ISSN 1818-1015
eISSN 2313-5417

2023 Volume 30 No 3

MODELING AND ANALYSIS OF INFORMATION SYSTEMS

SCIENTIFIC JOURNAL

Start date of publication — 1999
Published quarterly

FOUNDER
P.G. Demidov Yaroslavl State University

EDITORIAL OFFICE
14 Sovetskaya str., Yaroslavl 150003, Russian Federation

Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Phone: +7 (4852) 79-77-73

2023 Том 30 № 3

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ

НАУЧНЫЙ ЖУРНАЛ

Издается с 1999 года
Выходит 4 раза в год

УЧРЕДИТЕЛЬ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Ярославский государственный университет им. П. Г. Демидова»

РЕДАКЦИЯ
ул. Советская, 14, Ярославль, 150003, Российская Федерация
Website: http://mais-journal.ru
E-mail: mais@uniyar.ac.ru
Телефон: +7 (4852) 79-77-73

Свидетельство о регистрации СМИ ПИ № ФС77-66186 от 20.06.2016 выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций. Подписной индекс в каталоге «Урал-Пресс» – 31907. Технический редактор, компьютерная вёрстка – К. В. Лагутина. Подписано в печать 13.09.2023. Дата выхода в свет 30.09.2023. Формат 200×265 мм. Объем 96 с. Тираж 32 экз. Свободная цена. Заказ 23119. Издатель и его адрес: Ярославский государственный университет им. П. Г. Демидова; ул. Советская, 14, Ярославль, 150003, Россия. Типография и ее адрес: ООО «Филигрань»; ул. Свободы, 91, Ярославль, 150049, Россия.
Содержание предназначено для детей старше 12 лет.

Contents

Содержание

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

THEORY OF DATA

Generalization of typed include dependencies with null values in databases

S. V. Zykin[1]

DOI: 10.18255/1818-1015-2023-3-192-201

[1]Sobolev institute of mathematics SB RAS, 4 Acad. Koptyug avenue, 630090 Novosibirsk Russia.

MSC2020: 68P15
Research article
Full text in Russian

Received July 7, 2023
After revision August 1, 2023
Accepted August 2, 2023

The paper discusses a new type of dependency in databases, which is a generalization of inclusion dependencies. Traditionally, such dependencies are used in practice to ensure referential integrity. In this case, the restriction is established only between a pair of relations, the first of which is called the main, the second is external. In practice, referential integrity often needs to be established for a larger number of relations, where several main and several external relations participate in the same constraint. Such a structure corresponds to an ultragraph. The paper provides a rationale for generalized inclusion dependencies that take into account the presence of null values in external relations. Based on the study of the properties of typed dependencies, a system of axioms is obtained, for which consistency (soundness) and completeness are proved.

Keywords: database; inclusion dependencies; axiomatics; null values

INFORMATION ABOUT THE AUTHORS

Sergey V. Zykin
corresponding author

orcid.org/0000-0002-0576-2149. E-mail: szykin@mail.ru
leading researcher, Doctor in Technic.

Funding: The research was funded in accordance with the state task of the IM SB RAS, project FWNF-2022-0016.

For citation: S. V. Zykin, “Generalization of typed include dependencies with null values in databases”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 192-201, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

THEORY OF DATA

Обобщенные типизированные зависимости включения с неопределенными значениями в базах данных

С. В. Зыкин[1]

DOI: 10.18255/1818-1015-2023-3-192-201

[1]Институт математики им. С.Л. Соболева СО РАН, пр. ак. Коптюга, 4, 630090, г. Новосибирск, Россия.

УДК 004.652.4
Научная статья
Полный текст на русском языке

Получена 7 июля 2023 г.
После доработки 1 августа 2023 г.
Принята к публикации 2 августа 2023 г.

В статье рассматривается новый вид зависимостей в базах данных, являющийся обобщением зависимостей включения. Традиционно такие зависимости на практике используются для обеспечения ссылочной целостности. При этом, ограничение устанавливается только между парой отношений, первое из которых называется главным, второе — внешним. На практике ссылочную целостность часто требуется установить для большего числа отношений, где в одном ограничении участвуют несколько главных и несколько подчиненных отношений. Такая структура соответствует ультраграфу. В работе приведено обоснование обобщенных зависимостей включения, учитывающих наличие неопределенных значений во внешних отношениях. На основе исследования свойств типизированных зависимостей получена система аксиом, для которой доказана непротиворечивость (надежность) и полнота.

