpha \neq 0\} \tag{23}$$

группы $\mathrm{Aff}(\mathbb{F}_{2^n})$. Так как группа Y естественно изоморфна фактор–группе $\mathrm{Aff}(\mathbb{F}_{2^n})/\mathcal{H}$, то

$$\bigcup_{y\in Y}\mathcal{H}y = \mathrm{Aff}(\mathbb{F}_{2^n}).$$

Отметим, что группа $\mathrm{Aff}(\mathbb{F}_{2^n})$ не изоморфна прямой сумме $\mathcal{H}\oplus Y$. Построенная подгруппа Y является правой трансверсалью группы $\mathrm{Aff}(\mathbb{F}_{2^n})$ по подгруппе $\mathcal{H}$. В силу (14):

$$\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n}) = \bigoplus_{y\in Y}(\mathbb{F}_2\mathcal{H})y. \tag{24}$$

Таким образом, для декодирования кода $\mathbb{F}_2\mathrm{Aff}(\mathbb{F}_{2^n})\otimes_{\mathbb{F}_2\mathcal{H}}\varsigma(\mathrm{RM}_t)$ может быть применим алгоритм Decoder4.

Список литературы / References

[1] Федоренко С. В., *Методы быстрого декодирования линейных кодов*, ГУАП, СПб., 2008, 199 с.; [Fedorenko S. V., *Metody bystrogo dekodirovaniya lineynykh kodov*, GUAP, SPb., 2008, 199 pp., (in Russian).]

[2] Massey J. L., *Threshold Decoding*, MIT Press, Cambridge, 1963.

[3] Clark G. C. Jr., Cain J. B., *Error-Correction Coding for Digital Communications*, Springer, 1981., 432 pp.

[4] Сидельников В. М., "Открытое шифрование на основе двоичных кодов Рида–Маллера", *Дискретная математика*, **6**:2 (1994), 3–20; [English transl.: Sidelnikov V. M., "Open coding based on Reed–Muller binary codes", *Diskr. Mat.*, **6:2** (1994), 3–20]

[5] Берман С. Д., "К теории групповых кодов", *Кибернетика*, **3**:17 (1967), 31–39; [English transl.: Berman S. D., "On the theory of group codes", *Kibernetika*, **3**:1 (1967), 31–39]

[6] Грушко И. И., "О мажоритарном декодировании обобщенных кодов Рида–Маллера", *Пробл. передачи информ.*, **26**:3 (1990), 12–20; [English transl.: Grushko I. I., "Majority-logic decoding of generalized Reed-Muller codes", *Probl. Inf. Transm.*, **26**:3 (1990), 189–196]

[7] Логачев О. А., Ященко В. В., "Коды типа Рида–Маллера на конечной абелевой группе", *Пробл. передачи информ.*, **34**:2 (1998), 32–46; [English transl.: Logachev O. A., Yashchenko V. V., "Codes of the Reed-Muller type on a finite abelian group", *Probl. Inf. Transm.*, **34**:2 (1998), 121–133]

[8] Циммерман К.-Х., *Методы теории модулярных представлений в алгебраической теории кодирования*, МЦНМО, М., 2011, 246 с.; (Tsimmerman K.-Kh., *Metody teorii modulyarnykh predstavleniy v algebraicheskoy teorii kodirovaniya*, MTsNMO, M., 2011, 246 pp., [in Russian].)

[9] Деундяк В. М., Косолапов Ю. В., "О стойкости кодового зашумления к статистическому анализу наблюдаемых данных многократного повторения", *Моделирование и анализ информационных систем*, **19**:4 (2012), 110—127; (Deundyak V. M., Kosolapov Yu. V., "On the Firmness Code Noising to the Statistical Analysis of the Observable Data of Repeated Repetition", *Modeling and Analysis of Information Systems*, **19**:4 (2012), 110—127, [in Russian].)

[10] Косолапов Ю. В., "Коды для обобщенной модели канала с подслушиванием", *Проблемы передачи информации*, **51**:1 (2015), 23—28; [English transl.: Kosolapov Yu. V., "Codes for a Generalized Wire-Tap Channel Model", *Problems of Information Transmission*, **51**:1 (2015), 20—24]

[11] Деундяк В. М., Мкртичян В. В., "Исследование границ применения схемы защиты информации, основанной на РС-кодах", *Дискретный анализ и исследование операций*, **18**:3 (2011), 21—38; (Deundyak V. M., Mkrtichyan V. V., "Issledovanie granits primeneniya skhemy zashchity informatsii, osnovannoy na RS-kodakh", *Diskretnyy analiz i issledovanie operatsiy*, **18**:3 (2011), 21—38, [in Russian].)

[12] Сидельников В. М., *Теория кодирования*, ФИЗМАТЛИТ, 2011, 323 с.; (Sidel'nikov V. M., *Teoriya kodirovaniya*, FIZMATLIT, 2011, 323 pp., [in Russian].)

[13] Деундяк В. М., Дружинина М. А., Косолапов Ю. В., "Модификация криптоаналитического алгоритма Сидельникова–Шестакова для обобщенных кодов Рида–Соломона и ее программная реализация", *Известия высших учебных заведений. Северо-Кавказский регион*, Технические науки, 2006, 15–20; (Deundyak V. M., Druzhinina M. A., Kosolapov Yu. V., "Modifikatsiya kriptoanaliticheskogo algoritma Sidel'nikova–Shestakova dlya obobshchennykh kodov Rida–Solomona i ee programmnaya realizatsiya", *Izvestiya vysshikh uchebnykh zavedeniy. Severo-Kavkazskiy region*, Tekhnicheskie nauki, 2006, 15–20, [in Russian].)

[14] Minder L., Shokrollahi A., "Cryptanalysis of the Sidelnikov cryptosystem", *Eurocrypt 2007 of Lecture Notes in Computer Science*, **4515** (2007), 347—360.

[15] Зиновьев В. А., Зяблов В. В., "Коды с неравной защитой информационных символов", *Проблемы передачи информ.*, **15**:3 (1979), 50–60; [English transl.: Zinovev, V. A., Zyablov V. V., "Codes with unequal protection of information symbols", *Probl. Inf. Transm.*, **15**:15 (1979), 197–205]

[16] Curtis C. W., Reiner I., *Representation Theory of Finite Groups and Associative Algebras*, Intersclence Publishers, New York, 1962.

[17] Логачев О. А., Сальников А. Ю., Ященко В. В., *Булевы функции в теории кодирования и криптологии*, МЦМНО, М., 2004, 470 с.; [English transl.: Logachev O. A., Salnikov A. A., Yashchenko V. V., *Boolean Functions in Coding Theory and Cryptography*, AMS. Translations of Mathematical Monographs, 2012, 334 pp.

DOI: 10.18255/1818-1015-2015-4-464-482

Algorithms for Majority Decoding of Group Codes

Deundyak V. M., Kosolapov Y. V.

Received April 29, 2015

We consider a problem of constructive description and justification of the algorithms necessary for a practical implementation of the majority decoder for group codes specified as left ideals of group algebras. In addition to the algorithms needed to implement a classical decoder of J. Massey, it is built a generalization of the classical decoder for codes with unequal protection of characters, which in some cases could be better than the classic one. For use as a classical decoder of J. Massey and its generalization to group codes it was developed an algorithm for constructing decoding trees that lie at the core of these algorithms for majority decoding. Because group codes are defined as left ideals of group algebras, the decoding algorithm for constructing decoding trees allows to build all decoding trees from one tree. On the basis of the generalized decoding algorithm it was developed an algorithm for decoding group codes induced on the subgroup. Application of the developed decoders was illustrated by an example of Reed-Muller-Berman codes and group codes induced by them on a non-Abelian group of affine transformations. In particular, for Reed–Muller–Berman code description and justification of the algorithm for constructing one decoding tree are provided. This three is used for constructing all decoding trees and then it is a built decoder for Reed-Muller-Berman codes and codes induced by them.

Keywords: majority decoder, group algebra, group codes, Reed–Muller–Berman Codes

For citation: Deundyak V. M., Kosolapov Y. V., "Algorithms for Majority Decoding of Group Codes", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 464–482.

On the authors: Deundyak V. M., orcid.org/0000-0001-8258-2419, PhD,
FGNU NII "Specvuzavtomatika",
51 Gazetniy lane, Rostov-on-Don, 344002, Russia, e-mail: vlade@math.rsu.ru,

Kosolapov Y. V., orcid.org/0000-0002-1491-524X, PhD,
South Federal University,
105/42 Bolshaya Sadovaya Str., Rostov-on-Don, 344006, Russia, e-mail: itaim@mail.ru

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 483–499
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 483–499

DOI: 10.18255/1818-1015-2015-4-483-499

УДК 512.547.214

О конечных группах с большой степенью неприводимого характера

Казарин Л. С.[1], Поисеева С. С.

получена 6 июля 2015

Пусть G – конечная неединичная группа с неприводимым комплексным характером χ степени d. Согласно соотношениям ортогональности для неприводимых характеров, сумма квадратов степеней этих характеров равна порядку группы G. Н. Снайдером доказано, что если $|G| = d(d+e)$, то порядок группы G ограничен функцией e при $e > 1$. Я. Берковичем доказано, что в случае $e = 1$ группа G является группой Фробениуса с дополнением порядка d.

В данной работе изучается конечная неединичная группа G, обладающая неприводимым комплексным характером Θ, для которого $|G| \leq 2\Theta(1)^2$. Доказано, что в случае, когда $\Theta(1)$ – произведение двух различных простых чисел p и q, группа G является разрешимой группой с абелевой нормальной подгруппой K индекса pq. С помощью классификации простых конечных групп доказано, что простая неабелева группа, порядок которой делится на простое число p и не превышает $2p^4$, изоморфна одной из следующих групп: $L_2(q), L_3(q), U_3(q), Sz(8), A_7, M_{11}, J_1$.

Ключевые слова: конечная группа, характер конечной группы, степень неприводимого характера конечной группы

Для цитирования: Казарин Л. С., Поисеева С. С., "О конечных группах с большой степенью неприводимого характера", *Моделирование и анализ информационных систем*, **22**:4 (2015), 483–499.

Об авторах:
Казарин Лев Сергеевич, orcid.org/0000-0003-1836-0955, д-р физ.-мат. наук, профессор, зав. кафедрой алгебры и мат. логики, Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: kazarin@uniyar.ac.ru

Поисеева Саргылана Семеновна, orcid.org/0000-0003-2740-9845, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: pss.iii@mail.ru

Благодарности:
[1]Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 13-01-00469).

Введение

Пусть G – конечная группа, обладающая неприводимым представлением над полем комплексных чисел с характером Θ. Согласно соотношениям ортогональности для неприводимых характеров (глава 2 в [1]), сумма квадратов степеней этих характеров равна порядку группы. В частности, $\Theta(1)^2 < |G|$. В общем случае порядок группы существенно больше квадрата степени любого ее неприводимого характера. Однако, согласно [2], у любой простой неабелевой группы G имеется неприводимый характер степени, большей $|G|^{1/3}$. Н. Снайдером в [3] изучались группы, с неприводимым характером степени d, для которого $|G| = d(d + e)$. Им было доказано, что в этом случае порядок группы G ограничен функцией от e при $e > 1$. В случае $e = 1$ группа G является группой Фробениуса. Я. Берковичем в [4] классифицированы группы при $e = 1$ и $e = 2$. Для $e > 1$ М. Айзекс в [5] показал, что $|G| \leq Be^6$ для некоторой постоянной B. К. Дюрфи и С. Дженсен в [6] доказали, что $|G| \leq e^6 - e^4$. А по М. Льюису, $d \leq e^2 - e$ и $|G| \leq e^4 - e^3$ – наилучшее возможное ограничение [7].

Определение 1. *Конечную группу G порядка больше двух, обладающую неприводимым характером Θ, таким, что, $2\Theta(1)^2 \geq |G|$, будем называть $LC(\Theta)$-группой.*

Цель настоящей статьи – изучение конечных $LC(\Theta)$-групп с $\Theta(1) = pq$, где p и q – различные простые числа и $p > q$.

Так как $|G| \leq 2p^2q^2 < 2p^4$, то первым шагом в изучении $LC(\Theta)$-групп с нашим условием будет описание простых неабелевых групп G, обладающих силовской p-подгруппой порядка p, для которых $|G| \leq 2p^4$.

Заметим, что группы с силовской подгруппой порядка p изучались в серии работ Р. Брауэра и его учеников, начиная с 40-х годов прошлого века, а позднее в работах У. Фейта (см. библиографию в книге [8]). Используя классификацию конечных простых групп, получен следующий результат.

Теорема 1. *Пусть G – конечная простая неабелева группа с силовской p-подгруппой простого порядка p. Если $|G| \leq 2p^4$, то G изоморфна одной из следующих групп: $L_2(q), q > 3, L_3(q), U_3(q)$ (q – степень простого числа), $Sz(8), A_7, M_{11}, J_1$.*

Описание простых групп порядка $< p^3$ с силовской p-подгруппой порядка p получено в работе Брауэра и Туана (теорема VIII 5.1 и ее следствие в [8]). Для групп $L_3(q)$ соответствующее простое число $p = q^2 + q + 1$ может возникнуть при $q \not\equiv 1(mod\ 3)$, а для $U_3(q)$ такое простое число $p = q^2 - q + 1$ может появиться лишь при $q \not\equiv -1(mod\ 3)$. Для группы $Sz(8)$ будет $p = 13$, для A_7 $p = 7$, для M_{11} $p = 11$ и для J_1 $p = 19$.

Следующая теорема показывает, что в случае $\Theta(1) = pq$, где $p > q$ – простые числа, $LC(\Theta)$-группа будет p-разрешимой.

Теорема 2. *Пусть G является $LC(\Theta)$-группой. Если $\Theta(1) = pq$, где p и q – различные простые числа, то G – p-разрешимая группа.*

В доказательстве теоремы 2 также установлено, что при $|G| \notin \{54, 72\}$ порядок $|G|$ равен pq^bm, где $(pq, m) = 1$. Следующая теорема дает описание $LC(\Theta)$-групп.

Теорема 3. *Пусть G является $LC(\Theta)$-группой с $\Theta(1) = pq$, где $p > q$ и p, q – простые числа. Тогда G имеет абелеву нормальную подгруппу K индекса pq.*

Прямое произведение знакопеременной группы A_4 и симметрической группы S_3 имеет неприводимый характер Θ степени 6, так что $|G| = 2 \cdot 6^2$.

В общем случае задача описания строения группы, обладающей неприводимым характером Θ, таким, что $|G| \leq c\Theta(1)^2$ для небольшой константы c представляется довольно сложной. Таким характером при $c < 2,6$ обладает спорадическая группа Томпсона, при $c < 3,1$ все пять групп Матье, при $c < 4,4$ группа Янко порядка 175560 и группа Хигмана–Симса.

Все группы предполагаются конечными. Буква p везде используется для обозначения простого числа. Необходимые сведения, касающиеся обыкновенных и модулярных представлений конечных групп, можно найти в [9] и [1]. Для любого элемента a группы G через h_a обозначается число элементов группы G, сопряженных с a. Скалярное произведение характеров χ и ψ группы G обозначается $\langle \chi, \psi \rangle$. В частности, если эти характеры неприводимы, то $\langle \chi, \psi \rangle = \delta_{\chi,\psi}$. Множество неприводимых характеров группы G обозначается $Irr(G)$.

1. Вспомогательные результаты

Лемма 1. *(Клиффорд) Пусть $N \lhd G$ и $\chi \in Irr(G)$, а $\theta \in Irr(N)$. Тогда из $\langle \chi_N, \theta \rangle_N \neq 0$ следует, что*

$$\chi_N = e \sum_{i=1}^{t} \theta_i,$$

где θ_i – характеры, сопряженные θ. Число t различных сопряженных характеров к характеру θ равно индексу подгруппы инерции $I_G(\theta)$ характера θ (состоящей из всех $g \in G$, для которых $\theta^g = \theta$), а число $e = e(\chi)$ делит $|I_G(\theta) : N|$, если G/N – разрешимая группа.

Доказательство. См. теорему 6.2 из [1].

Лемма 2. *(Ито) Пусть $N \lhd G$ и N-абелева. Тогда $\chi(1) || G : N |$, для всех $\chi \in Irr(G)$.*

Доказательство. См. теорему 6.15 из [1].

Лемма 3. *Пусть $G = G'$– группа с силовской p–подгруппой P порядка $p > 3$ и $H = O_{p'}(G)$, причем число силовских p–подгрупп группы G равно $1 + np$. Тогда верно одно из следующих утверждений:*

(i) $G/H \simeq L_2(r)$, где либо $r = p$, либо $r = 2^a = p - 1$:

(ii) $n \geq (p + 3)/2$.

Доказательство. См. теорему 5.1, глава VIII из [8].

Лемма 4. *Пусть $G = G'$ – не p–разрешимая группа с силовской p–подгруппой P порядка $p > 3$, не имеющая нормальных подгрупп порядка 2. Если $|G| < p^3$, то $G \simeq L_2(r)$, где $r = p$ или $r = 2^a = p - 1$.*

Доказательство. См. следствие 5.2, глава VIII из [8].

Лемма 5. *(Брауэр и Туан). Пусть G – простая группа порядка pq^bm, где p, q – простые числа, b, m – натуральные числа и $m < p - 1$. Тогда $G \simeq L_2(r)$, где $q = 2$ и либо $r = p = 2^a \pm 1$, либо $r = 2^a = p - 1$.*

Доказательство. См. теорему 5.4, глава VIII из [8].

Лемма 6. *Степени неприводимых характеров группы $PGL_2(q)$, где q нечетно, $q > 3$ и группы $L_2(q), q = 2^a, a > 1$, состоят из чисел $1, q, q-1, q+1$.*

Доказательство. См. [10] и [9], с. 260.

Лемма 7. *Пусть G – конечная простая группа. Если силовская 2-подгруппа группы G абелева, то G изоморфна $L_2(q)$,$^2G_2(q)$ или J_1. Если силовская 2-подгруппа группы G имеет нетривиальный циклический коммутант, то G изоморфна группе $L_2(q), L_3(q), U_3(q) (q$ нечетно), A_7 или M_{11}.*

Доказательство. Содержится в [11] и в [12].

Лемма 8. *Пусть P – силовская p-подгруппа группы G, имеет порядок p. Если подгруппа H группы G содержит $N_G(P)$, то $|G : H| \equiv 1(mod\ p)$.*

Доказательство. Пусть $x \in G \setminus H$. Если $|H : H \cap H^x| \not\equiv 0(mod\ p)$, то найдется силовская p-подгруппа P_1 группы H, содержащаяся в $H \cap H^x$. По теореме Силова $P_1 = P^h$ для некоторого $h \in H$. Так как $P_1 \subseteq H^x$, то $P_1^{x^{-1}} \subseteq H$. Отсюда $P_1^{x^{-1}} = P_1^{h_1}$ для подходящего $h_1 \in H$. Следовательно, $h_1 x \in N_G(P) \leq H$ и $x \in H$ вопреки предположению. Значит, $|H : H \cap H^x| \equiv 0(mod\ p)$ для любого $x \in G \setminus H$. Так как $G = \cup_{y \in G} HyH$, то $|G| = |H| + pk|H|$ для некоторого целого неотрицательного k, что и доказывает лемму.

Лемма 9. *Пусть G – $LC(\Theta)$–группа и M – ее собственная нормальная подгруппа. Тогда характер Θ_M приводим.*

Доказательство. Допустим, что M нормальная в G и Θ_M неприводим. Тогда $\Theta(1)^2 \leq |M| - 1 < |M|$. Однако $2\Theta(1)^2 \geq |G| \geq 2|M|$, противоречие. Лемма доказана.

Лемма 10. *Пусть G – $LC(\Theta)$–группа и N – ее собственная нормальная подгруппа. Если $\Theta(1) = m$, то $(|G/N|, m) \neq 1$.*

Доказательство. Допустим, что N нормальная в G. По лемме 9 характер Θ_N приводим. Согласно лемме 1, $\Theta_N = e\sum_{i=1}^{s} \phi_i$, где ϕ_i – неприводимые характеры группы N, сопряженные с характером ϕ_1, а e и s делят $|G/N|$. Поэтому $m = \Theta(1) = es\phi_1(1)$. Если $(|G/N|, m) = 1$, то $\Theta_N = \phi_1 \in Irr(N)$. Однако $\Theta(1)^2 < |N| \leq |G|/2$, противоречие. Следовательно, $(|G/N|, m) \neq 1$. Лемма доказана.

2. Доказательство теоремы 1

Согласно теореме классификации конечных простых неабелевых групп (см. [13] для более подробной информации), все простые неабелевы группы принадлежат одному из следующих семейств:

I. Классические простые группы лиева типа: $L_n(q), n \geq 2, U_n(q), n \geq 3, S_{2n}(q), n \geq 2$, $P\Omega_{2n+1}(q), n \geq 2$, $P\Omega^{\pm}_{2n}, n \geq 4$.

II. Исключительные простые группы лиева типа: $G_2(q)$, $F_4(q)$, $E_6(q)$, $E_7(q)$, $E_8(q)$, $^2G_3(q)$, $(^2F_4(q))'$, $^3D_4(q)$, $^2B_2(q)$, q – степень простого числа.

III. Спорадические простые группы.

IV. Знакопеременные группы $A_n, n \geq 7$.

Доказательство будем проводить шаг за шагом для всех перечисленных групп.

I. Классические простые группы лиева типа.

(a) Пусть $G \simeq L_n(q), n \geq 3$. Легко видеть, что $|G| > q^{n^2-n}$, а наибольший простой делитель порядка группы не превосходит $q^n - 1 < q^n$. Допустим, что $2p^4 > q^{n^2-n}$. Тогда $2q^{4n} > q^{n^2-n}$ и потому $n^2 - 5n < 1$. Отсюда $n \leq 5$. При $n = 5$ имеем более точную оценку $|G| > q^{21} \geq 2q^{20}$. Поэтому $n \leq 4$. При $n = 4$ имеем:

$$|G| = 1/dq^6(q^4 - 1)(q^3 - 1)(q^2 - 1), \text{ где } d = (4, q - 1)$$

Теперь наибольший простой делитель порядка группы не превосходит $q^2 + q + 1 < 2q^2$. Если $2p^4 \geq |G| > q^{13}$, то $2(2q^2)^4 > q^{13}$, что исключает этот случай.

(b) Пусть $G \simeq U_n(q), n \geq 3$. Легко видеть, что $|G| > q^{n^2-n}$, а наибольший простой делитель порядка группы не превосходит q^n. Допустим, что $2p^4 > q^{n^2-n}$. Тогда $2q^{4n} > q^{n^2-n}$ и потому $n^2 - 5n < 1$. Отсюда $n \leq 5$. При $n = 5$ имеем более точную оценку $|G| > q^{21} \geq 2q^{20}$. Поэтому $n \leq 4$. При $n = 4$ имеем:

$$|G| = 1/dq^6(q^4 - 1)(q^3 + 1)(q^2 - 1), \text{ где } d = (4, q + 1)$$

Теперь наибольший простой делитель порядка группы (при $q > 2$) не превосходит $q^2 - q + 1 < 2q^2$. Если $2p^4 \geq |G| > q^{13}$, то $2(2q^2)^4 > q^{13}$, что исключает этот случай. Случай $q = 2$ также невозможен.

(c) Пусть $G \simeq S_{2n}(q)$ или $P\Omega_{2n+1}(q), n \geq 2$. Порядок G равен $1/dq^{n^2}(q^{2n} - 1)...(q^2 - 1)$, где $d = (2, q - 1)$. Если $n \geq 2$, то $|G| > q^{2n^2}$. Наибольший простой делитель порядка G не превосходит q^n+1. Легко видеть, что $2(q^n+1)^4 < q^{4n+3} < q^{2n^2}$ для $(q, n) \neq (2, 2)$. При $(q, n) = (2, 2)$ группа $G \simeq S_4(2)$ не проста и $S_4(2)' \simeq L_2(9)$.

(d) Пусть $G \simeq P\Omega^{\pm}_{2n}(q), n \geq 4$. Легко видеть, что $|G| > q^{2n^2-2n}$, а наибольший простой делитель ее порядка не превосходит $q^n + 1$. Так как $2(q^n + 1)^4 < q^{4n+3}$ при $n \geq 4$, то $2p^4 < |G|$.

I. Исключительные простые группы лиева типа.

(a) Пусть $G \simeq G_2(q)$. Порядок G равен $q^6(q^6 - 1)(q^2 - 1)$, так что наибольший простой делитель его не превосходит $q^2 + q + 1 < 2q^2$. Так как $|G| > q^{13}$, то этот случай исключен.

(b) Пусть $G \simeq F_4(q)$ порядка $q^{24}(q^{12} - 1)(q^8 - 1)(q^6 - 1)(q^2 - 1)$. Наибольший простой делитель порядка G не превосходит $q^4 + 1$. Так как $|G| > 2(q^4 + 1)^4$, то этот случай исключен.

(c) Пусть G – одна из групп $E_6(q), E_7(q), E_8(q)$ или ${}^2E_6(q)$. Порядок любой из этих групп больше q^{72}, а наибольший простой делитель порядка не превосходит q^9. Так что эти группы также исключены.

(d) Пусть $G \simeq^2 G_2(q)$. Порядок этой группы равен $q^3(q^3 + 1)(q - 1) > q^6$. При этом $q = 3^{2n+1}$, а наибольший простой делитель p порядка группы не превосходит $q + \sqrt{3q} + 1 < 2q$. Если $q > 3$, то $|G| > 2p^4$. Если же $q = 3$, то $G \simeq L_2(8).3$ – не простая группа.

(e) Пусть $G \simeq ({}^2F_4(q))'$. Ее порядок больше, чем $2q^{16}$, тогда как наибольший простой делитель p порядка группы не превосходит q^4-q^2+1. Этот случай исключен.

Пусть $G \simeq^3 D_4(q)$ порядка $q^{12}(q^8 + q^4 + 1)(q^6 - 1)(q^2 - 1) > q^{26}$, тогда как наибольший простой делитель p не превосходит $q^4 + q^2 + 1 < 2q^5$. Поэтому и этот случай исключен.

Пусть $G \simeq^2 B_2(q) \simeq Sz(q)$, где $q = 2^{2n+1}$. Наибольший простой делитель p порядка G не превосходит $q + \sqrt{2q} + 1$, так что $|G| > 2p^4$ при $q > 8$. Случай $q > 8$ исключен.

III. Спорадические простые группы. Все случаи рассматриваются с помощью [14]. Единственные возможности для p соответствуют группе Матье M_{11} и группе Янко J_1.

IV. Знакопеременные группы $A_n, n \geq 7$.

Случаи $5 \leq n \leq 6$ соответствуют группам $L_2(5)$ и $L_2(9)$. При $n = 7$ появляется группа A_7, где $p = 7$. Все остальные возможности исключены.

Теорема доказана.

3. Доказательство теоремы 2

Напомним, что G – $LC(\Theta)$–группа с $\Theta(1) = pq$, где p и q различные простые числа и $p > q$. Зафиксируем данные обозначения, которые будут употребляться на протяжении оставшейся части работы.

В силу равенства $\sum_{\chi \in Irr(G)} \chi(1)^2 = |G|$ и $2\chi(1)^2 \geq |G|$ характер Θ является единственным неприводимым характером группы G наибольшей степени. Поэтому все сопряженные с ним под действием группы Галуа расширения $\mathbb{Q}(\Theta)$ поля рациональных чисел $\mathbb{Q}$ с помощью присоединения к нему элементов $\Theta(g)$ для всех $g \in G$, совпадают с Θ. Отсюда все значения $\Theta(g)$, $g \in G$ являются целыми рациональными числами. В частности, $\overline{\Theta(g)} = \Theta(g)$ для всякого $g \in G$. В любом случае Θ является точным характером.

В [15] авторами доказано, что любой неприводимый характер $LC(\Theta)$-группы является конституентой характера $\Phi = \Theta^2$. Отметим, что $Z(G) = 1$ и Θ – точный характер.

Так как, $|G| \leq 2p^2q^2$, где p и q – различные простые числа и $p > q$, причем $p^2q^2 < |G| \leq 2p^2q^2 < 2p^4$, а поскольку $pq||G|$, то $|G| = p^aq^bm$, $a, b, m \in \mathbb{N}$ т.е. $p^2q^2 < p^aq^bm \leq 2p^2q^2 < 2p^4$. И так как $p > 2$, то p^4 не делит $|G|$. Отсюда, $a \leq 3$, т.е. силовская p-подгруппа группы G имеет порядок не больше p^3. Допустим, что $P \in Syl_p(G)$.

Начнем со случая, когда порядок силовской подгруппы равен p^3, т.е. $a = 3$.

Лемма 11. *Пусть G – $LC(\Theta)$–группа и силовская p–подгруппа P имеет порядок p^3. Тогда $p = 3$, $q = 2$ и $|G| = 54$.*

Доказательство. Так как $|P| = p^3$, то $|N_G(P)| = p^3n$, при $n \in \mathbb{N}$ и $|G| = p^3n(1 + kp) \leq 2p^2q^2$, k – целое неотрицательное, откуда $pn(1 + kp) \leq 2q^2$.

При $k \neq 0$: $n(1 + kp) < 2q$. Если $n \geq 2$, то $1 + kp < q$, что неверно. Значит, $n = 1$.

Пусть $k \geq 1$. $|G| = p^3(1 + kp) \leq 2p^2q^2$, причем q делит $1 + kp$. Очевидно, что $k = 1$ и q делит $p + 1$, причем $q < p$. При этом $q \leq \frac{p+1}{2}$. Значит, $p(1 + p) \leq 2\frac{(p+1)^2}{4}$, что неверно, поэтому $k = 0$ и $P \triangleleft G$. При этом $|G| = p^3qm \leq 2p^2q^2$, так, что $pm \leq 2q$. Так как $p > q$, то $m = 1$ и $p \leq 2q$.

Итак, $|G| = p^3q$ и $P \triangleleft G$. Если P абелева, то по лемме 2 степень характера $\chi(1)$ делит q для любого $\chi \in Irr(G)$. Противоречие.

Отсюда P неабелева и $P/\Phi(P)$ – элементарная абелева, $|P/\Phi(P)| = p^2$. Следовательно, $N_G(P)/P \leq Aut(P/\Phi(P)) \leq GL_2(p)$ и q делит $p \pm 1$.

Если q делит $p-1$, то $q \leq \frac{p-1}{2}$, а тогда $p \leq \frac{2(p-1)}{2}$, что невозможно, либо $q = p-1 = 2$, а тогда $p \leq 2q = 4$. Значит, $p = 3, q = 2$ и $|G| = 3^3 \cdot 2 = 54$

Пусть $q = \frac{p+1}{2}$. Тогда либо $q = 2, p = 3, |G| = 54$, либо $q = \frac{p+1}{2}$ нечетное простое число. В последнем случае $G/\Phi(P)$–группа Фробениуса порядка $p^2\frac{(p+1)}{2}$.

Так как в этом случае $Z(P) = \Phi(P)$ содержится в центре группы G, то получаем противоречие.

То есть $|P| = p^3$ возможно лишь при $p = 3, q = 2$ и $|G| = 54 = 3^3 \cdot 2$. Лемма доказана.

Все случаи при $a = 2$, т.е. когда силовская подгруппа имеет порядок p^2, сформулируем в виде следующей леммы.

Лемма 12. *Если G – $LC(\Theta)$–группа, то $|G| \not\equiv 0(mod\ p^2)$ или $|G| = 72$ либо $|G| = 54$.*

Доказательство. Случай, когда порядок силовской p-подгруппы равен p^3, рассмотрен в лемме 11. Пусть силовская p-подгруппа $LC(\Theta)$-группы G имеет порядок p^2 и $\Theta(1) = pq$. Тогда $|G| = p^2qm \leq 2p^2q^2$, откуда $m \leq 2q < 2p$. В свою очередь силовская q-подгруппа группы G либо имеет порядок $\leq q$, либо $q = 2$ и $|G| \leq 8p^2$, либо $|G| = p^2q^2$ или $2p^2q^2$.

Возможны подслучаи:

(1) $|G| \leq 8p^2$ и $\Theta(1) = 2p$, $q = 2$: Учитывая, что силовская p–подгруппа P группы G абелева, а пересечение любых двух силовских подгрупп нетривиально, имеем $O_p(G) > 1$. Если силовская p–подгруппа группы G нормальна, то степени любого неприводимого характера не делятся на p (следствие Ито 12.34 в [1]). Но тогда $n(1+kp) \leq 8$, где $n = |N_G(P) : P| > 1$ (теорема Бернсайда 7.1.1. в [16]). Значит, $1 + kp \leq 4$ и $kp \leq 3$, откуда $p = 3, k = 1$. В таком случае $|G| = 72$.

(2) $|G| = 2p^2q^2$, $(2, pq) = 1$: Если $|G : N_G(P)| \equiv 1(mod\ p)$, то $|G : N_G(P)| = 2q$ и $|N_G(P)| = p^2q$. Так как $1 + 2p > 2q$, то $1 + p = 2q$, т.е., $q = \frac{p+1}{2}$.

С другой стороны, $O_p(G) \neq 1$, ибо любые p–силовские подгруппы абелевы и имеют нетривиальное пересечение. Значит, $O_p(G) \cong C_p$-циклическая порядка p. Из неабелевости $N_G(P)$ следует, что q делит $p-1$. Отсюда $q = 2$, а это случай (1).

(3) $|G| = p^2q^2$: Пусть $P \in Syl_p(G), Q \in Syl_q(G)$. Если $N_G(P) = P$, то G – p–нильпотентна и $G = Q \rtimes P$. По лемме 2 степень характера $\chi(1)|p^2$, что неверно.

Если $N_G(P) = G$, то $G = P \rtimes Q$ и $\chi(1)|q^2$ для любого неприводимого χ, противоречие. Значит, $|N_G(P)| = p^2q$ и $|G : N_G(P)| = q \not\equiv 1(mod\ p)$. Таким образом, случай (3) невозможен.

(4) $|G| = p^2qm$, где $(m, pq) = 1$: Тогда $m \leq 2q$. В то же время $|N_G(P)| = np^2$, где q делит $n(1+kp) = qm$. Если q делит n, то $1 + kp \leq m$, что невозможно при $k > 1$. Значит, $k = 1$ и $n = q$. А тогда $m = 1 + p \leq 2q$. Как и в случае (3) q делит $p-1$. Если $q \leq \frac{p-1}{2}$, то $2\frac{p-1}{2} \geq 2q > m = 1 + p$ и $p - 1 = q$, так, что $q = 2, p = 3$. Отсюда $|G| = 9 \cdot 2 \cdot 4 = 72$, а это случай (1).

Теперь q делит $1 + kp$ и $(n, q) = 1$. Так как $O_p(G) \cong C_p = \langle x \rangle$, то $|G : C_G(x)|$ делит $|Aut(C_p)| = p - 1$.

По лемме 1 имеем $\Theta|_{C_G(x)} = e\sum_{i=1}^{s} \varphi_i$, где $\varphi_i \in Irr(C_G(x))$ и поэтому $\Theta(1) = pq = e\varphi_i(1)s$.

Так как $P \leq C_G(\langle x \rangle)$, $C_G(\langle x \rangle) \lhd G$, то по аргументу Фраттини $G = C_G(x)N_G(P)$. Поэтому $\frac{|C_G(x)||N_G(P)|}{|C_G(x) \cap N_G(P)|} = |G|$ и $|G : N_G(P)| = 1 + kp = |C_G(x) : C_G(x) \cap N_G(P)|$.

Поэтому q делит $|C_G(x) : C_G(x) \cap N_G(P)|$ и не делит $|N_G(P)|$. Но тогда q не делит $|G : C_G(x)|$. По лемме 1 имеет место разложение $\Theta_H = c\sum_{i=1}^{s} \varphi_i$, где φ_i – неприводимые сопряженные с $\varphi = \varphi_1$ характеры группы H и $c = \langle \Theta_H, \varphi_i \rangle$.

При этом s делит $|G : C_G(x)||p-1$ и c делит $p-1$.

Отсюда $pq = cs\varphi(1)$. Так как p не делит cs и q не делит cs, то $pq = \varphi(1)$ и $cs = 1$, т.е. $\varphi|_{C(x)}$ неприводимый характер группы $C_G(x)$. В этом случае, $|\Theta(x)| = pq$ противоречие. Лемма доказана.

Далее будем считать, что $a = 1$, т.е. $|G| = pq^b m$, где m взаимно просто с pq.

Лемма 13. *Пусть G – простая $LC(\Theta)$–группа порядка $|G| = pq^b m$, где $p > q$, $(m, pq) = 1$. Тогда $b \leq 2$ и $q > 2$.*

Доказательство. Так как $\Theta(1) = pq$ и $|G| = pq^b m \leq 2\Theta(1)^2$, то при $b \geq 3$ получим $m < 2p/(q^{b-2}) < p-1$, когда $q > 2$ или $b \geq 4$. Если же $q = 2$ и $b \leq 3$, то силовская 2-подгруппа группы G имеет порядок не более 8 и потому либо абелева, либо диедральна. По лемме 7 группа G изоморфна одной из следующих групп: $L_2(q)$,[2] $G_2(q)$, J_1, A_7. Остальные группы из списка этой леммы имеют силовскую 2-подгруппу порядка не меньше 16. Используя леммы 5, 6, порядки групп ${}^2G_2(q)$ и таблицы характеров групп J_1 и A_7 из [9], получаем противоречие. Поэтому $b \leq 2$ и $q > 2$. Лемма доказана.

Лемма 14. *Пусть G - $LC(\Theta)$–группа порядка $|G| = pq^b m$, где $p > q, (m, pq) = 1$. Если $|G| < p^3$, то $G \neq G'$.*

Доказательство. Допустим, что $G = G'$. Так как G является $LC(\Theta)$–группой, то $Z(G) = 1$. В частности, G не имеет нормальных подгрупп порядка 2. По следствию VIII (5.2) из [8] группа G изоморфна $L_2(r)$, где $q = 2$, $r = p$ или $r = p-1 = 2^a$, а эта группа не является $LC(\Theta)$–группой. Лемма доказана.

Лемма 15. *Пусть G – не p–разрешимая $LC(\Theta)$–группа порядка $|G| = pq^b m$, где $p > q, (m, pq) = 1$. Тогда $|G| > p^3$.*

Доказательство. Из леммы 14 известно, что $G \neq G'$. Предположим, что $H = O_{p'}(G)$ и G не p–разрешимая группа. Пусть M/H – минимальная нормальная подгруппа группы G/H. Тогда M/H не p–разрешимая и совпадает своим коммутантом. Так как силовская p–подгруппа группы G порядка p, то M/H – простая неабелева группа. По лемме 4 имеем $M/H \simeq L_2(r)$, где либо $r = p$, либо $r = p-1 = 2^a$. Допустим, что $M/H \simeq L_2(p)$. Тогда $G/M \simeq L_2(p)$ или $PGL_2(p)$. При $p > 3$ и $|H| \geq 3$ порядок G больше p^3. Тем самым, эта возможность исключена. Если $M/H \simeq L_2(p-1)$, то $|M/H| = p(p-1)(p-2) = p^3 - 3p^2 + 2p > 1/2p^3$ при $p \geq 7$ и $|G| > p^3$ для G, не являющейся простой группой. Если $p = 5$, то $|G/M||H| < 125/60$ только при $|H| = 2$, что уже было исключено выше, либо $G \simeq Aut(L_2(5)) \simeq S_5$. Таблица характеров в [9] исключает и эту возможность. Поэтому $|G| > p^3$. Лемма доказана.

Лемма 16. *Пусть G – не p-разрешимая $LC(\Theta)$–группа порядка $|G| = pq^b m$, где $p > q$, $(m, pq) = 1$. Тогда $b \leq 2$.*

Доказательство. В силу леммы 15 и условий, наложенных на группу G, имеем:

$$p^3 < |G| = pq^b m \leq 2p^2 q^2,$$

откуда $p < 2q^2, q^{b-2} \leq 2p/m$ и $q^{b-4} < 4/m$. При $m = 1$ или 2 и $b \geq 3$ группа G разрешима. Поэтому $m \geq 3$, откуда $b \leq 4$.

При $b = 4$ имеем $m < 4$. Следовательно, $m = 3$. Очевидно, $m < p - 1$ и по лемме 5 группа G имеет композиционный фактор, изоморфный $L_2(p)$ или $L_2(p-1)$, где $p = 2^a + 1$. При этом $q = 2$. Отсюда, учитывая, что $m = 3$, имеем: $p = 5$ или $p = 7$. В первом случае $|G| < 200$, а во втором – $|G| < 392$. Так как центр группы G тривиален, то она изоморфна $PGL_2(s)$, где $s \in \{5, 7\}$. По лемме 6 получаем противоречие.

Предположим, что $b = 3$. Тогда $q \leq 2p/m$. Если $q \geq 3$, то $m < p - 1$. По лемме 5 у группы имеется композиционный фактор M/N, изоморфный группе $L_2(p)$ или $L_2(p-1)$, где $p - 1 = 2^a$ для некоторого натурального числа $a \geq 2$. В обоих случаях $N = O_{p'}(G)$, а G/M изоморфна подгруппе $Out(M/N)$. Допустим, что $M/N \simeq L_2(p)$. Если q делит $p - 1/2$, то и q^3 делит $p - 1/2$. Но тогда $m \geq p + 1$, что неверно. Если q делит $p + 1$, то q^3 делит $(p + 1)/2$, откуда $m = p - 1$ что также неверно. Пусть $M/N \simeq L_2(p-1)$, где $p = 2^a + 1$. Так как q по предположению, нечетно, то q^3 делит $p - 2 = 2^a - 1$, откуда $m = 2^a = p - 1$, что приводит к противоречию.

Пусть теперь $b = 3, q = 2$. Тогда силовская 2-подгруппа группы G либо абелева, либо неабелева порядка 8. В силу леммы 7 группа G имеет композиционный фактор, изоморфный $L_2(r), A_7, {}^2G_2(q)$ или J_1. При этом $p < 2q^2 = 8$, а $m \leq p$. Поэтому, учитывая, что $(m, qp) = 1$, имеем либо $p = 5$, $m = 3$, либо $p = 7, m = 3$ или $m = 5$. Так как во всех случаях оказывается, что m – простое число, то $G = M$ – простая неабелева группа, $N = 1$. По лемме 5 группа G изоморфна $L_2(5)$ или $L_2(7)$. В обоих случаях $|G| < p^3$. По лемме 15 получили противоречие. Стало быть, $b \leq 2$. Лемма доказана.

Лемма 17. *Пусть G – не p–разрешимая $LC(\Theta)$–группа порядка $|G| = pq^b m$, где $p > q, (m, pq) = 1$. Тогда либо $b = 1$, G либо – простая группа порядка pq^2m.*

Доказательство. Предположим, что $b = 2$. В этом случае (ввиду лемм 15 и 16) имеем:

$$p^3 < |G| = pq^2m \leq 2p^2q^2.$$

Отсюда $2p \geq m, p < 2q^2$. Пусть группа G имеет неабелев композиционный фактор M/N. Понятно, что $|M/N| = pq^\lambda m_0$, где $\lambda \leq 2, m_0$ делит m. Если $\lambda = 0$, то силовская p–подгруппа группы M/N нормальна, что неверно. Значит,$\lambda \geq 1$.

При $m_0 < p - 1$, по лемме 5, группа M/N имеет композиционный фактор, изоморфный $L_2(p)$ или, $L_2(p-1), p = 2^a + 1$ и $N = O_{p'}(G)$. В обоих случаях $q = 2$. Так как $q = 2$, то $p < 8$ и G имеет абелеву силовскую 2-подгруппу (порядка 4). Отсюда $p = 5, q = 2, m_0 = 3$. Так что $M/N \simeq L_2(5)$. Следовательно, $|G/M||N| = m/m_0 < 4$. Так как G не имеет нормальных подгрупп порядка 2 и $|G| = pq^2m < 80$, то $G \simeq L_2(5)$, что неверно.

Таким образом, $m_0 \geq p - 1$ и $m/m_0 \leq 2$. Кроме того, $|G/M||N|$ делит $2q^{2-\lambda}$. Если $\lambda = 2$, то $N = 1$, ибо G не имеет нормальных подгрупп порядка 2. Если $m = m_0$, то $|N|$ делит q и $|G/C_G(N)| = 1$ и в этом случае $N = 1$. Если же $|N| = 2q$, то G изоморфна прямому произведению неабелевой группы порядка $2q$ и простой неабелевой группы G_1. Легко убедиться, что группа G_1 имеет неприводимый характер

степени pq. Ввиду теоремы 1 из [15] характер Θ^2 должен содержать любой характер диедральной группы порядка $2q$, что, очевидно, неверно.

Следовательно, в любом случае $N = 1$ и M – простая неабелева группа, причем $|G : M|$ делит $2q$.

Предположим, что G не является простой группой. По лемме 9 характер Θ_M содержит неприводимый характер ϕ, для которого $pq = e\phi(1)$, где e и s делят $2q$ и $\phi(1) < pq$. Так как M – простая неабелева группа, то $\phi(1) > 1$. Поэтому $\phi(1) = p$. Более того, можно считать, что ϕ имеет q сопряженных характеров (см. теорему (6.2) из [1]). В частности, $|M| > p^2q$. Поэтому $|G/M| = q$. Кроме того, $|M| = pqm$.

Если $y \in C_M(P)$ – элемент порядка, взаимно простого с p, то $|G : C_M(y)|\phi(y)/\phi(1)$ целое алгебраическое число. По лемме Бернсайда (лемма 4.3.1 в [16]) либо $y \in Z(\phi) \leq S(M) = 1$, либо $\phi(y) = 0$. Допустим, что $y \neq 1$ и $\phi(y) = 0$. Тогда скалярное произведение $\langle \phi, 1_{\langle y \rangle} \rangle = |\langle y \rangle|^{-1}\phi(1)$ – целое рациональное число, что исключено. Поэтому $C_M(P) = P$.

По лемме 3 группа M либо изоморфна $L_2(r)$, где $r = p$ или $r = p - 1 = 2^a$, либо $|M : N_M(P)| = 1 + np$ с $n \geq (p + 3)/2$. Если $M \simeq L_2(r)$ для $r = p$, то $|Out(M)| = 2$, откуда $p \leq 8$. По лемме 6 получаем противоречие. Если, $M \simeq L_2(2^a)$ где $a = \log_2(p - 1)$, то $|Out(M)| = a$. В этом случае из $2q^2 > p$ следует, что $4\log_2(p) > p$. Поэтому $p \leq 17$. Так как q – нечетное простое число, то эта возможность исключена.

Таким образом, $|M : N_M(P)| = 1 + np$ где $n \geq (p + 3)/2$. Обозначим

$$|N_M(P) : C_M(P)| = |N_M(P) : P| = e.$$

Так как $|M| = pmq = ep(1 + np)$, то $e(1 + np) = mq \leq 2pq$. Следовательно, $e < 2q/n < 4$. Так как M не является p–нильпотентной, то $e > 1$. Поэтому $e = 2$ или 3.

Элемент $u \in G \setminus M$ порядка q индуцирует автоморфизм группы M, нормализующий подгруппу $N_M(P)$. Если q не делит $(p - 1)/e$, то $C_G(u)$ имеет порядок, делящийся на epq^2. Отсюда $|G : C_G(u)| = mpq^2/epq^2 \leq m/2 \leq p$. По лемме 8 имеем $|G : C_G(u)| \equiv 1(mod\ p)$, что неверно.

Поэтому q делит $(p - 1)/e$. Отсюда из $e(1 + np) = mq \leq 2pq \leq 2p(p - 1)/e$ получаем: $n \leq 2(p-1)/e^2 \leq (p-1)/2$. Так как $n \geq (p+3)/2$, получили противоречие.

Следовательно, либо $b = 1$, либо G – простая группа порядка pq^2m. Лемма доказана.

Лемма 18. *Пусть G – не p-разрешимая $LC(\Theta)$-группа порядка $|G| = pq^bm$, где $p > q$, $(m, pq) = 1$. Тогда G – не простая группа.*

Доказательство. По теореме 1 группа G изоморфна одной из групп в ее заключении. При этом группы $L_2(q)$, $L_3(q)$, $U_3(q)$ не являются простыми $LC(\Theta)$-группами (их таблицы характеров имеются в [9]). Таблицы характеров групп $A_7, M_{11}, Sz(8)$ и J_1 содержатся в Приложении 1 книги [9]. Ни одна из перечисленных групп не является $LC(\Theta)$-группой при любом выборе неприводимого характера Θ. Лемма доказана.

Лемма 19. *Пусть G – не p-разрешимая $LC(\Theta)$-группа. Тогда $\tilde{G}$, главный не p-разрешимый фактор группы G, изоморфен $L_2(r)$ для натурального числа r, являющегося степенью простого числа s.*

Доказательство. Пусть $\tilde{G}$ – главный не p-разрешимый фактор группы G. В силу лемм 17 и 18 порядок G равен pqm, причем $m < 2pq$ и $p < 2q^2$. По теореме 1 $\tilde{G}$ изоморфна одной из следующих групп: $Sz(8)$, $p = 13$, A_7, $p = 7$, M_{11}, $p = 11$, J_1, $p = 19$, $L_2(r)$, $L_3(r)$, $p = r^2 + r + 1$ или $U_3(r)$, $p = r^2 - r + 1$.

Допустим, что $\tilde{G} \simeq Sz(8)$, $p = 13$. Если $q = 7$, то $\Theta(1) = 7 \cdot 13 = 91$ и $2 \cdot 91^2 < |\tilde{G}|$, что исключает эту возможность.

Пусть $\tilde{G} \simeq A_7$. Тогда $pq = 35$ и $|\tilde{G}| > 2 \cdot 35^2$, что исключает эту возможность.

Пусть $\tilde{G} \simeq M_{11}$, $p = 11$, $q = 5$. Так как $|\tilde{G}| > 2 \cdot 55^2$, то эта возможность исключена.

Пусть $\tilde{G} \simeq J_1$ $p = 19$, $q = 11$. Так как $|\tilde{G}| > 2 \cdot 209^2$, то эта возможность исключена.

Во всех рассмотренных выше примерах легко убедиться, что $pq||\tilde{G}|$.

Допустим, что $\tilde{G} \simeq U_3(r)$. Наибольший простой делитель p числа $|\tilde{G}|$ не превосходит $r^2 - r + 1$. Если q не делит $|\tilde{G}|$, то $|\tilde{G}| = pm'$, где $m' < m < 2p^2$, а $|\tilde{G}| < 2p^3$, что исключено. Поэтому $pq||\tilde{G}|$. Так как наибольший простой делитель порядка группы $\tilde{G}$, меньший p, не превосходит $r + 1$, то $pq < r^3 + 1$. Однако этот случай невозможен.

Допустим, что $\tilde{G} \simeq L_3(r)$. Наибольший простой делитель p числа $|\tilde{G}|$ не превосходит $r^2 + r + 1$. Если q не делит $|\tilde{G}|$, то $|\tilde{G}| = pm'$, где $m' < m < 2p^2$, а тогда $|\tilde{G}| < 2p^3$, что исключено. Поэтому $pq||\tilde{G}|$. Так как наибольший простой делитель порядка группы $\tilde{G}$, меньший p, не превосходит $r + 1 < 2(r - 1)$, то $pq < 2(r^3 - 1)$. Однако и этот случай невозможен.

Таким образом, можно считать, что $G \simeq L_2(r)$. Лемма доказана.

Перейдем к завершению доказательства теоремы 2. По лемме 18 главный не p-разрешимый фактор группы G изоморфен $L_2(r)$, где r – степень простого числа s. В силу лемм 17 и 18 порядок G равен pqm, причем $m < 2pq$ и $p < 2q^2$. Наибольший простой делитель порядка $\tilde{G}$ не превосходит $r + 1$.

Предположим, что q не делит $|\tilde{G}|$. Так как $m < 2p^2$, то $|\tilde{G}| = pm'$, где $m'|m < 2p^2 < 2(r + 1)^2$. Порядок группы $\tilde{G}$ равен $\lambda r(r^2 - 1)$, где $\lambda = 1$ для четного r и $\lambda = 1/2$ для нечетного r. Отсюда $m' \geq \lambda r(r - 1)$. Таким образом,

$$\frac{m}{m'} \leq \frac{2(r+1)^2}{\lambda r(r-1)} \leq \frac{4(r^2+2r+1)}{r^2-r} < 7$$

при $r \geq 7$. Поэтому $|G| \leq 7q|\tilde{G}|$. Так как $G/O_{p'}(G)$ изоморфна подгруппе группы $Aut(\tilde{G})$, содержащей $\tilde{G}$, то наибольший простой делитель числа $|Out(\tilde{G})|$ не превосходит $\log_s r$. Однако в этом случае $q \leq \log_s r$. Так как $p < 2q^2$ и $m < 2pq$, то $|G| < 4q^6 \leq 4(\log_s r)^6$, а $|\tilde{G}| = \lambda r(r^2 - 1) = pm' < 2(\log_s r)^5$. Очевидно, $q \geq 5$, так что $r = s^t$, где $t \geq 5$. Однако $r = s^t > 2t^2$ для любых допустимых значений $r = s^t$. Поэтому q не делит $|Out(\tilde{G})|$. Ввиду леммы 10 получаем, что $G/O_{p'}(G) \simeq \tilde{G}$. Так как q не делит $|\tilde{G}|$, то $|O_{p'}(G)| = m/m'q \leq 7q$. Если $q = m/m'$, то $|G/O_{p'}(G)|$ не делится на q и по лемме 2 группа G не может иметь характера, степень которого делится на q. Если $q > m/m'$, то G имеет характеристическую подгруппу порядка q и снова не может иметь характера степени, делящейся на q. Наконец, если $q < m/m'$, то группа $O_{p'}(G)$ централизуется G. Так как центр G тривиален, то G – прямое произведение группы $\tilde{G}$ и группы порядка $m/m'q$, где $q < m/m' \leq 7$, не имеющей абелевых нормальных подгрупп порядка q. В этом случае $q = 3$, противоречие.

Итак, $pq||\tilde{G}|$. По лемме 10 группа $G/O_{p'}(G)$ изоморфна $\tilde{G}$ и $G = G'$. Напомним, что $\tilde{G} \simeq L_2(r)$. Если r нечетно, то наибольший простой делитель p порядка $\tilde{G}$ делит либо $(r \pm 1)/2$, либо равен r.

Пусть $p = r$. Так как $|\tilde{G}| = pqm_0$, где $m_0|m$, то $m_0 \geq (p-1)$. Поэтому $|O_{p'}(G)| = m/m_0 \leq \frac{p(p+1)}{p-1} < p+3$. Если P – силовская p-подгруппа группы G, то $|O_{p'}(G) : (C_G(P) \cap O_{p'}(G))| \leq 1+p$. Поэтому P имеет на $O_{p'}(G)$ только орбиту длины p и потому $|O_{p'}(G)| = p + |C_G(P) \cap O_{p'}(G)|$. Отсюда $|O_{p'}(G)| = 1+p$. Последнее возможно лишь при $p+1 = 2^a$ и $q \leq (p-1)/2$. Тогда $m_0 \geq p+1$ и $|O_{p'}(G)| < p$. Так как $C_G(O_{p'}(G)) \supseteq P$, то $G = C_G(O_{p'}(G))O_{p'}(G)$. Так как $G = G'$ и центр G тривиален, то получили противоречие.

Пусть теперь p делит $(r+1)/2$. Так как $q < p$, то $q \leq (r-1)/2$. Стало быть, $|\tilde{G}| = pqm_0$, где $m_0 \geq r$. Отсюда

$$\tfrac{m}{m_0} \leq \tfrac{r^2-1}{4r} \leq p \leq (r+1)/2$$

Поэтому $O_{p'}(G) \leq C_G(P)$. Отсюда $G = O_{p'}(G)C_G(O_{p'}(G))$. По лемме 10 имеем $Z(G) = 1$. Так как $G = G'$, то $G = \tilde{G}$ и по лемме 4 получаем противоречие.

Если $s = 2^a$, то $p \leq r+1$, $q \leq r-1$. Так как $|\tilde{G}| = pqm_0$, где $m_0 \geq r$, то $m/m_0 \leq (r^2-1)/r < r < p$, то $O_{p'}(G)$ централизует группу G, что приводит к противоречию. Теорема 2 доказана.

4. Доказательство теоремы 3

В теореме 2 было установлено, что G является p-разрешимой группой и при $|G| \notin \{54, 72\}$ порядок G равен pq^bm, где $(pq, m) = 1$. По теореме 6.3.3 из [16] силовская p-подгруппа группы G содержится в $H = O_{p',p}(G)$. Поэтому $G = O_{p',p,p'}(G)$. Очевидно, что $H = K \rtimes P$, где $K = O_{p'}(G)$. Эти обозначения фиксируются до конца доказательства теоремы 3.

Лемма 20. *Если $H \neq G$, то $|G : H| = q$ и $\Theta_H = \sum_{i=1}^{q} \phi_i$, где $\phi_i \in Irr(H)$ – сопряженные характеры группы H степени p.*

Доказательство. Пусть $\phi \in Irr(H)$ – неприводимый характер группы H, входящий в разложение Θ_H и пусть $I_G(\phi)$ – его группа инерции. Согласно лемме 1,

$$\Theta_H = e\sum_{i=1}^{s} \phi_i,$$

где ϕ_i – неприводимые сопряженные с $\phi = \phi_1$ характеры группы H, e – натуральное число, делящее $|I_G(\phi_1) : H|$, а s делит $|G : I_G(\phi)|$. Из леммы 9 получаем, что Θ_H – приводимый характер, так что $s > 1$. Так как $pq = \Theta(1) = es\phi(1)$ и $|G : H|$ взаимно просто с p, то $e = 1, s = q$ и $\phi(1) = p$.

Из сказанного выше ясно, что индекс H в G делится на q. Предположим, что $|G : H| = aq$ для некоторого натурального числа a. Так как H имеет q сопряженных в G неприводимых характеров степени p, то

$$|H| \geq 1 + \sum_{i=1}^{q} \phi_i(1)^2 = p^2q.$$

С другой стороны, $|H| = |G|/|G : H| \leq 2p^2q^2/(aq) = (2/a)p^2q$. Поэтому $a = 1$ и $|G : H| = q$. Лемма доказана.

Лемма 21. *Если $H \neq G$, то для любого неприводимого характера $\phi = \phi_i$ из разложения Θ_H существует p неприводимых характеров λ_j, сопряженных с характером $\lambda = \lambda_1$, таких, что $\phi_K = \sum_{j=1}^{p} \lambda_j$. Группа K абелева.*

Доказательство. Пусть λ – неприводимый характер группы K, входящий в разложение характера ϕ_K, где $\phi = \phi_i$ – любой неприводимый характер группы H, определенный в лемме 20. Согласно лемме 1,

$$\phi_K = e' \sum_{i=1}^{t} \lambda_i,$$

где λ_i – характеры, сопряженные с λ, e' и t делят $|H : K| = p$. В частности,

$$p = \phi(1) = e' t \lambda(1).$$

Если ϕ остается неприводимым при ограничении на K, то это же верно и для любого его сопряженного ϕ_j, а тогда порядок подгруппы K не меньше, чем $qp^2+1 \leq |G|/|G : K| \leq 2pq$, что противоречиво. Следовательно, $t = p, e' = 1$ и $\lambda_j(1) = 1$ для всех $j \leq p$. В частности, $\ker \lambda_j$ содержит коммутант группы K. Отсюда $\lambda_j(y) = 1$ для всех λ_j. Но тогда $\phi(y) = p$ для любого характера, сопряженного с ϕ_1. Как результат, $\Theta(y) = pq$. Напомним, что Θ является точным характером. Поэтому $y = 1$ и $K' = 1$, что и требовалось доказать. Лемма доказана.

Заметим, что в случае $O^q(G) \neq G$ заключение леммы 21 остается в силе. Поэтому далее можно считать, что $H = O_{p'p}(G) = G$ и $G' = O_{p'}(G) = K$. Мы предполагаем далее, что $K = G'$. В силу леммы 1,

$$\Theta_K = v \sum_{i=1}^{u} \psi_i,$$

где ψ_i – характеры, сопряженные с ψ_1, v и u делят $|G : K| = p$. В частности, $pq = \Theta(1) = vu\psi(1)$. Так как u и v делят $|G : K| = p$, то либо $\Theta_K = \Theta$, либо Θ_K – приводимый характер. По лемме 9 характер Θ_K приводимый. Следовательно, $u = p, v = 1$. Таким образом, характер $\psi = \psi_1$ имеет степень q и имеется p характеров степени q группы K, сопряженных с ψ. Напомним, что группа G имеет порядок $pq^b m$, где $(pq, m) = 1$.

Лемма 22. *Группа $G = K \rtimes P$ содержит подгруппу QP порядка $q^b p$, где P – силовская p-подгруппа группы G, а Q – силовская q-подгруппа G. Если $b \geq 3$, то $m < p$. В частности, K имеет нормальную $\{p, q\}$- подгруппу, порядка, делящегося на p.*

Доказательство. В самом деле, при $b \geq 3$ имеем $pq^b m < 2p^2q^2$, то $m < 2pq^{2-b} \leq p$. Поэтому $m \leq p - 1$. Как сказано выше, группа $G = K \rtimes P$, где $|K| = q^b m$ и $(p, qm) = 1$. Так как P действует с помощью сопряжений на множестве силовских q-подгрупп группы K, а их число не делится на p, то имеется силовская q-подгруппа Q группы K, инвариантная относительно P. Таким образом, в G содержится подгруппа $L = QP$, имеющая индекс m в G. Пусть γ – гомоморфизм группы G в группу

правых смежных классов по L. Очевидно, ядро γ содержится в L и имеет порядок, делящийся на p. Поэтому группа G имеет нормальную подгруппу $M \subseteq L$, порядок которой делится на p. Лемма доказана.

Покажем, что порядок силовской $q-$подгруппы группы G не превосходит q^2.

Лемма 23. *$b \leq 2$.*

Доказательство. Допустим, что $b \geq 3$. В силу доказанного леммы 22 группа G имеет нормальную подгруппу M, содержащуюся в $L = QP$, порядка, делящегося на p. Из теоремы Н. Ито (Лемма 2) следует, что $M \neq P$. Поэтому, не ограничивая общности, можно считать, что $M = Q_1 \rtimes P$, где Q_1 – подгруппа группы Q, на которой P действует нетривиально. Допустим сначала, что $q > 2$. Так как $q < p$, то $|Q_1| = q^a, a \geq 2$, причем $q^a - 1 \geq 2p$. Следовательно, $|G/M| \leq 2p^2q^2/(p(2p+1)) < q^2$. Поэтому либо $m < q$, либо $b = a$ и $m < q^2$. Допустим, что $|G/M|$ делится на q. Так как $|G/M| < q^2$, то $|G/M| = mq$, где $m < q$. В силу теорем Силова имеем $Q \triangleleft G$. При этом имеется холлова подгруппа T порядка m, на которой P действует тождественно. Таким образом, $G = Q \rtimes (T \times P)$. Поэтому при $m > 1$ имеем $G' \neq K$. В силу леммы 21 получаем противоречие.

Предположим, что $q = 2$. Очевидно, в этом случае $a \geq 3$, причем $q^a - 1 \geq p$. Поэтому $|G/M| \leq 2p^2q^2/p(p+1) = 8p/(p+1) < 8$ и либо G/M имеет подгруппу индекса $q = 2$, что неверно по предположению, либо $|G : G'|$ не совпадает с p. Последнее также исключено.

Если $|G/M|$ не делится на q, то $M = L = G$, причем $m = 1, a = b$. Тогда существует минимальная ненильпотентная подгруппа S порядка q^cp, где $q^c \equiv 1(modp)$ и $c \leq b$ (см. доказательство теоремы в [17]). Так как $q^c - 1$ делится на $2p$, то $q^c \geq 2p$ и потому $b \leq c = 1$. Отсюда либо $b = c$ и G – минимальная ненильпотентная группа с абелевыми силовскими подгруппами, либо G имеет нормальную подгруппу индекса q. Применяя лемму 2, получаем противоречие. Таким образом, случай $b > 2$ невозможен. Лемма доказана.

Лемма 24. *$|G| = mpq$, где $(m, pq) = 1$ и $m < 2pq$.*

Доказательство. Допустим, что $b = 2$, т.е. силовская $q-$подгруппа Q группы G имеет порядок q^2. В силу леммы 22 группа $G = K \rtimes P$, причем G содержит подгруппу $L = Q \rtimes P$, где P порядка p.

Пусть y – элемент группы Q и $h_y = |G : C_G(y)|$. Заметим, что группа $L = QP$ абелева. Действительно, в противном случае p делит порядок группы автоморфизмов Q. Так как последняя имеет порядок, делящий $q^2(q^2-1)(q-1)$ и $p > q$, то p не делит $|Aut(Q)|$. Таким образом, h_y взаимно прост с $\Theta(1)$. Согласно лемме 4.3.1 в [16], либо $\Theta(y) = 0$, либо $\Theta(y) = \mu\Theta(1)$ для некоторого числа μ, являющегося корнем q-ой степени из единицы. Так как $\Theta-$ точный характер, то $\ker\Theta = 1$, а так как $Z(G) = 1$, то $\Theta(y) = 0$ для любого $y \in Q^{\#}$. Поэтому

$$\langle \Theta_Q, 1_Q \rangle = 1/|Q| \sum_{y \in Q} \Theta(y) = \frac{pq}{q^2} = p/q$$

– целое число. Противоречие. Поэтому q^2 не делит $|G|$. То есть $|G| = pqm$, где $(pq, m) = 1$. Так как $|G| \leq 2\Theta(1)^2 = 2p^2q^2$, то $m \leq 2pq$. Лемма доказана.

Лемма 25. *Пусть ψ – неприводимый характер группы K степени q, входящий в разложение Θ_K. Тогда $\psi(y) = 0$ для любого q-элемента группы G. Группа K не является простой неабелевой группой и для $R \lhd K$ характер ψ_R приводим.*

Доказательство. Так как $|K|$ не делится на q^2, то характер ψ степени q содержится в q-блоке группы K дефекта 0. Поэтому $\psi(y) = 0$ для любого q-сингулярного элемента группы K. В частности, для элемента порядка q.

Допустим, что K – простая неабелева группа. Так как p делит $|Out(K)|$, то по лемме 8 из [18] получаем, что либо $K \simeq L_3(q), U_3(q)$ или $L_2(q)$, либо $|K| \geq p^6$. Так как $|K| = mq \leq 2pq^2 < p^4$ в любом из случаев, то K не может быть простой неабелевой группой.

Допустим, что R – наибольшая собственная нормальная подгруппа группы K. По лемме 1 имеем:

$$\psi_R = e \sum_{s=1}^{d} \lambda_s,$$

где d, e – натуральные числа, делящие $|K : R|$, и λ_s – неприводимые характеры группы R, сопряженные с $\lambda_1 = \lambda$. Отсюда $q = \psi(1) = ed\lambda(1)$. Если ψ_R неприводим для любого ψ, входящего в разложение Θ_K, то подгруппа R имеет p неприводимых характеров степени q. Но тогда ее порядок равен не меньше, чем q^2p, тогда как $|R| \leq |K|/(|K : R|) \leq pq^2$. Противоречие. Значит, $\psi_R = \sum_{s=1}^{p} \lambda_s$, где λ_s – линейные характеры. Лемма доказана.

Закончим доказательство теоремы 3. Из леммы 25 следует, что пересечение ядер характеров λ_s, входящих в разложение Θ_R, содержит коммутант R и потому тривиально. Поэтому подгруппа R группы G, имеющая индекс pq в G абелева, что и утверждалось. Теорема доказана.

5. Примеры

1. Пусть $p = 2^a - 1$ и $q = 2^b - 1$ – два различных простых числа Мерсенна. Тогда группа $G = M \times K$, являющаяся прямым произведением групп Фробениуса порядков $p(p+1)$ и $q(q+1)$ соответственно имеет неприводимый характер Θ с $\Theta(1) = pq$.

2. Пусть $p = 5$ и $q = 3$ и G – группа Фробениуса порядка $16 \cdot 15$. Она является $LC(\Theta)$-группой.

3. Пусть $p = 2^a - 1$ – простое число Мерсенна, $q = 2$. Тогда группа $G = M \times K$, являющаяся прямым произведением групп Фробениуса порядков $(p+1)p$ и 6 является $LC(\Theta)$-группой.

Список литературы / References

[1] Isaacs I. M., *Character theory of finite groups*, Academic press, New York, San Francisco, London, 1976, 305 pp.

[2] Казарин Л. С., Сагиров И. А., "О степенях неприводимых характеров конечных простых групп", Тр. ИММ УрО РАН, **7**, 2001, 113–123; English transl.:Kazarin L. S., Sagirov I. A., "On the degrees of irreducible characters of finite simple groups", *Proc. of the Steklov Inst. Math. Suppl.(Supplementary issues)*, 2001, suppl. 2, №2, S71–S81.

[3] Snyder N., "Groups with a character of large degree", Proc. Amer. Math. Soc., **136**, 2008, 1893–1903.

[4] Berkovich Y., "Groups with few characters of small degree", *J. Math.*, **110** (1999), 325–332.

[5] Isaacs I. M., "Bounding the order of a group with a large degree character", *J. Algebra*, **348**:1 (2011), 264–275.

[6] Durfee C., Jensen S., "A bound on the order of a group having a large character degree", *J. Algebra*, **338** (2011), 197–206.

[7] Lewis M. L., "Bounding group order by large character degrees: A question of Snyder", *Journal of Group Theory*, **17**:6 (2014), 1081–1116.

[8] Feit W., *The representation theory of finite groups*, North-Holland Publishing Company, Amsterdam, New York, Oxford, 1982, 510 pp.

[9] Белоногов В. А., *Представления и характеры в теории конечных групп*, УрО АН СССР, Свердловск, 1990; [Belonogov V. A., *Predstavlenija i haraktery v teorii konechnyh grupp*, UrO AN SSSR, Sverdlovsk, 1990 (in Russian)].

[10] Bierbrauer J., "The uniformly 3-homogeneus subsets in $PGL(2, q)$", *J.Algebraic Combinatorics*, **4** (1995), 99–102.

[11] Walter J., "The characterization of finite groups with abelian Sylow 2-subgroups", *Ann. Math.*, **8** (1969), 405–514.

[12] Chabot P., "Groups whose Sylow 2-subgroups have cyclic commutator groups, II, III", *J.Algebra*, **19, 21, 29** (1971, 1972, 1974), 21–30, 312–320, 455–458.

[13] Gorenstein D., *Finite Simple Groups. An introduction to their classification*, Plenum Publishing Corporation, New York, 1982, 333 pp.

[14] Conway J. H., Curtis R. T., Norton S. P., Parker R. A., Wilson R. A., *Atlas of finite groups: maximal subgroups and ordinary characters for simple groups*, Clarendon Press, Oxford, 1985, 255 pp.

[15] Казарин Л. С., Поисеева С. С., "Конечные группы с большим неприводимым характером", *Матем. заметки*, **98**:2 (2015), 237–246; English transl.:Kazarin L. S., Poiseeva S. S., "Finite groups with large irreducible character", *Mathematical Notes*, **98**:2 (2015), 265–272.

[16] Gorenstein D., *Finite Group*, Chelsea publishing company, New York, 1968, 522 pp.

[17] Шмидт О. Ю., "Группы, все подгруппы которых специальные", *Матем. сб.*, **31**:3–4 (1924), 366—372; English transl.: Schmidt O. Y., "Groups all whose subgroups are nilpotent", *Mat. Sb.*, **31** (1924), 366—372.

[18] Amberg B., Kazarin L. S., "ABA-groups with cyclic subgroup B", Труды института математики и механики УрО РАН, **18**, 2012, 10–22.

DOI: 10.18255/1818-1015-2015-4-483-499

On Finite Groups with an Irreducible Character Large Degree

Kazarin L. S.[1], Poiseeva S. S.

Received July 6, 2015

Let G be a finite nontrivial group with an irreducible complex character χ of degree $d = \chi(1)$. It is known from the orthogonality relation that the sum of the squares of degrees of irreducible characters of G is equal to the order of G. N. Snyder proved that if $|G| = d(d+e)$, then the order of G is bounded in terms of e, provided $e > 1$. Y. Berkovich proved that in the case $e = 1$ the group G is Frobenius with the complement of order d. We study a finite nontrivial group G with an irreducible complex character Θ such that $|G| \leq 2\Theta(1)^2$ and $\Theta(1) = pq$, where p and q are different primes. In this case we prove that G is solvable groups with abelian normal subgroup K of index pq. We use the classification of finite simple groups and prove that the simple nonabelian group whose order is divisible by a prime p and of order less than $2p^4$ is isomorphic to $L_2(q), L_3(q), U_3(q), Sz(8), A_7, M_{11}$ or J_1.

Keywords: finite group, character of a finite group, irreducible character degree of a finite group

For citation: Kazarin L. S., Poiseeva S. S., "On Finite Groups with an Irreducible Character Large Degree", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 483–499.

On the authors:
Kazarin Lev Sergeevich, orcid.org/0000-0003-1836-0955, doctor of science, professor
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: kazarin@uniyar.ac.ru

Poiseeva Sargylana Semenovna, orcid.org/0000-0003-2740-9845, graduate student,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: pss.iii@mail.ru

Acknowledgments:

[1]This work was supported by the Russian Foundation for Basic Research (RFBR), №13-01-00469

Моделирование и анализ информационных систем. Т. 22, №4 (2015), с. 500–506
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 500–506

DOI: 10.18255/1818-1015-2015-4-500-506

УДК 512.543

О финитной отделимости подгрупп в расщепляемых расширениях

Кряжева А. А.

получена 21 апреля 2015

В 1973 году Аленби и Грегорас доказали следующее утверждение. Пусть G – расщепляемое расширение конечно порожденной группы A с помощью группы B. 1) Если в группах A и B все подгруппы (все циклические подгруппы) финитно отделимы, то и в группе G все подгруппы (все циклические подгруппы) финитно отделимы; 2) если в группе A все подгруппы финитно отделимы, а в группе B все конечно порожденные подгруппы финитно отделимы, то в группе G все конечно порожденные подгруппы финитно отделимы. Напомним, что группа G называется расщепляемым расширением группы A с помощью группы B, если группа A является нормальной подгруппой группы G, B – подгруппа группы G, $G = AB$ и $A \cap B = 1$. Напомним также, что подгруппа H группы G называется финитно отделимой, если для каждого элемента g группы G, не принадлежащего подгруппе H, существует гомоморфизм группы G на некоторую конечную группу, при котором образ элемента g не принадлежит образу подгруппы H. В данной работе получено обобщение теоремы Аленби и Грегораса за счет замены условия конечной порожденности группы A более общим: для любого натурального числа n число всех подгрупп группы A индекса n конечно. В действительности при этом условии удалось получить необходимое и достаточное условие финитной отделимости всех подгрупп (всех циклических подгрупп, всех конечно порожденных подгрупп) в группе G.

Ключевые слова: расщепляемое расширение, финитная отделимость подгруппы, конечно порожденная группа

Для цитирования: Кряжева А. А., "О финитной отделимости подгрупп в расщепляемых расширениях", *Моделирование и анализ информационных систем*, **22**:4 (2015), 500–506.

Об авторах:
Кряжева Анастасия Алексеевна, orcid.org/0000-0002-5985-5172, аспирант каф. алгебры и математической логики,
Ивановский государственный университет,
ул. Ермака, 39, г. Иваново, 153025 Россия, e-mail: Stasia.07.10@mail.ru

Благодарности:
Работа выполнена при финансовой поддержке Минобрнауки России по государственному заданию.

Введение

Напомним, что подгруппа H группы G называется финитно отделимой [1], если для каждого элемента g группы G, не принадлежащего подгруппе H, существует гомоморфизм группы G на некоторую конечную группу, при котором образ элемента g не принадлежит образу подгруппы H. Из [2, п. 1. 3. 10] известно, что в полициклической группе все подгруппы финитно отделимы. Однако условие финитной отделимости всех подгрупп является достаточно жестким ограничением. Менее жестким ограничением является финитная отделимость всех конечно порожденных подгрупп. Р. Бернс [3] для обозначения групп с финитно отделимыми конечно порожденными подгруппами ввел термин LERF, использующийся в иностранной литературе. Исследования таких групп были начаты М. Холлом в 1949 году. Он доказал [4], что произвольная конечно порожденная подгруппа любой свободной группы финитно отделима.

Рипсом [5] был построен пример группы, содержащей неотделимую конечно порожденную подгруппу и являющуюся обобщенным свободным произведением с циклическими объединенными подгруппами двух групп, все конечно порожденные подгруппы каждой из которых финитно отделимы. Позже был приведен более простой аналогичный пример свободного произведения двух конечно порожденных нильпотентных групп с циклическим объединением [6]. Вопрос финитной отделимости конечно порожденных подгрупп рассматривался также в работах [7–12].

Исследования финитной отделимости циклических подгрупп являются также весьма интересными. Здесь можно использовать те же методы, что и для свойства финитной аппроксимируемости. Основной в данном направлении является работа Стиба [13], а также работы Логиновой Е. Д. [14], Соколова Е. В. [15].

Расщепляемые расширения и их аппроксимационные свойства рассматривались в работах Мальцева А. И. [1], Азарова Д. Н. [16–19].

В данной статье рассматривается вопрос финитной отделимости подгрупп в расщепляемых расширениях. Напомним, что группа G называется расщепляемым расширением группы A с помощью группы B, если группа A является нормальной подгруппой группы G, B – подгруппа группы G, $G = AB$ и $A \cap B = 1$.

Хорошо известна следующая теорема Аленби и Грегораса [7].

Теорема 1. *Пусть G – расщепляемое расширение конечно порожденной группы A с помощью группы B.*

1. Если в группах A и B все подгруппы (все циклические подгруппы) финитно отделимы, то и в группе G все подгруппы (все циклические подгруппы) финитно отделимы.

2. Если в группе A все подгруппы финитно отделимы, а в группе B все конечно порожденные подгруппы финитно отделимы, то в группе G все конечно порожденные подгруппы финитно отделимы.

Пример прямого произведения двух свободных групп ранга 2 [7] показывает, что пункт 2 нельзя сформулировать по аналогии с пунктом 1. Легко заметить, что п. 1 теоремы Аленби и Грегороса можно обратить, но второй пункт в таком виде обратить нельзя. Поэтому возник вопрос о нахождении необходимого и достаточного условия, при котором в группе G все конечно порожденные подгруппы финитно

отделимы. Это условие содержится в пункте 2 следующей теоремы, доказанной автором.

Теорема 2. *Пусть G – расщепляемое расширение группы A с помощью группы B. И пусть группа A удовлетворяет следующему условию: для любого натурального числа n число всех подгрупп группы A индекса n конечно. Тогда*

1. В группе G все подгруппы (все циклические подгруппы) финитно отделимы тогда и только тогда, когда в группах A и B все подгруппы (все циклические подгруппы) финитно отделимы.

2. В группе G все конечно порожденные подгруппы финитно отделимы тогда и только тогда, когда в группе B все конечно порожденные подгруппы финитно отделимы, и в группе A финитно отделимы все подгруппы, высекаемые в A конечно порожденными подгруппами группы G. В частности, если в группе A все подгруппы финитно отделимы, а в группе B все конечно порожденные подгруппы финитно отделимы, то в группе G все конечно порожденные подгруппы финитно отделимы.

М. Холл [20, с. 250] доказал, что в любой конечно порожденной группе существует только конечное число подгрупп данного конечного индекса. Поэтому доказанная Аленби и Грегоросом теорема 1 является частным случаем теоремы 2.

1. Доказательство теоремы 2

Лемма 1. *Пусть G – расщепляемое расширение конечной группы A с помощью группы B. Если в группе B все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы, то и в группе G все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы.*

Доказательство. Так как конечная группа является конечно порожденной и так как в ней все подгруппы финитно отделимы, то лемма 1 является следствием сформулированной выше теоремы Аленби и Грегораса.

Лемма 2. *Пусть G – расщепляемое расширение группы A с помощью группы B и в группе B все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы. Пусть H – подгруппа (циклическая подгруппа, конечно порожденная подгруппа) группы G. И пусть для произвольного элемента g из группы G, не принадлежащего подгруппе H, существует нормальная подгруппа N группы G, лежащая в группе A и имеющая конечный индекс в A такая, что $g \notin HN$. Тогда подгруппа H финитно отделима в G.*

Доказательство. Рассмотрим элемент $g \in G$, не принадлежащий подгруппе H. Тогда по условию леммы 2 для него существует нормальная подгруппа N, лежащая в группе A и имеющая конечный индекс в A такая, что $g \notin HN$. Рассмотрим факторгруппу G/N и естественный гомоморфизм $\varphi : G \longrightarrow G/N$. Так как элемент g не принадлежит подгруппе HN, то $gN \notin HN/N$, т. е. $g\varphi \notin H\varphi$.

Поскольку G – расщепляемое расширение группы A с помощью группы B и подгруппа N содержится в A, то легко видеть, что G/N является расщепляемым

расширением группы A/N с помощью группы BN/N, причем группа A/N – конечна, так как подгруппа N имеет конечный индекс в группе A. Так как группа BN/N изоморфна группе B, то по условию леммы 2 в группе BN/N все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы. Поэтому, в силу леммы 1, в группе G/N все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы.

Отсюда и из того, что HN/N наследует от подгруппы H свойства цикличности и конечной порожденности, следует, что HN/N финитно отделима в G/N, т. е. $H\varphi$ финитно отделима в G/N. А поскольку $g\varphi \notin H\varphi$, то существует гомоморфизм ψ группы G/N в некоторую конечную группу такой, что $(g\varphi)\psi \notin (H\varphi)\psi$. Таким образом, подгруппа H финитно отделима в группе G. Лемма доказана.

Лемма 3. *Если в группе все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы, то в любой ее подгруппе все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы.*

Доказательство. Пусть в группе G все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) финитно отделимы, H – подгруппа группы G. И пусть M – подгруппа (циклическая подгруппа, конечно порожденная подгруппа) группы H. Докажем, что подгруппа M финитно отделима в H.

Возьмем элемент $h \in H$, не принадлежащий подгруппе M. Так как M – подгруппа группы G, то по условию леммы 3 M финитно отделима в группе G. Значит, существует гомоморфизм φ группы G в некоторую конечную группу такой, что $h\varphi \notin M\varphi$. Пусть φ_1 – ограничение гомоморфизма φ на подгруппу H. Тогда φ_1 – гомоморфизм группы H на конечную группу и $h\varphi_1 \notin M\varphi_1$. Значит, подгруппа M финитно отделима в H. Лемма доказана.

Лемма 4. *Пусть G – группа. И пусть A, H – подгруппы группы G. Если подгруппа H финитно отделима в группе G, то $A \cap H$ финитно отделима в A.*

Доказательство. Пусть подгруппа H финитно отделима в группе G. Докажем, что $A \cap H$ финитно отделима в A. Возьмем элемент g из подгруппы A такой, что $g \notin A \cap H$. Тогда $g \notin H$. Подгруппа H финитно отделима в G, значит, существует гомоморфизм φ группы G на некоторую конечную группу X такой, что $g\varphi \notin H\varphi$. Следовательно, $g\varphi \notin (A \cap H)\varphi$. Пусть φ_1 – ограничение гомоморфизма φ на подгруппу A. Тогда φ_1 – гомоморфизм группы A в конечную группу X и $g\varphi_1 \notin (A \cap H)\varphi_1$. Таким образом получаем, что $A \cap H$ финитно отделима в A.

Теперь перейдем непосредственно к доказательству теоремы 2.

Докажем сначала достаточность в первом и втором утверждениях теоремы 2. Предположим, что в группе B финитно отделимы все подгруппы (все циклические подгруппы, все конечно порожденные подгруппы) и что в группе A финитно отделимы все подгруппы (все циклические подгруппы, все подгруппы, высекаемые в A конечно порожденными подгруппами группы G).

Возьмем произвольную подгруппу (циклическую подгруппу, конечно порожденную подгруппу) H группы G и элемент $g \in G \setminus H$. Докажем, что существует такая нормальная подгруппа N группы G, лежащая в A и имеющая конечный индекс в

A, что $g \notin HN$. Тогда в силу леммы 2 получим, что подгруппа H будет финитно отделима в G, и тем самым будет доказана достаточность в теореме.

Если элемент g не принадлежит подгруппе HA, тогда искомой подгруппой N является подгруппа A.

Пусть теперь элемент $g \in HA$, тогда он представим в виде $g = ha$, где $h \in H$, $a \in A$. Заметим, что элемент a не принадлежит подгруппе $H_1 = H \cap A$. Покажем, что подгруппа H_1 финитно отделима в A.