Ключевые слова: база данных; зависимости включения; аксиоматика; неопределенные значения

ИНФОРМАЦИЯ ОБ АВТОРАХ

Сергей Владимирович Зыкин
автор для корреспонденции

orcid.org/0000-0002-0576-2149. E-mail: szykin@mail.ru
ведущий научный сотрудник, доктор техн. наук.

Финансирование: Работа выполнена в рамках государственного задания ИМ СО РАН, проект FWNF-2022-0016.

Для цитирования: S. V. Zykin, “Generalization of typed include dependencies with null values in databases”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 192-201, 2023.

Введение

Ссылочная целостность в базах данных (referential integrity) является реализацией бизнес-правил в прикладной области, то есть отражает логику функционирования информационной системы с использованием структурной целостности базы данных (БД). В современных системах управления базами данных (СУБД) такие ограничения поддерживается за счет связей (relationship) на схеме БД.

В теории ссылочная целостность была формализована в виде зависимостей включения. Проблема развития теории зависимостей включения до сих пор вызывает интерес со стороны исследователей, прежде всего с точки зрения взаимодействия с другими видами зависимостей в БД. Кроме того, на практике появляются новые виды ссылочных ограничений целостности, которые требуют исследования и встраивания их в общую теорию проектирования БД.

Пример 1. Рассмотрим пример схемы БД, в котором требуется объединить одной связью несколько отношений. Пусть у некоторой фирмы есть два типа клиентов: собственные сотрудники и внешние (сторонние) клиенты. Эти два типа клиентов не целесообразно объединять в один класс объектов, поскольку у них различные атрибуты и различные правила обслуживания. Допустим, что фирма предоставляет два вида услуг для своих клиентов: создание (и сопровождение) накопительных счетов клиентов и регистрация (и исполнение) заказов клиентов. В следующем фрагменте схемы БД фирмы представлены соответствующие отношения:
R_1 = *Внутренние клиенты* (**№ клиента**, № сотрудника),
R_2 = *Внешние клиенты* (**№ клиента**, ФИО клиента, Адрес клиента),
R_3 = *Счета клиентов* (**№ счета**, № клиента, Остаток средств),
R_4 = *Заказы клиентов* (**№ заказа**, № клиента, Содержание заказа),
где полужирным шрифтом выделены ключевые атрибуты отношений.

В примере 1 атрибут «№ клиента» является первичным ключом в отношениях R_1 и R_2. В отношениях R_3 и R_4 этот атрибут является внешним ключом. Для представленных отношений должны быть установлены ссылочные ограничения целостности: в отношении R_3 атрибут «№ клиента» может принимать только те значения, которые есть либо в R_1 либо в R_2. Аналогичная ситуация для отношения R_4, в котором атрибут «№ клиента» может принимать только те значения, которые есть либо в R_1 либо в R_2. В общем случае совпадающие значения атрибута «№ клиента» могут быть как в отношениях R_1 и R_2, так и в отношениях R_3 и R_4. Таким образом, все четыре отношения должны быть объединены одним ограничением целостности, в котором R_1 и R_2 являются главными отношениями, а отношения R_3 и R_4 — внешними. По структуре такое объединение является ребром ультраграфа, в котором R_1 и R_2 являются входящими узлами, а R_3 и R_4 исходящими узлами.

В технологиях БД значения внешних ключей могут принимать неопределенные значения (*Null*), если они не являются компонентами первичного ключа. Для рассмотренного примера 1 это означает, что в отношении R_3 атрибут «№ клиента» может принять неопределенное значение, например, для остатка невостребованных средств. В отношении R_4 атрибут «№ клиента» может принять неопределенное значение для предзаказа. Если появляется необходимость замены неопределенного значения, то соответствующее определенное значение атрибута «№ клиента» должно быть выбрано в отношениях R_1 или R_2. Формальная теория должна учитывать данную ситуацию.

Формализацией представленного вида ссылочных ограничений целостности будем считать обобщенные зависимости включения. В данной работе предложена система аксиом, соответствующая свойствам обобщенных зависимостей включения. Для этой системы доказана полнота и непротиворечивость. Представленная система аксиом является обобщением теории типизированных зависимостей включения при наличии неопределенных значений.