Если подгруппа H произвольная, то и H_1 – произвольная подгруппа группы A, и, следовательно, по условию теоремы H_1 финитно отделима в A.

Если подгруппа H циклическая, то подгруппа H_1 наследует от H свойство цикличности. Тогда в силу условия теоремы, H_1 финитно отделима в A.

Если подгруппа H конечно порожденная, то подгруппа H_1, являющаяся пересечением подгруппы A с конечно порожденной подгруппой H, финитно отделима в A в силу нашего предположения.

Таким образом, имеем, что подгруппа H_1 финитно отделима в A, и элемент a не принадлежит подгруппе H_1. Поэтому элемент $a \notin H_1N$ для некоторой нормальной подгруппы N конечного индекса группы A.

Так как для любого натурального числа n число всех подгрупп группы A индекса n конечно, то любая подгруппа конечного индекса группы A содержит в себе характеристическую подгруппу конечного индекса. Поэтому без потери общности можем считать подгруппу N характеристической в A. А поскольку A нормальна в G, то и N нормальна в группе G. Остается доказать, что $g \notin HN$.

Если, напротив, элемент g принадлежит подгруппе HN, тогда элемент g представим в виде $g = h_1x$, где $h_1 \in H$, $x \in N$. С другой стороны, $g = ha$, где $h \in H$, $a \in A$. Тогда $g = ha = h_1x$. Следовательно, получаем равенство $h^{-1}h_1 = ax^{-1}$, где $h^{-1}h_1 \in H$, $ax^{-1} \in A$. Значит, $h^{-1}h_1 \in H_1$ и поэтому элемент $a = h^{-1}h_1x \in H_1N$, но это противоречит выбору подгруппы N. Следовательно, $g \notin HN$, и подгруппа N является искомой. Таким образом, достаточность в теореме доказана.

Необходимость в первом и во втором утверждениях теоремы 2 обеспечивается леммами 3 и 4. Тем самым теорема полностью доказана.

Автор благодарен Д. Н. Азарову за помощь при написании этой статьи.

Список литературы / References

[1] Мальцев А. И., “О гомоморфизмах на конечные группы”, *Учен. зап. Иван. гос. пед. ин-та.*, **18(5)** (1958), 49–60; (Mal’cev A. I., “O gomomorfizmah na konechnye gruppy”, *Uchen. zap. Ivan. gos. ped. in-ta*, **18(5)** (1958), 49–60, [in Russian].)

[2] Lennox J., Robinson D., *The theory of infinite soluble groups*, Clarendon Press, Oxford, 2004.

[3] Burns R. C., “On finitely generated subgroups of free products”, *J. Austral. Math. Soc.*, **12** (1971), 358–364.

[4] Hall M., “Coset representations in free groups”, *Trans. Amer. Math. Soc.*, **67** (1949), 421–432.

[5] Rips E., “An example of a non-LERF group which is a free ptoduct of LERF groups with an amalgamated cyclic subgroup”, *Israel J. of math.*, **70**:1 (1990), 104–110.

[6] Allenby R., Doniz D., “A free product of finitely generated nilpotent groups amalgamating a cycle that is not subgroup separable”, *Proc. Amer. Math. Soc.*, **124**:4 (1996), 1003–1005.

[7] Allenby R., Gregorac R., "On locally extended residually finite groups", *Lecture Notes Math.*, **319** (1973), 9–17.

[8] Allenby R., Tang C., "Subgroup separability of generalized free products of freeby-finite groups", *Canad. Math. Bull.*, **36**:4 (1993), 385–389.

[9] Bruuner R. M., Burns R. G., Solitar D., "The subgroup separability of free products of two free groups with cyclic amalgamation", *Contributions to group theory. Contemp. Math.*, **33** (1984), 90–115.

[10] Baumslag G., "On the residual finiteness of generalized free products of nilpotent groups", *Trans. Amer. Math. Soc.*, **106**:2 (1963), 193–209.

[11] Романовский Н. С., "О финитной аппроксимируемости свободных произведений относительно вхождения", *Известия АН СССР. Сер. Мат.*, **33**:6 (1969), 1324–1329; Romanovskij N. S., "O finitnoj approksimiruemosti svobodnyh proizvedenij otnositel'no vhozhdenija", *Izvestija AN SSSR. Ser. Mat.*, **33**:6 (1969), 1324–1329, [in Russian].)

[12] Молдаванский Д. И., Ускова А. А., "О финитной отделимости подгрупп обобщенных свободных произведений групп", *Чебышевский сб.*, **14**, 2013, 81–87; (Moldavanskij D. I., Uskova A. A., "O finitnoj otdelimosti podgrupp obobshhennyh svobodnyh proizvedenij grupp", *Chebyshevskij sb.*, **14**, 2013, 81–87, [in Russian].)

[13] Stebe D., "Residual finiteness of a class of knot groups", *Comm. Pure and Applied Math.*, **21** (1968), 563–583.

[14] Логинова Е. Д., "Финитная отделимость циклических подгрупп свободного произведения двух групп с коммутирующими подгруппами", *Науч. тр. Иван. гос. ун-та*, Математика, **3**, 2000, 49–55; (Loginova E. D., "Finitnaja otdelimost' ciklicheskih podgrupp svobodnogo proizvedenija dvuh grupp s kommutirujushhimi podgruppami", *Nauch. tr. Ivan. gos. un-ta*, Matematika, **3**, 2000, 49–55, [in Russian].)

[15] Соколов Е. В., "Финитная отделимость циклических подгрупп в некоторых обобщенных свободных произведениях групп", *Вестник молодых ученых ИвГУ*, **2** (2002), 7–10; (Sokolov E. V., "Finitnaja otdelimost' ciklicheskih podgrupp v nekotoryh obobshhennyh svobodnyh proizvedenijah grupp", *Vestnik molodyh uchenyh IvGU*, **2** (2002), 7–10, [in Russian].)

[16] Азаров Д. Н., "О группах конечного общего ранга", *Вестн. Иван. гос. ун-та*, **3** (2004), 100–103; (Azarov D. N., "O gruppah konechnogo obshego ranga", *Vestn. Ivan. gos. un-ta*, **3** (2004), 100–103, [in Russian].)

[17] Азаров Д. Н., "О финитной аппроксимируемости HNN-расширений и обобщенных свободных произведений групп конечного ранга", *Сиб. мат. журн.*, **54**:6 (2013), 1203–1215; (Azarov D. N., "O finitnoj approksimiruemosti HNN-rasshirenij i obobshhennyh svobodnyh proizvedenij grupp konechnogo ranga", *Sib. mat. zhurn.*, **54**:6 (2013), 1203–1215, [in Russian].)

[18] Азаров Д. Н., "О почти аппроксимируемости конечными p-группами", *Чебышевский сб.*, **11**, 2010, 11–21; (Azarov D. N., "O pochti approksimiruemisti konechnymi p-gruppami", *Chebyshevskij sb.*, **11**, 2010, 11–21, [in Russian].)

[19] Азаров Д. Н., Чуракова Е. И., "Об аппроксимируемости конечными p-группами некоторых расщепляемых расширений", *Вестн. Иван. гос. ун-та*, **2** (2009), 68–71; (Azarov D. N., Churakova E. I., "Ob approksimiruemosti konechnymi p-gruppami nekotoryh rasshhepljaemyh rasshirenij", *Vestn. Ivan. gos. un-ta*, **2** (2009), 68–71, [in Russian].)

[20] Курош А. Г., *Теория групп*, Наука, М., 1967; (Kurosh A. G., *Teorija grupp*, Nauka, M., 1967, [in Russian].)

DOI: 10.18255/1818-1015-2015-4-500-506

On Residual Separability of Subgroups in Split Extensions

Krjazheva A. A.

Received April 21, 2015

In 1973, Allenby and Gregoras proved the following statement. Let G be a split extension of a finitely generated group A by the group B. 1) If in groups A and B all subgroups (all cyclic subgroups) are finitely separable, then in group G all subgroups (all cyclic subgroups) are finitely separable; 2) if in group A all subgroups are finitely separable, and in group B all finitely generated subgroups are finitely separable, then in group G all finitely generated subgroups are finitely separable. Recall that a group G is said to be a split extension of a group A by a group B, if the group A is a normal subgroup of G, B is a subgroup of G, $G = AB$ and $A \cap B = 1$. Recall also that the subgroup H of a group G is called finitely separable if for every element g of G, which does not belong to the subgroup H, there exists a homomorphism of G on a finite group in which the image of an element g does not belong to the image of the subgroup H. In this paper we obtained a generalization of the Allenby and Gregoras theorem by replacing the condition of the finitely generated group A by a more general one: for any natural number n the number of all subgroups of the group A of index n is finite. In fact, under this condition we managed to obtain a necessary and sufficient condition for finite separability of all subgroups (of all cyclic subgroups, of all finitely generated subgroups) in the group G.

Keywords: split extensions, finitely separable subgroups, finitely generated group

For citation: Krjazheva A. A., "On Residual Separability of Subgroups in Split Extensions", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 500–506.

On the authors:
Krjazheva Anastasia Alekseevna, orcid.org/0000-0002-5985-5172, graduate student,
Voronezh State Technology University,
Universitetskaya str., 7, Voronezh, 394016, Russia, e-mail: Stasia.07.10@mail.ru

Acknowledgments:

This work was supported by the Federal targeted Program.

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 507–520
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 507–520

DOI: 10.18255/1818-1015-2015-4-507-520

УДК 517.9

О выразительности подхода к построению ПЛК-программ по LTL-спецификации

Кузьмин Е. В.[1], Рябухин Д. А., Соколов В. А.

получена 3 августа 2015

Статья посвящена подходу к построению и верификации «дискретных» программ логических контроллеров (ПЛК) по LTL-спецификации. Этот подход обеспечивает возможность анализа корректности программ логических контроллеров с помощью метода проверки модели (Model Checking). В рамках подхода в качестве языка спецификации программного поведения используется язык темпоральной логики LTL. Анализ корректности LTL-спецификации относительно программных свойств производится автоматически с помощью программного средства символьной проверки модели Cadence SMV.

В статье демонстрируется состоятельность подхода к построению и верификации ПЛК-программ по LTL-спецификации с точки зрения тьюринговой мощности. Доказывается, что в соответствии с этим подходом для произвольной счётчиковой машины Минского может быть построена LTL-спецификация, по которой осуществляется её программная реализация на любом из языков программирования ПЛК стандарта МЭК 61131-3. Поскольку счётчиковые машины Минского равномощны машинам Тьюринга, то и рассматриваемый подход к программированию ПЛК будет обладать тьюринговой мощностью.

В доказательстве основное внимание уделяется заданию поведения счётчиковой машины в виде набора LTL-формул и сопоставлению этим формулам конструкций языков ST и SFC. SFC представляет интерес с точки зрения специфики графического языка, а язык ST рассматривается в качестве базового в том смысле, что реализация счётчиковой машины на языках IL, FBD/CFC и LD сводится к переписыванию на них конструкций ST-программы.

Идея доказательства демонстрируется на примере трехсчетчиковой машины Минского, реализующей функцию возведения числа в квадрат.

Ключевые слова: программируемые логические контроллеры (ПЛК), построение и верификация ПЛК-программ, LTL-спецификация, счётчиковые машины Минского

Для цитирования: Кузьмин Е. В., Рябухин Д. А., Соколов В. А., "О выразительности подхода к построению ПЛК-программ по LTL-спецификации", *Моделирование и анализ информационных систем*, **22**:4 (2015), 507–520.

Об авторах:
Кузьмин Егор Владимирович, orcid.org/0000-0003-0500-306X, доктор физ.-мат. наук, доцент, профессор кафедры теоретической информатики, Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: kuzmin@uniyar.ac.ru

Рябухин Дмитрий Александрович, orcid.org/0000-0002-0799-8868, аспирант,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: dmitriy_ryabukhin@mail.ru

Соколов Валерий Анатольевич, orcid.org/0000-0003-1427-4937, доктор физ.-мат. наук, профессор, зав. кафедрой теоретической информатики, Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: sokolov@uniyar.ac.ru

Благодарности:

[1]Работа проводилась при финансовой поддержке РФФИ, грант №12-01-00281-а.

Введение

Статья посвящена предложенному ранее [1–5, 11–13] подходу к построению и верификации «дискретных» программ логических контроллеров по LTL-спецификации. Этот подход обеспечивает возможность анализа корректности программ логических контроллеров с помощью метода проверки модели (Model Checking) [8, 9]. В рамках подхода в качестве языка спецификации программного поведения используется язык темпоральной логики LTL. Анализ корректности LTL-спецификации относительно программных свойств производится автоматически с помощью программного средства символьной проверки модели Cadence SMV [16].

Программируемый логический контроллер (ПЛК) — «реагирующая» система, имеющая множество входов, подключаемых посредством датчиков к объекту управления, и множество выходов, подключаемых к исполнительным устройствам [7, 14]. Программа ПЛК выполняется в рабочем цикле: считывание входов, выполнение программы и выставление выходов. Применение ПЛК в системах управления сложными производственными процессами предъявляет строгие требования корректности к программам ПЛК.

Языки программирования ПЛК определяются стандартом МЭК 61131-3. Этот стандарт включает в себя описание пяти языков: SFC, IL, ST, LD и FBD. Язык IL (Instruction list) — ассемблер с аккумулятором и переходами по меткам. Язык ST (Structured Text) — высокоуровневый язык, синтаксически представляющий собой несколько адаптированный язык Паскаль. Язык релейных диаграмм LD (Ladder Diagram) — графический язык, реализующий структуры электрических цепей. FBD (Function Block Diagram) — графический язык диаграмм принципиальных схем электронных устройств на микросхемах. Разновидностью FBD является язык непрерывных функциональных схем CFC (Continuous Function Chart), позволяющий свободно размещать компоненты и соединения. SFC (Sequential Function Chart) — последовательные функциональные схемы. Диаграммы SFC являются высокоуровневым графическим инструментом, они состоят из шагов и переходов между ними, которые разделяют задачи на простые этапы с формально определенной логикой работы системы. Разрешение перехода определяется условием. С шагом связаны определенные действия, которые описываются на любом из языков МЭК 61131-3.

В данной статье демонстрируется состоятельность подхода к построению и верификации ПЛК-программ по LTL-спецификации с точки зрения тьюринговой мощности. Доказывается, что в соответствии с этим подходом для произвольной счётчиковой машины Минского может быть построена LTL-спецификация, по которой осуществляется её программная реализация на любом из языков программирования ПЛК стандарта МЭК 61131-3. Поскольку счётчиковые машины Минского равномощны машинам Тьюринга [6, 15], то и рассматриваемый подход к программированию ПЛК будет обладать тьюринговой мощностью. В доказательстве основное внимание уделяется заданию поведения счётчиковой машины в виде набора LTL-формул и сопоставлению этим формулам конструкций языков ST и SFC. SFC представляет интерес с точки зрения специфики графического языка, а язык ST рассматривается в качестве базового в том смысле, что реализация счётчиковой машины на языках IL, FBD/CFC и LD сводится к переписыванию на них конструкций ST-программы. Идея доказательства демонстрируется на примере.

1. Основные понятия и определения

1.1. Построение программ ПЛК по LTL-спецификации

Смысл концепции программирования ПЛК по LTL-спецификации [1–5,11–13] состоит в обеспечении возможности анализа корректности ПЛК-программ методом проверки модели [8,9]. В соответствии с этой концепцией значение каждой переменной должно изменяться только в одном месте программы и не более одного раза за одно полное выполнение программы при прохождении рабочего цикла ПЛК. Поэтому изменение значения каждой программной переменной задаётся с помощью пары явных LTL-формул и одной неявной. Первая LTL-формула описывает ситуации, при которых происходит возрастание значения соответствующей переменной, вторая LTL-формула определяет условия, приводящие к уменьшению значения переменной. Третья LTL-формула в явном виде не прописывается, поскольку всегда имеет жёсткую конструкцию, составленную из элементов первых двух формул. Она описывает условия, при которых переменная не меняет своего значения за один проход рабочего цикла ПЛК. Рассматриваемые для спецификации поведения переменных LTL-формулы являются конструктивными в том смысле, что по ним производится построение ПЛК-программы, которая соответствует темпоральным свойствам, выраженным этими формулами. Таким образом, программирование ПЛК сводится к построению LTL-спецификации поведения каждой программной переменной.

Для описания ситуаций, которые приводят к увеличению и уменьшению значения целочисленной переменной V, используются LTL-формулы вида

$$\mathbf{G}\,\mathbf{X}(\ V{>}\_V \Rightarrow \mathit{OldValCond}\ \wedge \mathit{FiringCond}\ \wedge V = \mathit{NewValExpr}\ ); \tag{1}$$

$$\mathbf{G}\,\mathbf{X}(\ V{<}\_V \Rightarrow \mathit{OldValCond}' \wedge \mathit{FiringCond}' \wedge V = \mathit{NewValExpr}'\ ). \tag{2}$$

Символ лидирующего подчеркивания «_» в обозначении переменной $\_V$ воспринимается как псевдооператор, позволяющий обратиться к значению переменной V, которое она имела в предыдущем состоянии (после последнего полного прохода рабочего цикла ПЛК). При этом псевдооператор может использоваться только под действием темпорального оператора **X**.

Условия *FiringCond* и *OldValCond* являются логическими выражениями над программными переменными и константами, которые строятся с применением операторов сравнения, логических и арифметических операторов и псевдооператора «_» (который по определению может быть применим только к переменным). Выражение *FiringCond* описывает ситуации, при которых возникает необходимость изменения значения переменной V, если это, конечно, допускается условием *OldValCond*. Выражение *NewValExpr*, определяющее новое значение переменной V, строится с помощью переменных и констант, операторов сравнения, логических, арифметических операторов и псевдооператора «_».

Для описания всех возможных ситуаций, при которых происходит возрастание значения переменной V, в формуле (1) после символа оператора импликации $\Rightarrow$ может потребоваться несколько наборов рассмотренных конъюнктивных членов $\mathit{OldValCond}_i \wedge \mathit{FiringCond}_i \wedge V = \mathit{NewValExpr}_i$ объединенных в дизъюнкцию.

Аналогичный смысл имеют выражения $\mathit{FiringCond}'$, $\mathit{OldValCond}'$ и $\mathit{NewValExpr}'$.

В случае с переменной логического типа данных для спецификации ее поведения используются более простые LTL-формулы:

$\mathbf{G}\,\mathbf{X}(\ \neg\_V \wedge \ \ V \Rightarrow \mathit{FiringCond}\ )$;

$\mathbf{G}\,\mathbf{X}(\ \ \_V \wedge \neg V \Rightarrow \mathit{FiringCond}'\,)$;

которые означают, что всякий раз, когда новое значение переменной V оказывается больше или меньше ее предыдущего значения, записанного в переменной $\_V$, из этого следует, что было выполнено условие соответствующего внешнего воздействия $\mathit{FiringCond}$ или $\mathit{FiringCond}'$.

Неявная LTL-формула сохранения прежнего значения переменной V имеет вид

$$\mathbf{G}\,\mathbf{X}(\ V{=}\_V \Rightarrow \neg(\mathit{OldValCond} \wedge \mathit{FiringCond}) \wedge \neg(\mathit{OldValCond}' \wedge \mathit{FiringCond}')\ ). \qquad (3)$$

Для логической переменной V эта неявная LTL-формула выглядит как

$\mathbf{G}\,\mathbf{X}(\ V{=}\_V \Rightarrow \neg(\neg\_V \wedge \mathit{FiringCond}) \wedge \neg(\_V \wedge \mathit{FiringCond}')\ )$.

При построении спецификации важно учитывать то, в каком порядке располагаются темпоральные формулы, описывающие поведение переменных. Некоторая переменная без псевдооператора «_» может быть задействована в спецификации поведения другой переменной, только если спецификация ее поведения уже произведена и находится выше по тексту.

1.2. Счётчиковая машина Минского

Счетчиковая машина Минского M представляет собой набор (q_0, q_n, Q, X, Δ), где $Q = \{q_0, \ldots, q_n\}$ — конечное непустое множество состояний машины; $q_0 \in Q$ — начальное состояние; $q_n \in Q$ — финальное состояние; $X = \{x_1, \ldots, x_m\}$ — конечное непустое множество счетчиков, которые могут принимать значения из $\mathbb{N} \cup \{0\}$; $\Delta = \{\delta_0, \ldots, \delta_{n-1}\}$ — набор правил переходов по состояниям машины; δ_i — правило переходов для состояния q_i. Состояния q_i, $0 \leqslant i \leqslant n-1$, подразделяются на два типа. Состояния первого типа имеют правила переходов вида:

$$(\delta_i)\ \ q_i\colon\ x_j := x_j + 1;\ \mathsf{goto}\ q_k,$$

где $1 \leqslant j \leqslant m, 0 \leqslant k \leqslant n$. Для состояний второго типа имеем, $1 \leqslant j \leqslant m, 0 \leqslant k, l \leqslant n$:

$$(\delta_i)\ \ q_i\colon\ \mathsf{if}\ x_j > 0\ \mathsf{then}\ (x_j := x_j - 1;\ \mathsf{goto}\ q_k)\ \mathsf{else}\ \mathsf{goto}\ q_l.$$

Для финального состояния q_n правило перехода не предусмотрено. Это означает, что при попадании в состояние q_n машина Минского M завершает свою работу.

Конфигурация машины Минского — это набор $(q_i, c_1, \ldots, c_m)$, где q_i — состояние машины, $c_1, \ldots, c_m$ — натуральные числа (включая ноль), являющиеся значениями соответствующих счетчиков.

Исполнением машины Минского называется последовательность конфигураций $s_0\, s_1\, s_2\, s_3\, s_4\, \ldots$, индуктивно определяемая в соответствии с правилами переходов. Счетчиковая машина имеет одно исполнение из начальной конфигурации s_0, так как для каждого состояния предусмотрено не более одного правила переходов. Машина, получив на вход некоторый набор значений счетчиков, стартует из состояния q_0 и либо останавливается в состоянии q_n с выходным набором значений счетчиков, либо зацикливается, реализуя тем самым частичную числовую функцию.

2. Реализация счётчиковой машины Минского

Теорема 1. *Любая счётчиковая машина Минского может быть реализована на языках программирования логических контроллеров IL, ST, FBD/CFC, LD и SFC в соответствии с подходом к построению и верификации программ логических контроллеров по LTL-спецификации.*

Доказательство. Рассмотрим произвольную счётчиковую машину Минского M. Сопоставим каждому счётчику x_j и каждому состоянию q_i машины M программную переменную `xj` типа данных «беззнаковое целое» и логическую программную переменную `qi` соответственно. При этом все переменные-состояния, кроме `q0`, инициализируются нулём, а переменной-состоянию `q0` изначально присваивается логическое значение 1 (логическая истина). Переменные-счётчики при инициализации получают начальные значения соответствующих счётчиков машины M. Реализация машины Минского M осуществляется в виде программы на языке программирования логических контроллеров. Переход из одной конфигурации машины в другую будет соответствовать исполнению программы в рамках одного прохода рабочего цикла ПЛК.

В соответствии с подходом к построению и верификации программ по LTL-спецификации доказательство теоремы проводится в два этапа. Первый этап предполагает создание LTL-спецификации счётчиковой машины Минского M. Второй этап – построение по этой LTL-спецификации ПЛК-программ (на разных языках программирования), реализующих поведение машины M.

Первый этап. Запишем требуемое поведение каждой переменной программной реализации счётчиковой машины M с помощью пары LTL-формул.

Начнём с описания поведения переменных-счётчиков. LTL-формула, учитывающая ситуации, которые приводят к увеличению значения переменной-счётчика `xj`, имеет следующий вид:

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} > \texttt{\_xj} \ \Rightarrow\ (\texttt{\_qi} \vee \texttt{\_qr} \vee \cdots \vee \texttt{\_qs}) \wedge \texttt{xj=\_xj+1}\ ),$$

где переменные `_qi`, `_qr`, …, `_qs` соответствуют таким машинным состояниям первого типа q_i, q_r, …, q_s, в правилах перехода которых участвует счётчик x_j. Например, для состояния первого типа q_i в описании счётчиковой машины M должно быть правило перехода (δ_i) $q_i\colon\ x_j := x_j + 1;$ goto q_k, которое порождает условие LTL-спецификации вида `_qi` $\wedge$ `xj=_xj+1`.

Если в описании машины M для счётчика x_j правил перехода первого типа нет, то LTL-формула «возрастания» для программной переменной `xj` следующая:

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} > \texttt{\_xj} \ \Rightarrow\ \text{false}\ ).$$

LTL-формула, учитывающая ситуации, которые приводят к уменьшению значения переменной-счётчика `xj`, имеет вид

$$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} < \texttt{\_xj} \ \Rightarrow\ (\texttt{\_qi} \vee \texttt{\_qr} \vee \cdots \vee \texttt{\_qs}) \wedge \texttt{\_xj}>0 \wedge \texttt{xj=\_xj}-1\ ),$$

где переменные `_qi`, `_qr`, …, `_qs` соответствуют машинным состояниям второго типа q_i, q_r, …, q_s, в правилах перехода которых участвует счётчик x_j, т. е. для состояния второго типа q_i в описании машины M должно быть правило перехода (δ_i) $q_i\colon$ if $x_j > 0$ then $(x_j := x_j - 1;$ goto $q_k)$ else goto q_l, которое порождает условие LTL-спецификации вида `_qi` $\wedge$ `_xj`$>0 \wedge$ `xj=_xj`-1.

Если в описании машины M для счётчика x_j правил перехода второго типа нет, то LTL-формула «убывания» для программной переменной `xj` строится просто как

$\mathbf{G}\,\mathbf{X}(\ \texttt{xj} < \texttt{\_xj} \ \Rightarrow\ \text{false}\ )$.

Теперь опишем построение LTL-спецификации поведения программных переменных-состояний. LTL-формула, учитывающая условия, при которых счётчиковая машина M переходит из некоторого текущего состояния в новое состояние q_k, т. е. логическая программная переменная `qk` получает значение 1, имеет следующий вид:

$\mathbf{G}\,\mathbf{X}(\ \neg\texttt{\_qk} \wedge \texttt{qk} \ \Rightarrow\ \texttt{\_qi} \wedge \texttt{\_xj}{>}0 \vee \cdots \vee \texttt{\_qr} \wedge \neg(\texttt{\_xt}{>}0) \vee \cdots \vee \texttt{\_qs}\ )$,

где условия вида $\texttt{\_qi} \wedge \texttt{\_xj}{>}0$ соответствуют правилам перехода второго типа

$(\delta_i)\ \ q_i\colon\ \mathsf{if}\ x_j > 0\ \mathsf{then}\ (x_j := x_j - 1;\ \mathsf{goto}\ \mathbf{q_k})\ \mathsf{else\ goto}\ q_l$,

а условия вида $\texttt{\_qr} \wedge \neg(\texttt{\_xt}{>}0)$ – правилам

$(\delta_r)\ \ q_r\colon\ \mathsf{if}\ x_t > 0\ \mathsf{then}\ (x_t := x_t - 1;\ \mathsf{goto}\ q_l)\ \mathsf{else\ goto}\ \mathbf{q_k}$.

Условия с одной программной переменной вида `_qs` соответствуют правилам перехода первого типа $(\delta_s)\ \ q_s\colon\ x_g := x_g + 1;\ \mathsf{goto}\ \mathbf{q_k}$.

Важно отметить, что все программные переменные приведённой LTL-формулы, стоящие в условиях после оператора импликации, должны быть отличными от переменной-состояния `qk`, т. е. переход в то же самое состояние не специфицируется.

Если в состояние q_k машины M нет ни одного перехода, то для переменной `qk` LTL-формула «активации» строится как

$\mathbf{G}\,\mathbf{X}(\ \neg\texttt{\_qk} \wedge \texttt{qk} \ \Rightarrow\ \text{false}\ )$.

LTL-формула деактивации (выхода из) состояния q_k, которому соответствует правило перехода без петель (только с переходами в другие состояния), для программной переменной `qk` имеет простой вид:

$\mathbf{G}\,\mathbf{X}(\ \texttt{\_qk} \wedge \neg\texttt{qk} \ \Rightarrow\ \text{true}\ )$,

где логическая константа true означает, что переменная `qk` должна быть сброшена в ноль уже на следующем проходе рабочего ПЛК после её активации, т. е. после присваивания значения 1. Несмотря на то, что импликация внутри LTL-формулы является тавтологией, построение этой формулы имеет смысл, поскольку условие срабатывания в виде константы true участвует в «неявной» LTL-формуле, описывающей ситуации, при которых переменная сохраняет своё значение после одного исполнения программы. Эта неявная LTL-формула представлена ниже и по сути запрещает переменной `qk` иметь значение 1 дольше одного рабочего цикла программы:

$\mathbf{G}\,\mathbf{X}(\ \texttt{\_qk} = \texttt{qk} \ \Rightarrow\ \neg(\neg\texttt{\_qk} \wedge \mathit{FiringCond}) \wedge \neg(\texttt{\_qk} \wedge \text{true})\ )$.

Если правило перехода для q_k имеет петлю $(\delta_k)\ \ \mathbf{q_k}\colon\ x_h := x_h + 1;\ \mathsf{goto}\ \mathbf{q_k}$ или две петли $(\delta_k)\ \ \mathbf{q_k}\colon\ \mathsf{if}\ x_h > 0\ \mathsf{then}\ (x_h := x_h - 1;\ \mathsf{goto}\ \mathbf{q_k})\ \mathsf{else\ goto}\ \mathbf{q_k}$, или же состояние q_k является финальным состоянием, то программная переменная `qk` будет иметь следующую LTL-формулу «деактивации»:

$\mathbf{G}\,\mathbf{X}(\ \texttt{\_qk} \wedge \neg\texttt{qk} \ \Rightarrow\ \text{false}\ )$.

Если состояние второго типа q_k имеет в правиле перехода с переменной x_h только одну петлю, то для программной переменной `qk` выбирается соответствующий вариант LTL-формулы

$\mathbf{G}\,\mathbf{X}(\ \texttt{\_qk} \wedge \neg\texttt{qk} \ \Rightarrow\ \texttt{\_xh}{>}0\ )$ или $\mathbf{G}\,\mathbf{X}(\ \texttt{\_qk} \wedge \neg\texttt{qk} \ \Rightarrow\ \neg(\texttt{\_xh}{>}0)\ )$.

Второй этап. Имея спецификацию поведения каждой переменной-счётчика и каждой переменной-состояния в виде пары LTL-формул, опишем сначала способ построения программной реализации счётчиковой машины M на языке ST.

Если счётчик x_j машины Минского M участвует в правилах перехода обоих типов, LTL-спецификация поведения программной переменной-счётчика `xj` будет иметь следующий вид:

$\mathbf{G}\,\mathbf{X}($ `xj>_xj` $\Rightarrow$ (`_qi` $\vee$ `_qk` $\vee \cdots \vee$ `_ql`) $\wedge$ `xj=_xj+1` $)$;
$\mathbf{G}\,\mathbf{X}($ `xj<_xj` $\Rightarrow$ (`_qr` $\vee$ `_qs` $\vee \cdots \vee$ `_qt`) $\wedge$ `_xj>0` $\wedge$ `xj=_xj-1` $)$.

По этой паре LTL-формул с учётом третьей неявной LTL-формулы спецификации может быть построен следующий программный код на языке ST:

```
IF    (_qi OR _qk OR ... OR _ql)           THEN xj:=_xj+1;
ELSIF (_qr OR _qs OR ... OR _qt) AND _xj>0 THEN xj:=_xj-1; END_IF;.
```

Если одна из формул LTL-спецификации имеет вид $\mathbf{G}\,\mathbf{X}($ `xj>_xj` $\Rightarrow$ false $)$ или $\mathbf{G}\,\mathbf{X}($ `xj<_xj` $\Rightarrow$ false $)$, то соответствующая ветка блока IF-ELSIF не строится. Если обе формулы такого вида, то им блок IF-ELSIF не сопоставляется.

Если состояние q_k счётчиковой машины M имеет входы и выходы без петель, то LTL-спецификация поведения программной переменной-состояния `qk` выглядит следующим образом:

$\mathbf{G}\,\mathbf{X}($ $\neg$`_qk` $\wedge$ `qk` $\Rightarrow$ `_qi` $\wedge$ `_xj>0` $\vee \cdots \vee$ `_qr` $\wedge \neg($`_xt>0`$) \vee \cdots \vee$ `_qs` $)$;
$\mathbf{G}\,\mathbf{X}($ `_qk` $\wedge \neg$`qk` $\Rightarrow$ true $)$.

По этим двум формулам также с учётом третьей неявной LTL-формулы спецификации строится следующий программный код на языке ST:

```
IF NOT _qk AND (_qi AND     _xj>0  OR ... OR
                _qr AND NOT(_xt>0) OR ... OR _qs) THEN qk:=1;
ELSIF  _qk                                        THEN qk:=0; END_IF;.
```

Для формул вида $\mathbf{G}\,\mathbf{X}($ $\neg$`_qk` $\wedge$ `qk` $\Rightarrow$ false $)$ или $\mathbf{G}\,\mathbf{X}($ `_qk` $\wedge \neg$`qk` $\Rightarrow$ false $)$ соответствующие ветки IF-ELSIF блока не строятся. Если из состояния q_k нельзя перейти в отличное от него состояние и не существует перехода в него из другого состояния, то переменной-состоянию `qk` ST-код не сопоставляется. В том случае, если вторая LTL-формула («деактивации») имеет вид $\mathbf{G}\,\mathbf{X}($ `_qk` $\wedge \neg$`qk` $\Rightarrow$ `_xh>0` $)$ или $\mathbf{G}\,\mathbf{X}($ `_qk` $\wedge \neg$`qk` $\Rightarrow \neg($`_xh>0`$)$ $)$, то во вторую ветку блока IF-ELSIF добавляется через оператор `AND` условие `_xh>0` или `NOT(_xh>0)` соответственно.

Отметим, что порядок IF-ELSIF блоков в ST-программе, соответствующей счётчиковой машине M, не имеет значения, поскольку новое значение каждой переменной зависит от предыдущих значений переменных, полученных на последнем проходе рабочего цикла ПЛК. После построения всех IF-ELSIF блоков (по одному на программную переменную) необходимо в конец ST-программы добавить псевдооператорный раздел, реализующий оператор лидирующего подчёркивания «_»:

```
_x1:=x1; _x2:=x2; ... _xm:=xm;
_q0:=q0; _q1:=q1; ... _qn:=qn;.
```

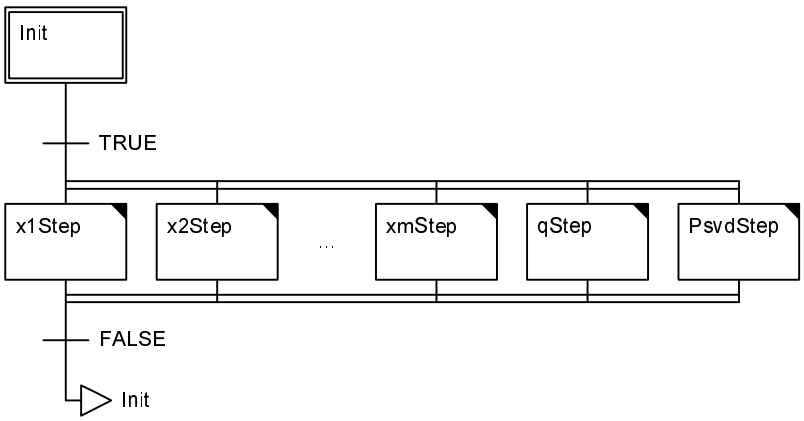
Таким образом, ST-программа машины M будет содержать переменные-счётчики `x1`, `x2`, ..., `xm`, переменные-состояния `q0`, `q1`, ..., `qn` и псевдооператорные переменные `_x1`, `_x2`, ..., `_xm`, `_q0`, `_q1`, ..., `_qn`. При этом переменная-состояние `q0` инициализируется единицей, а переменные-счётчики при инициализации получают

начальные значения соответствующих счётчиков машины M. Остальные переменные, включая псевдооператорные переменные, инициализируются нулём.

Для языков IL, FBD/CFC и LD ограничимся лишь замечанием, что реализация машины M на этих языках главным образом (с небольшими поправками на специфику) сводится к переписыванию на них IF-ELSIF блоков и псевдооператорного раздела ST-программы.

Рассмотрим реализацию машины M на графическом языке последовательных функциональных схем SFC. В качестве языка SFC будем рассматривать упрощенный SFC, имеющийся, например, в инструментальном средстве программирования CoDeSys [10]. В упрощенном SFC каждому шагу могут быть сопоставлены действия трех типов – текущее, входное и выходное. Шаги, содержащие действие, на схеме отличаются тем, что верхний правый угол прямоугольника шага закрашен. Пока шаг активен, текущее действие будет выполняться один раз в каждом рабочем цикле. Выход из шага, т. е. его деактивация, осуществляется при выполнении условий на переходе из шага. Входное действие обозначается сегментом «E» (Entry) в нижнем левом углу прямоугольника шага и выполняется однократно при активации шага. Выходное действие обозначается сегментом «X» (eXit) в нижнем правом углу прямоугольника шага и выполняется однократно при завершении работы шага (при выходе из шага, но на следующем проходе рабочего цикла до активации следующего шага). Каждое из действий, а также громоздкие условия на переходах, могут быть реализованы на любом из языков стандарта МЭК 61131-3.

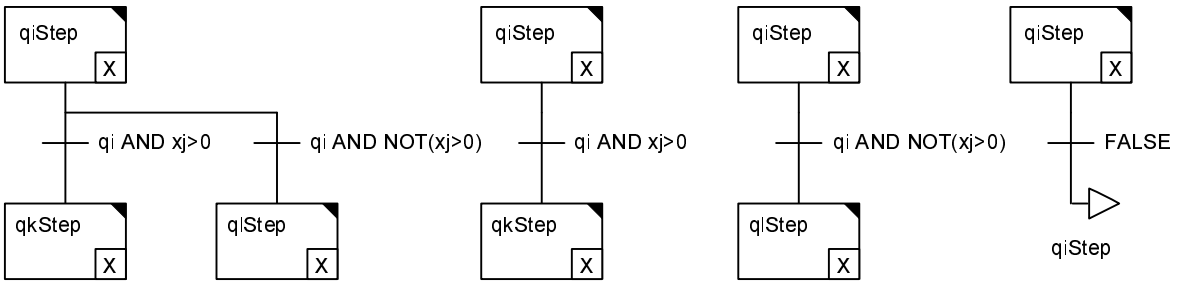
Реализация машины M на упрощенном SFC состоит из двух схем – основной и вложенной. Основная SFC-схема представлена ниже.



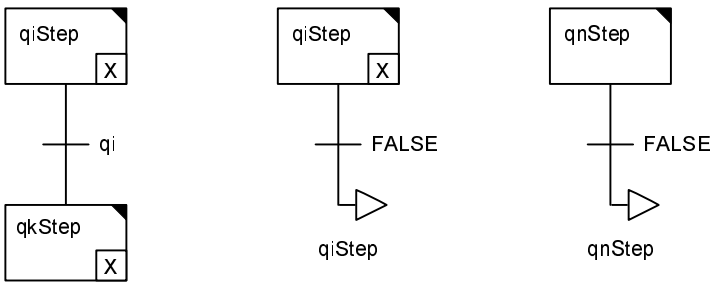
Действию шага x1Step соответствует программная реализация для пары LTL-формул спецификации поведения переменной-счетчика `x1`. Например, действием этого шага может быть уже рассмотренная IF-ELSIF конструкция на языке ST. Аналогичным образом действиям шагов x2Step, ..., xmStep сопоставляются программные реализации для пар LTL-формул спецификации поведения переменных-счетчиков `x2`, ..., `xm` соответственно. Действием шага PsvdStep является весь псевдооператорный раздел, поскольку этот шаг в рамках одного прохода рабочего цикла в программе будет выполняться последним.

Действию шага qStep соответствует вложенная SFC-схема, которая строится из следующих фрагментов с привязкой к переменным-состояниям. Для состояния

второго типа q_i машины M в зависимости от наличия петель в правилах перехода имеем фрагменты (для правил без петель, с одной петлёй и с двумя петлями соответственно):



Для состояния первого типа q_i машины M в зависимости от наличия петли в правиле перехода, а также для финального состояния q_n, имеем фрагменты (для правила без петли, с петлёй и для финального состояния соответственно):



Начальным шагом вложенной SFC-схемы будет являться шаг q0Step. Текущему и выходному действиям шага qiStep соответствует одна и та же программная реализация (на любом из языков IL, ST, FBD/CFC и LD) пары LTL-формул спецификации поведения переменной-состояния `qi`. Выходное действие «X» (eXit) необходимо для сбрасывания значения переменой `qi` в 0 после срабатывания условия выхода из шага qiStep. Вложенная SFC-схема в точности будет соответствовать графу переходов счётчиковой машины M.

Таким образом, мы показали, что для произвольной n-счётчиковой машины Минского M может быть задана LTL-спецификация её поведения, по которой осуществляется построение программной реализации машины M на любом из стандартных языков программирования ПЛК. □

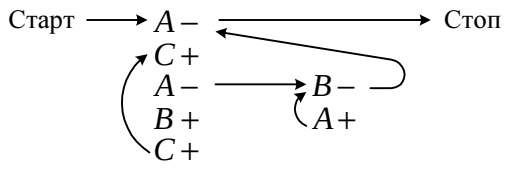
3. Счетчиковая машина возведения в квадрат

Рассмотрим в качестве примера трехсчетчиковую машину Минского $3cM$, реализующую функцию возведения числа N в квадрат. Эта счетчиковая машина имеет восемь состояний q_0, q_1, …, q_7, где q_0 является начальным состоянием, а q_7 – это финальное состояние. В начальной конфигурации счетчик A получает значение N, а начальные значения двух других счетчиков B и C равны нулю. В финальной конфигурации результат вычисления будет содержаться в счетчике C, а значения остальных счетчиков A и B будут равны нулю. Правила перехода по состояниям машины $3cM$ представлены ниже:

(δ_0) q_0: if $A > 0$ then $(A := A - 1;$ goto $q_1)$ else goto q_7;
(δ_1) q_1: $C := C + 1;$ goto q_2;
(δ_2) q_2: if $A > 0$ then $(A := A - 1;$ goto $q_3)$ else goto q_5;
(δ_3) q_3: $B := B + 1;$ goto q_4;
(δ_4) q_4: $C := C + 1;$ goto q_1;
(δ_5) q_5: if $B > 0$ then $(B := B - 1;$ goto $q_6)$ else goto q_0;
(δ_6) q_6: $A := A + 1;$ goto q_5.