1. Связанные работы

Классическая теория проектирования реляционных баз данных основывается на четырех видах зависимостей: функциональные зависимости, многозначные зависимости, зависимости соединения [1, 2], и зависимости включения [3]. Обозначим все множество атрибутов $U = \{A_1, A_2, \ldots, A_n\}$,

на которых определены все отношения (таблицы) R_i в БД, $[R_i]$ — схема отношения R_i (атрибуты, на которых определено отношение R_i), $[R_i] \subseteq U$, $1 \leq i \leq k$, $\mathbf{R} = (R_1, R_2, \ldots, R_k)$ — БД, $S = \{[R_1], [R_2], \ldots, [R_k]\}$ — схема БД без учета ограничений ссылочной целостности. В работе [3] приведено следующее определение зависимостей включения.

Определение 1. Пусть R_i и R_j — два отношения БД (возможно $i = j$), заданы два множества $V \subseteq [R_i]$ и $W \subseteq [R_j]$, для которых $|V| = |W|$, тогда соотношение на множествах кортежей $R_i[V] \subseteq R_j[W]$ называется зависимостью включения.

В определении 1 $|V|$ — кардинальность множества V, $R_i[V]$ — проекция (столбцы) отношения R_i, соответствующие атрибутам V. Зависимость включения является типизированной (typed), если выполнено равенство $V = W$, иначе — нетипизированной.

В работе [3] представлена система из трех аксиом зависимостей включения: рефлексивность, проецирование и перестановка, транзитивность. Относительно этих аксиом доказана полнота (непротиворечивость очевидна). Неоднократно исследователями осуществлялись попытки построить совместную систему аксиом для функциональных зависимостей и зависимостей включения. В результате было получено, что полная аксиоматизация существует по отдельности для этих видов зависимостей, тогда как совместная аксиоматизация отсутствует [3—5]. В некоторых частных случаях полная аксиоматизация [6, 7] была получена.

Исследования зависимостей включения и их взаимодействие с функциональными зависимостями были проанализированы в работах [7—9]. В результате были сформулированы условия существования нормальной формы зависимостей включения (IDNF), в основе которой лежит нормальная форма Бойса-Кодда (BKNF), дополненная условием ацикличности. Интерпретация зависимостей включения в виде ориентированного графа и полиномиальный алгоритм поиска избыточных зависимостей представлены в работе [10].

В работе [11] рассматривается проблема взаимодействия функциональных зависимостей и зависимостей включения с частным случаем многозначных зависимостей, компоненты базиса которых названы в статье независимыми атомами. Основной результат заключается в том, что как конечные, так и неограниченные задачи выводимости для комбинированного класса атомов независимости, унарных функциональных зависимостей и унарных зависимостей включения аксиоматизируемы и разрешимы за полиномиальное время.

Заметим, что полная аксиоматизация необходима для определения выводимости ранее установленной зависимости. Это является основой для построения не избыточного и непротиворечивого проекта БД.

Значительный объем исследований посвящен разработке алгоритмов поиска зависимостей включения в существующей БД. Различные алгоритмы поиска зависимостей включения представлены в работах [12—14]. В качестве источников информации используются свойства схемы БД, например, первичный ключ одного отношения может быть внешним ключом в другом отношении. Кроме того, используются трудоемкие алгоритмы анализа сохраненных данных в БД, и даже обработка входящих запросов. Такие алгоритмы дают неточные результаты, требующие дополнительной проверки обнаруженных зависимостей включения. Их достоинство в том, что частично автоматизируется процесс поиска зависимостей включения.

Полная автоматизация поиска зависимостей включения в общем случае нереализуема, поскольку специфика бизнес-правил в прикладной области требует ручной корректировки. Улучшить качество и полноту построения зависимостей позволяют алгоритмы, которые используют определения пороговых значений качества [15, 16].

Значительные усилия в настоящее время уделяется направлению поиска и анализа зависимостей включения. Для этого используются алгоритмы проверки и диагностики соблюдения правил описания семантики данных на основе так называемых бизнес-данных, в том числе при модификации структуры хранения данных и наличии неточных данных [17—21].