В качестве пояснения отметим, что при построении этой счетчиковой машины использовался тот факт, что $N^2 = (2N - 1) + (2N - 3) + \cdots + 3 + 1$.

Правила перехода машины $3cM$ можно наглядно представить в следующем графическом виде, где для некоторой переменной V обозначение «$V+$» соответствует безусловному увеличению счетчика на единицу, а «$V-$» используется для обозначения условного вычитания единицы с переходом в другое состояние по правосторонней стрелке в случае нулевого значения счетчика V.



Построим LTL-спецификацию трехсчетчиковой машины Минского $3cM$ возведения числа N в квадрат. В этой спецификации переменная-счетчик A при инициализации получает значение N. Переменная-состояние q0 инициализируется единицей, т. е. изначально q0 = 1. Остальные переменные (в том числе и псевдооператорные) при инициализации обнуляются.

Init(A) = N;
A+ : **GX**(A > _A ⇒ _q6 ∧ A = _A + 1);
A− : **GX**(A < _A ⇒ (_q0 ∨ _q2) ∧ _A > 0 ∧ A = _A − 1);
B+ : **GX**(B > _B ⇒ _q3 ∧ B = _B + 1);
B− : **GX**(B < _B ⇒ _q5 ∧ _B > 0 ∧ B = _B − 1);
C+ : **GX**(C > _C ⇒ (_q1 ∨ _q4) ∧ C = _C + 1);
C− : **GX**(C < _C ⇒ false);
Init(q0) = 1;
q0+ : **GX**(¬_q0 ∧ q0 ⇒ _q5 ∧ ¬(_B > 0));
q0− : **GX**(_q0 ∧ ¬q0 ⇒ true);
q1+ : **GX**(¬_q1 ∧ q1 ⇒ _q0 ∧ _A > 0 ∨ _q4);
q1− : **GX**(_q1 ∧ ¬q1 ⇒ true);
q2+ : **GX**(¬_q2 ∧ q2 ⇒ _q1);
q2− : **GX**(_q2 ∧ ¬q2 ⇒ true);
q3+ : **GX**(¬_q3 ∧ q3 ⇒ _q2 ∧ _A > 0);
q3− : **GX**(_q3 ∧ ¬q3 ⇒ true);
q4+ : **GX**(¬_q4 ∧ q4 ⇒ _q3);
q4− : **GX**(_q4 ∧ ¬q4 ⇒ true);
q5+ : **GX**(¬_q5 ∧ q5 ⇒ _q2 ∧ ¬(_A > 0) ∨ _q6);
q5− : **GX**(_q5 ∧ ¬q5 ⇒ true);

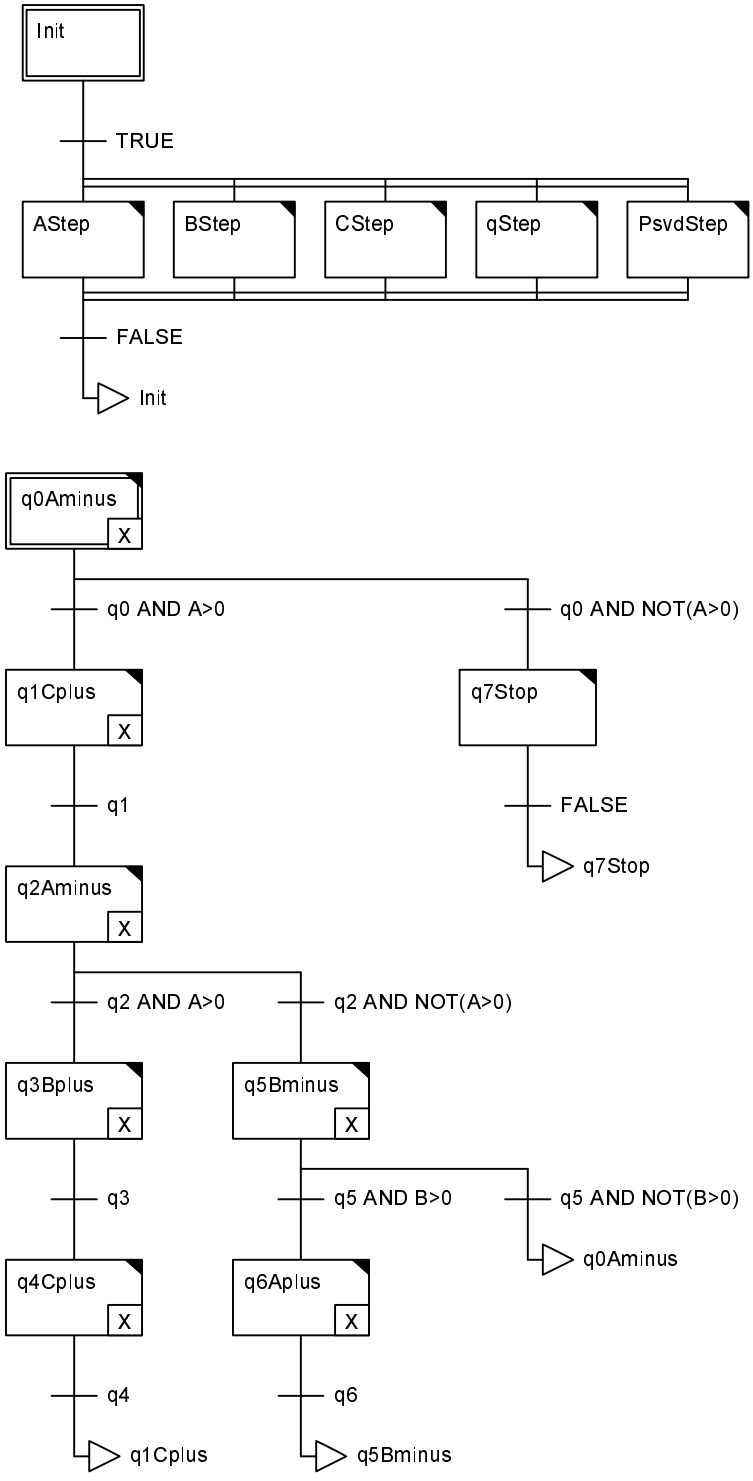
$\texttt{q6+}\ :\ \mathbf{GX}(\ \neg\texttt{\_q6} \wedge\ \ \texttt{q6}\ \Rightarrow\ \texttt{\_q5} \wedge \texttt{\_B} > 0\ );$
$\texttt{q6}-\ :\ \mathbf{GX}(\ \ \texttt{\_q6} \wedge \neg\texttt{q6}\ \Rightarrow\ \texttt{true}\ );$
$\texttt{q7+}\ :\ \mathbf{GX}(\ \neg\texttt{\_q7} \wedge\ \ \texttt{q7}\ \Rightarrow\ \texttt{\_q0} \wedge \neg(\texttt{\_A} > 0)\ );$
$\texttt{q7}-\ :\ \mathbf{GX}(\ \ \texttt{\_q7} \wedge \neg\texttt{q7}\ \Rightarrow\ \texttt{false}\ ).$

Реализация счётчиковой машины $3cM$ на языке программирования ST по приведенной выше LTL-спецификации имеет следующий вид.

```
VAR (* раздел описания переменных *)
   A  : INT :=N; (* N - входное значение, возводимое в квадрат *)
   q0 : BOOL:=1; (* начальное состояние *)
   q1, q2, q3, q4, q5, q6 : BOOL; (* состояния *)
   q7 : BOOL; (* финальное состояние *)
   _A, _B, _C : INT; (* псевдооператорные переменные *)
   _q0, _q1, _q2, _q3, _q4, _q5, _q6, _q7 : BOOL;
END_VAR
   (* Переменные *)
   IF     _q6                   THEN A:=_A+1;          (* A+ *)
   ELSIF (_q0 OR _q2) AND _A>0 THEN A:=_A-1; END_IF; (* A- *)
   IF    _q3           THEN B:=_B+1;                   (* B+ *)
   ELSIF _q5 AND _B>0 THEN B:=_B-1; END_IF;           (* B- *)
   IF    (_q1 OR _q4) THEN C:=_C+1; END_IF;           (* C+ *)
   (* Состояния *)
   IF NOT _q0 AND _q5 AND NOT(_B>0) THEN q0:=1;                   (* q0+ *)
   ELSIF  _q0                       THEN q0:=0; END_IF;           (* q0- *)
   IF NOT _q1 AND (_q0 AND _A>0 OR _q4) THEN q1:=1;               (* q1+ *)
   ELSIF  _q1                           THEN q1:=0; END_IF;       (* q1- *)
   IF NOT _q2 AND _q1 THEN q2:=1;                                 (* q2+ *)
   ELSIF  _q2         THEN q2:=0; END_IF;                         (* q2- *)
   IF NOT _q3 AND _q2 AND _A>0 THEN q3:=1;                        (* q3+ *)
   ELSIF  _q3                  THEN q3:=0; END_IF;                (* q3- *)
   IF NOT _q4 AND _q3 THEN q4:=1;                                 (* q4+ *)
   ELSIF  _q4         THEN q4:=0; END_IF;                         (* q4- *)
   IF NOT _q5 AND (_q2 AND NOT(_A>0) OR _q6) THEN q5:=1;          (* q5+ *)
   ELSIF  _q5                                THEN q5:=0; END_IF; (* q5- *)
   IF NOT _q6 AND _q5 AND _B>0 THEN q6:=1;                        (* q6+ *)
   ELSIF  _q6                  THEN q6:=0; END_IF;                (* q6- *)
   IF NOT _q7 AND _q0 AND NOT(_A>0) THEN q7:=1; END_IF;           (* q7+ *)
   (* Псевдооператорный раздел *)
   _A:=A; _B:=B; _C:=C;
   _q0:=q0; _q1:=q1; _q2:=q2; _q3:=q3; _q4:=q4; _q5:=q5; _q6:=q6; _q7:=q7.
```

Рассмотрим теперь реализацию счетчиковой машины $3cM$ на упрощённом SFC также по описанной ранее LTL-спецификации. Основная и вложенные SFC-схемы представлены ниже (по порядку). Действию шага AStep соответствует программная реализация пары LTL-формул спецификации поведения переменной-счетчика A. Например, действием этого шага может быть взят IF-ELSIF блок на языке ST, помеченный парой A+ и A−. Аналогичным образом действиям шагов BStep и CStep сопоставляются программные реализации пар LTL-формул спецификации поведения переменных-счетчиков B и C соответственно. Действием шага PsvdStp является

псевдооператорный раздел. Действию шага qStep соответствует вложенная SFC-схема. Во вложенной SFC-схеме текущим и выходным действиями шага, имя которого начинается с префикса `qi`, является программная реализация (на IL, ST, FBD/CFC или LD) LTL-формул с метками `qi+` и `qi−`. Вложенная SFC-схема повторяет графическое представление правил перехода машины $3cM$.



Список литературы / References

[1] Кузьмин Е. В., Рябухин Д. А., Соколов В. А., "Моделирование согласованного поведения ПЛК-датчиков", *Моделирование и анализ информационных систем*, **21**:4 (2014), 75–90; [Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "Modeling a Consistent Behavior of PLC-Sensors", *Modeling and Analysis of Information Systems*, **21**:4 (2014), 75–90, (in Russian).]

[2] Рябухин Д. А., Кузьмин Е. В., Соколов В. А., "Построение IL-программ ПЛК по LTL-спецификации", *Моделирование и анализ информационных систем*, **21**:2 (2014), 26–38; [Ryabukhin D. A., Kuzmin E. V., Sokolov V. A., "Construction of PLC IL-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **21**:2 (2014), 26–38, (in Russian).]

[3] Кузьмин Е. В., Соколов В. А., Рябухин Д. А., "Построение и верификация LD-программ ПЛК по LTL-спецификации", *Моделирование и анализ информационных систем*, **20**:6 (2013), 78–94; [Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC LD-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **20**:6 (2013), 78–94, (in Russian).]

[4] Кузьмин Е. В., Соколов В. А., Рябухин Д. А., "Построение и верификация ПЛК-программ по LTL-спецификации", *Моделирование и анализ информационных систем*, **20**:4 (2013), 5–22; [Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC-programs by LTL-specification", *Modeling and Analysis of Information Systems*, **20**:4 (2013), 5–22, (in Russian).]

[5] Кузьмин Е. В., Соколов В. А., "Моделирование, спецификация и построение программ логических контроллеров", *Моделирование и анализ информационных систем*, **20**:2 (2013), 104–120; [Kuzmin E. V., Sokolov V. A., "Modeling, Specification and Construction of PLC-programs", *Modeling and Analysis of Information Systems*, **20**:2 (2013), 104–120, (in Russian).]

[6] Минский М., *Вычисления и автоматы*, М.: Мир, 1971; [Minsky M., *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc., 1967.]

[7] Петров И. В., *Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования*, М.: СОЛОН-Пресс, 2004; [Petrov I. V., *Programmiruemye kontrollery. Standartnye jazyki i priemy prikladnogo proektirovanija*, M.: SOLON-Press, 2004, (in Russian).]

[8] Baier C., Katoen J.-P., *Principles of Model Checking*, The MIT Press, 2008.

[9] Clark E. M., Grumberg O., Peled D. A., *Model Checking*, The MIT Press, 2001.

[10] CoDeSys, *Controller Development System*, http://www.3s-software.com/.

[11] Kuzmin E. V., Sokolov V. A., Ryabukhin D. A., "Construction and Verification of PLC LD Programs by the LTL Specification", *Automatic Control and Computer Sciences*, **48**:7 (2014), 424–436.

[12] Kuzmin E. V., Sokolov V. A., "Modeling, Specification and Construction of PLC-programs", *Automatic Control and Computer Sciences*, **48**:7 (2014), 554–563.

[13] Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "Modeling a Consistent Behavior of PLC-Sensors", *Automatic Control and Computer Sciences*, **48**:7 (2014), 602–614.

[14] Parr E. A., *Programmable Controllers. An engineer's guide*, Newnes, 2003.

[15] Schroeppel R., *A Two Counter Machine Cannot Calculate* 2^N, Memo 257. Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1972.

[16] SMV, *The Cadence SMV Model Checker*, http://www.kenmcmil.com/smv.html.

DOI: 10.18255/1818-1015-2015-4-507-520

On the Expressiveness of the Approach to Constructing PLC-programs by LTL-Specification

Kuzmin E. V.[1], Ryabukhin D. A., Sokolov V. A.

Received August 3, 2015

The article is devoted to the approach to constructing and verification of discrete PLC-programs by LTL-specification. This approach provides an ability of correctness analysis of PLC-programs by the model checking method. The linear temporal logic LTL is used as a language of specification of the program behavior. The correctness analysis of LTL-specification is automatically performed by the symbolic model checking tool Cadence SMV.

The article demonstrates the consistency of the approach to constructing and verification of PLC programs by LTL-specification from the point of view of Turing power. It is proved, that in accordance with this approach for any Minsky counter machine can be built an LTL-specification, which is used for machine implementation in any PLC programming language of standard IEC 61131-3. Minsky machines equipollent Turing machines, and the considered approach also has Turing power.

The proof focuses on representation of a counter machine behavior in the form of a set of LTL-formulas and matching these formulas to constructions of ST and SFC languages. SFC is interesting as a specific graphical language. ST is considered as a basic language because an implementation of a counter machine in IL, FBD/CFC and LD languages is reduced to rewriting blocks of ST-program.

The idea of the proof is demonstrated by an example of a Minsky 3-counter machine, which implements a function of squaring.

Keywords: programmable logic controllers (PLC), construction and verification of PLC-programs, LTL-specification, Minsky counter machines

For citation: Kuzmin E. V., Ryabukhin D. A., Sokolov V. A., "On the Expressiveness of the Approach to Constructing PLC-programs by LTL-Specification", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 507–520.

On the authors:
Kuzmin Egor Vladimirovich, orcid.org/0000-0003-0500-306X, doctor of science, associate professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: kuzmin@uniyar.ac.ru

Ryabukhin Dmitriy Aleksandrovich, orcid.org/0000-0002-0799-8868, graduate student,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: dmitriy_ryabukhin@mail.ru

Sokolov Valery Anatolievich, orcid.org/0000-0003-1427-4937, doctor of science, professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: sokolov@uniyar.ac.ru

Acknowledgments:

[1]This work was supported by the Russian Foundation for Basic Research (RFBR), №12-01-00281-a.

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 521–532
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 521–532

DOI: 10.18255/1818-1015-2015-4-521-532

УДК 517.9

Разработка механизма адаптивной маршрутизации трафика в программно-конфигурируемых сетях

Носков А. Н., Манов И. А.

получена 20 сентября 2015

Цель данной работы – разработать единый механизм адаптивной маршрутизации трафика различного рода, опираясь на текущие требования к качеству предоставления сервисов.

Программное конфигурирование сети – это технологии будущего. Современная тенденция развития систем связи постоянно подтверждает этот факт. Однако на сегодняшний день применение этой технологии в ее текущем виде оправдано лишь в больших сетях технологических гигантов и операторов связи.

К настоящему времени создано большое количество динамических протоколов маршрутизации трафика в сетях связи. Наша задача создать такое решение, которое сможет использовать возможности каждого узла принимать решение о передаче информации всеми возможными способами для каждого типа трафика.

Достижение поставленной цели возможно путем решения задачи разработки обобщенной метрики, подробно характеризующей каналы связи между устройствами в сети, и задачи разработки механизма адаптивного перестроения логической топологии сети (управления маршрутами) для обеспечения качественной работы всей сети в целях удовлетворения текущим требованиям к качеству предоставления того или иного вида сервиса.

Ключевые слова: программно-конфигурируемые сети, информационный поток

Для цитирования: Носков А. Н., Манов И. А., "Разработка механизма адаптивной маршрутизации трафика в программно-конфигурируемых сетях", *Моделирование и анализ информационных систем*, **22**:4 (2015), 521–532.

Об авторах:
Носков Антон Николаевич, orcid.org/0000-0003-4569-6905, старший преподаватель
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: lantoni@uniyar.ac.ru

Манов Иван Алексеевич, orcid.org/0000-0002-9943-4954, магистрант
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: ivanmanov@live.com

1. Введение

Современное общество живет в эпоху стремительного развития вычислительной техники. По прогнозам аналитиков, телекоммуникационная отрасль в недалеком будущем постепенно перейдёт к концепции объединения всех устройств планеты в единую вычислительную сеть связи, основная задача которой – качественное предоставление сервисов любого рода. Различные сервисы – это различные типы трафика (реального / не реального времени), качество передачи которых на данный момент регламентируется в рекомендациях сектора телекоммуникаций Международного Союза Электросвязи (МСЭ-Т) [1], [2]. В связи с этим встает вопрос о необходимости наличия единого механизма управления трафиком в объединенных сетях связи, предоставляющих различные типы сервисов.

На сегодняшний день создано большое количество динамических протоколов маршрутизации трафика в сетях связи. В качестве примера можно привести динамическое протоколы внутреннего шлюза EIGRP и OSPF, установленные почти на всех современных сетевых маршрутизаторах, протоколы NICE, ZigZag со структурированным представлением сети, наиболее часто используемые в пиринговых P2P-сетях. Все они обеспечивают оптимальную маршрутизацию трафика по тому или иному признаку, полагаясь на конкретный набор базовых рабочих характеристик, описывающих каналы между узлами связи. Но ни один из них не обладает гибким механизмом управления маршрутами, опираясь на текущие потребности всей системы связи в целом. К примеру, протоколы NICE и ZigZag, применение которых оправдано в задачах обеспечения оптимальной маршрутизации трафика реального времени, не могут, в свою очередь, обеспечить оптимальную маршрутизацию больших объемов документальных данных, качество передачи которых не сильно зависит от задержек на каналах связи [4]. Протоколы EIGRP и OSPF, в свою очередь, принято считать универсальными при решении задач передачи трафика любого вида, однако при создании таблиц маршрутизации они используют скудный по сегодняшним меркам набор первичных статичных рабочих характеристик (статичные значения, принятые для каналов различного типа, т.е. не вычисляются динамически), описывающих каналы связи. Помимо всего прочего, их применение в P2P-сетях является неоправданным по причине больших требований к вычислительным устройствам, на которых они работают. Всё же это специализированные протоколы, создававшиеся для работы на специализированных сетевых устройствах. Протокол MPLS был одним из первых протоколов, созданных для решения проблем адаптивной маршрутизации. Его механизмы в полной мере могут удовлетворить текущие потребности в грамотном перераспределении потоков данных. Однако данный протокол требует от сетевых специалистов больших навыков в его настройке и не обладает автоматизированным механизмом распределения потоков согласно текущим потребностям сети [5].

Программное конфигурирование сети связи можно назвать технологией будущего. Современная тенденция развития систем связи постоянно подтверждает этот факт. Однако на сегодняшний день применение этой технологии в ее текущем виде оправдано лишь в больших сетях технологических гигантов и операторов связи. Но, позаимствовав основные принципы работы программно-конфигурируемых сетей (централизованное управление, виртуализация сетевой функции и т.д.) и спро-

ецировав их на ограниченное количество самостоятельных устройств, с единой политикой маршрутизации трафика, можно добиться гибкости в предоставлении сервисов различного рода [6]. Всё дальнейшее рассмотрение будет спроецировано на совокупности устройств, представляющих собой P2P систему связи.

Цель данной работы – разработать единый механизм адаптивной маршрутизации трафика различного рода, опираясь на текущие требования к качеству предоставления сервисов, регламентируемые МСЭ-Т [2]. Достижение поставленной цели возможно путем решения задачи разработки обобщенной метрики, подробно характеризующей каналы связи между устройствами в сети и задачи разработки механизма адаптивного перестроения логической топологии сети (управления маршрутами) для обеспечения качественной работы всей сети в целях удовлетворения текущим требованиям на качество предоставления того или иного вида сервиса. Таблица маршрутизации – один из базовых элементов любого протокола маршрутизации трафика. Она регламентирует основные правила, на которые опирается сетевая функция при передаче трафика. Каждое сетевое устройство использует таблицу маршрутизации, в которой отражена топология сети на данный момент времени. Базовая единица представления таблицы маршрутизации – маршрутная информация (маршрут). Маршрут, как принято считать, состоит из четырёх основных частей:

- адрес (и префикс) сети или узла назначения;
- шлюз, обозначающий адрес маршрутизатора в сети, на который необходимо отправить пакет, следующий до указанного адреса назначения;
- интерфейс (в зависимости от системы это может быть порядковый номер, GUID или символьное имя устройства);
- метрика — числовой показатель, задающий предпочтительность маршрута. Чем меньше число, тем более предпочтителен маршрут (интуитивно представляется как расстояние).

2. Разработка метрики маршрута

Метрика маршрута – важнейшая часть любого протокола маршрутизации, являющаяся условием добавления того или иного набора маршрутов в таблицу маршрутизации. Чем лучше метрика, тем больше вероятность того, что сетевое устройство примет решение отправить пакет по маршруту, который она характеризует. Современные протоколы маршрутизации при вычислении метрик маршрутов часто опираются на использование до трех основных параметров, регламентируемых МСЭ-Т. Это процент потерянных пакетов (IPLR) на сквозном канале между двумя устройствами в сети, полоса пропускания (Bandwidth) и задержка при передаче сетевых пакетов (IPTD) [1]. Каждый из них по тем или иным причинам оказывает непосредственное влияние друг на друга. Данные параметры дают лишь общую оценку каналов связи, а изменения их значений могут являться лишь следствием влияния других сопутствующих факторов. Учет таких факторов в будущем поможет создавать наиболее подробную картину состояния связей в сети и всей системы в целом.

В качестве одного из таких факторов можно считать загруженность (Load) узлов связи (с учетом загруженности его отдельных вычислительных модулей, запаса оперативной памяти и т.д.), которые будут заниматься обработкой пакетов, отправленных по связанному с ним маршруту. Обратный к загруженности параметр – вычислительная эффективность. Проводимые исследования показывают, что учет данного параметра позволит в перспективе снизить вероятность потери пакетов на канале связи и, как следствие, задержки при передаче трафика с подтверждением доставки (сервисы обмена файлами).

Для получения информативной картины, характеризующей систему связи, введем интегральную метрику $W_{(i,j)}$, учитывающую четыре параметра: процент потерянных пакетов, полосу пропускания, задержку между узлами i и j и вычислительную эффективность узла связи j, задача которого заключается в обработке трафика от узла i.

Рассмотрим сеть связи, состоящую из N устройств. Численные значения метрики $W_{(i,j)}$ можно представить в виде матрицы смежности A^W как:

$$A^W = \begin{pmatrix} W_{1,1} & \cdots & W_{1,n} \\ \vdots & \ddots & \vdots \\ W_{n,1} & \cdots & W_{n,n} \end{pmatrix} \tag{1}$$

Стоит отметить, что матрица параметров A^W не является симметричной ($W_{(i,j)} \neq W_{(j,i)}$).

Минимизация некой целевой функции $W_(i,j)$, на которую могут быть наложены несколько ограничений или предельных значений, называется многокритериальной оптимизацией. Главная возникающая сложность в решении задач такого рода – неоднозначность оптимального решения в точке, где один из критериев достигает своего максимума, а другой может быть очень далек не только от максимума, но и даже от какой-либо приемлемой величины. Свертка данных метрики методом «идеальной точки» требует одинаковой размерности всех величин. Для этого зададим характеристики канала в следующем виде:

- для количества потерянных пакетов s

$$P_{(i,j)} \to min, P_{(i,j)} = \left(1 - \frac{N_{(i,j)}}{M_{(i,j)}}\right), \tag{2}$$

  где $N_{i,j}$ – количество принятых пакетов, $M_{i,j}$ – количество переданных пакетов и $M_{i,j} > 0$;

- для времени задержки сигнала

$$D_{(i,j)} \to min, D_{(i,j)} = \left(1 - \frac{t_{min})}{t_{(i,j)}}\right), \tag{3}$$

  где t_{min} – минимальное значение задержки в матрице смежности, и $t_{min}, t_{i,j} > 0$;

- для ширины полосы пропускания

$$G_{(i,j)} \to min, G_{(i,j)} = \left(1 - \frac{B_{(i,j)})}{B_{max}}\right), \tag{4}$$

где B_{max} – максимальное значение ширины полосы пропускания в матрице смежности.

- для вычислительной эффективности узла-обработчика информации

$$CE_{(j)} \to min, CE_{(j)} = \left(1 - \frac{L_i}{L_{(i,max)}}\right), \tag{5}$$

где L_{jmax} – максимальное значение загруженности узла связи в матрице смежности.

Таким образом, метрика на основе четырех параметров будет иметь следующий вид:

$$W_{(i,j)} = \sqrt[4]{X_1 \left(P_{i,j}\right)^2 + X_2 \left(D_{i,j}\right)^2 + X_3 \left(G_{i,j}\right)^2 + X_4 \left(\frac{1}{CE_j}\right)^2}, \tag{6}$$

где X_1, X_2, X_3, X_4 – весовые коэффициенты, изменяющиеся в диапазоне от 0 до 1, и их сумма должна быть равна единице:

$$\begin{cases} X_1 + X_2 + X_3 + X_4 = 1 \\ X_1 \leq 1 \\ X_2 \leq 1 \\ X_3 \leq 1 \\ X_4 \leq 1 \end{cases} \tag{7}$$

Варьируя значения весовых коэффициентов в метрике $W_{(i,j)}$, мы, тем самым, создаем аппарат для управления значимостью того или иного параметра метрики в итоговой оценке канала передачи данных от узла i к узлу j.

С математической точки зрения в качестве «идеальной» выбирается точка с координатами, соответствующими t_{min}, P_{min}, B_{max}, CE_{max} (или L_{min}) при фиксированных весовых коэффициентах X_1, X_2, X_3, X_4.

В реальности, с учетом динамики работы всей системы связи, выбор весовых коэффициентов обусловлен характером первостепенных типов трафика, обеспечивающих предоставление первостепенных сервисов.

МСЭ-Т в рекомендации [2] регламентирует пять основных классов качества предоставления сервисов (IP QoS), к которым можно отнести систему связи. Сети класса 0 наиболее требовательны к значениям рабочих характеристик каналов связи. Как правило, это сети, обеспечивающие предоставления сервисов реального времени, наиболее чувствительные к задержкам (VoIP, видеоконференцсвязь, онлайн игры и т.д.). В настоящее время нет единой концепции к построению систем и сетей связи, обеспечивающих предоставление всех сервисов с наивысшим качеством [7]. В связи с этим возникает необходимость в разработке гибкого механизма маршрутизации трафика отдельно выбранного сервиса с учетом текущих возможностей системы связи и требований на качество его предоставления. Качество предоставления сервисов неразрывно связано с основными параметрами, фигурирующими в метрике $W_{(i,j)}$. Основная идея разработанного механизма обеспечения гибкой эффективной маршрутизации заключается в манипуляции весовыми коэффициентами

таким образом, чтобы добиваться наилучших возможных значений рабочих характеристик, определяющих класс IP QoS. В реалиях программно-конфигурируемых сетей возможность реализация такого механизма наиболее перспективна. Ядром такой сети является управляющее устройство – программный контроллер, управляющий структурой сети [6]. В данном случае – это устройство, способное оптимально подбирать весовые коэффициенты метрики $W_{(i,j)}$. МСЭ-Т в рекомендации [3] регламентирует пять основных классов качества предоставления сервисов (QoS), к которым можно отнести систему связи. Диапазоны значений, присущие основным трем классам, представлены в таблице 1.

Параметр рабочей характеристики сети	**Классы QoS**		
	Класс 0	Класс 1	Класс 2
IPTD	100 мс	400 мс	100 мс
IPDV	50 мс	50 мс	Н
IPLR	1 x 10-3	1 x 10-3	1 x 10-3
IPER	1 x 10-4	1 x 10-4	1 x 10-4

Таблица 1. Классы качества облуживания

3. Разработка алгоритма маршрутизации

Качество предоставления сервисов неразрывно связано с основными параметрами, фигурирующими в метрике $W_{(i,j)}$. Основная идея разрабатываемого алгоритма обеспечения гибкой эффективной маршрутизации заключается в манипуляции весовыми коэффициентами метрики таким образом, чтобы добиваться наилучших возможных значений рабочих характеристик, определяющих класс QoS.

В таблице 2 представлен набор весовых коэффициентов, оказывающих непосредственное влияние на качество передачи соответствующего вида трафика [3].

Тип трафика	X_2,X_3	X_2,X_4	X_3	X_1, X_3, X_4
	Задержка	Джиттер	Полоса	Потеря пакетов
Реального времени	+	+	+	+
обмен документов			+	+
Сигнализация/WEB				+
Потоковое видео			+	

Таблица 2. Влияние весовых коэффициентов на определенный тип трафика

На базе программно-конфигурируемых сетей возможность реализации такого механизма наиболее перспективна. Ядром такой сети является управляющее устройство – программный контроллер, определяющий структуру сети. В данном случае – это устройство, способное гибко подбирать весовые коэффициенты метрики $W_{(i,j)}$, опираясь на текущие возможности системы.

Далее приведем блок-схему разработанного алгоритма (рис. 1) и описание работы его функциональных блоков. Работа алгоритма начинается с определения базовых значений весовых коэффициентов метрики. Для систем общего пользования,

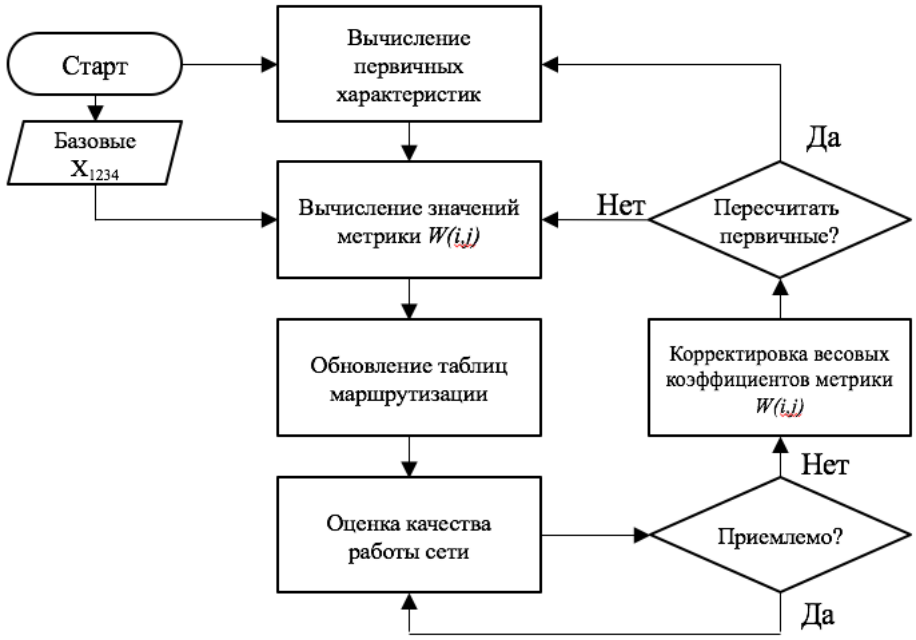


Рис. 1. Блок-схема разработанного алгоритма

которые единовременно требуют равноправного вклада каждого из первичных параметров метрики, базовые коэффициенты можно задавать равнозначными. При приоритезации конкретного вида трафика выбор значений весовых коэффициентов может определяться согласно таблице 2, учитывающей их взаимное влияние на качество предоставления этого трафика.

Параллельно запускается процесс, отвечающий за вычисление первичных характеристик, определяющих состояние всех существующих каналов связи. Полученные результаты поступают в блок «Вычисления значений метрики $W_{(i,j)}$». В нем по формуле (6) производится вычисление соответствующих значений метрики для каждого канала. Значения записываются в матрицу смежностей AW, использующуюся для определения стоимости маршрутов таблиц маршрутизации.

Рекурсивность алгоритма позволяет добиваться постоянного улучшения качества работы сети, путем проверки текущих значений рабочих характеристик сети на соответствие текущим требованиям, возлагаемым на всю систему связи в целом. В варианте несоответствия значений рабочих характеристик заданным требованиям к системе связи возможно, при необходимости, производить корректировку значений весовых коэффициентов, с целью настроить текущую топологию для работы с максимальной эффективностью.

Учитывая, что изменение значений может влиять на текущие значения первичных характеристик каналов связи, возможно осуществлять пересчет метрик, с целью поддержки актуальности характеристик каналов. Принятие решений по пересчету первичных параметров сети и непосредственной корректировке значений весовых коэффициентов метрики – основная задача контроллера программно-конфигурируемой сети. В реальности подбор значений весовых коэффициентов – не тривиальная задача, поскольку, учитывая динамику работы IP-систем связи, нет четких правил по ее управлению, например, в случае возникновения непредвиден-

ных ситуаций. Работа контроллера представляет собой интеллектуальную систему, которая, принимая во внимание текущее состояние сети, способна управлять всей сетью с целью достижения требуемых наилучших результатов передачи трафика. Разработанный алгоритм дает необходимый инструмент для упрощения управления системой связи, путем манипуляции весовыми коэффициентами метрики [6], тем самым сводя задачу к подбору их «наилучших» значений.

В следующем пункте будет рассмотрен процесс реализации разработанного алгоритма и произведен его анализ и сравнение с общепризнанными аналогами.

4. Разработка имитационной модели

В процессе разработки была создана имитационная модель P2P-сети, состоящая из 18 узлов. Первичные параметры, характеризующие разработанную метрику, задавались с условием достижимости критических ситуаций, разрешение которых возможно путем логического перераспределения потоков данных. Задача, решаемая системой связи, сводилась к обеспечению наилучшего качества передачи потока данных высокой плотности из сети A в сеть B через транзитную сеть C, состоящую, в основном, из каналов с низкой пропускной способностью (рис. 1).

В качестве эталонных алгоритмов маршрутизации трафика были выбраны алгоритмы OSPF и EIGRP, оперирующие лишь численными заданными значениями полосы пропускания и задержки на каналах связи.

В ходе проведенных исследований были получены значения задержки, проценты потерянных пакетов и джиттера для эталонных алгоритмов OSPF и EIGRP. После, в процессе установления соединения, из значений метрики $W_{(i,j)}$ была составлена матрица смежностей, описывающая текущее состояние всей системы связи при заданных первоначальных коэффициентах, равных $X_{1234} = 0.25$, гарантирующих равнозначный вклад каждого из основных параметров метрики. В процессе моделирования было выявлено, что средняя задержка по отношению к OSPF снизилась на 34%, к EIGRP – на 20%. Процент потерянных пакетов в свою очередь снизился на 80% по отношению к OSPF и на 98% – по отношению к EIGRP

Параметр / Алгоритм	Средняя задержка	% потреяных пакетов	Джиттер
OSPF	1341,45	10,82%	219,69
EIGRP	1115,45	84,60%	573
Prime $X_1 = X_2 = X_3 = X_4 = 0,25$	890,93	2,14%	55,46
Prime $X_2 = [0.58; 0.99]X_1 = X_3 = X_4 = (1 - (X_3)/3)$	612,52	84,44%	570,7
Prime $X_3 = [0.3; 0.35]X_1 = X_2 = X_4 = (1 - (X_2)/3)$	803,71	0%	12,99

Таблица 3. Влияние весовых коэффициентов на определенный тип трафика

При приоритезации параметра, характеризующего задержку на канале связи в

диапазоне значений $X_2 = [0,58; 0,99]$, при равных $X_1 = X_3 = X_4$, средняя задержка за весь сеанс работы сети снизилась на 54% по отношению к OSPF и на 45% по отношению к EIGRP, что на 31% эффективнее по сравнению с работой алгоритма при равных значениях весовых коэффициентов. При этом процент потерянных пакетов составил 84%, что может очень негативно сказываться на предоставлении сервисов, чувствительных к потерям пакетов.

При приоритезации параметра, характеризующего полосу пропускания каналов в диапазоне значений $X_3 = [0,3; 0,35]$, при равных $X_1 = X_2 = X_4$, средняя задержка снизилась на 40% по отношению к эталонному алгоритму OSPF и на 28% к EIGRP, что на 10% эффективнее по сравнению с работой алгоритма при равных значениях X_{1234}. При этом потери пакетов снизились до нуля, что говорит о том, что был найден оптимум, удовлетворяющий текущим потребностям сети. Алгоритм определил оптимальный маршрут для потока высокой плотности, минуя при этом низкопроизводительные узлы и каналы с низкой пропускной способностью и высокой вероятностью потерь пакетов. Характер работы алгоритма можно проследить на рис. 2. Из него видно, что при различных методах оценки стоимости каналов связи меняется маршрут потока. На практике задача разработанного механизм – адаптивно подбирать маршрут, максимизирующий качественные характеристики работы сети, исходя из потребностей системы и текущего состояния всей сети.

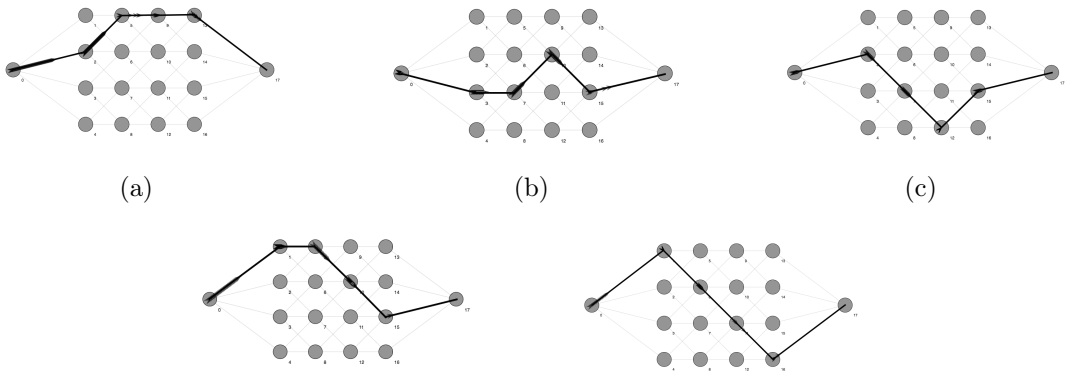


Рис. 2. Изменение движения потоков при различных значениях метрики: (a) эталонная модель OSPF; (b) эталонная модель EIGRP; (c) модель при равных весовых коэффициентах; (d) модель с приоритезацией по задержке; (e) с приоритезацией по полосе пропускания

В динамике процесса работы системы необходим пересчет первоначальных параметров метрики в текущих условиях работы сети и, как следствие, изменение значений весовых коэффициентов X_{1234}, согласно текущим потребностям системы связи.

5. Заключение

Было показано, что, имея совокупность сетевых узлов, можно увеличить эффективность работы сети путем грамотного управления маршрутной информацией.

Существенным достоинством предложенного подхода к управлению механизмами маршрутизации является его возможность развертывания как в рамках текущих программно-конфигурируемых систем, так и в рамках отдельно взятой совокупности узлов связи, объединенной единой логикой организации сетевых ресурсов.

Дальнейшие исследования должны быть направлены на разработку интуитивного механизма принятия решений, суть которого – в автоматическом режиме изменять весовые коэффициенты метрики с целью оптимизации передачи отдельных потоков данных, основываясь на текущих возможностях системы связи и возлагаемых на нее требований.

Список литературы / References

[1] *ITU-T. Recommendation Y.1540 Internet protocol aspects – Quality of service and network performance*, 2011.

[2] *ITU-T. Recommendation Y.1541 Network Performance Objectives for IP-Based Services*, 2011.

[3] *ITU-T. Y.1541: Network performance objectives for IP-based services.*

[4] John F. Buford, Heather Yu, Eng Lua, *P2P Networking and Applications*, The Morgan Kaufmann Series in Networking, 2009.

[5] Luc De Ghein, *MPLS Fundamentals*, Cisco Press, 2006.

[6] Thomas D. Nadeau, Ken Gray, *SDN: Software Defined Networks*, O'Reilly Media, 2013.

[7] Вешегна Ш., *Качество облуживания в IP-сетях*, Cisco Press, 2003, 356 с.; [Veshegna Sh., *Kachestvo obslujivaniya v IP-setyah*, Cisco Press, 2003, 356 pp., (in Russian).]

[8] Blei D. M., Ng A. Y., Jordan M. I., "Latent Dirichlet Allocation", *Journal of Machine Learning Research*, **3** (2002), 993–1022.

[9] Daud A, Li J. ,Zhou L., Muhammad F., "Knowledge discovery through directed probabilistic topic models", *Frontiers of Computer Science in China*, **4**:2 (2010), 280–301.

[10] Gelman A., Carlin J. B., Stern H. S., Rubin D. B., *Bayesian Data Analysis*, Chapman and Hall/CRC, 2013.

[11] Vapnik V., *Statistical Learning Theory*, Wiley, 1998.

[12] Ferguson T. S., "A bayesian analysis of some nonparametric problems", *The Annals of Statistics*, **1**:2 (1973), 209–230.

[13] Kintsch W., *Handbook of Latent Semantic Analysis*, Erlbaum, Hillsdale, NJ, 2007.

[14] Knorr E. M., Ng R. T., "Algorithms for Mining Distance-Based Outliers in Large Datasets", *Proceedings of the 24th International Conference on Very Large Data Bases*, **1** (1998), 392–403.

[15] Knorr E. M., Ng R. T., "Finding Intensional Knowledge of Distance-based Outliers", *Proceedings of the 25th International Conference on Very Large Data Bases*, **1** (1999), 211–222.

[16] Minka T., Lafferty J., "Expectation-propagation for the generative aspect model", *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002.

[17] Saurabh Ganeriwal, Laura K. Balzano, Mani B., "Reputation-based Framework for High Integrity Sensor Networks", *ACM Transactions on Sensor Networks*, **5** (2007).

[18] Paramasivan B. A, Prakash M. J, Kaliappan M., "Development of a secure routing protocol using game theory model in mobile ad hoc networks", *Journal of Communications and Networks*, **1**:15 (2015), 75–83.

[19] Samsudin N. A. a, Bradley A. P. b, "Extended na?ve bayes for group based classification Advances in Intelligent Systems and Computing", *1st International Conference on Soft Computing and Data Mining*, **287** (2014), 497–506.

[20] Jolliffe I. T., *Principal components analysis*, Springer-Verlag, New York, 1986.

[21] Shipman C. M., Hopkinson K. M., Lopez J., "Con-resistant trust for improved reliability in a smart-grid special protection system", *IEEE Transactions on Power Delivery*, **13**:1 (2015), 455–462.

DOI: 10.18255/1818-1015-2015-4-521-532

Development of an Adaptive Routing Mechanism in Software-Defined Networks

Noskov A. N., Manov I. A.

Received September 20, 2015

The purpose of this work is to develop a unitary mechanism of adaptive routing of different kinds, basing on the current requirements on the quality of service. The software configuration of a network is the technology of the future. The trend in communication systems constantly confirms this fact. However, the application of this technology in its current form is justified only in large networks of technology giants and telecom operators. Today we have a large number of dynamic routing protocols to route big volume traffic in communication networks. Our task is to create the solution that can use the opportunities of each node to make a decision on the transmission of information by all possible means for each type of traffic. Achieving this goal is possible by solving the problem of the development of generalized metrics, which details the links between devices in the network, and the problem of establishing a framework of adaptive logical network topology (route management) to ensure the quality of the whole network in order to meet the current requirements on the quality of a particular type service.

Keywords: software defined network, information flow

For citation: Noskov A. N., Manov I. A., "Development of an Adaptive Routing Mechanism in Software-Defined Networks", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 521–532.

On the authors:
Noskov Anton Nikolaevich, orcid.org/0000-0003-4569-6905, Lecturer,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: lantoni@uniyar.ac.ru

Manov Ivan Alekseevich, orcid.org/0000-0002-9943-4954, magister,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: ivanmanov@live.com

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 533–545
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 533–545

DOI: 10.18255/1818-1015-2015-4-533-545

УДК 519.179.2, 519.854.3

Задача о наибольшем кратном потоке в делимой сети и ее частные случаи

Смирнов А. В.

получена 10 июля 2015

В статье рассматривается задача о наибольшем кратном потоке в сети произвольной натуральной кратности k. Определяется три типа дуг в сети: обычная дуга, кратная дуга, мультидуга. Каждая кратная и мультидуга представляет собой объединение k связанных дуг, согласованных между собой. Задаются правила построения сети.

Вводится понятие делимой сети и ряд связанных определений. Отмечается важная особенность делимых сетей – возможность разделить их на k частей, согласованных на связанных дугах кратных и мультидуг, таким образом, что каждая часть является обычной транспортной сетью.

Основным результатом статьи является выделение следующих подклассов задачи о наибольшем кратном потоке в делимой сети.

1. Делимая сеть с ограничениями на мультидуги. Если в $k-1$ части делимой сети имеется только одна вершина, являющаяся концом мультидуги, то задача о наибольшем потоке полиномиально разрешима.

2. Делимая сеть со слабыми ограничениями на мультидуги. Если в s частях делимой сети ($1 \leq s < k-1$) имеется только одна вершина, являющаяся концом мультидуги, а в остальных частях таких вершин несколько, то размерность задачи о наибольшем кратном потоке может быть понижена до $k-s$.

3. Делимая сеть параллельной структуры. Пусть компонента делимой сети, состоящая из всех кратных дуг, может быть разделена на субкомпоненты, содержащие ровно по одной вершине-началу мультидуги. Пусть при этом каждая пара субкомпонент пересекается только в источнике сети x_0. Если $k = 2$, то задача о максимальном кратном потоке разрешима за полиномиальное время. Если $k \geq 3$, то задача NP-полна.

Для каждого из полиномиальных подклассов получены алгоритмы. Также сформулирован алгоритм понижения размерности задачи для делимой сети со слабыми ограничениями на мультидуги.

Ключевые слова: кратные сети, кратные потоки, делимые сети, NP-полнота, полиномиальный алгоритм

Для цитирования: Смирнов А. В., "Задача о наибольшем кратном потоке в делимой сети и ее частные случаи", *Моделирование и анализ информационных систем*, **22**:4 (2015), 533–545.

Об авторах:
Смирнов Александр Валерьевич, orcid.org/0000-0002-0980-2507, канд. физ.-мат. наук., доцент кафедры теоретической информатики,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: alexander_sm@mail.ru

Благодарности:
Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 15-07-03038 А).

Введение

Задача о наибольшем кратном потоке в сети $G(X,U)$ натуральной кратности k (см. [1]) является обобщением классической задачи о наибольшем потоке в транспортной сети (см. [2]). Она имеет связь со многими задачами целочисленного линейного программирования (см. [3]).

В частности, поиск наибольшего кратного потока используется при решении задачи целочисленного сбалансирования трехмерной матрицы с ограничениями первого и второго рода (см. [4] – [6]). В трехмерной вещественной матрице A размерности $(n+1)\times(m+1)\times(p+1)$ элементы внутренней части (все индексы положительны) просуммированы по каждому сечению матрицы и по каждому направлению, а также найдена общая сумма. Указанные суммы размещаются в элементах с одним или несколькими нулевыми индексами (условия баланса). Требуется так округлить элементы внутренней части матрицы до целого сверху или снизу, чтобы общая сумма итоговой матрицы была округлена по стандартным правилам, а все остальные суммы в итоговой матрице отличались от исходных менее чем на 1 (ограничения первого рода) или менее чем на 2 (ограничения второго рода).

Очевидно, что задача целочисленного сбалансирования k-мерной матрицы является частным случаем многоиндексных задач линейного программирования (см. [7] – [8]). Часто такие задачи являются NP-полными (например, рассмотренные в работе [9] аксиальная и планарная трехиндексная задачи), однако существуют и полиномиально разрешимые многоиндексные задачи (например, трехиндексные задачи линейного программирования с вложенной структурой, см. [10]).

Относительно задачи целочисленного сбалансирования известно, что в случае двух индексов она может быть решена за полиномиальное время с помощью сведения к задаче о максимальном потоке в транспортной сети (см. [11] – [12]). В случае же трех и более индексов такое сведение невозможно, соответствующая теорема доказывается в статье [13] на основе результатов из работ [14] – [16]. Более того, задача целочисленного сбалансирования трехмерной матрицы является NP-полной (см. [17]). В работах [4] – [6] рассматривались кратные сети целочисленного сбалансирования кратности 2, которые использовались для поиска решения задачи целочисленного сбалансирования с ограничениями первого и второго рода. Кратные сети произвольной натуральной кратности k рассматривались в работе [1].

В данной статье будет рассмотрен класс кратных сетей специального вида – *делимых сетей*. Необходимые определения приведены в разделе 1.

Поскольку задача о максимальном потоке в делимой сети является NP-полной, важным является вопрос о полиномиальных подклассах этой задачи. Два таких подкласса будут рассмотрены в разделах 2 и 4.

В разделе 3 будет показано, каким образом можно понизить размерность задачи о наибольшем кратном потоке для одного частного случая делимой сети.

1. Делимая кратная сеть

Понятие делимой сети вводилось в работах [1], [18]. Напомним несколько определений, которые понадобятся нам в дальнейшем.

Сначала введем натуральное $k > 1$, которое назовем *кратностью потока*. В качестве сети рассматривается ориентированный мультиграф $G(X, U)$, между вершинами которого могут быть дуги одного из 3 видов:

1) *обычная дуга* u^o с пропускной способностью $c(u^o)$, поток по которой не связан с потоком по другим дугам; множество обычных дуг обозначим через U^o;

2) *кратная дуга* u^k между двумя вершинами, которая состоит из k дуг одной ориентации с одинаковой пропускной способностью $c(u^k)$ и одинаковым потоком по каждой из них; множество кратных дуг обозначим через U^k;

3) *связанная дуга* u между двумя вершинами, которая связана с еще $k-1$ дугой, имеющей одинаковый один из концов; множество связанных дуг, выходящих из одной вершины или входящих в одну вершину, будем называть *мультидугой* u^m; пропускная способность всех связанных дуг одной мультидуги одинакова; поток по каждой связанной дуге одной мультидуги одинаков; множество мультидуг обозначим через U^m.

Множество выходящих из вершины дуг может быть либо только кратными дугами, либо только одной мультидугой (k связанных дуг), либо только обычными дугами. Из источника x_0 сети выходят только кратные дуги, а в сток z сети входит только одна мультидуга. Если из вершины выходят связанные дуги мультидуги, то в нее обязательно входит кратная дуга. Если в вершину входит мультидуга, то из нее может выходить только кратная дуга. Определенный таким образом мультиграф $G(X, U)$ с целочисленными пропускными способностями дуг назовем *кратной (транспортной) сетью*.

Кратным потоком по сети называется целочисленная функция, определенная на множестве дуг $U = U^o \cup U^k \cup U^m$, для которой выполнены условия неотрицательности, ограниченности (пропускными способностями дуг) и неразрывности потока (в каждой вершине).

Величиной кратного потока называется сумма φ_z входящего потока для стока z, равная сумме выходящего из источника потока. В силу того, что поток по каждой обычной дуге и по каждой связанной дуге каждой кратной и мультидуги должен быть целочислен, величина φ_z должна быть кратна k. Как и обычно, обозначим через $c(u)$ *пропускную способность* дуги u, а через $f(u)$ – *поток* на ней.

Отметим, что при $k = 1$ кратная сеть превращается в обычную транспортную сеть, а кратный поток превращается в обычный поток по этой сети. *Задача о наибольшем кратном потоке* для кратной сети является обобщением задачи о наибольшем потоке для обычной транспортной сети.

Пусть имеется кратная сеть произвольной кратности k. Пусть при удалении всех мультидуг сеть распадается на $k+2$ слабо связных компоненты, при этом одна компонента состоит только из вершины z, компонента, содержащая вершину x_0, содержит только кратные дуги, а остальные k компонент содержат только обычные дуги. Если при этом каждая мультидуга имеет ровно один конец в каждой из k компонент, содержащих обычные дуги, то такую сеть мы будем называть *делимой*. Примером делимой сети может служить кратная сеть целочисленного сбалансирования трехмерной матрицы, подробно рассмотренная в [4] – [6].

Обозначим через P_0 компоненту делимой сети, содержащую кратные дуги, через $P_1, \ldots, P_k$ обозначим компоненты, содержащие обычные дуги. Тогда *частью* G_i делимой сети ($i \in \overline{1,k}$) назовем объединение соответствующей компоненты P_i с

инцидентными ей связанными дугами всех мультидуг, кроме мультидуги с концом в z, а также с i-ой связанной дугой каждой кратной дуги компоненты P_0. Заметим, что возможность выделения частей G_i является особенностью делимых сетей, в общем случае это не всегда возможно.

Обозначим начальные вершины мультидуги с концом в z через $z_1, \ldots, z_k$. Вершины, являющиеся началом остальных мультидуг, будем обозначать через y_j.

Объединение мультидуги u_z^m, идущей из вершин $z_1, \ldots, z_k$ в z и k путей μ_r $(r \in \overline{1,k})$, каждый из которых является ориентированным путем в соответствующей части G_r из вершины x_0 в вершину z_r, назовем *обобщенным путем*, если каждый из путей μ_r проходит через одну и ту же вершину y_j.

Кратный поток назовем *полным*, если любой обобщенный путь из x_0 в z имеет дугу u (обычную, кратную или мультидугу), поток по которой равен ее пропускной способности $f(u) = c(u)$.

Проекция C_i $(i \in \overline{1,k})$ подграфа C на часть сети G_i – это часть подграфа C, образованная его вершинами и дугами, принадлежащими G_i.

Так как части G_i $(i \in \overline{1,k})$ сети G представляют собой обычные транспортные сети с источником x_0 и стоком z_i, то будем называть некоторый путь из x_0 в z_i *путем прорыва* в части G_i, если $f(u) < c(u)$ на прямых дугах и $f(u) > 0$ на обратных дугах этого пути.

Пусть в делимой сети определен некоторый поток φ величины $\varphi_z \geq 0$. *Кратным циклом* в делимой сети $G(X,U)$, где X – множество вершин, U – множество дуг (обычных, кратных или мультидуг), назовем такой подграф $C(X',U')$, $X' \subseteq X$, $U' \subseteq U$, для которого:

1) проекции $C_1, \ldots, C_k$ на части $G_1, \ldots, G_k$ соответственно есть объединение некоторых циклов, причем дуги, поток по которым ненулевой, могут проходиться в обратном направлении;

2) проекции $C_1, \ldots, C_k$ согласованы (одинаковы) на общей части подграфов $G_1, \ldots, G_k$;

3) C_1 представим в виде $C_1 = \cup\{C_1^j\}$, где C_1^j – некоторые циклы и $C_1^j \not\subseteq C_1^k \; \forall j \neq k$; при этом для любой дуги u из G_1 выполняется неравенство

$$0 \leq f(u) + a^+(u) - a^-(u) \leq c(u),$$

где $a^+(u)$ – это число циклов C_1^j, в которых дуга u проходится в прямом направлении, а $a^-(u)$ – это число циклов C_1^j, в которых дуга u проходится в обратном направлении. Такое же условие должно выполняться и для $C_2, \ldots, C_k$.

Обобщенным путем прорыва в делимой сети $G(X,U)$ для некоторого кратного потока φ назовем такой подграф $S(X',U')$, $X' \subseteq X$, $U' \subseteq U$, для которого:

1) каждая из проекций $S_1, \ldots, S_k$ на части $G_1, \ldots, G_k$ соответственно есть объединение ровно одного пути прорыва из x_0 в z_i и некоторых циклов, причем дуги, поток по которым ненулевой, могут проходиться в обратном направлении;

2) проекции $S_1, \ldots, S_k$ согласованы (одинаковы) на общей части подграфов $G_1, \ldots, G_k$;

3) S_1 представим в виде $S_1 = \mu_1 \cup \{C_1^j\}$, где μ_1 – путь прорыва, C_1^j – некоторые циклы и $C_1^j \not\subseteq C_1^k \; \forall j \neq k$; при этом для любой дуги u из G_1 выполняется неравенство

$$0 \leq f(u) + a^+(u) - a^-(u) \leq c(u),$$

где $a^+(u)$ – это число элементов множества $\mu_1 \cup \{C_1^j\}$, в которых дуга u проходится в прямом направлении, а $a^-(u)$ – это число элементов множества $\mu_1 \cup \{C_1^j\}$, в которых дуга u проходится в обратном направлении. Такое же условие должно выполняться и для $S_2, \ldots, S_k$;

4) Дуга $(\{z_1, \ldots, z_k\}, z) \in U'$ и $f(\{z_1, \ldots, z_k\}, z) < c(\{z_1, \ldots, z_k\}, z)$.

5) S не содержит кратного цикла.

Обозначим $G_\varphi = G(X, U_\varphi)$; $U_\varphi = \{u \mid f(u) \geq 0\}$. Справедливо следующее утверждение.

Теорема 1. *Пусть $\varphi^1(U)$, $\varphi^2(U)$ – два полных потока в делимой сети $G(X, U)$, причем $\varphi_z^1 \leq \varphi_z^2$. Пусть $\varphi^0(U) = \varphi^2(U) - \varphi^1(U)$. Тогда граф $G_{\varphi^0} = \{S_i\} \cup \{C_j\}$, где $\{S_i\}$ – множество всех обобщенных путей прорыва из x_0 в z, а $\{C_j\}$ – множество всех кратных циклов. В случае, когда $\varphi_z^1 = \varphi_z^2 = \varphi_z^{\max}$, $\{S_i\} = \varnothing$.*

Теорема 1 была доказана в статье [1] на базе результатов из работы [4].

Отметим также, что если в делимой сети существует поток величины kT (T – натуральное число), то в ней всегда существует поток любой другой величины kS ($1 \leq S < T$, S – натуральное число). Для произвольной кратной сети это условие выполнено не всегда.

На данном свойстве основан алгоритм нахождения максимального потока в делимой сети, который был подробно рассмотрен в работе [1]. Здесь мы ограничимся только идеей алгоритма.

Работа алгоритма происходит в два этапа:

1) *этап построения полного потока*;

2) *этап увеличения потока*. Если полученный полный поток не является максимальным, то увеличиваем его при помощи *обобщенного алгоритма пометок* до тех пор, пока поток не станет максимальным.

Алгоритм нахождения полного потока прост. На данном этапе увеличения потока можно добиваться, находя все возможные обобщенные пути прорыва из x_0 в z через вершины y_j без обратных дуг и увеличивая поток по ним. В итоге будет получен полный поток.

Рассмотрим обобщенный алгоритм пометок. Идея алгоритма состоит в следующем: в частях $G_1, \ldots, G_k$ поочередно строятся пути прорыва $\mu_1, \ldots, \mu_k$ (возможно, в объединении с некоторыми циклами) до тех пор, пока пути во всех частях не станут согласованными, либо же не останется вариантов для продолжения построения пути. В первом случае объединение $\mu_1, \ldots, \mu_k$ с мультидугой с концом в z даст обобщенный путь прорыва, во втором случае производится откат до *«точки ветвления»*, после чего выполнение алгоритма возобновляется.

Отметим, что указанный алгоритм является экспоненциальным, а задача нахождения максимального потока в кратной сети является NP-полной (обоснование см. в [1]). Поэтому важным является вопрос поиска подклассов полиномиально разрешимых задач о максимальном кратном потоке.

2. Делимая кратная сеть с дополнительным условием для мультидуг

Следующее утверждение устанавливает полиномиальную разрешимость задачи о максимальном кратном потоке для одного подкласса делимых сетей.

Теорема 2. *Пусть $G(X,U)$ – делимая сеть кратности k. Пусть в каждой компоненте P_i ($i \in \overline{1,k-1}$) существует только одна вершина y_0^i, являющаяся концом мультидуги, а в компоненте P_k таких вершин может быть несколько. Тогда задача нахождения максимального кратного потока разрешима за полиномиальное время.*

Доказательство. Очевидно, что любой обобщенный путь прорыва из x_0 в z в сети $G(X,U)$ указанного вида будет проходить через вершины y_0^1, …, y_0^{k-1} (то же справедливо и для обобщенных путей из x_0 в z). Каждая из компонент P_i ($i \in \overline{1,k-1}$) представляет собой обычную транспортную сеть с источником y_0^i и стоком z_i. С помощью алгоритма Форда–Фалкерсона (см. [2]) найдем максимальный поток в каждой из таких сетей. Величины соответствующих потоков обозначим через φ_i.

В части G_k сети $G(X,U)$, являющейся обычной транспортной сетью с источником x_0 и стоком z_k, также найдем максимальный поток величины φ_k.

Пусть

$$F = \min\left\{\varphi_1, \ldots, \varphi_k, \frac{c\left(\{z_1, \ldots, z_k\}, z\right)}{k}\right\}. \tag{1}$$

Очевидно, что в силу построения сети $G(X,U)$ величина максимального потока в ней

$$\varphi_z^{\max} \le kF. \tag{2}$$

Покажем, что в сети $G(X,U)$ можно получить поток величины kF. Последовательно применим следующие действия.

1. В каждой из компонент P_i ($i \in \overline{1,k-1}$) и в части G_k уменьшим потоки на соответствующие величины $d_i = \varphi_i - F$. Для этого последовательно будем находить в этих транспортных сетях пути из источника в сток с ненулевым потоком на всех дугах и уменьшать этот поток.

2. В частях G_i ($i \in \overline{1,k-1}$) сети $G(X,U)$ установим поток на всех связанных дугах всех кратных и мультидуг равным соответствующим величинам из части G_k.

3. Установим $f(\{z_1, \ldots, z_k\}, z) = kF$.

Очевидно, что полученный кратный поток φ величины $\varphi_z = kF$ будет удовлетворять условиям неотрицательности, ограниченности пропускными способностями дуг и неразрывности в каждой вершине. При этом величина потока на каждой кратной и мультидуге будет кратна k, а потоки во всех частях G_i ($i \in \overline{1,k}$) сети $G(X,U)$ будут согласованы. В силу (2) поток максимален.

Все проводимые нами операции были полиномиальны, а для поиска максимальных потоков в обычных транспортных сетях мы использовали полиномиальный алгоритм Форда–Фалкерсона.

Теорема 2 доказана.

В доказательстве теоремы фактически содержится следующий простой полиномиальный алгоритм поиска наибольшего кратного потока в делимой сети указанного вида.

Алгоритм 1.

1. С помощью алгоритма Форда–Фалкерсона найдем максимальные потоки в компонентах P_i ($i \in \overline{1,k-1}$) и в части G_k сети $G(X,U)$, обозначим их через φ_i.

2. Найдем F по формуле (1). Уменьшим потоки в компонентах P_i ($i \in \overline{1,k-1}$) и в части G_k сети $G(X,U)$ до величины F и согласуем их. Для этого произведем действия 1–3 из доказательства теоремы. В результате получим максимальный кратный поток φ величины kF.

3. Понижение размерности в задаче о наибольшем кратном потоке для некоторых делимых сетей

Пусть теперь $G(X,U)$ – делимая сеть кратности k, в которой в каждой компоненте P_i ($i \in \overline{1,s}$, $1 \le s < k-1$) существует только одна вершина y_0^i, являющаяся концом мультидуги, а в компонентах P_i ($i \in \overline{s+1,k}$) таких вершин несколько. Для такой сети можно предложить следующий алгоритм нахождения максимального потока (получаемый как следствие из теоремы 2, доказанной в предыдущем разделе).

Алгоритм 2.

1. С помощью алгоритма Форда–Фалкерсона находим максимальный поток величины φ_i в компонентах P_i ($i \in \overline{1,s}$), каждая из которых является обычной транспортной сетью с источником y_0^i и стоком z_i. Обозначим

$$F = \min_{i \in \overline{1,s}} \varphi_i.$$

2. Объединение

$$G' = \cup_{i=s+1}^{k} G_s \cup (\{z_{s+1}, \ldots, z_k\}, z)$$

частей G_s сети $G(X,U)$ и $k-s$ связанных дуг мультидуги с концом в z является кратной сетью кратности $k-s$ с источником x_0 и стоком z.

С помощью общего алгоритма поиска наибольшего потока в кратной делимой сети будем искать максимальный поток в сети G'. Поскольку максимальный поток φ в сети $G(X,U)$ не превышает величины kF, то при достижении величины потока в сети G' равной $(k-s)F$ процедуру поиска можно остановить. Таким образом, мы заведомо получим, что величина найденного в сети G' потока $\varphi_0 \le (k-s)F$.

3. В компонентах P_i ($i \in \overline{1,s}$) уменьшим потоки на соответствующие величины $d_i = \varphi_i - \frac{\varphi_0}{k-s}$. Для этого последовательно будем находить пути из y_0^i в z_i с ненулевым потоком на них и уменьшать этот поток.

4. Поток на всех связанных дугах всех кратных и мультидуг в частях G_i ($i \in \overline{1,s}$) сети $G(X,U)$ установим равным соответствующим величинам из частей G_i ($i \in \overline{s+1,k}$).

5. Поток на мультидуге с концом в z сделаем равным $\frac{k}{k-s}\varphi_0$.

Получившийся кратный поток φ в делимой сети $G(X,U)$ величины $\varphi_z = \frac{k}{k-s}\varphi_0$ является максимальным.

Таким образом, для делимой сети указанного вида с помощью алгоритма 2 удается понизить размерность задачи о максимальном кратном потоке до $k-s$. При

этом кратный поток кратности $k-s$ по-прежнему ищется с помощью экспоненциального алгоритма, но количество вычислительных операций и общее время решения сокращается.

4. Делимая сеть параллельной структуры

Пусть делимая сеть $G(X,U)$ устроена таким образом, что для любой пары вершин y_j, y_t $(j \neq t)$ произвольный путь из x_0 в y_j и произвольный путь из x_0 в y_t не имеют общих вершин, кроме x_0. В этом случае будем говорить, что сеть $G(X,U)$ имеет *параллельную структуру*.

Пусть для определенности $j \in \overline{1,m}$. Компоненту P_0 сети $G(X,U)$ можно разделить на субкомпоненты P_0^j, каждая из которых состоит из всех возможных путей из x_0 в y_j. Очевидно, что

$$P_0^1 \cup P_0^2 \cup \ldots \cup P_0^m = P_0; \quad P_0^j \cap P_0^t = \{x_0\}, \quad j \neq t.$$

При этом каждая субкомпонента P_0^j – это обычная транспортная сеть с источником x_0 и стоком y_j, содержащая только кратные дуги. Следовательно, в каждой такой субкомпоненте можно применить (с учетом кратности дуг) полиномиальный алгоритм Форда–Фалкерсона для нахождения максимального потока.

Рассмотрим следующий алгоритм нахождения наибольшего кратного потока в делимой сети $G(X,U)$ параллельной структуры кратности $k=2$.

Алгоритм 3.

1. В каждой из субкомпонент P_0^j $(j \in \overline{1,m})$ с помощью алгоритма Форда–Фалкерсона найдем максимальный поток, его величину обозначим через φ_j.

2. Выполним сведение задачи о наибольшем кратном потоке в сети $G(X,U)$ к задаче о наибольшем потоке в обычной сети $G_1(X_1,U_1)$ по следующим правилам.

2.1. $X_1 = X \setminus \{x_0\}$.

2.2. Добавим в U_1 все дуги из компонент P_1 и P_2 сети $G(X,U)$, дугам из компоненты P_1 при этом поменяем ориентацию на противоположную.

2.3. Соединим вершину z_2 с вершиной z дугой пропускной способности

$$c(z_2,z) = \frac{c(\{z_1,z_2\},z)}{2}.$$

2.4. Каждую мультидугу вида $\left(y_j, \left\{y_p^1, y_q^2\right\}\right) \in U$ преобразуем в пару дуг (y_p^1, y_j), (y_j, y_q^2) пропускной способности

$$c(y_p^1,y_j) = c(y_j,y_q^2) = \frac{1}{2}\min\left\{\varphi_j, c\left(y_j,\left\{y_p^1,y_q^2\right\}\right)\right\}.$$

Полученные таким образом дуги включим в U_1.

3. На шаге 2 нами была получена сеть $G_1(X_1,U_1)$ с источником z_1 и стоком z. С помощью алгоритма Форда–Фалкерсона найдем максимальный поток в сети $G_1(X_1,U_1)$ величины φ_0.

4. Перенесем поток из сети $G_1(X_1,U_1)$ в сеть $G(X,U)$ по следующим правилам.

4.1. Поток на каждой обычной дуге (a,b) установим равным величине потока на соответствующей дуге из U_1 (для компоненты P_1 это дуги (b,a), для компоненты P_2 – (a,b)).

4.2. Поток на мультидуге с концом в z установим равным

$$f(\{z_1, z_2\}, z) = 2f_1(z_2, z) = 2\varphi_0.$$

Здесь $f_1(z_2, z)$ – поток на дуге (z_2, z) в сети $G_1(X_1, U_1)$.

4.3. Поток на каждой мультидуге, начинающейся в вершине y_j ($j \in \overline{1, m}$), установим равным $2f_1(y_j)$, где $f_1(y_j)$ – величина потока, проходящего через вершину y_j в сети $G_1(X_1, U_1)$.

4.4. Для нахождения потока на кратных дугах в каждой субкомпоненте P_0^j возьмем максимальный поток в этой субкомпоненте, найденный на шаге 1, и уменьшим его на величину $d_j = \varphi_j - 2f_1(y_j)$. Для этого будем последовательно находить пути из x_0 в y_j с ненулевым потоком на них и уменьшать этот поток.

Полученный в итоге кратный поток φ обладает свойствами неотрицательности, ограниченности пропускными способностями дуг и непрерывности в каждой вершине; поток на каждой кратной и мультидуге согласован и кратен 2. Этот поток является максимальным, его величина $\varphi_z = 2\varphi_0$.

Теорема 3. *В делимой кратной сети $G(X, U)$ параллельной структуры можно найти максимальный кратный поток за полиномиальное время, если кратность сети $k = 2$.*

Доказательство. Справедливость теоремы следует непосредственно из алгоритма 3. Действительно, переход от кратной сети $G(X, U)$ к обычной сети $G_1(X_1, U_1)$ осуществляется таким образом, что максимальный поток в сети $G_1(X_1, U_1)$ всегда индуцирует максимальный кратный поток в сети $G(X, U)$. При этом действия на каждом из шагов алгоритма полиномиальны.

Теорема 3 доказана.

Рассмотрим теперь задачу распознавания следующего вида: требуется определить, существует ли в делимой сети $G(X, U)$ параллельной структуры кратности k кратный поток величины не меньшей K. Эту задачу мы будем нвзывать *задачей КПk_ пар.*

В работе [1] была рассмотрена аналогичная задача распознавания КПk, поставленная для произвольной кратной сети, и обоснована ее NP-полнота. Задача КПk_пар является частным случаем задачи КПk. При этом задача КП2_пар полиномиальна (теорема 3) в отличие от задачи КП2. Покажем, что при $k \geq 3$ задача КПk_пар NP-полна. Для этого рассмотрим следующую классическую NP-полную *задачу о трехмерном сочетании* (см. [19] – [20]).

Задано 3 множества I, J, P с n элементами (можно считать, что $I = J = P = \{1, \ldots, n\}$) и подмножество M прямого произведения $I \times J \times P$.

Требуется определить, можно ли из M выбрать подмножество M', у элементов которого значение каждой координаты i, j или p присутствует ровно 1 раз.

Теорема 4. *Задача КПk_ пар NP-полна при $k \geq 3$.*

Доказательство. В работе [1] показано, что задача КПk принадлежит классу NP для любых натуральных значений k, следовательно, задача КПk_пар также принадлежит классу NP.

Пусть сначала $k = 3$.

Построим полиномиальное сведение задачи о трехмерном сочетании к задаче КПЗ_пар.

1. Каждому элементу $(i, j, p) \in M$ поставим в соответствие вершину x_{ijp} кратной сети.

2. Соединим источник сети x_0 с каждой из вершин x_{ijp} кратной дугой пропускной способности 3.

3. Добавим в сеть вершины x_{i00}, x_{0j0}, x_{00p}, соответствующие всем ненулевым значениям i, j и p. Соединим каждую вершину x_{ijp} $(i > 0, j > 0, p > 0)$ с вершинами x_{i00}, x_{0j0}, x_{00p} мультидугой $(x_{ijp}, \{x_{i00}, x_{0j0}, x_{00p}\})$ пропускной способности 3.

4. Добавим в сеть вершины z_1, z_2, z_3. Каждую вершину x_{i00} $(i > 0)$ соединим с вершиной z_1 обычной дугой пропускной способности 1. Каждую вершину x_{0j0} $(j > 0)$ соединим с вершиной z_2 обычной дугой пропускной способности 1. Каждую вершину x_{00p} $(p > 0)$ соединим с вершиной z_3 обычной дугой пропускной способности 1.

5. Соединим вершины z_1, z_2, z_3, со стоком сети z мультидугой $(\{z_1, z_2, z_3\}, z)$ пропускной способности $3n$.

6. Установим $K = 3n$.

Полученная делимая сеть $G(X, U)$ кратности 3 имеет параллельную структуру, причем величина потока в этой сети не может превысить $3n$.

Если задача о трехмерном сочетании разрешима, то ее решение $M' \subseteq M$ индуцирует решение задачи КПЗ_пар для сети $G(X, U)$ указанного вида, если положить

$$f(x_0, x_{ijp}) = f\left(x_{ijp}, \{x_{i00}, x_{0j0}, x_{00p}\}\right) = 3, \quad (i, j, p) \in M';$$

$$f(x_0, x_{ijp}) = f\left(x_{ijp}, \{x_{i00}, x_{0j0}, x_{00p}\}\right) = 0, \quad (i, j, p) \notin M', \quad i > 0,\ j > 0,\ p > 0;$$

$$f(x_{i00}, z_1) = f(x_{0j0}, z_2) = f(x_{00p}, z_3) = 1, \quad i > 0,\ j > 0,\ p > 0;$$

$$f\left(\{z_1, z_2, z_3\}, z\right) = 3n.$$

В силу построения сети $G(X, U)$ и требования целочисленности потока на всех дугах, если $f(x_{i_1 j_1 p_1}) = f(x_{i_2 j_2 p_2}) = 3$ для ненулевых значений индексов, то обязательно $i_1 \neq i_2$, $j_1 \neq j_2$ и $p_1 \neq p_2$ (иначе либо $f(x_{i_1 00}) > c(x_{i_1 00})$, либо $f(x_{0 j_1 0}) > c(x_{0 j_1 0})$, либо $f(x_{00 p_1}) > c(x_{00 p_1})$). Поэтому, если в задаче КПЗ_пар для сети $G(X, U)$ указанного вида есть решение при $K = 3n$, то это решение индуцирует решение задачи о трехмерном сочетании

$$M' = \{(i, j, p) \mid f(x_{ijp}) = 3,\ i > 0,\ j > 0,\ p > 0\}.$$

Таким образом, мы построили сведение известной NP-полной задачи о трехмерном сочетании к задаче КПЗ_пар. Поскольку шаги 1–6 полиномиальны, данное сведение является полиномиальным. Следовательно, задача КПЗ_пар NP-полна.

Пусть теперь $k > 3$. Процедура построения полиномиального сведения задачи о трехмерном сочетании к задаче КПk_пар будет аналогична изложенной выше со следующими дополнениями.

1. Добавим в сеть вершины $y_4, \ldots, y_k$, $z_4, \ldots, z_k$.

2. Все имеющиеся мультидуги $(x_{ijp}, \{x_{i00}, x_{0j0}, x_{00p}\})$ заменим на мультидуги $(x_{ijp}, \{x_{i00}, x_{0j0}, x_{00p}, y_4, \ldots, y_k\})$ пропускной способности k.

3. Соединим все вершины y_s $(s \in \overline{4, k})$ с вершинами z_s обычными дугами пропускной способности n.

4. Заменим мультидугу $(\{z_1, z_2, z_3\}, z)$ на мультидугу $(\{z_1, \ldots, z_k\}, z)$ пропускной способности kn.

5. Установим пропускную способность всех кратных дуг равной k.

6. Установим $K = kn$.

Нетрудно убедиться, что при $k > 3$ задача КПk_пар NP-полна (рассуждения аналогичны изложенным выше).

Теорема 4 доказана.

Следствие 1. *Задача о наибольшем кратном потоке в делимой сети $G(X, U)$ параллельной структуры NP-полна при кратности сети $k \geq 3$.*

Доказательство. Задача КПk_пар может быть решена при помощи следующего алгоритма:

1. Поиск максимального кратного потока φ в сети $G(X, U)$ величины φ_z.

2. Проверка $\varphi_z \geq K$ (соответствует ответу «да» в задаче КПk_пар).

Проверка шага 2 выполняется за константное время, а задача КПk_пар NP-полна при $k \geq 3$. Следовательно, задача о наибольшем кратном потоке в делимой сети $G(X, U)$ параллельной структуры также является NP-полной при $k \geq 3$.

Следствие 1 доказано.

Список литературы / References

[1] Рублев В. С., Смирнов А. В., “Потоки в кратных сетях”, *Ярославский педагогический вестник*, **3**:2 (2011), 60–68; [Rublev V. S., Smirnov A. V., “Flows in Multiple Networks”, *Yaroslavsky Pedagogichesky Vestnik*, **3**:2 (2011), 60–68, (in Russian).]

[2] Ford L. R., Fulkerson D. R., *Flows in Networks*, Princeton University Press, 1962.

[3] Papadimitriou Ch. H., Steigliz K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, 1982.

[4] Рублев В. С., Смирнов А. В., “Задача целочисленного сбалансирования трехмерной матрицы и алгоритмы ее решения”, *Моделирование и анализ информационных систем*, **17**:2 (2010), 72–98; [Roublev V. S., Smirnov A. V., “The Problem of Integer-Valued Balancing of a Three-Dimensional Matrix and Algorithms of Its Solution”, *Modeling and Analysis of Information Systems*, **17**:2 (2010), 72–98, (in Russian).]

[5] Смирнов А. В., “Некоторые классы разрешимости задачи целочисленного сбалансирования трехмерной матрицы с ограничениями второго рода”, *Моделирование и анализ информационных систем*, **20**:2 (2013), 54–69; [Smirnov A. V., “Some Solvability Classes for The Problem of Integer Balancing of a Three-dimensional Matrix with Constraints of Second Type”, *Modeling and Analysis of Information Systems*, **20**:2 (2013), 54–69, (in Russian).]

[6] Smirnov A. V., “Some Solvability Classes for the Problem of Integer Balancing of a Three-Dimensional Matrix with Constraints of the Second Type”, *Automatic Control and Computer Sciences*, **48**:7 (2014), 543–553.

[7] Корбут А. А., Финкельштейн Ю. Ю., *Дискретное программирование*, Наука, 1969; [Korbut A. A., Finkelstein J. J., *Diskretnoe programmirovanie*, Nauka, 1969, (in Russian).]

[8] Раскин Л. Г., Кириченко И. О., *Многоиндексные задачи линейного программирования*, Радио и связь, 1982; [Raskin L. G., Kirichenko I. O., *Mnogoindeksnye zadachi lineynogo programmirovaniya*, Radio i svyaz, 1982, (in Russian).]

[9] Spieksma F. C. R., “Multi index assignment problems: complexity, approximation, applications”, *Nonlinear Assignment Problems. Algorithms and Applications*, eds. P. M. Pardalos, L. S. Pitsoulis, Kluwer Academic Publishers, 2000, 1–11.

[10] Афраймович Л. Г., "Трехиндексные задачи линейного программирования с вложенной структурой", *Автоматика и телемеханика*, 2011, № 8, 109–120; English transl.: Afraimovich L. G., "Three-index linear programs with nested structure", *Automation and Remote Control*, **72**:8 (2011), 1679–1689.

[11] Кондаков А. С., Рублев В. С., "Задача сбалансирования матрицы плана", *Доклады Одесского семинара по дискретной математике*, Астропринт, 2005, 24–26; [Kondakov A. S., Roublev V. S., "Zadacha sbalansirovaniya matritsy plana", *Doklady Odesskogo seminara po diskretnoy matematike*, Astroprint, 2005, 24–26, (in Russian).]

[12] Коршунова Н. М., Рублев В. С., "Задача целочисленного сбалансирования матрицы", *Современные проблемы математики и информатики*, ЯрГУ, 2000, 145–150; [Korshunova N. M., Roublev V. S., "Zadacha tselochislennogo sbalansirovaniya matritsy", *Sovremennye problemy matematiki i informatiki*, Yaroslavl State University, 2000, 145–150, (in Russian).]

[13] Смирнов А. В., "Задача целочисленного сбалансирования трехмерной матрицы и сетевая модель", *Моделирование и анализ информационных систем*, **16**:3 (2009), 70–76; [Smirnov A. V., "The Problem of Integer-valued Balancing of a Three-dimensional Matrix and Network Model", *Modeling and Analysis of Information Systems*, **16**:3 (2009), 70–76, (in Russian).]

[14] Афраймович Л. Г., Прилуцкий М. Х., "Многоиндексные задачи распределения ресурсов в иерархических системах", *Автоматика и телемеханика*, 2006, № 6, 194–205; English transl.: Afraimovich L. G., Prilutskii M. Kh., "Multiindex resource distributions for hierarchical systems", *Automation and Remote Control*, **67**:6 (2006), 1007–1016.

[15] Афраймович Л. Г., Прилуцкий М. Х., "Многопродуктовые потоки в древовидных сетях", *Известия РАН. Теория и системы управления*, 2008, № 2, 57–63; English transl.: Afraimovich L. G., Prilutskii M. Kh., "Multicommodity flows in tree-like networks", *Journal of Computer and Systems Sciences International*, **47**:2 (2008), 214–220.

[16] Hoffman A. J., Kruskal J. B., "Integral Boundary Points of Convex Polyhedra", *Linear Inequalities and Related Systems*, eds. H. W. Kuhn, A. W. Tucker, Princeton University Press, 1972, 223–246.

[17] Рублев В. С., Смирнов А. В., "NP-полнота задачи целочисленного сбалансирования трехмерной матрицы", *Доклады Академии Наук*, **435**:3 (2010), 314–316; English transl.: Roublev V. S., Smirnov A. V., "NP-Completeness of the Integer Balancing Problem for a Three-Dimensional Matrix", *Doklady Mathematics*, **82**:3 (2010), 912–914.

[18] Смирнов А. В., "Некоторые полиномиальные подклассы задачи о наибольшем кратном потоке в делимой сети", *Дискретные модели в теории управляющих систем: IX Международная конференция: труды*, МАКС Пресс, 2015, 229–231; [Smirnov A. V., "Nekotorye polinomialnye podklassy zadachi o naibolshem kratnom potoke v delimoy seti", *Discrete Models in Control Systems Theory: IX International Conference: Proceedings*, MAKS Press, 2015, 229–231, (in Russian).]

[19] Garey M. R., Johnson D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[20] Karp R., "Reducibility among combinatorial problems", *Complexity of Computer Computations*, eds. R. E. Miller, J. W. Thatcher, Plenum, 1972, 85–103.

DOI: 10.18255/1818-1015-2015-4-533-545

The Problem of Finding the Maximal Multiple Flow in the Divisible Network and its Special Cases

Smirnov A. V.

Received July 10, 2015

In the article the problem of finding the maximal multiple flow in the network of any natural multiplicity k is studied. There are arcs of three types: ordinary arcs, multiple arcs and multi-arcs. Each multiple and multi-arc is a union of k linked arcs, which are adjusted with each other. The network constructing rules are described. The definitions of a divisible network and some associated subjects are stated. The important property of the divisible network is that every divisible network can be partitioned into k parts, which are adjusted on the linked arcs of each multiple and multi-arc. Each part is the ordinary transportation network. The main results of the article are the following subclasses of the problem of finding the maximal multiple flow in the divisible network. 1. The divisible networks with the multi-arc constraints. Assume that only one vertex is the ending vertex for a multi-arc in $k-1$ network parts. In this case the problem can be solved in a polynomial time. 2. The divisible networks with the weak multi-arc constraints. Assume that only one vertex is the ending vertex for a multi-arc in s network parts ($1 \leq s < k-1$) and other parts have at least two such vertices. In that case the multiplicity of the multiple flow problem can be decreased to $k-s$. 3. The divisible network of the parallel structure. Assume that the divisible network component, which consists of all multiple arcs, can be partitioned into subcomponents, each of them containing exactly one vertex-beginning of a multi-arc. Suppose that intersection of each pair of subcomponents is the only vertex-network source x_0. If $k = 2$, the maximal flow problem can be solved in a polynomial time. If $k \geq 3$, the problem is NP-complete. The algorithms for each polynomial subclass are suggested. Also, the multiplicity decreasing algorithm for the divisible network with weak multi-arc constraints is formulated.

Keywords: multiple networks, multiple flows, divisible networks, NP-completeness, polynomial algorithm

For citation: Smirnov A. V., "The Problem of Finding the Maximal Multiple Flow in the Divisible Network and its Special Cases", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 533–545.

On the authors:
Smirnov Alexander Valeryevich, orcid.org/0000-0002-0980-2507, PhD,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: alexander_sm@mail.ru

Acknowledgments:

This work was supported by the Russian Foundation for Basic Research under the Grant No 15-07-03038 A.

Моделирование и анализ информационных систем. Т. 22, №4 (2015), с. 546–562
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 546–562

DOI: 10.18255/1818-1015-2015-4-546-562

УДК 004.72, 004.057.4, 004.023

Инструментальная система для поддержки разработки и исследования программно-конфигурируемых сетей подвижных объектов

Соколов В. А., Корсаков С. В., Смирнов А. В., Башкин В. А., Никитин Е. С.

получена 4 сентября 2015

В данной статье рассмотрены принципы организации беспроводных mesh-сетей — программно-конфигурируемых сетей подвижных объектов. Основное внимание уделяется вопросам построения эффективных алгоритмов маршрутизации для подобных сетей.

Математической моделью системы является стандартная транспортная сеть. В качестве ключевого параметра системы маршрутизации рассматривается коэффициент доступности узла — функция, зависящая от ряда основных и дополнительных параметров («mesh-факторов»), характеризующих маршрут между двумя узлами сети. Каждой паре (дуга, узел) сопоставляется композитный параметр, характеризующий «доступность» узла по маршруту, начинающемуся данной дугой. Лучшим («кратчайшим») маршрутом между двумя узлами считается маршрут с наибольшим коэффициентом доступности.

Описаны правила построения и обновления таблиц маршрутизации узлами сети. Получая анонс от соседа, узел имеет сведения об энергетике соединения, надежности соединения, времени получения анонса, отсутствии промежуточных узлов, а также располагаемой пропускной способности. На основании этой информации ко всем маршрутам, проходящим через данного соседа, может быть применена пенализация (наложение штрафа) или поощрение (увеличение коэффициента доступности). Указанная схема пенализации / поощрения складывается из отдельных аспектов:

1. Пенализация за актуальность информации.
2. Пенализация / вознаграждение за надежность узла.
3. Пенализация за энергетику соединения.
4. Пенализация за располагаемую пропускную способность.

На основе предложенных эвристических алгоритмов маршрутизации построен симулятор беспроводной mesh-сети подвижных объектов, описание и характеристики которого приведены в статье. Также рассмотрены особенности программной реализации симулятора.

Ключевые слова: mesh-сеть, сетевой протокол, маршрутизация, пенализация, эвристический алгоритм, симулятор

Для цитирования: Соколов В. А., Корсаков С. В., Смирнов А. В., Башкин В. А., Никитин Е. С., "Инструментальная система для поддержки разработки и исследования программно-конфигурируемых сетей подвижных объектов", *Моделирование и анализ информационных систем*, **22**:4 (2015), 546–562.

Об авторах:
Соколов Валерий Анатольевич, orcid.org/0000-0003-1427-4937,
докт. физ.-мат. наук., профессор, заведующий кафедрой теоретической информатики,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: valery-sokolov@yandex.ru

Корсаков Станислав Валентинович, orcid.org/0000-0003-2349-7980,
директор, ООО «Нетше лаб»,
ул. Белинского, 28-75, г. Ярославль, 150047 Россия,
ассистент кафедры теоретической информатики,

Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: sta@stasoft.net

Смирнов Александр Валерьевич, orcid.org/0000-0002-0980-2507,
канд. физ.-мат. наук., доцент кафедры теоретической информатики,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: alexander_sm@mail.ru

Башкин Владимир Анатольевич, orcid.org/0000-0002-2534-1026,
докт. физ.-мат. наук., доцент, доцент кафедры теоретической информатики,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: v_bashkin@mail.ru

Никитин Евгений Сергеевич, orcid.org/0000-0002-2341-9950, студент,
Ярославский государственный университет им. П.Г. Демидова,
ул. Советская, 14, г. Ярославль, 150000 Россия, e-mail: nik.zhenya@gmail.com

Благодарности:

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект № 15-07-03038 А).

Введение

Стремительное развитие Интернета вещей, автономных либо удаленно управлямых беспилотных аппаратов (исследовательских, спасательных и других робототехнических комплексов) и средств телеприсутствия переводит в разряд важнейших проблем организацию и поддержание эффективной работы беспроводных сетей передачи данных, узлами которых являются подвижные объекты.

Подобные подвижные беспроводные сети обладают рядом существенных особенностей, таких как:

— множественность узлов;

— перемещение узлов сети относительно друг друга и относительно неподвижного наблюдателя с некоторой скоростью;

— недетерминированное во времени, а возможно, и в пространстве появление новых узлов сети и отключение имеющихся;

— неоднородность физической среды в пределах сети;

— изменяющиеся во времени условия взаимной радиовидимости между узлами такой сети;

— возможные топологии сети с двумя и более возможными маршрутами между узлами, а также с узлами, достижимыми исключительно через транзитные узлы;

— равноправность (в сетевом смысле) узлов в сетях;

— ограниченность вычислительных ресурсов узлов.

В то же время к рассматриваемым нами сетям предъявляются достаточно «стандартные» требования:

— эффективное использование частотного ресурса и имеющейся пропускной способности сети;

— потребность в значительной располагаемой пропускной способности на всем протяжении маршрута передачи информации в связи с необходимостью передачи видео, голосовой, телеметрической и иных видов информации.

Классические централизованные беспроводные сети (сети с главным устройством — точкой доступа или базовой станцией) непригодны для подобных ситуаций ввиду потери работоспособности при отключении центрального устройства. Однако такие сети широко применяются в системах телеприсутствия и автономных аппаратах в конфигурациях «одно устройство — один пульт управления», т. е. для сетей топологии «точка-точка».

Классические беспроводные одноранговые (они же Ad-Hoc) сети непригодны для подобных ситуаций ввиду отсутствия механизмов маршрутизации и, как следствие, невозможности передачи информации между узлами, не являющимися соседями.

Модификация одноранговых беспроводных сетей посредством внедрения в них функции маршрутизации выглядит наиболее перспективным вариантом решения проблемы. На данный момент существует немало решений для организации беспроводных сетей передачи данных с равноправными узлами (mesh-сетей). Большинство из них ориентировано на организацию и поддержку функционирования неподвижных mesh-сетей. Среди наиболее известных и доступных можно назвать batman_adv [1] и стандарт 802.11s [2]. Их недостатком является учет ограниченного числа факторов (число хопов, пропускная способность) выбора маршрута. В

них совершенно не учитываются факторы изменения топологии сети и надежности узлов.

Существуют реализации (см. [3] – [4]) средств управления подвижными mesh-сетями. Однако все известные решения либо являются закрытыми и в настоящее время ориентированными на военное применение, либо являются предметом академических исследований. По причине закрытости работ над средствами управления подвижными mesh-сетями практически невозможно судить о методах организации процесса маршрутизации, используемого в этих решениях.

Тем не менее, анализ решений для неподвижных mesh-сетей, а также математические соображения подсказывают, что в процессе маршрутизации для подвижных mesh-сетей оптимальное решение должно учитывать факторы времени (возраста маршрутной информации) и поведения узлов сети (предсказание состояния узла на основе статистических данных) наряду с такими классическими факторами, как число хопов и пропускная способность.

В данной статье мы рассмотрим принципы построения mesh-сетей указанного вида.

В разделе 1 предложены алгоритмы маршрутизации в mesh-сети.

Разделы 2–3 посвящены принципам организации и особенностям программной реализации симулятора mesh-сети.

Напомним также несколько определений, использующихся для рассматриваемых нами mesh-сетей.

Тайм-слот — временной интервал, в течение которого беспроводным компьютерным устройством будет на выбранной скорости гарантированно завершен процесс передачи одного кадра-маяка и, как минимум, одного кадра данных максимального размера с защитными интервалами перед передачей кадра и после нее. Тайм-слот (продолжительность этого временного интервала) фиксирован для любого узла сети. Любые два действующих узла в сети могут использовать один и тот же тайм-слот при условии, что они не принадлежат одному фрагменту сети. Для тайм-слотов в сети определены три состояния: свободный, занятый, спорный.

Сеть оперирует N тайм-слотами, где N может принимать значения от 2 до 64. Количество тайм-слотов неизменно для конкретной работающей сети и может быть изменено только перезапуском сети с иными настройками.

Раунд — время, включающее в себя все N тайм-слотов.

Доступ к среде с временным мультиплексированием — механизм неколлизионного доступа к среде, при котором каждому узлу сети для передачи назначаются определенные, как правило, непересекающиеся временные интервалы (тайм-слоты).

1. Эвристические алгоритмы как часть инструментальной системы для исследования сетей подвижных объектов

Существует большое количество протоколов маршрутизации в мобильных сетях (см., например, обзор [5]), тем или иным способом решающих проблему выбора оптимального маршрута. Разнообразие подходов объясняется разнородностью об-

ластей применения рассматриваемых сетей и, соответственно, различиями в целевых функциях, характеристиках среды, природе помех, скорости изменения сетевой конфигурации и т.д. Для сравнения успешности того или иного подхода в конкретной системе используются, как правило, численные и натурные эксперименты. В качестве примера можно привести исследование [6], посвященное сравнению наиболее известных алгоритмов маршрутизации в мобильных сетях (DSDV, TORA, DSR и AODV).

Особенностью рассматриваемого нами случая является достаточно высокая скорость изменения топологии сети и энергетических характеристик соединений между узлами, а также используемая схема разделения времени соединения на основе тайм-слотов. Необходимость «быстрого» установления соединения и достаточно небольшие временные интервалы устойчивости конфигурации сети требуют высокой адаптивности системы маршрутизации. Кроме того, отсутствие диспетчера в сети приводит к обязательности использования децентрализованных (мультиагентных) алгоритмов.

Для классических ad hoc сетей адаптивный децентрализованный подход к маршрутизации не является чем-то новым, однако известные нам работы рассматривают достаточно специфические предметные области. В частности, в [7] предлагается основанный на случайных блужданиях алгоритм для сенсорных сетей (передача данных в основном от сенсора к серверу); в [8] рассматривается локальный подход, использующий оценку остаточной пропускной способности соединения (для проводных сетей); в [9] приводится основанный на потенциалах алгоритм организации доступа внутренних узлов mesh–сети к внешним шлюзам.

Перечисленные эвристики обладают рядом важных достоинств, однако, на наш взгляд, ни одна из них не подходит для случая беспроводных сетей подвижных объектов. Основной причиной является то, что в наших сетях возникает целый ряд новых факторов, влияние которых мы хотим каким-то образом учитывать в алгоритме при выборе того или иного маршрута. Кроме того, заметим, что характеристики «выгодности» можно описать множеством способов. Именно в этой множественности проявляется принципиальная трудность формализации задачи маршрутизации в подвижных mesh-сетях. На практике возможен лишь подбор эмпирических формул и коэффициентов, в «нужной» степени учитывающих «нужные» характеристики реальной системы.

Для упрощения задачи в наших алгоритмах мы используем единый композитный (вычисляемый) параметр, отвечающий за «оптимальность» маршрута с точки зрения «mesh-факторов» (его характеристики будут сформулированы далее). Если называть этот параметр «ценой» («длиной») маршрута, то задача маршрутизации сводится к известной проблеме поиска потока с наименьшими затратами (наименьшей длины).

Основными критериями при выборе алгоритма являлись:

— простота реализации;

— эффективность (в среднем);

— возможность мультиагентной реализации (одноранговость сети, отсутствие узла-диспетчера);

— скорость реагирования на изменения конфигурации сети (как топологии сети, так и загруженности узлов/соединений).

Были рассмотрены следующие варианты:

1. Алгоритмы на основе метода проталкивания предпотока (ПП) — push-relabel.

2. Алгоритмы на основе метода последовательного насыщения кратчайших путей (КП) — successive shortest path.

Алгоритмы на основе проталкивания предпотока в настоящее время считаются одними из наиболее эффективных среди полиномиальных по времени. Описания таких алгоритмов можно найти во множестве классических книг (например, [10]). К их недостаткам можно отнести достаточно сложную реализацию и большое число предварительных служебных сообщений (согласований) в мультиагентном варианте алгоритма.

Алгоритмы последовательного насыщения кратчайших путей, напротив, имеют достаточно простую реализацию и не требуют подробного описания. Поток пускается по кратчайшему (самому дешевому) маршруту до его насыщения, далее по следующему кратчайшему маршруту до его насыщения, и т.д. до полного исчерпания потока.

При предварительном сравнительном анализе двух подходов выяснились следующие преимущества КП перед ПП:

1. Объем хранимой на узле информации для КП может быть существенно меньше, поскольку его можно реализовать с использованием только лишь таблицы маршрутизации (не храня на узле всю известную структуру оптимальных путей). Схемы на основе таблиц, возможно, менее универсальны, однако достаточно эффективны. Например, в [11] показано, что «табличный» алгоритм AODV эффективнее более «интеллектуального» DSR в зашумленных и загруженных сетях, а также в сетях с динамически изменяющейся топологией. Кроме того, как показано в [12], для табличных схем маршрутизации достаточно просто реализуются методы защиты от атак «византийского» типа (когда в качестве злоумышленника выступают внутреннние узлы mesh-сети).

2. При мультиагентной реализации КП позволяет начать передачу данных сразу после обнаружения адреса узла-получателя узлом-отправителем в хотя бы одной из таблиц маршрутизации узлов-соседей (и выбранный маршрут будет достаточно эффективен) — то есть скорость реагирования на смену конфигурации сети максимальна.

3. Алгоритм КП легко адаптировать для сети с большим количеством разнонаправленных потоков (с учетом изменения коэффициентов доступности узлов и неполного насыщения кратчайших путей). Локальные методы оценки оптимальности маршрутов уже достаточно хорошо себя зарекомендовали на практике [13].

Однако очень существенным недостатком КП по сравнению с ПП и другими полиномиальными алгоритмами является его неэффективность: в худшем случае требуется экспоненциальное число операций для нахождения оптимального потока между двумя узлами. Тем не менее, в ходе обзора литературы были обнаружены следующие факты:

1. В работе [14] исследованы проблемы эффективного перераспределения электроэнергии в сетях, состоящих из множества активных агентов (производителей и потребителей). Как и в нашем случае, элементы сети (узлы и дуги) обладают и пропускной способностью, и «ценой» (производства, потребления и передачи электроэнергии). Авторы при помощи математического моделирования сравнивают стра-

тегии маршрутизации потоков, основанные на ПП и КП, и на основе анализа статистики приходят к выводу, что в среднем ПП является более эффективным для «радиальных» сетей (древовидной топологии), а КП — для «mesh-сетей» (так в статье названы графы произвольной топологии).

2. В работе [15] проведен анализ КП методами теории гладкого анализа вычислительной сложности алгоритмов (Smoothed Analysis). Эта теория возникла относительно недавно [16] и позволяет не учитывать при оценке трудоемкости алгоритмов крайние (вырожденные) случаи индивидуальных алгоритмических проблем. Авторы работы [15] доказывают, что с точки зрения Smoothed Analysis алгоритм КП является полиномиальным (более того, он даже более эффективен, чем многие считающиеся эффективными классические полиномиальные алгоритмы).

На основании проведенного анализа в качестве основы ПО маршрутизации в подвижных mesh-сетях предлагается использовать алгоритм последовательного насыщения кратчайших путей.

Математической моделью системы является стандартная транспортная сеть. Задача маршрутизации формализуется как задача поиска потока в сети с минимальными затратами. В качестве алгоритма решения выбран алгоритм последовательного насыщения кратчайших путей, где в качестве аналога «длины» пути рассматривается коэффициент доступности маршрута на первой транзитной вершине.

Ключевой параметр системы маршрутизации — *коэффициент доступности* (КД) узла — функция, зависящая от ряда основных и дополнительных параметров («mesh-факторов»), характеризующих маршрут между двумя узлами сети. Каждой паре (дуга, узел) сопоставляется композитный параметр, характеризующий «доступность» узла по маршруту, начинающемуся данной дугой. Таким образом, у каждой дуги есть ровно столько таких характеристик (коэффициентов доступности узлов), сколько узлов достижимы от вершины-окончания данной дуги (т.е. если эта вершина выступает в качестве первой транзитной вершины всех соответствующих маршрутов). Неформально, если КД пары $((s,u),t)$ выше, чем КД пары $((s,v),t)$, то, исходя из комбинации дополнительных факторов, при передаче данных из s в t «лучше» использовать в качестве первой транзитной вершины u, а не v.

Рассмотрим структуру функции вычисления КД узла. Отдельные компоненты данной функции будем называть *штрафами (пенализацией)*. Объектами наложения штрафов могут являться:

— *надежность узла* — $F_1(f) = F_f$;

— *число промежуточных узлов* — $F_2(n) = F_n$;

— *энергетика соединения* с учетом истории ее изменения — $F_3(e) = F_e$;

— *располагаемая пропускная способность* с учетом ее изменения — $F_4(p) = F_p$;

— *возраст данных о маршруте* — $F_5(t) = F_t$. Заметим, что поскольку мы имеем дело с подвижной сетью, а процесс распространения данных растянут по времени, наиболее свежие данные должны иметь преимущество.

Таким образом, коэффициент доступности узла можно выразить абстрактной формулой

$$K = F(F_f, F_n, F_e, F_p, F_t).$$

Будем считать, что значение данной функции находится в некотором интервале $[K_{\min}, K_{\max}]$, где $K_{\max}$ — значение, соответствующее максимально возможной до-

ступности узла. Маршруты, достигшие значения $K_{\min}$, будем исключать из таблицы маршрутизации (либо игнорировать) и, соответственно, не будем анонсировать.

Лучшим («кратчайшим») маршрутом между двумя узлами будем считать маршрут с наибольшим коэффициентом доступности. Именно этот маршрут будет насыщаться первым по выбранной стратегии алгоритма последовательного насыщения кратчайших путей.

Ключевые структурные характеристики архитектуры системы маршрутизации:

— информация для определения топологии сети и построения маршрутов в реактивном режиме распространяется в кадрах-маяках;

— перерасчет таблицы маршрутизации происходит каждый раз по приходу кадра-маяка, кадра данных, а также квитанции о доставке кадра;

— локальная таблица маршрутизации узла имеет ограниченное время жизни, записи в таблице маршрутизации постоянно «устаревают»;

— на каждом узле-ретрансляторе выполняется проактивная часть маршрутизации.

Используется минимальный по расходуемым ресурсам вариант таблицы маршрутизации, в котором для каждого MAC-адреса в сети, известного конкретному узлу, поставлен в соответствие MAC-адрес соседа-ретранслятора, обеспечивающего (по мнению этого самого узла) маршрут до получателя, а также некоторое свойство (коэффициент доступности), позволяющее определять этот «лучший маршрут».

Исходя из ограничений на целочисленную математику и 32-разрядную архитектуру, в качестве коэффициента доступности используется 16-разрядная величина, а для штрафов — два 16-разрядных значения (числитель и знаменатель). Все операции укладываются в 32-разрядные требования.

Опишем процесс формирования сведений о сети и маршрутах в ней.

Для начала рассмотрим процесс формирования информации на отдельно взятом узле сети, не имеющем локально подключенных устройств и не имеющем связи с другими устройствами в сети. Такое устройство просто анонсирует свой MAC-адрес в сеть с максимальным коэффициентом доступности $K_{\max}$.

Теперь рассмотрим процесс формирования информации на отдельно взятом узле сети, имеющем локально подключенные устройства и не имеющем связи с другими устройствами в сети. Данный узел анонсирует свой MAC-адрес с коэффициентом $K_{\max}$, а также MAC-адреса всех локально подключенных устройств с тем же коэффициентом $K_{\max}$.

Рассмотрим поведение соседа, который получает анонсы от вышеуказанного узла.

Получая анонс, узел имеет сведения об энергетике соединения, надежности соединения, времени получения анонса, отсутствии промежуточных узлов, а также располагаемой пропускной способности. Т.е. сосед может рассчитать все штрафы, налагаемые на такое соединение, а как следствие, на маршруты, доступные через данного соседа. Соответственно, если мы немного изменим функцию расчета коэффициента достижимости для учета предыдущего значения, мы сможем пересчитывать коэффициент, получаемый от соседа $K' = F(F_f, F_n(1), F_e, F_p, K)$.

Поскольку мы заявили принцип преимущества «свежих данных», нам требуется функция перерасчета коэффициента от времени $K'' = F'(F_t(n), K)$, где n — число микросекунд, прошедших с момента последнего пересчета по времени.

Рассмотрим теперь *реактивную часть* процесса маршрутизации.

Каждый узел анонсирует собственную таблицу маршрутизации, в виде доступных через него MAC-адресов и коэффициентов достижимости, где свой MAC-адрес и MAC-адреса локально подключенных устройств имеют коэффициент $K_{\max}$, а другие узлы имеют текущий расчетный коэффициент, находящийся в интервале $K_{\min} < K < K_{\max}$.

При получении анонса узел производит перерасчет коэффициентов доступности для узлов, имеющих коэффициент меньше $K_{\max}$, из локальной таблицы маршрутизации по формуле $K'' = F'(F_t(n), K)$ (где n — число микросекунд с момента предыдущего пересчета).

Производится исключение маршрутов, достигших значения коэффициента $K_{\min}$.

Коэффициенты для маршрутов из анонса пересчитываются по формуле $K' = F(F_f, F_n(1), F_e, F_p, K)$, т.е. применяются штрафы.

Далее выполняется объединение двух таблиц (локальной и анонсируемой) по правилу:

— если анонсируемый MAC-адрес отсутствует в локальной таблице маршрутизации, он копируется из анонса вместе с обновленным коэффициентом доступности. Транзитным узлом для него назначается анонсирующий данный адрес сосед;

— если для анонсируемого MAC-адреса имеется маршрут, то маршрут из анонса добавляется в таблицу маршрутизации;

— обновляются временные метки в локальной таблице маршрутизации.

В результате работы предлагаемой схемы на каждом узле сети имеется таблица маршрутизации, в которой для каждого MAC-адреса в сети имеется минимум одна запись с MAC-адресом непосредственного соседа узла, начинающего маршрут к данному MAC-адресу.

В предлагаемой схеме используется пенализация (наложение штрафа) и поощрение (увеличение коэффициента доступности) соседей в зависимости от их «надежности». Под надежностью узла в данном контексте понимается пропуск кадров-маяков от узла-соседа, а также отсутствие квитанций от соседа о доставке кадров.

В предлагаемой схеме отсутствует обновление маршрутов через соседа при пропуске кадра-маяка и, как следствие, уменьшается коэффициент доступности зависимых маршрутов ввиду пересчета по времени. Уменьшение коэффициента доступности проводится также при неполучении квитанции от узла соседа. При этом данное уменьшение весьма существенно.

В качестве поощрения вводится увеличение коэффициента доступности маршрута при получении квитанции о доставке кадра от узла-соседа. Поощрение пропорционально размеру доставленного кадра.

Данная стратегия позволяет быстро реагировать на изменение условий соединения: прекратить использование маршрута в случаях выпадения кадров либо выявить и активно использовать соединение с высокой пропускной способностью.

Ниже рассмотрим отдельные виды пенализации.

1. Пенализация за актуальность информации.

Поскольку мы имеем дело с подвижной mesh-сетью, для определения лучшего маршрута нам нужна максимально свежая информация. В связи с этим логичной выглядит пенализация маршрутов по мере устаревания информации о них.

Предлагается ввести пенализацию маршрутов за каждую микросекунду, прошедшую с момента их получения. Степень пенализации должна зависеть от времени (прямо пропорциональна) и расчетной средней скорости взаимного перемещения объектов в целевой сети (прямо пропорциональна). Она является эмпирической величиной и должна быть установлена в ходе экспериментов (в том числе на симуляторе). Например, для малоподвижной сети (ССВПО до 20 км/ч) и при размере тайм-слота в 10 миллисекунд, максимальная пенализация за раунд не должна превышать 10% от $K_{\max}$.

2. Пенализация / вознаграждение за надежность узла.

Основывается на статистических данных по пропуску кадров-маяков от узла и доставке / недоставке кадров к узлу. Эмпирическая величина, устанавливаемая в ходе экспериментов. В связи с подвижностью сети, нет смысла собирать сведения о «надежности» узла в течение значительного времени (по меркам сети).

Содержит два компонента пенализации:

— *временной* (все маршруты через узел значительно штрафуются при непоступлении крайнего кадра-маяка);

— *отсутствие квитанции* (маршруты через узел значительно штрафуются при неполучении квитанции о доставке кадра)

и один компонент поощрения:

— *получение квитанции* (коэффициенты доступности для маршрутов через узел увеличиваются пропорционально размеру успешно переданного кадра).

3. Пенализация за энергетику соединения.

Данный вид пенализации является наиболее важной и сложной частью в системе штрафов.

Для вычисления штрафов предлагается оперировать четырьмя сущностями:

1) средний уровень сигнала;

2) тренд изменения среднего уровня сигнала;

3) текущий уровень сигнала;

4) скорость и направление изменения уровня сигнала.

Уровень сигнала — это величина, находящаяся в определенном диапазоне. Приближение к границам диапазона вредно, поскольку ведет к переусилению сигнала или потере связи. Соответственно, пенализация при приближении текущего уровня сигнала к границам диапазона должна быть максимальной. Пенализация при приближении среднего уровня к границам диапазона должна быть значительной. В обоих случаях при нахождении уровней в середине диапазона пенализация должна отсутствовать. Характер соответствующих функций нелинейный. Правила пенализации по энергетике:

1) устойчивый тренд на снижение среднего уровня сигнала должен приводить к увеличению штрафа; устойчивый тренд на рост — к уменьшению штрафа; отсутствие изменений — не влиять на штраф;

2) высокая скорость уменьшения сигнала при наличии сигнала в нижней части диапазона должна приводить к высокому штрафу;

3) высокая скорость уменьшения сигнала при наличии сигнала в верхней части диапазона должна приводить к среднему штрафу;

4) высокая скорость уменьшения сигнала при наличии сигнала в середине диа-

пазона должна приводить к среднему штрафу. При этом уровень штрафа должен быть больше предыдущего;

5) средняя скорость уменьшения сигнала при наличии сигнала в нижней части диапазона должна приводить к высокому штрафу. При этом штраф должен быть меньше аналогичного за высокую скорость;

6) средняя скорость уменьшения сигнала при наличии сигнала в верхней части диапазона не должна приводить к штрафу;

7) средняя скорость уменьшения сигнала при наличии сигнала в середине диапазона должна приводить к среднему штрафу. При этом уровень штрафа должен быть меньше соответствующего за высокую скорость;

8) низкая скорость уменьшения сигнала при наличии сигнала в нижней части диапазона должна приводить к среднему штрафу;

9) низкая скорость уменьшения сигнала при наличии сигнала в верхней части диапазона не должна приводить к штрафу;

10) низкая скорость уменьшения сигнала при наличии сигнала в середине диапазона должна приводить к малому штрафу;

11) высокая скорость увеличения сигнала при наличии сигнала в нижней части диапазона должна приводить к среднему штрафу;

12) высокая скорость увеличения сигнала при наличии сигнала в верхней части диапазона должна приводить к высокому штрафу;

13) высокая скорость увеличения сигнала при наличии сигнала в середине диапазона должна приводить к малому штрафу;

14) средняя скорость увеличения сигнала при наличии сигнала в нижней части диапазона должна приводить к малому штрафу;

15) средняя скорость увеличения сигнала при наличии сигнала в верхней части диапазона должна приводить к среднему штрафу;

16) средняя скорость увеличения сигнала при наличии сигнала в середине диапазона не должна приводить к штрафу;

17) низкая скорость увеличения сигнала при наличии сигнала в нижней части диапазона должна приводить к среднему штрафу;

18) низкая скорость увеличения сигнала при наличии сигнала в верхней части диапазона не должна приводить к штрафу;

19) низкая скорость увеличения сигнала при наличии сигнала в середине диапазона не должна приводить к штрафу.

Уровни штрафных коэффициентов являются эмпирическими величинами и устанавливаются в процессе экспериментов.

4. Пенализация за располагаемую пропускную способность.

Пенализация за располагаемую пропускную способность соединения на данном этапе выглядит достаточно простой — штраф обратно пропорционален располагаемой пропускной способности.

Для сети с временным разделением доступа к среде предложенная схема реактивного режима одновременно является практически полной реализацией *проактивного режима.*

На начало процесса передачи кадра данных на узле сформирована таблица маршрутизации, имеющая «лучший маршрут» до узла назначения.

Полная реализация проактивного режима заключается в реализации процедуры замены адреса ретранслятора в заголовке кадра на соответствующий адрес из локальной таблицы маршрутизации, а также установки актуальной скорости передачи непосредственно перед передачей кадра.

2. Характеристика и принципы организации симулятора

Для моделирования различных ситуаций, возникающих в mesh-сети указанного вида, был разработан симулятор, предназначенный для работы под операционными системами Ubuntu 14.04 LTS и Windows 7. Он состоит из двух компонентов:

— «Редактор»;

— «Интерпретатор».

Компонент «Редактор» позволяет определять и изменять конфигурацию сети, для которой необходимо выполнить моделирование. Конфигурация сети представляется с помощью набора параметров:

1) *количество узлов сети* (от 3 до 16);

2) *матрица смежности / энергетики* — квадратная матрица, в которой для каждой пары узлов задается величина энергетики соединения между ними (целое число в диапазоне от -127 до 12 dBm); в случае, когда соединение отсутствует, указывается значение 13. При этом для любой пары узлов $\{a, b\}$ допускается несовпадение значений энергетики на соединениях (a, b) и (b, a);

3) *вектор длин очередей* (неотрицательные целые значения) — показывает, сколько кадров находится в очереди на каждом узле;

4) *вектор времени* (целые значения в диапазоне от 0 до 8388 мс) — возраст информации о каждом узле.

Компонент «Интерпретатор» позволяет выполнять моделирование различных ситуаций в mesh-сети указанной конфигурации. Функциональные возможности «Интерпретатора»:

1) построение таблиц маршрутизации для отдельного узла либо для всех узлов в сети;

2) отслеживание продвижения одного кадра между двумя выбранными узлами сети.

Компонент «Интерпретатор» является консольной утилитой, поддерживающей 4 режима работы:

1. *Инициализация сети с консольным выводом.* Инициализация сети по конфигурации из файла; вывод осуществляется на консоль; результатом выполнения является набор таблиц маршрутизации для всех узлов сети.

2. *Инициализация сети с файловым выводом.* Инициализация сети по конфигурации из файла; вывод осуществляется в файл; результатом выполнения является набор таблиц маршрутизации для всех узлов сети.

3. *Инициализация узла с файловым выводом.* Инициализация сети по конфигурации из файла; вывод осуществляется в файл; результатом выполнения является таблица маршрутизации для выбранного узла сети.

4. *Отслеживание кадра.* Инициализация сети по конфигурации из файла и отслеживание продвижения информационного кадра от узла-отправителя к узлу-получателю; вывод осуществляется в файл; результатом выполнения является набор таблиц маршрутизации для всех узлов сети и информация по отслеживанию кадра.

Разработанный симулятор предназначен прежде всего для анализа и оценки используемой модели пенализации и алгоритма маршрутизации без проведения натурных испытаний. При этом все методы расчета таблиц маршрутизации и выбора конкретного маршрута эквивалентны тем, что применяются в реальных устройствах (описание методов см. в разделе 1.).

В текущей версии симулятора инициализация сети и построение таблиц маршрутизации осуществляется следующим образом.

1. Узлы поочередно «включаются», получая при этом индивидуальный номер тайм-слота.

2. Каждый узел анонсирует себя путем отправки кадра-маяка в момент включения, а также в начале своего тайм-слота в каждом раунде. Кадр-маяк содержит информацию об узле, в том числе таблицу маршрутизации, получаемую после применения всех штрафов на этом узле.

3. Все узлы-соседи, включенные в момент передачи кадра-маяка, получают этот кадр и обновляют свои таблицы маршрутизации на основе полученной информации (правила обновления таблицы см. в разделе 1.). Узел b считается соседом узла a, если в матрице смежности / энергетики указано наличие соединения (a, b).

После включения всех узлов сети шаги 2–3 повторяются циклически до завершения симуляции.

При обновлении таблиц маршрутизации к записям применяются следующие элементы пенализации:

1. Пенализация записей по времени на основе вектора времени. Чем больше значение некоторого компонента вектора времени, тем «старее» информация о соответствующем узле и больше штраф.

2. Пенализация по энергетике соединения. Поскольку в данной версии мы работаем только с одним снимком сети (конфигурация связей и энергетика не меняются), то используется простая пенализация по энергетике, не учитывающая характер изменения топологии сети и энергетики соединений во времени.

3. Пенализация за пропускную способность соединения. В данной версии симулятора предусмотрена возможность передачи только одного информационного кадра, поэтому пенализация за пропускную способность сводится к пенализации по длине очереди (чем больше очередь на узле, тем меньше пропускная способность соединения). Соответственно, размер накладываемого штрафа определяется вектором длин очередей — чем больше значение, тем больше пенализация.

В симуляторе не используется пенализация / поощрение за «надежность» узла, поскольку в данной реализации невозможно собрать статистику по передаче информационных кадров (отслеживается только один кадр).

При работе симулятора в режиме «Отслеживание кадра» используется следующий алгоритм продвижения информационного кадра.

Пусть необходимо передать кадр от узла a узлу b. С данным кадром ассоции-

руется параметр TTL (Time To Live, время жизни), в начальный момент времени равный 255.

Пусть в некоторый момент времени кадр находится на некотором узле x и $TTL \neq 0$. Во время тайм-слота, соответствующего узлу x, этот узел ищет в своей таблице маршрутизации все маршруты до узла b, выбирает среди них тот, который обладает максимальным коэффициентом доступности, и отправляет кадр тому узлу y, который указан в качестве ретранслятора для выбранного в таблице маршрута.

Получая кадр, узел y проверяет его. Если $y = b$, то кадр доставлен. Иначе узел y уменьшает значение TTL на 1. Если $TTL = 0$, кадр уничтожается.

3. Особенности программной реализации

Как уже отмечалось выше, нами был реализован симулятор, состоящий из двух компонентов: «Редактор» и «Интерпретатор».

Компонент «Редактор» является приложением с графическим интерфейсом. Основная задача этого приложения — создание и редактирование конфигураций имитируемой сети. Исходный код компонента «Редактор» представляет собой проект на языке C++ с использованием кроссплатформенного инструментария Qt и соответствует известному шаблону проектирования “Model–View–Controller” («Модель–Вид–Контроллер»).

Напомним, что согласно данному шаблону основная логика программы делится на модель, вид и контроллер. Модель в данном случае подразумевает под собой внутреннюю логику программного обеспечения, вид отвечает за отображение состояния модели пользователю, а контроллер обеспечивает связь между пользователем и системой.

Компонент «Интерпретатор» реализован в виде консольного приложения. Данный компонент позволяет выполнять симуляцию сети на основе файла конфигурации, созданного при помощи приложения «Редактор». Исходный код интерпретатора написан на языке C. Функционал программы логически разделен на уровни (таблица 1).

Таблица 1. Разделение логики в компоненте «Интерпретатор»

Файл	Функционал
nsutils.h	Подключение необходимых системных библиотек
list.h	Linux-реализация двусвязных списков
nstypes.h	Описание структур, используемых в программе
nsalloc.h	Реализация работы с памятью
nsio.h	Реализация операций ввода-вывода информации
nsrouting.h	Реализация работы узлов и обмена кадрами
nssimulator.h	Симуляция сети
tested.h	Тестируемый функционал

Остановимся подробнее на реализации работы узлов и симуляции работы сети в целом.

Интерпретатор в начале своей работы производит инициализацию сети, включающую в себя поочередное включение узлов сети с последующим обменом маршрутной информацией между включенными узлами.

Инициализация, как и симуляция работы сети, производится пошагово, один шаг соответствует одному тайм-слоту. По прохождению тайм-слотов всех узлов отсчитывается раунд. Сеть считается проинициализированной, если с момента включения всех узлов в сети прошло количество раундов, равное количеству задействованных в симуляции узлов. После инициализации в сети применяются временные задержки на основе вектора времени. На этом инициализация завершается.

Симуляция сети проходит тем же образом, что и инициализация. Во время симуляции работы сети производится генерация и обмен кадрами между узлами с сохранением маршрутов передачи сгенерированных кадров. Симуляция считается законченной, если в сети не происходит обмена ни одним сгенерированным кадром и в то же время отсутствуют записи о генерации кадров в будущем. Рассмотрим симуляцию сети на уровне обработки узлов. Следует заметить, что каждый узел в нашей реализации содержит следующие элементы:

— два потока (входной и выходной потоки), реализованные при помощи двусвязного списка;

— номер-идентификатор в сети;

— таблицу маршрутизации.

Во время текущего тайм-слота производится обработка узла, за которым закреплен данный тайм-слот. Обработка узла реализована в виде последовательности шагов:

1. *Обработка входящих кадров.*

На данном этапе происходит обработка кадров, так или иначе оказавшихся во входном потоке. В случае появления кадров, которые необходимо переслать другому узлу, производится перенос такого кадра в выходной поток. Если же во входном потоке встречается кадр-маяк, то его содержимое используется для обновления таблицы маршрутизации узла.

2. *Обновление таблицы маршрутизации.*

На данном этапе производится обновление таблицы маршрутизации узла, пенализация и сортировка записей в таблице.

3. *Генерация кадра-маяка.*

В процессе генерации создается кадр, содержащий информацию о наиболее выгодных маршрутах от данного узла. Данный кадр помещается в выходной поток. Поскольку этот кадр всегда рассматривается как первоочередной, он помещается в начало очереди.

4. *Обработка исходящих кадров.*

На данном этапе производится отправка кадров из выходного потока. В случае, если связность между узлом-отправителем и узлом-получателем в данный момент нарушена, кадр удаляется, иначе кадр попадает во входной поток узла-получателя.

Если узел в данный момент неактивен, то его обработка пропускается.

Список литературы / References

[1] "B.A.T.M.A.N. Advanced Documentation Overview", 2014, http://www.open-mesh.org/projects/batman-adv/wiki.

[2] Conner W. S. et al., "IEEE 802.11s Tutorial", 2006, http://www.ieee802.org/802_tutorials/06-November/802.11s_Tutorial_r5.pdf.

[3] "Mobile Mesh Networks for Military, Defense and Public Safety", 2015, http://meshdynamics.com/military-mesh-networks.html.

[4] Shen W. L. et al., "Autonomous Mobile Mesh Networks", *IEEE Transactions on Mobile Computing*, **13**:2 (2014), 364–376.

[5] Boukerche A. et al., "Routing protocols in ad hoc networks: A survey", *Computer Networks*, **55**:13 (2011), 3032–3080.

[6] Broch J. et al., "A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols", *Proc. of ACM/IEEE International Conference on Mobile Computing and Networking*, 1998, 85–97.

[7] Servetto S. D., Barrenechea G., "Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks", *Proc. of 1st ACM International Workshop on Wireless Sensor Networks and Applications*, 2002, 12–21.

[8] Nelakuditi S. et al., "Adaptive Proportional Routing: A Localized QoS Routing Approach", *IEEE/ACM Transactions on Networking*, **10**:6 (2002), 790–804.

[9] Baumann R. et al., "Routing in Large-Scale Wireless Mesh Networks Using Temperature Fields", *IEEE Network*, 2008, 25–31.

[10] Cormen T. H. et al., *Introduction to Algorithms*, MIT Press, 2009.

[11] Perkins Ch. E. et al., "Performance Comparison of Two On-demand Routing protocols for Ad Hoc Networks", *IEEE Personal Communications*, 2001, 16–28.

[12] Awerbuch B. et al., "On the Survivability of Routing Protocols in Ad Hoc Wireless Networks", *CERIAS Tech Report 2005–121*, 2005.

[13] Nelakuditi S., Zhang Zh.-L., "On Selection of Paths for Multipath Routing", *Proc. of IWQoS*, 2001, 170–186.

[14] Nguyen Ph. H. et al., "Distributed routing algorithms to manage power flow in agent-based active distribution network", *Innovative Smart Grid Technologies Conference Europe (ISGT Europe)*, IEEE PES, 2010, 1–7.

[15] Brunsch T. et al., "Smoothed Analysis of the Successive Shortest Path Algorithm", 2015, arXiv: 1501.05493v1.

[16] Spielman D. A., Teng Sh.-H., "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time", *Journal of the ACM*, **51**:3 (2004), 385–463.

DOI: 10.18255/1818-1015-2015-4-546-562

Instrumental Supporting System for Developing and Analysis of Software-Defined Networks of Mobile Objects

Sokolov V. A., Korsakov S. V., Smirnov A. V., Bashkin V. A., Nikitin E. S.

Received September 4, 2015

This article describes the organization principles for wireless mesh-networks (software-defined networks of mobile objects). The emphasis is on the questions of getting effective routing algorithms for such networks. The mathematical model of the system is the standard transportation network. The key parameter of the routing system is the node reachability coefficient — the function depending on several basic and additional parameters ("mesh-factors"), which characterize the route between two network nodes. Each pair (arc, node) is juxtaposed to a composite parameter which characterizes the "reachability" of the node by the route which begins with this arc. The best ("shortest") route between two nodes is the route with the maximum reachability coefficient. The rules of building and refreshing the routing tables by the network nodes are described. With the announcement from the neighbor the node gets the information about the connection energy and reliability, the announcement time of receipt, the absence of transitional nodes and also about the connection capability. On the basis of this information the node applies the penalization (decreasing the reachability coefficient) or the reward (increasing the reachability coefficient) to all routes through this neighbor node. The penalization / reward scheme has some separate aspects: 1. Penalization for the actuality of information. 2. Penalization / reward for the reliability of a node. 3. Penalization for the connection energy. 4. Penalization for the present connection capability. The simulator of the wireless mesh-network of mobile objects is written. It is based on the suggested heuristic algorithms. The description and characteristics of the simulator are stated in the article. The peculiarities of its program realization are also examined.

Keywords: mesh-network, network protocol, routing, penalization, heuristic algorithm, simulator

For citation: Sokolov V. A., Korsakov S. V., Smirnov A. V., Bashkin V. A., Nikitin E. S., "Instrumental Supporting System for Developing and Analysis of Software-Defined Networks of Mobile Objects", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 546–562.

On the authors:
Sokolov Valery Anatolyevich, orcid.org/0000-0003-1427-4937, Doctor, Professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: valery-sokolov@yandex.ru

Korsakov Stanislav Valentinovich, orcid.org/0000-0003-2349-7980,
CEO, NETSHe Lab Ltd, Belinskogo str., 28-75, Yaroslavl, 150047, Russia,
Assistant Professor, P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: sta@stasoft.net

Smirnov Alexander Valeryevich, orcid.org/0000-0002-0980-2507, PhD, Associate Professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: alexander_sm@mail.ru

Bashkin Vladimir Anatolyevich, orcid.org/0000-0002-2534-1026, Doctor, Associate Professor,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: v_bashkin@mail.ru

Nikitin Evgeny Sergeevich, orcid.org/0000-0002-2341-9950, student,
P.G. Demidov Yaroslavl State University,
Sovetskaya str., 14, Yaroslavl, 150000, Russia, e-mail: nik.zhenya@gmail.com

Acknowledgments:

This work was supported by the Russian Foundation for Basic Research under the Grant No 15-07-03038 A.

Моделирование и анализ информационных систем. Т. 22, № 4 (2015), с. 563–577
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 563–577

DOI: 10.18255/1818-1015-2015-4-563-577

UDC 519.681.2

A Method of Sample Models of Program Construction in Terms of Petri Nets

Kharitonov D.I., Golenkov E.A., Tarasov G.V., Leontyev D.V.

Received September 1, 2015

In the article a method of automated construction of Petri nets simulating the behaviour of imperative programs is considered from the formal point of view. Petri net samples with certain characteristics are necessary in programming new algorithms for program analysis; in particular, they can be used for developing or optimizing algorithms of Petri nets compositions and decompositions, building the reachability tree, checking invariants and so on. The generation process consists of two stages. At the first stage, construction templates for a resulting net and parameters for construction are described. With the help of these parameters it is possible to regulate the final size and the absolute or relative amount of certain structures in the resulting net. At the second stage, iterative process of automated net construction is used for Petri net generation of any size, limited only by an available computer memory. In the first section of the article the minimum necessary definitions are given and a new version of Petri nets composition operation by places is introduced. Commutative and associative properties of introduced binary operation allow to synchronize any number of Petri nets in arbitrary order. Then construction template is defined as a marked Petri net with input and output interfaces and rules for templates composition using this interfaces. A number of construction templates can be united in a collection, for which the evolution rules are defined. The completeness property of a collection guarantees that the collection evolution results in a Petri net that simulates the imperative program behavior. The article provides a version of the construction templates complete collection and an example of Petri net simulating sequential imperative program construction.

The article is published in the author's wording.

Keywords: program model, control flow model, Petri net object

For citation: Kharitonov D.I., Golenkov E.A., Tarasov G.V., Leontyev D.V., "A Method of Sample Models of Program Construction in Terms of Petri Nets", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 563–577.

On the authors:

Kharitonov Dmitry Ivanovich, orcid.org/0000-0003-3359-2383, PhD, senior researcher,
Institution of Russian Academy of Sciences Institute of Automation and Control Processes Far Eastern Branch of the RAS, 5 Radio str., Vladivostok, Russia, 690041, e-mail: demiurg@dvo.ru

Golenkov Evgeny Alexandrovich, orcid.org/0000-0002-8148-3504, PhD, senior researcher,
Institution of Russian Academy of Sciences Institute of Automation and Control Processes Far Eastern Branch of the RAS, 5 Radio str., Vladivostok, Russia, 690041, e-mail: golenkov@dvo.ru

Tarasov Georgiy Vitalievich, orcid.org/0000-0001-8855-7388, research officer,
Institution of Russian Academy of Sciences Institute of Automation and Control Processes Far Eastern Branch of the RAS, 5 Radio str., Vladivostok, Russia, 690041,
Far-Eastern Federal University, 8 Suhanova st., Vladivostok, Russia, 690950, e-mail: george@dvo.ru

Leontiev Denis Valerievich, orcid.org/0000-0002-5116-3008, postgraduate student,
Institution of Russian Academy of Sciences Institute of Automation and Control Processes Far Eastern Branch of the RAS, 5 Radio str., Vladivostok, Russia, 690041,
Far-Eastern Federal University, 8 Suhanova st., Vladivostok, Russia, 690950, e-mail: devozh@dvo.ru

Acknowledgments:

This work was supported by the Russian academy of sciences fundamental research program "Fundamental problems of mathematical modelling", project 0262-2014-0157.

Introduction

Automatics and machinery in the modern world more and more relies on software. In many areas of human activity software errors may cost human lives. For example, in 2000 an erroneous calculation of the radiation dose led to several deaths [4]. However, among the variety of existing programs only very few have been formally verified, proving their correctness. This situation caused by the necessity of the human intellect to describe the programs examined in terms suitable to analysis. Petri nets are one of the few formal languages allowing to automate the process of software systems behavior models construction. In some cases, Petri nets are extremely suitable for modeling due to the distributed nature of the systems described, such as the development of multimedia streams scenarios [11]. In other cases, Petri nets analysis tools meet the stated objectives, like in the development of the process managing web services [9]. Software engineering and Petri nets crossed several times in the past resulting in interesting ideas in both areas [6]. Certain steps have been done by the authors of this article towards imperative programs modeling [7,17,18]. Nevertheless, there is a serious concern that the advantages of Petri nets as a formalism for distributed systems description with a clear graphical presentation will be lost, when describing the programs of actual complexity. At first, nets with more than a thousand of elements can not be represented on the screen or on the printed page in a readable form. At second, more significantly, nets analysis algorithms, for example, reachability tree construction algorithm, are to be adapted for Petri nets with a large number of elements. The classic reachability tree construction algorithm [1] for the nets greater than of 10^5 places and transitions requires more than 10Gb of memory that can be considered is a threshold for personal computers. In the international competition "Model Checking Contest @ Petri Net" for the comparison of Petri nets analysis tools a set of predefined models is used, and in 2015 the largest, by the number of places and transitions, model had about 34 thousand elements [20]. This number of elements corresponds to the Petri nets modeling imperative programs of less than 10 thousand lines of code, while the larger software systems can have hundreds of thousands of lines. For the adaptation of the algorithms dealing with Petri nets and their quality investigation there is a demand for the nets with a predefined number of elements and with known properties. The authors concluded that automatic generation of such nets is an important issue.

Material in the article is presented in the next way. The first section provides the minimum of the necessary definitions and a simple Petri nets composition operation by places is introduced. The second section describes the notion of the construction template and defines the rules of Petri nets automatic generation. The third section provides a complete set of construction templates and an example of generation of Petri net simulating behaviour of imperative program. Finally, conclusions on the applicability the method proposed are drawn.

1. Simple Petri net composition by places

Let $A = \{a_1, a_2, ..., a_k\}$ is a set. Multiset on set A is a function $\mu : A \to \{0, 1, 2, \dots\}$, that assigns a non-negative integer to each element of the set A. Multiset is conveniently written as a formal sum $n_1a_1 + n_2a_2 + ... + n_ka_k$ or $\Sigma n_i a_i$, where $n_i = \mu(a_i)$ is the number

of occurrences of the $a_i \in A$ in the multiset. Normally, when recording the sum, its zero elements $n_i = 0$ are omitted. The arithmetical sum and difference of multisets μ_1 and μ_2 are defined, respectively, as

$$(\mu_1 + \mu_2)(a) = \mu_1(a) + \mu_2(a),$$

$$(\mu_1 - \mu_2)(a) = \begin{cases} \mu_1(a) - \mu_2(a), & \text{если } \mu_2(a) \le \mu_1(a); \\ 0, & \text{otherwise.} \end{cases}$$

Comparing multisets μ_1 and μ_2 it is right to write: $\mu_1 \le \mu_2$, if $\forall a \in A : \mu_2(a) \le \mu_1(a)$, and $\mu_1 \ge \mu_2$, if $\forall a \in A : \mu_2(a) \ge \mu_1(a)$. If $n_i = 0$ for all i, then this multiset will be denoted as $\mathbf{0}$. We will also write that $a \in \mu$, if $\exists n > 0 : (a, n) \in \mu$. The set of all finite multisets on the set A will be denoted as $\mathcal{M}(A)$.

Let's define a sequence s on the set A as a function $\mathbb{N}_0 \to A \cup \emptyset$, associating with a positive integer one element of the set A or the empty set element $\emptyset$, if the number is greater than the size of the sequence $|s|$. The sequence is written as $(a_i)_{i=0}^n$ or shorter (a_i). Sequence element a_i is written as the function value of integer argument $s(i)$. The set of all finite sequences in the set A is written as (A). Let's also define a linearly ordered subset B of the set A with a linear order relation $\prec_A$ as $(b_i \mid \forall i < j \le n \Rightarrow b_i \prec_A b_j)$. Linearly ordered subset is written as $[b_i]_{i=0}^n$ or shortly $[b_i]$. The set of all finite linearly ordered subsets of the set A is written as $[A]$.

Definition 1. *Petri net is a tuple* $\Sigma = \langle S, T, {}^\bullet(), ()^\bullet \rangle$, *where*

1. S — *a finite set of places;*
2. T — *a finite set of transitions such that* $S \cap T = \emptyset$*;*
3. ${}^\bullet() : T \to \mathcal{M}(S)$ — *input incidence function;*
4. $()^\bullet : T \to \mathcal{M}(S)$ — *output incidence function.*

Multisets ${}^\bullet t$ and $t^\bullet$ are called input and output multisets of transition $t \in T$ accordingly.

Definition 2. *Formal union of Petri nets.*
Let us given two Petri nets $\Sigma_1 = \langle S_1, T_1, {}^\bullet()_1, ()_1^\bullet \rangle$ *and* $\Sigma_2 = \langle S_2, T_2, {}^\bullet()_2, ()_2^\bullet \rangle$. *Formal union of the Petri nets* Σ_1 *and* Σ_2 *is the net* $\Sigma = \Sigma_1 \oplus \Sigma_2 = \langle S, T, {}^\bullet(), ()^\bullet \rangle$, *such that*

$$S = S_1 \cup S_2, \qquad T = T_1 \cup T_2.$$

$${}^\bullet(t) = \begin{cases} {}^\bullet(t)_1, & \textit{if } t \in T_1; \\ {}^\bullet(t)_2, & \textit{if } t \in T_2. \end{cases}$$

$$(t)^\bullet = \begin{cases} (t)_1^\bullet, & \textit{if } t \in T_1; \\ (t)_2^\bullet, & \textit{if } t \in T_2. \end{cases}$$

Petri nets defined in such a way quite rarely used for modeling real systems, because as the number of places and transitions increases so raises the complexity of model perception as a whole. To simplify modeling of complex systems the compositional approach to build whole model from the simpler models of its subsystems is widely

used in practice. The most widely used is a nets composition by transitions [3, 14], but there are variations of nets composition operations by places [10], and also by places and transitions [15]. We introduce the operation of Petri nets composition by places using the scheme proposed in articles [2, 5]. In our case, the goal is to minimize the algorithmic complexity of the operation implementation.

Definition 3. *Simple access point by places to Petri net.*
Let's call the tuple $\iota = \langle id_\iota, \varrho_\iota \rangle$ *simple access point by places to Petri net* $\Sigma = \langle S, T, {}^\bullet(), ()^\bullet \rangle$, *where* id_ι *— unique identifier of access point, a* $\varrho_\iota \in [S]$ *— linearly ordered subset of places, used by access point.*

The name "simple point of access" is used to distinguish this access point from the ones introduced in the articles [2, 5]. Further in the text instead of the full name "simple access point by places" abbreviation "simple access point" may be used or even just "access point".

Definition 4. *Merge of Petri net simple access points.*
Let us given Petri net $\Sigma_1 = \langle S_1, T_1, {}^\bullet()_1, ()^\bullet_1 \rangle$ *and two its simple access points* $\iota_1 = \langle id_1, \varrho_1 \rangle$, $\iota_2 = \langle id_2, \varrho_1 \rangle$, *such that* $|\varrho_1| = |\varrho_2|$ *u* $\varrho_1 \cap \varrho_2 = \emptyset$. *Then merge operation of simple access points* ι_1 *and* ι_2 *of Petri net* Σ_1 *forms new net* $\Sigma = \Sigma_1|^{\iota_1}_{\iota_2} = \langle S, T, {}^\bullet(), ()^\bullet \rangle$, *so that*

1. $S = S_{const} \cup S_{syn}$, *where*

   - $S_{const} = S_1 \setminus (\varrho_1 \cup \varrho_2)$,
   - $S_{syn} = \{\langle \varrho_1(i), \varrho_2(i) \rangle \mid 0 \le i \le |\varrho_1|\}$,

   and there is a mapping surjection between a source and a finite set of places $\Upsilon^{\iota_1}_{\iota_2} : S_1 \to S$, *such that* $\Upsilon^{\iota_1}_{\iota_2}(s) = \begin{cases} s, & \forall s' \in S_{const} \\ s' = \langle s, \varrho_2(i) \rangle, & \forall s = \varrho_1(i) \in \varrho_1 \\ s' = \langle \varrho_1(j), s \rangle, & \forall s = \varrho_2(j) \in \varrho_2. \end{cases}$

2. $T = T_1$,

3. $\forall t \in T, s \in S_{const} : {}^\bullet(t)(s) = {}^\bullet(t)_1(s)$ *u*
   $\forall t \in T, s = \langle s', s'' \rangle \in S_{syn} : {}^\bullet(t)(s) = {}^\bullet(t)_1(s') + {}^\bullet(t)_1(s'')$,

4. $\forall t \in T, s \in S_{const} : (t)^\bullet(s) = (t)^\bullet_1(s)$ *u*
   $\forall t \in T, s = \langle s', s'' \rangle \in S_{syn} : (t)^\bullet(s) = (t)^\bullet_1(s') + (t)^\bullet_1(s'')$,

5. $\forall s \in S_{const} : M_0(s) = M_{01}(s)$,
   $\forall s = \langle s', s'' \rangle \in S_{syn} : M_0(s) = M_{01}(s') + M_{01}(s'')$.

Less formally merge of Petri nets simple access points by places performs "joining" of places, used by the access points, on the principle "one access point place merge another access point place with the same sequence number". Transitions of the original net do not change, and the arcs are restored from the original net, connecting transitions with the mapping of the original incident places. Using a list representation of places, transitions and arcs sets, software implementation of the access points merge operation

can be performed along with copying elements from original to destination nets in no more than $O(N)$ CPU operations where N is the number of elements in the original net. The mapping between the source and a finite set of places allows also to convert other simple access points by places not involved in the merge operation.

Definition 5. *Let us given Petri net $\Sigma = \Sigma_1|_{\iota_2}^{\iota_1}$, resulting from the simple access poing merge of the net Σ_1. Then simple access point $\iota = \langle id_\iota, \varrho_\iota \rangle$ by places of net Σ is the convertion of access point $\iota' = \langle id'_\iota, \varrho'_\iota \rangle$ by places of net Σ_1 as the result of merging, if $id'_\iota = id_\iota$ and $\forall i \Rightarrow \varrho_\iota(i) = \Upsilon_{\iota_2}^{\iota_1}(\varrho'_\iota(i))$.*

In practice, two Petri nets merge operation is more frequently used, which is defined as follows.

Definition 6. *Binary Petri nets merge operation by simple access points.*
Let us given two Petri nets $\Sigma_1 = \langle S_1, T_1, {}^\bullet()_1, ()_1^\bullet \rangle$, $\Sigma_2 = \langle S_2, T_2, {}^\bullet()_2, ()_2^\bullet \rangle$ and two their simple access points by places $\iota_1 = \langle id_1, \varrho_1 \rangle$, $\iota_2 = \langle id_2, \varrho_2 \rangle$, such that $|\varrho_1| = |\varrho_2|$. Then Petri nets Σ_1 and Σ_2 merge operation by simple access points ι_1 and ι_2 forms new net $\Sigma = \Sigma_1 \underset{\iota_1\ \iota_2}{\oplus} \Sigma_2 \equiv (\Sigma_1 \oplus \Sigma_2)|_{\iota_2}^{\iota_1}$.

Software implementation of the binary Petri nets merge operation can be performed, similarly to unary, in no more than $O(N_1 + N_2)$ CPU operations, where N_1 and N_2 are numbers of elements of the original nets. Taking into account the above-described conversion of simple access points by places, let's assume that source net access points are applicable to the net resulting from merging. Then merge operations properties can be written that follow directly from the definitions:

1. Unary operation commutativity

   $$\Sigma|_{\iota_2}^{\iota_1} = \Sigma|_{\iota_1}^{\iota_2}$$

   indicates that the result of merging the access point does not depend on the access points order.

2. Binary operation commutativity

   $$\Sigma_1 \underset{\iota_1\ \iota_2}{\oplus} \Sigma_2 = \Sigma_2 \underset{\iota_2\ \iota_1}{\oplus} \Sigma_1$$

   allows not to worry about the order of nets in the operation.

3. Unary operation associativity

   $$\Sigma|_{\iota_2}^{\iota_1}|_{\iota_4}^{\iota_3} = \Sigma|_{\iota_4}^{\iota_3}|_{\iota_2}^{\iota_1}$$

   allows to perform a number of merge operations over one net in any order.

4. Binary operation associativity

   $$\Sigma_1 \underset{\iota_1\ \iota_2}{\oplus} \Sigma_2 \underset{\iota_3\ \iota_4}{\oplus} \Sigma_3 = (\Sigma_1 \underset{\iota_1\ \iota_2}{\oplus} \Sigma_2) \underset{\iota_3\ \iota_4}{\oplus} \Sigma_3 = \Sigma_1 \underset{\iota_1\ \iota_2}{\oplus} (\Sigma_2 \underset{\iota_3\ \iota_4}{\oplus} \Sigma_3)$$

   allows to merge several Petri nets in random order.

2. Construction templates in terms of Petri nets

With the use of simple access points by places to Petri nets and composition operations, introduced in the previous section, we formulate object-oriented approach to the automatic generation of Petri nets. This approach based on the concept of the construction template.

Definition 7. *A template of imperative construction in terms of Petri nets (for short PN-template) is the tuple* $X = \langle \Sigma, I, O, M_0 \rangle$, *where*

1. $\Sigma = \langle S, T, {}^\bullet(), ()^\bullet \rangle$ — *Petri net, called object structure;*
2. $I = \{\iota_1, \iota_2, \ldots, \iota_n\}$ — *the set of simple access points, called input interface;*
3. $O = \{\phi_1, \phi_2, \ldots, \phi_n\}$ — *the set of simple access points, called output interface;*
4. $M_0 \in \mathcal{M}(S)$ — *initial marking.*

An imperative construction template is a marked Petri net, having part of the places assigned for merging with "superior" nets as the input interface, and another part of the places - to merge with the "subordinate" nets as the output interface. Let's formalize construction templates merge operation, "superior" nets are built with the help of.

Definition 8. *Formal union of PN-templates.*
Let us given two imperative construction templates $X_1 = \langle \Sigma_1, I_1, O_1, M_{01} \rangle$ *and* $X_2 = \langle \Sigma_2, I_2, O_2, M_{02} \rangle$. *Formal union of* X_1 *and* X_2 *is the template* $X = X_1 \oplus X_2 = \langle \Sigma, I, O, M_0 \rangle$, *such that*

$$\Sigma = \Sigma_1 \oplus \Sigma_2, \qquad I = I_1 \cup I_2, \qquad O = O_1 \cup O_2, \qquad M_0 = M_{01} + M_{02}.$$

Operation of PN-templates formal union makes new template by simple union of the sets and markings of the initial templates. To change the structure of the template the merge operation of simple access points is used.

Definition 9. *Merge of PN-template simple access points.*
Let us given a PN-template $X_1 = \langle \Sigma_1, I_1, O_1, M_{01} \rangle$ *and two its simple points* $\iota \in I_1, \phi \in O_1$, *where* $\iota = \langle id_1, \varrho_1 \rangle, \phi = \langle id_2, \varrho_2 \rangle$ *and* $\Sigma_1 = \langle S_1, T_1, {}^\bullet()_1, ()_1^\bullet \rangle$. *If subsets of places of both simple access points have equal cardinality* $|\varrho_1| = |\varrho_2|$, *than merge of simple access points operation of imperative construction* X_1 *by simple access points* ι *u* ϕ *forms new template* $X = \langle \Sigma, I, O, M_0 \rangle$, *where* $I = I_1 \setminus \{\iota\},\ O = O_1 \setminus \{\phi\}$ *and* $\Sigma = \Sigma_1|_\phi^\iota$.

Merge of a PN-template simple access points operation (in unary form) is denoted as $X = X_1|_\phi^\iota$.

More common used, and usefull for us, binary form of templates merge operation is defined by consecutive application of the two above operations.

Definition 10. *PN-templates merge operation by simple access points.*
Let us given two construction templates $X_1 = \langle \Sigma_1, I_1, O_1, M_{01} \rangle$, $X_2 = \langle \Sigma_2, I_2, O_2, M_{02} \rangle$ *and two their access points* $\iota \in I_1, \phi \in O_2$, *where* $\iota = \langle id_1, \varrho_1 \rangle, \phi = \langle id_2, \varrho_2 \rangle$. *And subsets of places of both simple access points have equal cardinality* $|\varrho_1| = |\varrho_2|$. *Then templates* X_1 *and* X_2 *merge operation by simple access points* ι *and* ϕ *forms new template* $X = \langle \Sigma, I, O, M_0 \rangle$, *so that* $X = (X_1 \oplus X_2)|_\phi^\iota$.

Now it is possible to formulate the necessary requirements to imperative construction templates in order to build program simulations in terms of Petri nets.

Definition 11. *PN-templates collection.*
A set $\Pi = \{X_i = \langle \Sigma_i, I_i, O_i, M_i \rangle\}$ *is called PN-templates collection, if:*

1. *There is the start template* $X_o \in \Pi$*, such that* $M_o > \mathbf{0}, |I_o| = 0, |O_o| \geq 0$.
2. *There are building templates:* $|\{X_k \mid X_k \in \Pi, |I_k| = 1, |O_k| > 0\}| \geq 1$.
3. *All simple access point in the input interfaces has a pair in the output interfaces, and vice versa:*
   - $\forall X_i \in \Pi, \phi \in O_i \rightarrow \exists X_j \in \Pi, \iota \in I_j : |\phi| = |\iota|$,
   - $\forall X_i \in \Pi, \iota \in I_i \rightarrow \exists X_j \in \Pi, \phi \in O_j : |\phi| = |\iota|$.

Practically, a templates collection - is a system in which the result of the start template merging with any of the others gives a new start template.

Definition 12. *PN-templates collection evolution.*
PN-templates collection $\Pi_{n+1} = \{X_0^{n+1}, X_1, ..., X_k\}$ *is an evolution of PN-templates collection* $\Pi_n = \{X_0^n, X_1, ..., X_k\}$*, if* $\exists\ \iota, \phi$*, such that* $X_0^{n+1} = X_0^n \underset{\iota\ \phi}{\oplus} X_i$.

It should be noted that the software implementation of templates collection *evolution* can be made, using a list representation of Petri net elements sets, in no more than $O(N)$ CPU operations, where N - the number of elements in the final net. To do this, at each step of the *evolution*, instead of creating new start template, all changes should be done in current one. Then, in each of the binary merges from $O(N_1 + N_2)$ CPU operations only $O(N_2)$ operations, related to copying second net elements and "gluing" places, remain.

Finally, with regard to the program behaviour simulations building, it is possible to formulate the final requirements to PN-templates set.

Definition 13. *Complete PN-templates collection.*
PN-templates collection $\Pi = \{X_i = \langle \Sigma_i, I_i, O_i, M_i \rangle\}$ *is considered to be complete, if:*

1. *All templates have no more than one access point in input interface:* $\forall X_i \in \Pi \rightarrow |I_i| \leq 1$.
2. *There is sufficient number of terminator templates:*
   $\forall X_i \in \Pi, \phi \in O_i \rightarrow \exists X_j \in \Pi, \iota \in I_j : |O_j| = 0, |\phi| = |\iota|$. [1]

Definition 14. *The resulting template.*
Template $X = \langle \Sigma, I, O, M_0 \rangle$ *is called the resulting temlate, if* $|I| = 0, |O| = 0$.

[1] Theoretically, the existence condition of sufficient number of terminator templates can be refined to reduce the number of templates. So, if there is some set of terminator templates, then can be built a set consisting of all the possible merges of initial terminators with building templates, and the resulting set tested for sufficiency.

With a complete collection of templates as defined in 13, it is possible, using a merge by access points operation 10, to build the resulting template of any predefined size. It is easy to verify that, using start template, each merge operation will result in a new start template that does not have an input interface. Available building templates and access point pairs in interfaces allow to continue build procedure. And merging start template with terminator templates reduces the amount of access points in output interface at the start template until it becames the resulting template.

3. A generation example of Petri net simulating imperative program

Consider as an example the generation of Petri net simulating the behavior of a simple sequential imperative program. The complete collection $\Pi_x = \{X_1..X_{10}\}$ consisting of ten templates is used to build sample net. Let us give drawings of templates and describe each of them in order. The following designations are used in templates representations. Petri net describing the structure of the template is placed in a rectangle. Petri nets are drawn using usual graphical notation in the form of a bipartite directed graph, where places are represented by circles and transitions — by rectangles. Places and transitions are connected by arcs representing input and output incidence functions. At the boundaries of the rectangle, framing template structure, the symbolic images of simple access points by places are drawn in the form of a circles with a sign of the interface it is belonged inside. In this article, all access points of the input interface are placed on the top edge of the rectangle, and all of the output — on the bottom edge. Each place of the access point ordered subset of places is connected by a thin dotted line with the symbol of the access point. Formal descriptions of the template input and output interfaces and its marking are placed inside the rectangle.

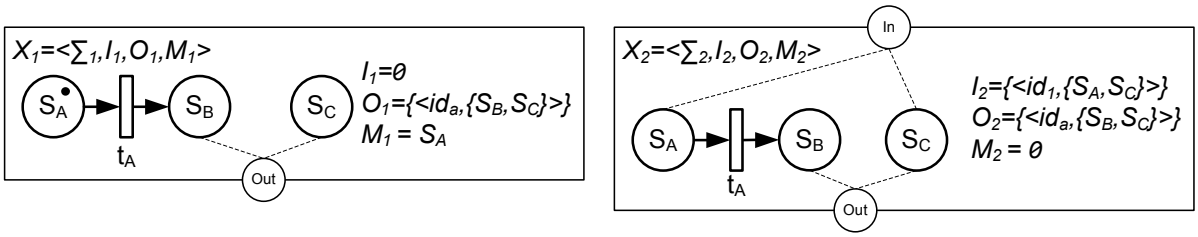


Fig 1: Templates of process (left) and linear section (right)

Figure 1 shows first two templates. The first template X_1, called the process template, is modelling begin and end of a sequential process. This is the only start template with a nonzero marking in the described collection, it has no input interface. The initial place with the token is the starting point of the program model, where the program begins its work, and the only transition in template simulates start of the process. Template X_2, called linear section, simulates simple mathematical expression in the imperative program, it differs from start template by absence of marking and presence of input interface.

Next template X_3 is drawn on figure 2 and designed to simulate the behaviour of cycles in the imperative program. First access point of the template input interface is

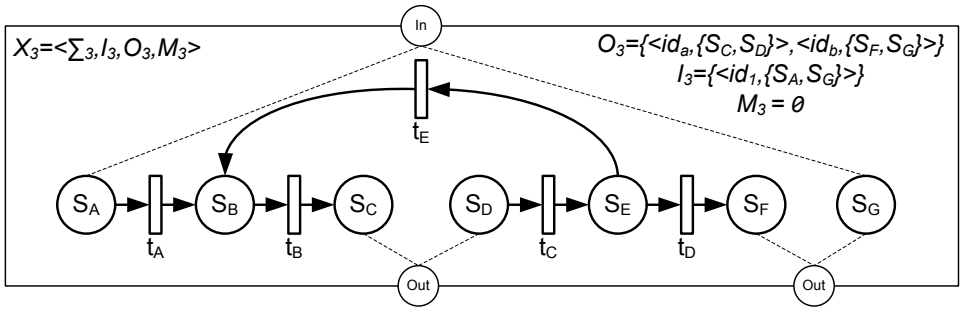


Fig 2: Template of cycle

used to form the cycle body. And second access point - to continue the program after the cycle.

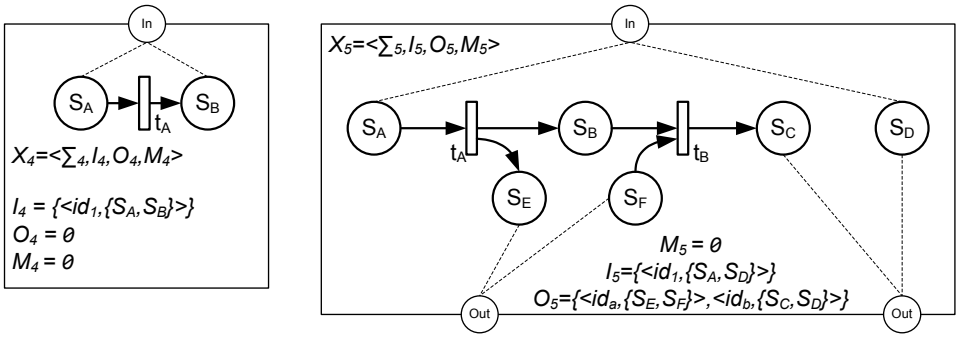


Fig 3: Templates of stub (left) and function call (right) constructions

At the left side of the figure 3 terminator template X_4, called stub, is represented. This template has only one access point of a pair of places in input interface and no access points in output interface, so after the merging with this template the resulting net would have one access point less in output interface. At the right side of the figure there is template X_5, modelling function call in the imperative program. This template has a single access point in input interface and two access points in the output interface. The first access point of the output interface is designed to form the body of the function, the second access point — to continue the program after the function call.

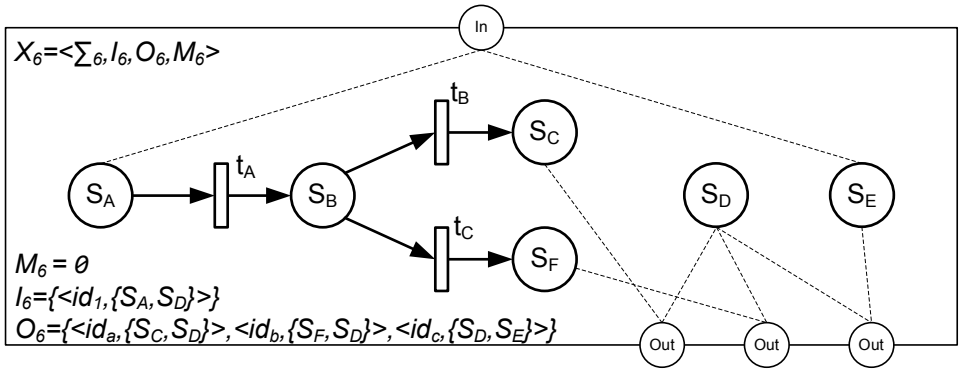


Fig 4: Template of branching operator construction

Figure 4 depicts template X_6 that models an imperative programming language **branching operator** construction. This template has one simple access point in the input interface, consisting of the begin and end places of the template. Three access points in the output interface, each consisting of a pair of places, are designed to simulate the program parts of *then* branch, *else* branch, and to continue the program after the branching operator.

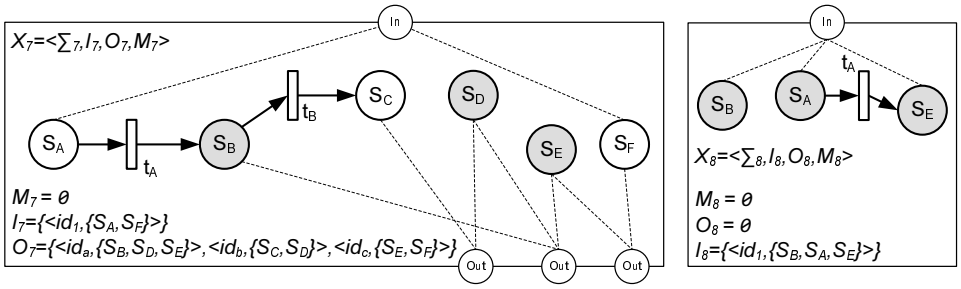


Fig 5: The main template of switch (right) and completing as *default* case (right)

Templates X_7, X_8, X_9, X_{10} simulate parts of syntax construction **switch** of imperative programming languages: main template — begin and end of the construction, completing template as *default* case, templates to continue after break operator and continue without break operator accordingly. This templates are shown of figures 5 and 6. Main template X_7 have one access point of a pair of places in input interface and three access points in output interface, designed to continue switch construction, building body of the first execution case of switch construction and to continue program after switch operator. The access point for the continuation of the switch construction has three places, and other two access points — two. Template X_8 has a single access point of the input interface of three places and no output, so it is the terminator for the switch construction, because after the merge operation of the switch construction with template X_8 addition of new cases will be impossible.

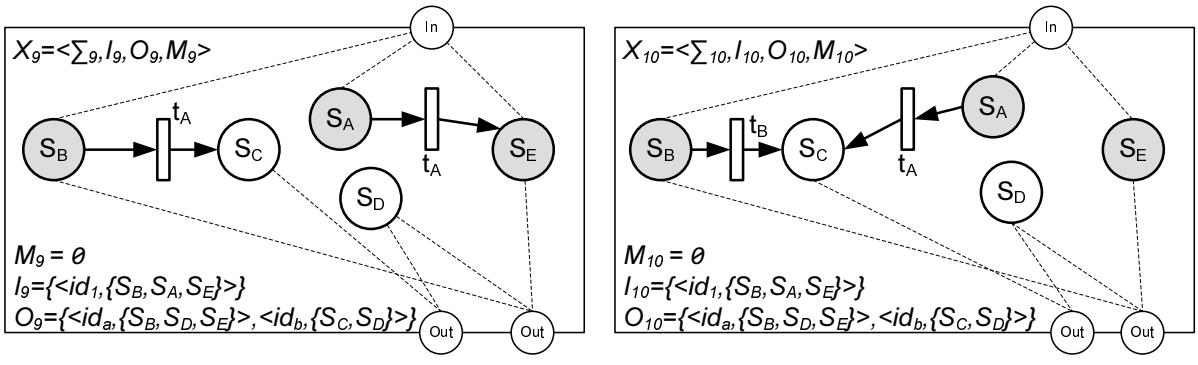


Fig 6: Templates of switch - continue after *break* operator (left), continue without *break* operator (right)

Figure 6 shows two options X_9, X_{10} of adding a new case to the switch construction. It is due to the single access point of three places in the input interface, this templates can be merged only with a templates from the set of switch construction template. The output interface of these templates has two access points: first access point of three

places is designed for the developing of the switch construction, second access point of two places – for constructing the case control flow. Thus, to simulate the behavior of the program in the switch construction it is necessary to use the main switch template, merge it step by step with the required number of cases templates and finish by merging with completing template.

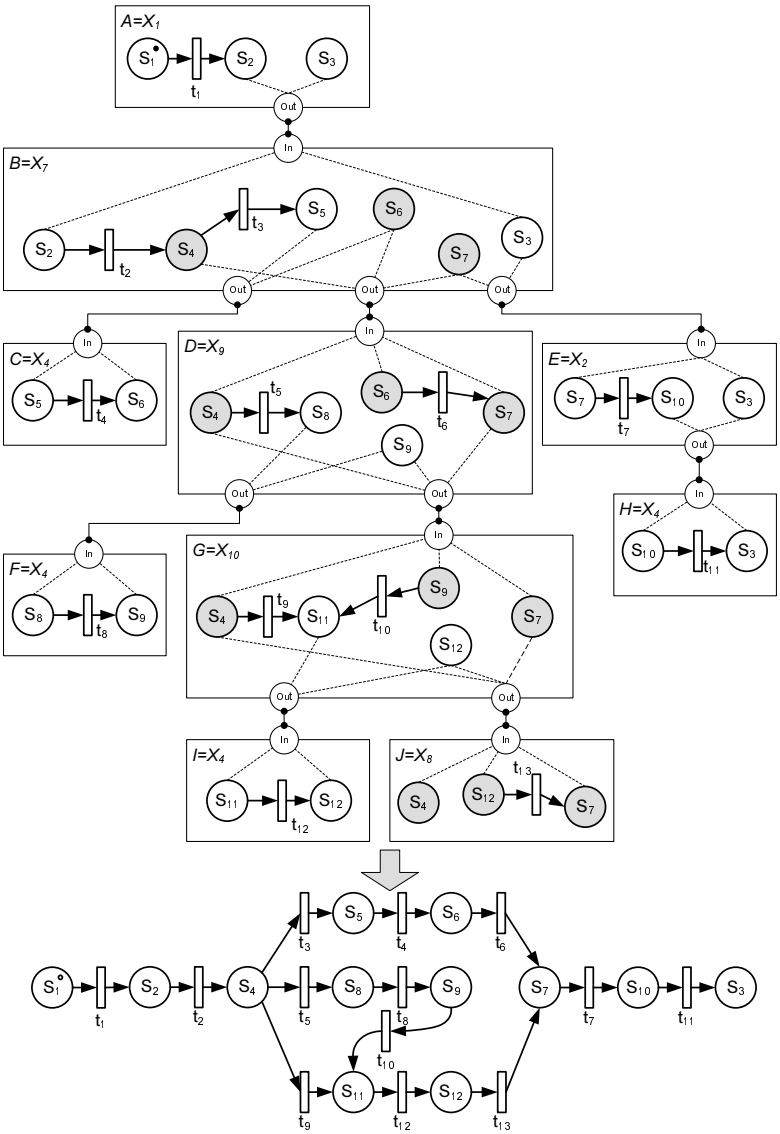


Fig 7: An example of Petri Net construction

Templates collection Π_x described above is a minimal collection for modelling of imperative programs. Let's consider the building process of Petri net simulating imperative program, using this templates collection. For the example of the net building the next templates were used: the process, the switch with three different branches and the stub. Figure 7 shows the diagram of the net construction, with the next used conventions:

- Each template is framed by a rectangle and signed by the ordinal character of the English alphabet and template number. English character indicates the order of

the templates merge operations that represents the evolution 12 of a templates collection.

- The lines between simple access points by places show what access points are used in merge operation.
- All access points of the input interface are placed on the upper edge of the rectangle, and of the output — at the bottom. Therefore, the order of templates merging coincides with the rules of reading - from top to bottom, left to right.
- For convenience all the places and transitions of construction templates are renamed to coincide with the resultant places and transitions.

Bottom part of the figure shows the result of templates evolution. This net is similar in behavior to the real program, consisting for the most part of a switch operator, which has three branches: the upper branch with break operator, the middle branch without break operator (without break operator the process continues in next branch) and the default branch.

4. Conclusion

Automatic Petri nets generation is quite often used in the modeling of objects of different application areas, for example, in railway interlocking design [16], large scale biological networks [12], semiconductors manufacturing [8], flexible manufacturing systems [13]. Typically, the description of the object in terms of domain-specific languages is used as the input data for translate procedure, that builds model of the object in terms of Petri nets. The authors proposed a new statement of the problem, when the input data, and the final result are described by Petri nets. Publications conforming that formulation have not yet been met by the authors.

This paper formally describes the method of Petri nets generation on the base of templates, having input and output interfaces in form of sets of simple access points by places, that allows templates merging. A distinctive feature of the method proposed is low computational complexity, since to implement the Petri nets merge operation it is necessary to perform a simple copy of nets elements with the subsequent gluing of beforehand known pairs of places. In the second half of this article a complete templates collection is given and a generation example of Petri net, simulating the behavior of an imperative program. A templates collection used to generate Petri nets, defines the behavior characteristics of the resulting network, so the method proposed has a certain flexibility, which allows to use it not only to build examples of imperative program models, but also to generate other different by behavior Petri nets. In particular, the authors see one of the interesting direction of the proposed method development in its adaptation to generate nets used for verification of analysis tools in “Model Checking Contest @ Petri Net”.

References

[1] Peterson, James Lyle, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981, 290 pp.

[2] Anisimov N.A., Kovalenko A.A., "Towards Petri Net Calculi based on Synchronization via Places", *Proc. of the 1995 IEEE Symposium on Parallel Algorithms/Architecture Synthesis*, IEEE Press, Japan, 1995, 264–270.

[3] Best, Eike and Devillers, Raymond and Koutny, Maciej, *Petri Net Algebra*, Springer-Verlag New York, Inc., USA, 2001.

[4] International Atomic Energy Agency, A Panel of Experts (2001), *Investigation of an Accidental Exposure of Radiotherapy Patients in Panama/Report of a Team of Experts, 26 May – 1 June, 2001 (PDF)*, Austria: International Atomic Energy Agency, Vienna, 2001.

[5] Anisimov N.A., Golenkov E.A., Kharitonov D.I., "Kompozitsionnal'nyy podkhod k razrabotke parallel'nykh i raspredelennykh sistem na osnove setey Petri", *Programmirovanie*, 2001, № 6, 30–43, (in Russian).

[6] Denaro G. and M. Pezzè, "Petri nets and software engineering", *Lectures on Concurrency and Petri Nets*, **3098**, Springer-Verlag, 2004, 439–466.

[7] Golenkov E.A., Sokolov A.S., "Metod avtomaticheskogo postroeniya modeli parallel'noy programmy v terminakh setey Petri", *Vychislitel'nye metody i programmirovanie*, **6**:2 (2005), 77–82, (in Russian).

[8] Mueller Ralph et al., "Automatic Generation of Simulation Models for Semiconductor Manufacturing", *Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best is Yet to Come*, WSC '07, IEEE Press, Piscataway, NJ, USA, 2007, 648–657.

[9] Desel J., "Controlling Petri Net Process Models", Web Services and Formal Methods, 4th International Workshop, WS-FM (2007, Brisbane, Australia, September 28-29, ed. Marlon Dumas and Reiko Heckel), 2008, 17–30.

[10] Laure Petrucci, "Aggregating views for Petri net model construction", *In Proc. workshop on Petri Nets and Distributed Systems (PNDS08, associated with Petri Nets 2008)*, 2008, 17–31.

[11] Abdelghani Ghomari, Chaabane Djeraba, "Modelling Multimedia Synchronization using a Time Petri Net Based Approach", *Advances in Petri Net: Theory and Applications*, eds. Tauseef Aized, Intech, 2010, http://www.intechopen.com/books/advances-in-petri-net-theory-and-applications/modeling-multimedia-synchronization-using-a-time-petri-net-based-approach-.

[12] Chen Ming et al., "Petri net models for the semi-automatic construction of large scale biological networks", *Natural Computing*, **10**:3 (2011), 1077–1097.

[13] Ballarini P. et al., "Petri Nets Compositional Modeling and Verification of Flexible Manufacturing Systems", In 7th Annual IEEE Conference on Automation Science and Engineering (CASE 2011), 2011.

[14] Alekseyev Arseniy et al., "Improved Parallel Composition of Labelled Petri Nets", *ACSD*, eds. Caillaud, Benoît and Carmona, Josep and Hiraishi, Kunihiko, IEEE Computer Society, 2011, 131–140.

[15] Ivan Petko and Stefan Hudák, "General composition for high level Petri nets and its properties", *Central Europ. J. Computer Science*, **2**:3 (2012), 222–235.

[16] Durmus M. S., Yildirim U., Soylemez M. T., "Automatic Generation of Petri Net Supervisors for Railway Interlocking Design", *Control Conference (AUCC), 2012 2nd Australian*, IEEE, 2012, 180–185.

[17] Tarasov G.V., Kharitonov D.I, Golenkov E.A.., "Ob odnom predstavlenii funktsii v modeli imperativnoy programmy, zadannoy setyami Petri", *Modelirovanie i analiz informatsionnykh sistem*, **18**:2 (2011), 18–38, (in Russian).

[18] Kharitonov D., Tarasov G., "Modeling function calls in program control flow in terms of Petri Nets", *ACSIJ Advances in Computer Science: an International Journal*, **3**:6 12 (November 2014), 82–91.

[19] B. Esther Sunanda, P. Seetharamaiah, "Modeling of Safety-Critical Systems Using Petri Nets", *SIGSOFT Softw. Eng. Notes*, **40**:1 (2015), 1–7.

[20] Kordon F. et al., "Complete Results for the 2015 Edition of the Model Checking Contest", 2015, http://mcc.lip6.fr/2015/results.php.

DOI: 10.18255/1818-1015-2015-4-563-577

Метод генерации примеров моделей программ в терминах сетей Петри

Харитонов Д.И., Голенков Е.А., Тарасов Г.В., Леонтьев Д.В.

получена 1 сентября 2015

В данной работе рассматривается с формальной точки зрения метод построения сетей Петри, имитирующих поведение императивных программ. Примеры сетей Петри с заданными характеристиками являются необходимыми в процессе программирования новых алгоритмов анализа моделей программ, в частности, они могут использоваться для разработки и оптимизации алгоритмов композиции и декомпозиции сетей Петри, построения дерева достижимости, проверки инвариантов и т.д. Способ построения состоит из двух стадий. На первой стадии описываются шаблонные конструкции, из которых будет состоять результирующая сеть, и параметры, с которыми будет выполняться построение. С помощью этих параметров можно регулировать конечный размер, а также абсолютное или относительное количество определённых конструкций в результирующей сети. На второй стадии с помощью автоматического итерационного процесса может быть сгенерирована сеть Петри любого размера, ограниченного оперативной памятью компьютера. В первом разделе статьи приводится необходимый минимум определений и вводится новый вариант операции композиции сетей Петри по местам. Свойства коммутативности и ассоциативности бинарного вида предложенной операции позволяют сливать несколько сетей Петри в произвольном порядке. Далее вводится понятие шаблонной конструкции в виде маркированной сети Петри, обладающей входным и выходным интерфейсами, а также правилами композиции шаблонных конструкций с использованием этих интерфейсов. Множество шаблонных конструкций объединяются в набор, для которого определяются правила эволюции. Свойство полноты набора гарантирует, что в результате эволюции набора будет получена сеть Петри, имитирующая поведение императивной программы. В статье приводится вариант полного набора шаблонных конструкций и пример генерации сети Петри, имитирующей последовательную императивную программу.

Статья публикуется в авторской редакции.

Ключевые слова: модель программы, модель потока управления, теория сетей Петри, объект сети Петри

Для цитирования: Харитонов Д.И., Голенков Е.А., Тарасов Г.В., Леонтьев Д.В., "Метод генерации примеров моделей программ в терминах сетей Петри", *Моделирование и анализ информационных систем*, **22**:4 (2015), 563–577.

Об авторах:
Харитонов Дмитрий Иванович, orcid.org/0000-0003-3359-2383, канд. техн. наук, ст. науч. сотр.,
Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения РАН, ул. Радио, д. 5, г. Владивосток, Россия, 690041, e-mail: demiurg@dvo.ru

Голенков Евгений Александрович, orcid.org/0000-0002-8148-3504, канд. физ.-мат. наук, ст. науч. сотр.,
Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения РАН, ул. Радио, д. 5, г. Владивосток, Россия, 690041, e-mail: golenkov@dvo.ru

Тарасов Георгий Витальевич, orcid.org/0000-0001-8855-7388, науч. сотр.
Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения РАН, ул. Радио, д. 5, г. Владивосток, Россия, 690041,
Дальневосточный Федеральный Университет, ул. Суханова, д. 8, г. Владивосток, Россия, 690950, e-mail: george@dvo.ru

Леонтьев Денис Валерьевич, orcid.org/0000-0002-5116-3008, аспирант,
Федеральное государственное бюджетное учреждение науки Институт автоматики и процессов управления Дальневосточного отделения РАН, ул. Радио, д. 5, г. Владивосток, Россия, 690041
Дальневосточный Федеральный Университет, ул. Суханова, д. 8, г. Владивосток, Россия, 690950, e-mail: devozh@dvo.ru

Благодарности:
Работа выполнена при финансовой поддержке программы фундаментальных исследований РАН по приоритетным направлениям "Фундаментальные проблемы математического моделирования", проект № 0262-2014-0157.

Моделирование и анализ информационных систем. Т. 22, №4 (2015), с. 578–589
Modeling and Analysis of Information Systems. Vol. 22, No 4 (2015), pp. 578–589

DOI: 10.18255/1818-1015-2015-4-578-589

UDC 004.052.42

Automation of Formal Verification of Programs in the Pifagor Language

Ushakova M. S., Legalov A. I.

Received May 18, 2015

Nowadays, due to software sophistication, programs correctness is more often proved by means of formal verification. The method of deduction based on Hoare logic could be used for any programming language and it has the capability of partial automation of the proof process. However, the method of deduction is not widely used for verification of parallel programs because of high complexity of the process. The usage of the functional data-flow paradigm of parallel programming allows to decrease the complexity of the proof process. In this article a proof process of correctness of functional data-flow parallel programs in the Pifagor language is considered. The proof process of a program correctness is considered as a tree where each node is a program data-flow graph, whose edges are marked with formulas in a specification language. The tree root is the initial program data-flow graph with a precondition and a postcondition, which describe restrictions on input variables and correctness conditions of the result of the program execution, respectively. Basic transformations of the data-flow graph are edge marking, equivalent transformation, splitting, folding of the program. By means of these transformations the data-flow graph is transformed and finally is reduced to a set of formulas in the specification language. If all these formulas are identically true, the program is correct. Several modules is distinguished in the system: “Program correctness prover”, “Axioms and theorems library management system” and “Errors analysis and output of information about errors”. According to this architecture, the toolkit for supporting formal verification was developed. The main functionality of the system implementation is considered.

Keywords: functional data-flow parallel programming, Pifagor programming language, programs formal verification, toolkit for supporting formal verification

For citation: Ushakova M. S., Legalov A. I., "Automation of Formal Verification of Programs in the Pifagor Language", *Modeling and Analysis of Information Systems*, **22**:4 (2015), 578–589.

On the authors:
Ushakova Maria Sergeevna, orcid.org/0000-0003-4234-2714, graduate student,
Siberian Federal University, Institute of Space and Information Technology,
26 Kirenskogo street, Krasnoyarsk, 660074, Russia, e-mail: ksv@akadem.ru

Legalov Alexander Ivanovich, orcid.org/0000-0002-5487-0699, PhD,
Siberian Federal University, Institute of Space and Information Technology,
26 Kirenskogo street, Krasnoyarsk, 660074, Russia, e-mail: legalov@mail.ru

Introduction

Nowadays there is a tendency to software sophistication that leads to debugging complexity and increase of the system failure risks. As the result, methods of program verification, particularly formal verification, started to develop rapidly. *Formal verification* is a proof of programs correctness by finding a correspondence between the program and its specification, which describes the aim of the development [1]. The correspondence between the program and its specification is established by the rigorous mathematical proof. The main advantage of formal verification is the ability to prove the absence of errors in the program formally.

The method of deduction based on Hoare logic [2, 3] is the most universal method of formal verification. Hoare logic is an extension of a formal system $\mathfrak{I}$ with certain formulas called Hoare triples. A *Hoare triple* is a program, namely the source code, and two formulas of the theory $\mathfrak{I}$, which describe restrictions on input variables and correctness conditions of the result of the program execution. These formulas are called *precondition* and *postcondition*, respectively. The extended formal system is distinguished from $\mathfrak{I}$ by additional axioms and inference rules, which allow to deduce certain program properties, particularly the program correctness. The program is correct if its Hoare triple is identically true. So the main idea of this approach is to derive a formula of formal system $\mathfrak{I}$ from the Hoare triple and then prove the truth of this formula within the formal system $\mathfrak{I}$.

The described method could be used for any programming language. Its main advantage is the capability of partial automation of the proof process. It should be pointed out that axioms and inference rules are individual for each programming language, since they are generated on the base of the language semantics. So each programming language has its individual corresponding formal system, therefore the difficulty of the proof process is also different.

Nowadays there are some achievements in practical use of the deduction method for verification of sequential imperative programs. There exist several systems that aid the proof process, for example, Boogie [4] and Spectrum [5] for programs in the C language, and LOOP [6] and KeY [7] for programs written in an object-oriented language. Boogie generates verification conditions for the theorem prover Simplify [8], and LOOP uses the interactive theorem prover PVS [9], for which it generates proof obligations that look like Hoare triples.

The development of formal verification methods are topical especially for parallel programming. Formal verification complexity for parallel imperative programs increases rapidly in comparison with sequential ones. The main problem is system resource conflicts, for example the misuse of shared memory or deadlocks of processes in distributed memory.

An alternative to imperative programming is the functional data-flow paradigm and its implementation the Pifagor (Parallel Informational and Functional AlGORthmic) language [10], [11]. This language allows to exclude resource conflicts. Each program is a function, so in Pifagor there are neither variables nor loops, and function execution starts only on data readiness. Another distinguishing feature of this language is the ability to achieve the maximum parallelism of the program, as parallelism is implemented at the level of operations. So after formal verification the correct program could be transferred to a system with specific architecture and limited resources, with the reduction of

parallelism if needed.

Hoare logic for the Pifagor language, which allows to prove programs correctness, has been already considered [12], [13]. First-order logic is used as the specification language to specify preconditions and postconditions. However the proof process is rather tedious, as proving the truth of a single triple is usually reduced to proving the truth of several transformed triples. The necessity to take into account a great number of triples makes the proof process quite complicated. So the aim of this work is to develop a toolkit for supporting the formal verification of functional data-flow parallel programs.

1. Transformations of a Labeled Data-Flow Graph

A program written in Pifagor is more demonstrable when represented as a *data-flow graph*. It is an acyclic directed graph that represents the data-flow of the program. The nodes of this graph represent program operators and the edges represent channels of data-flow between incident nodes [11]. Let us call a data-flow graph, whose edges are marked with formulas in the specification language, as a *labeled data-flow graph* (LDFG). If in a data-flow graph only the input and output edges are marked, then this graph corresponds to the Hoare triple. In this triple the program is represented by the graph, the precondition is the formula that marks the input edge and the postcondition is the formula that marks the output edge. Let us define the following transformations of a labeled data-flow graph:

1) edge marking;
2) modification of a data-flow graph:
   (a) equivalent transformation,
   (b) splitting;
3) folding of the program.

The process of proving the correctness of functional data-flow programs could be considered as a sequence of transformations of the initial labeled data-flow graph. The "initial labeled data-flow graph" is the LDFG that corresponds to the initial Hoare triple, namely the graph in which the input and output edges are marked with the user-defined precondition and postcondition respectively. Sequential transformations of the initial LDFG result in the set of *fully marked LDFG*. Each edge of a fully marked LDFG is marked with a singe formula. By the folding transformation these graphs are transformed into Hoare triples, which could be directly transformed into formulas in the specification language. The latter formulas are checked for truth. If all derived formulas are identically true, then the initial Hoare triple is also identically true, and the program is correct.

1.1. Edge Marking

The principle of marking graph edges with formulas is described in [12], [13]. A formula, attached to the edge of the graph, describes properties of data transferred through this edge.

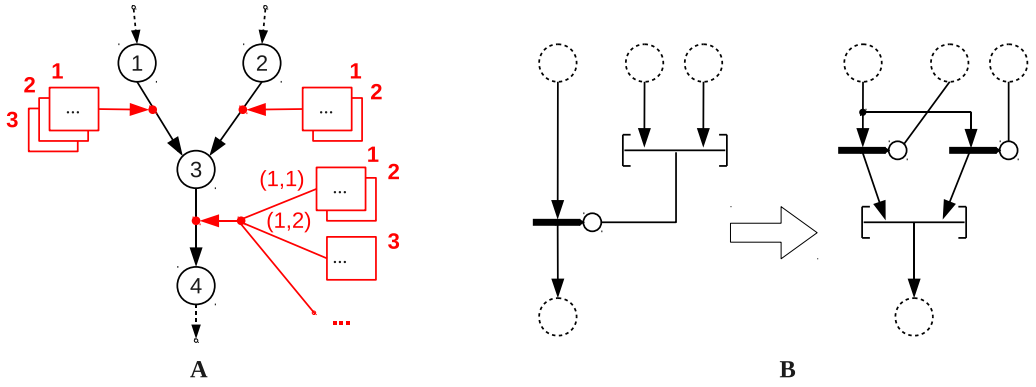


Fig. 1. The scheme of the labeled data-flow graph transformations. A — a part of a program data-flow graph marked with formulas. The graph nodes (denoted by circles) are program operators, the edges are data-flow links and are marked with formulas (denoted by rectangles); numbers near the formulas are formula indices, and parent formula indices are in parentheses. B — the equivalent transformation called the "parallel list release". The parallel list is denoted by a horizontal line in square brackets, the operator of interpretation is denoted by a bold horizontal line with a circle (the circle indicates a function input).

Edges of a data-flow graph are marked on basis of axioms (for built-in functions of the Pifagor language) and theorems (for user-defined functions with proved correctness). Several axioms or theorems could describe one function, so after edge marking several new graphs are derived.

Let us introduce the relation of hierarchy on the set of formulas. A formula attached to an edge (a, b) is called parent to a formula attached to an edge (b, c) for graph nodes a, b, c.

Edges marking does not change the data-flow graph of the program, so if it gives several new LDFGs, then they differ between each other only in one formula of the marked edge. That is why for the sake of compactness these LDFGs could be united into a single LDFG with the edge marked with several formulas simultaneously. Then the relation of hierarchy between formulas allows us to split the compact representation into the initial LDFGs.

Let us illustrate the compact representation of LDFG by means of the following example. Consider a graph of some program, depicted in the figure 1.A. The four operators of this program are connected with data-flow links. The node 3 receives input data from the nodes 1 and 2, the node 4 receives data from the node 3. Initially, the edges of the graph are not marked.

At first let us mark the edge $(1, 3)$. If some three axioms for the operator 1 are suitable for transforming the graph, then the result of the transformation gives us three new LDFGs, whose compact representation is equal to the initial graph with edge $(1, 3)$ marked with three formulas simultaneously. Then let us mark the edge $(2, 3)$. The number of suitable axioms for each of three graphs, obtained on the previous step, is the same, since the operators 1 and 2 are independent. If there exist two axioms for the operator 2

suitable for the transformation of these three DFMGs, then the result of transformation gives $6 = 3 \cdot 2$ new graphs. Their compact representation is obtained from the compact representation from the first step by marking the edge $(2, 3)$ with two new formulas. The operator 3 depend on the operators 1 and 2, so different number of axioms could appear applicable to each of the six graphs. Let this number be equal to two for the first LDFG (so its transformations gives two new LDFGs) and one for the rest five LDFGs. Thus, we obtain $7 = 1 \cdot 2 + 5 \cdot 1$ graphs after the transformation. This number equals the total number of suitable axioms as well as the number of formulas attached to the edge $(3, 4)$ of the compact representation. In the indices (i, j) of the formulas, marking the edge $(3, 4)$ in the figure 1, i is the index of the formula attached to the edge $(1, 3)$, parent to $(3, 4)$, and j is the index of the formula attached to the edge $(2, 3)$, also parent to $(3, 4)$.

1.2. Modification of a Data-Flow Graph

The second type of LDFG transformations changes the program graph. This equivalent transformation is done according to the rules of equivalent transformations for operators and links of the Pifagor language (see [10], [14] for details). The result of such transformations is a LDFG with a modified graph. The example of the equivalent transformation called the "parallel list release" is given in the figure 1.B. A parallel list allows to declare explicitly that all its input operators could be executed simultaneously. In the left graph the parallel list of two elements is the functional input of the interpretation operator. An operator of interpretation has two inputs: one for an argument and another for a function. The operator applies the function to the argument. In the equivalent graph on the right each element of the initial parallel list is the functional input of its own operator of interpretation, and the result of both interpretations is transfered to the parallel list.

Another transformation of the second type is splitting. Application of this transformation to a single LDFG results in two or more LDFGs with modified graphs. Splitting could be used for proof simplification, for example, if we know all the choices in the "select list element" operator. Thus it is possible to simplify not only the data-flow graph, but also marking formulas.

1.3. Folding

The third type of transformations is folding of the program. The folding transformation is applied to the marked edge (except the input and output edges of the program). The whole induced subgraph with nodes, from which the beginning of the concerned edge is reachable (except the input argument), is replaced by a new variable, and formula attached to the concerned edge is added to the precondition of the initial LDFG. The program becomes "shorter" when a part of the code is replaced by a variable. The folding transformation could be applied when graph is completely or partly marked with formulas. If folding is applied to the all marked edges of the graph, then there is a Hoare triple equal to the resulting LDFG. Let us call such folding as *complete folding*. When a LDFG has all edges marked, complete folding transforms its graph into a single variable. The properties of this variable are described in precondition, which also describes the properties of all the data transfered through the edges of the initial graph. At the same time this variable is the return value of the program. Such program of one variable

is called the *empty program*. To prove its correctness it is sufficient to show that the precondition implies the postcondition.

As the result we could make the following conclusion: the whole proof process could be represented as a tree whose root is the initial LDFG, child nodes are derived from parent nodes by one of the transformations 1)–3), and leaves are fully marked LDFGs which are transformed into formulas by the complete folding. Let us call such tree as a *proof tree* or a proof of program correctness.

2. Architecture of the Toolkit

A general scheme of the system for supporting formal verification of functional data-flow parallel programs is given in the figure 2.

Several modules could be distinguished in the system: "Program correctness prover", "Axioms and theorems library management system" and "Errors analysis and output of information about errors". The "Formula prover" module (formula verifier) is isolated from the system as it is a third-party application and could be excluded from the proof process, in this case its operations should be done by a user (manually or with the help of a more convenient verifier). More information about provers could be found in [15, 16, 17, 18].

The principle of the system operation is the following. A user passes a program in the Pifagor language and program specification in the specification language to the system. The "Program correctness prover" module generates the initial DGFM (which is equal to the initial Hoare triple) and starts the proof process, i.e. marking edges with formulas. Axioms and proved theorems from the "Axioms and theorems library" are used in this process. The "Program correctness prover" module sends requests to the "Axioms and theorems library management system". In case the requested function is not present in the library, the error of marking impossibility is returned, which is processed by the "Errors analysis" module. If axioms (theorems) for the considered function are present in the library, then the "Program correctness prover" module makes a selection among them. For each axiom (theorem) available for the considered function the applicability condition in the specification language is formed. This condition is passed to the formula verifier, and if it is true or satisfiable then the axiom (theorem) is used for marking. Otherwise the axiom (theorem) is rejected. After all edges in the data-flow graph are marked, the complete folding is performed. Hoare triples corresponding to the obtained LDFGs are transformed into formulas (let us call them *final formulas*) and are passed to the formula verifier to check their truth. If all the formulas are identically true, then the program is correct and its initial Hoare triple is a theorem. The "Program correctness prover" module passes this theorem (together with the proof) to the "Axioms and theorems library management system" to save the theorem in the library. If the truth of the formula is not proved then the "Errors analysis" module detects the error and informs the user.

Let us consider the operation of the "Program correctness prover" module in detail (see the figure 2). The "Proof process control" block is the main one, it interacts with the user, takes his commands and visualizes the proof process. This block forms the proof tree. Having received a LDFG this block searches operators with unmarked edges

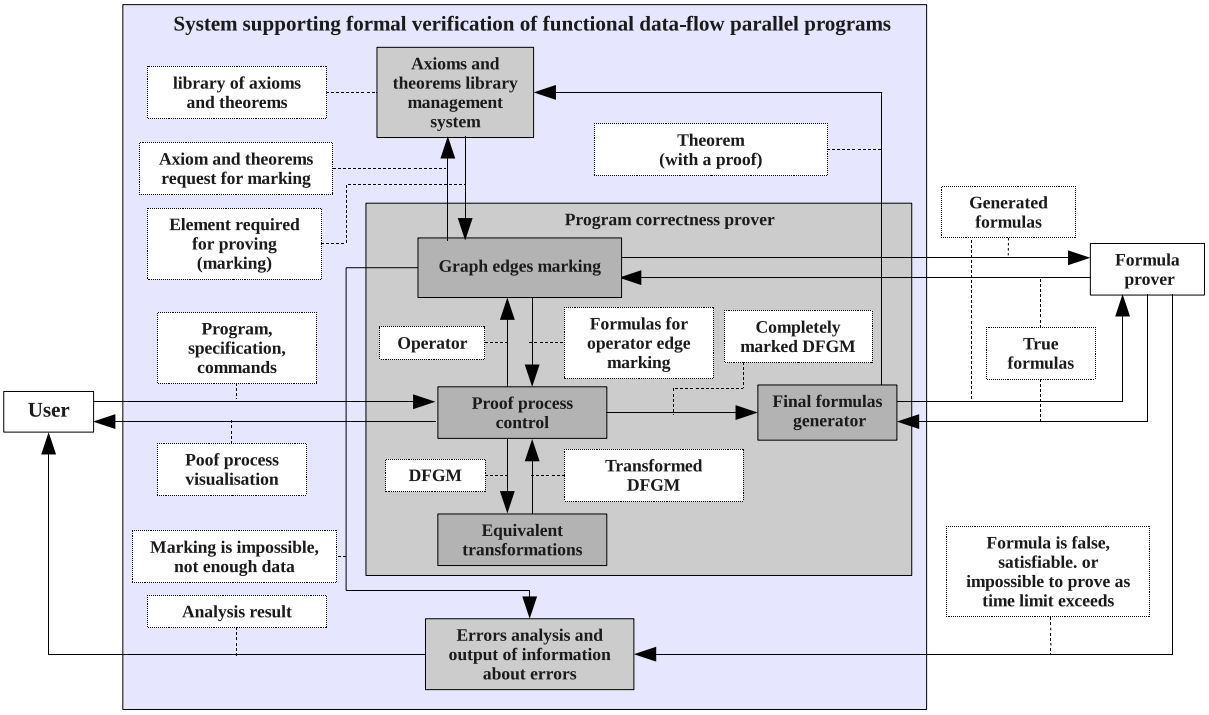


Fig. 2. General scheme of the toolkit for supporting formal verification of functional data-flow parallel programs.

and passes them, firstly, to the "Equivalent transformations" block and, secondly, to the "Graph edges marking" block, which generates marking formulas. After the LDFG is fully marked it is passed to the "Final formulas generator" block that performs the complete folding and generates the set of final formulas.

The considered system could work in several modes:

1) completely manual;

2) partially automated;

3) automated.

In the first case the user uses only the "Proof process control" and "Final formulas generator" blocks and also the "Errors analysis" module. The user is to mark the edges with formulas and prove the truth of the final formula manually. In the second case only the "Formula prover" module is not used. And in the automated mode all modules are used.

3. System Implementation

According to the architecture considered above, the toolkit for supporting formal verification of functional data-flow parallel programs was developed. This toolkit allows to construct a proof tree of programs in the Pifagor language.

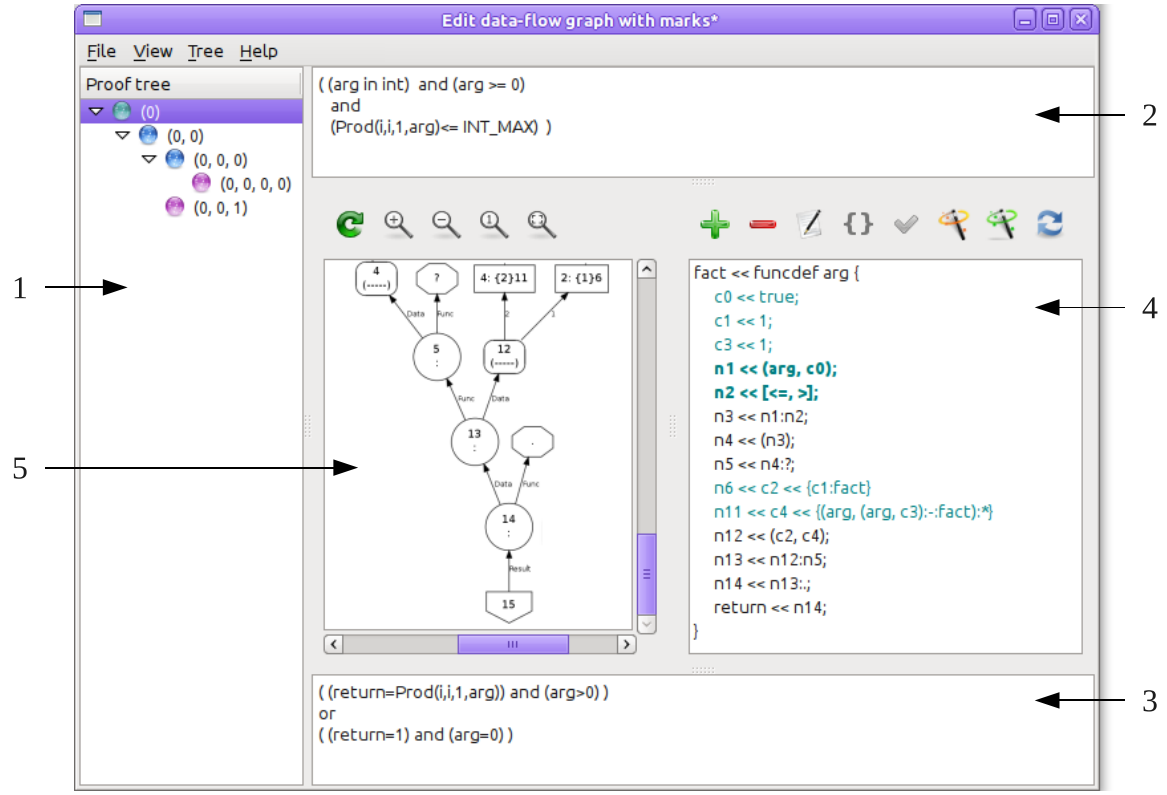


Fig. 3. The main window of the toolkit for supporting formal verification of functional data-flow parallel programs; 1 — the proof tree nodes editor, 2 — the precondition entry field, 3 — the postcondition entry field, 4 — the program code editor, 5 — the graphical representation of the reversed data-flow graph of the program

The system has a graphical user interface for editing a proof tree. As an input data, the system receives a program in the reversed data-flow graph (RDFG) format [19, 20], which represent data dependencies in the program (this graph is the same as the data-flow graph of the program except the edges that has the opposite direction), and the formulas marking its edges. Also a previously saved LDFG or a proof tree could be loaded.

The main window of the system (shown in the figure 3) is the proof tree editor. On the left part it has a tree nodes editor and on the rigth it has the LDFG editor, which shows the current tree node. The proof tree editor allows to insert new or already existing LDFGs, copy and delete current LDFGs and insert a copied LDFG as a child to the current LDFG.

The LDFG editor has precondition and postcondition entry fields in its top and bottom parts respectively. The graphical representation of the reversed data-flow graph of the program is on the left part of the editor and the program code editor is on the right. A third-party program GraphViz is used to generate the graphical representation of the RDFG. This program uses DOT files (DOT is a plain text graph description language). The program code editor has a tool bar and a text box for a program source code. The source code of a program is divided into lines, where each line is equal to a

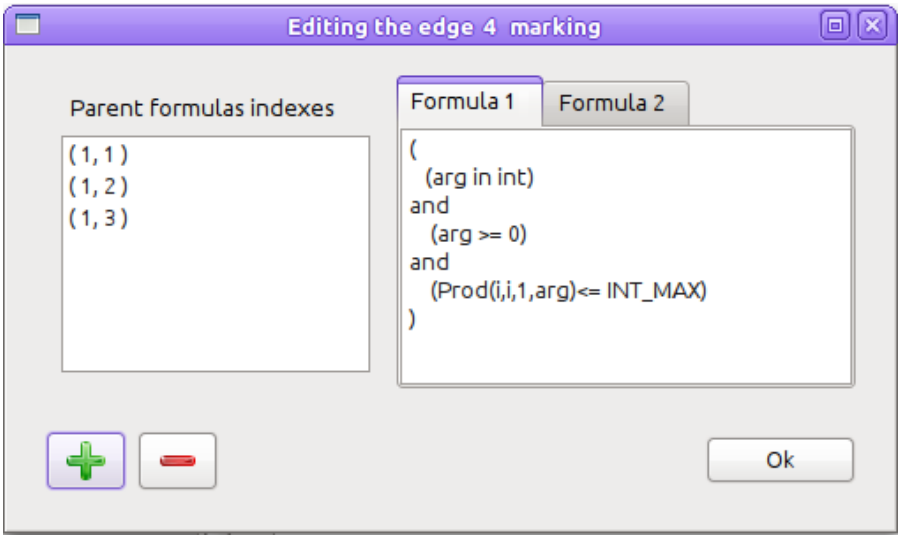


Fig. 4. The window to edit formulas hierarchy for the marking edge

single node of the program RDFG. The RDFGs are in a text format and are created according to the program code. Such RDFG text format has the right order of function calls, which means that a function is called only after its definition. So this RDFG text format allows to restore the program code unambiguously.

The program code editor allows to insert, delete and edit RDFG nodes. At the same time certain restrictions do not allow the user to use an operand before its definition. The editor is also capable of editing delay lists. The operator of grouping into the delay list (briefly, the delay list) contains a subgraph of the program. Operators from the delay list are not executed even if all their arguments are ready. Their execution starts only after the delay release, when the delayed subgraph becomes a part of the whole graph [10], [11]. The program code editor allows to add any operator to the delay list together with all operators that are used by the concerned operator. A delay list is treated as a constant, so editing delayed operators is possible only after the delay release.

Edges marking could be done manually, by means of the LDFG editor, or by calling the function of automatic edges marking. The LDFG editor uses the compact representation of LDFGs, which was described above. If an operator output edge was not marked then the readiness of marks for all operator input edges is checked. If some graph in the compact representation has an unmarked edge, then error message of marking impossibility for the current edge is returned to the user. If all input edges are marked then the window (see figure 4) with formulas hierarchy for the output edge of the current node is shown. All formulas of the edge are divided into groups according to different combinations of their parent nodes formulas. There is a list of parent formulas indices on the left part of the window. Marking formulas of the edge currently being marked are displayed on the pages of the tab widget on the right. The user could add new formulas, and the system automatically changes the indices of all child nodes (if they are already marked) and the necessary number of empty pages on the tab widget are added for new formulas. If a formula is deleted, all child formulas for the current formula are also deleted, and indices of the rest formulas are changed respectively.

Also available are the following functions: autotransforming, LDFG automarking and

final formulas generating.

The autotransforming of the current LDFG (namely, equivalent transformations that are performed automatically) is done in case the current LDFG is a leaf of the proof tree, otherwise its child subtree should be deleted. Furthermore, only *ready* operators, whose all input edges are marked and an output edge is not marked, are transformed. The applicability of the equivalent transformation is checked for all ready operators. If the equivalent transformation is applicable to a LDFG, then a new obtained LDFG becomes a child of the initial LDFG in the proof tree.

The function of LDFG automarking is applied to the current graph. At first equivalent transformations are applied, then a ready operator is searched to perform the marking of its output edge. If the found operator is a list, then the output edge is marked with the constant *true*. If it is the operator of interpretation, then a request for a list of axioms (for built-in functions) or a list of theorems (for user-defined functions) is send to the library of axioms and theorems. If the function is present in the library, then for each axiom (theorem) the applicability condition is formed. The truth or satisfiability of this condition should be checked manually. If this condition is identically false for some axiom or theorem, then it is rejected. The remaining axioms and theorems are used for marking the edge of the considered operator of interpretation with several formulas. Further, another ready operator is searched.

If all edges of current LDFG are marked, then the complete folding is applied. The result is a set of Hoare triples with the empty program. They are transformed into formulas by the final formulas generating function. The user should check whether all these formulas are identically true to prove the program correctness.

Thus, the completely manual and partially automated modes of the system described in the previous section have already been implemented.

4. Conclusion

This paper describes main transformations of a labeled data-flow graph during the program correctness proving. Main concepts of the architecture of the toolkit for supporting formal verification of functional data-flow parallel programs were developed. The task of implementing the toolkit is partially solved. At present the work is underway to enhance the partially automated mode functionality and implement the automated mode.

References

[1] Nepomnyaschiy V. A., Ryakin O. M., *Prikladnyie metodyi verifikatsii programm*, Radio i svyaz, Moscow, 1988, 255 pp., [in Russian].

[2] Hoare C. A. R., "An Axiomatic Basis for Computer Programming", *Communications of the ACM*, **10**:12 (1969), 576–585.

[3] Floyd R. W., "Assigning meaning to programs", Mathematical Aspects of Computer Science, ed. J. T. Schwartz, 1967, 19–32.

[4] Barnett M., Chang B. Y. E., Deline R., et al., "Boogie: A Modular Reusable Verifier for Object-Oriented Programs", LNCS, **4111**, 2006, 364–387.

[5] Nepomniaschy V. A., Anureev I. S., Atuchin M. M., "C Program Verification in SPECTRUM Multilanguage System", *Automatic Control and Computer Sciences*, **45**:7 (2011), 413–420.

[6] Van den Berg J., Jacobs B., "The LOOP compiler for Java and JML", LNCS, **2031**, 2001, 299–312.

[7] Ahrendt W., Baar T., Beckert B., "The KeY Tool", *Software and System Modeling*, **4**:1 (2005), 32–54.

[8] Detlefs D., Nelson G., Saxe J. B., "Simplify: a theorem prover for program checking", *Journal of the ACM*, **52**:3 (2005), 365–473.

[9] Owre S., Rajan S., Rushby J. M., "PVS: Combining Specification, Proof Checking, and Model Checking", LNCS, **1102**, 1996, 411–414.

[10] Legalov A. I., "Funktsionalnyiy yazyik dlya sozdaniya arhitekturno-nezavisimyih parallelnyih programm", *Vyichislitelnyie tehnologii*, **10**:1 (2005), 71–89, [in Russian].

[11] Legalov A. I., Kuzmin D. A., Kazakov F. A., "Na puti k perenosimyim parallelnyim programmam", *Otkryityie sistemyi*, **5** (2003), 36–42, [in Russian].

[12] Kropacheva M. S., Legalov A. I., "Formal Verification of Programs in the Functional Data-Flow Parallel Language", *Automatic Control and Computer Sciences*, **47**:7 (2013), 373–384.

[13] Kropacheva M. S., Legalov A. I., "Formal Verification of Programs in the Pifagor Language", *Parallel Computing Technologies (PaCT-2013)*, 12th International Conference (September 30 - October 4, 2013. Saint-Petersburg, Russia), LNCS, **7979**, 2013, 80–89.

[14] Kropacheva M. S., "Formalizatsiya semantiki funktsionalno-potokovogo yazyika parallelnogo programmirovaniya Pifagor", *Problemyi informatizatsii regiona (PIR-2011): Materialyi XII Vserossiyskoy nauchno-prakticheskoy konferentsii* (Krasnoyarsk, 22 – 23 noyabrya 2011), Publishing of the Siberian Federal University, Krasnoyarsk, 2011, 144–148, [in Russian].

[15] *The Seventeen Provers of the World*, AI Systems, LNAI, **3600**, ed. Wiedijk F., 2006, 169 pp.

[16] Gordon M., Melham T., *Introduction to HOL: a theorem proving environment for higher order logic*, Cambridge University Press, 1993.

[17] Bertot Y., Casteran P., *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*, Springer, 2004, 494 pp.

[18] Nipkow T., Paulson L., Wenzel M., *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, LNCS, **2283**, 2002, 205 pp.

[19] Legalov A. I., Savchenko G. V., Vasilev V. S., "Sobyitiynaya model vyichisleniy, podderzhivayuschaya vyipolnenie funktsionalno-potokovyih parallelnyih programm", *Sistemyi. Metodyi. Tehnologii*, **1**:13 (2012), 113–119, [in Russian].

[20] Matkovskiy I. V., "Parallelnaya sobyitiynaya mashina dlya funktsionalno-potokovogo yazyika "Pifagor"", *Informatsionnyie i matematicheskie tehnologii v nauke i upravlenii: Sbornik trudov XVVII Baykalskoy Vserossiyskoy konferentsii s mezhdunarodnyim uchastiem* (Irkutsk – Baykal, 30 iyunya – 9 iyulya 2012), **2**, Publishing of the Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences, Irkutsk, 2012, 186–193, [in Russian].

DOI: 10.18255/1818-1015-2015-4-578-589

Автоматизация формальной верификации программ на языке Пифагор

Ушакова М.С., Легалов А.И.

получена 18 мая 2015

В связи с увеличением сложности программного обеспечения корректность программы всё чаще доказывается с помощью методов формальной верификации. Дедуктивный анализ на основе исчисления Хоара применим для произвольных языков программирования и допускает частичную автоматизацию процесса. Однако дедуктивный анализ не нашёл широкого применения для верификации параллельных программ из-за высокой сложности процесса. Использование функционально-потоковой парадигмы параллельного программирования позволяет снизить сложность доказательства. В работе рассматривается процесс доказательства корректности функционально-потоковых параллельных программ на языке Пифагор и предлагается архитектура инструментального средства для поддержки процесса доказательства. Процесс доказательства корректности программы представляется в виде дерева, каждый узел которого — информационный граф программы, в котором дуги размечены формулами на языке спецификации. Корнем дерева является информационный граф программы с предусловием и постусловием, которые описывают ограничения на входные переменные и условия корректности результата работы программы соответственно. Основные преобразования, применимые к информационному графу программы: разметка дуг, эквивалентное преобразование, расщепление, свертка программы. Посредством данных преобразований информационный граф модифицируется и, в конечном счете, сводится к набору формул на языке спецификации, истинность которых будет свидетельствовать о корректности программы. Предложена архитектура системы поддержки процесса доказательства, которая позволяет строить дерево доказательства. В системе выделено несколько основных модулей: «Модуль доказательства корректности программы», «Система управления библиотекой аксиом и теорем» и «Модуль анализа ошибок и выдачи информации об ошибках». Согласно описанной архитектуре, разработано инструментальное средство для поддержки формальной верификации, которая позволяет строить дерево доказательства. Описана основная функциональность реализации системы.

Ключевые слова: функционально-потоковое параллельное программирование, язык программирования Пифагор, формальная верификация программ, инструментальные средства для поддержки формальной верификации

Для цитирования: Ушакова М.С., Легалов А.И., "Автоматизация формальной верификации программ на языке Пифагор", *Моделирование и анализ информационных систем*, **22**:4 (2015), 578–589.

Об авторах:
Ушакова Мария Сергеевна, orcid.org/0000-0003-4234-2714, аспирант,
Институт космических и информационных технологий, Сибирский федеральный университет,
ул. Академика Киренского, 26, г. Красноярск, 660074 Россия, e-mail: ksv@akadem.ru

Легалов Александр Иванович, orcid.org/0000-0002-5487-0699, д-р техн. наук, профессор, заведующий кафедрой вычислительной техники,
Институт космических и информационных технологий, Сибирский федеральный университет,
ул. Академика Киренского, 26, г. Красноярск, 660074 Россия, e-mail: legalov@mail.ru