Особое направление исследований посвящено наличие неопределенных значений в БД. Проблема выводимости при наличии неопределенных значений рассматривается в работах [22—24]. При этом неопределенные значения учитываются совместно для функциональных зависимостей и зависимостей включения. Полная и непротиворечивая система аксиом получена только для частного случая. Причиной для невозможности получения результатов в общем случае является взаимодействие функциональных зависимостей и нетипизированных зависимостей включения.

В данной работе рассматривается новый вид зависимостей - обобщенные типизированные зависимости включения. В работе [25] показано, что решение семантических проблем на этапе проектирования БД позволяет избавиться от нетипизированных зависимостей включения. С учетом отсутствия взаимодействия типизированных зависимостей включения с функциональными зависимостями появляется возможность построения неизбыточного и непротиворечивого проекта БД в общем случае.

2. Предварительные сведения

В примере 1 общий атрибут «№ клиента» дублируется в качестве первичного ключа в отношениях R_1 и R_2, и в качестве внешнего ключа в отношениях R_3 и R_4. Такой результат невозможно получить, используя классическую теорию проектирования БД [1, 2], хотя необходимость такого результата неоднократно отмечалась разработчиками БД. Это, отчасти, обуславливает критическое отношение к классической теории проектирования БД. В работе [26] предложено решение указанной проблемы за счет введения понятия области определения функциональной зависимости. Для полноты формальных построений кратко рассмотрим полученные результаты.

В работе используется ранее введенное разделение значения *Null* на два значения: «не определено» и «отсутствует». Для последнего значения предложен специальный маркер *nov*.

Определение 2. Пусть R — произвольная реализация отношения на множестве атрибутов U и $X \subseteq U$. Множество кортежей $t \in R$ будем называть областью определения $dom(X)$ множества атрибутов X, если $t[A_j] \neq nov$ для всех атрибутов $A_j \in X$.

Из определения 2 следуют очевидные свойства.

Свойство 1. Если $X = \cup_{i=1}^{m} X_i$, тогда $dom(X) = \cap_{i=1}^{m} dom(X_i)$.

Свойство 2. Пусть $Y \subseteq X$, тогда $dom(X) \subseteq dom(Y)$.

Определение 3. Областью определения $dom(X \to Y)$ функциональной зависимости $X \to Y$ называется множество кортежей $t \in R$, для которых $t[A_j] \neq nov$, если $A_j \in X \cup Y$, и для любой пары кортежей $t_1, t_2 \in dom(X \to Y)$, если $t_1[X] = t_2[X]$, то $t_1[Y] = t_2[Y]$.

Свойство 3. Если $F = \{F_1, F_2, \ldots, F_k\}$ множество функциональных зависимостей, то

$$dom(F) = \cap_{i=1}^{k} dom(F_i).$$

Свойство 4. Для любой зависимости $X \to Y$ выполнено $dom(X \to Y) = dom(X) \cap dom(Y)$.

На основе исследованных свойств была получена следующая система аксиом:

Аксиома. (Рефлексивность.) Если $Y \subseteq X \subseteq U$, то $X \to Y$ и $dom(X \to Y) = dom(X)$.

Аксиома. (Пополнение.) Если $X \to Y$ и $Z \subseteq U$, то $XZ \to YZ$ и $dom(XZ \to YZ) = dom(X) \cap dom(Y) \cap dom(Z)$.

Аксиома. (Транзитивность.) Если $X \to Y$ и $Y \to Z$, то $X \to Z$ и $dom(X \to Z) = dom(X) \cap dom(Z)$.

Для представленной системы аксиом доказана полнота и непротиворечивость (надежность). Основное достоинство данной системы в том, что она является истинным обобщением классической системы аксиом [1]: если область определения зависимостей будет совпадать со всей прикладной областью (отсутствует значение *nov*), то предложенная система будет совпадать с классической. Кроме того, в [26] рассмотрены алгоритмы построения замыкания множества атрибутов и построения минимального покрытия множества функциональных зависимостей с доказательством их корректности.

Использование областей определения функциональных зависимостей согласуется с синтетическим подходом к формированию схемы БД [1] на основе минимального покрытия множества функциональных зависимостей. Отличие заключается только в том, что объединяемые функциональные зависимости должны иметь не только совпадающие левые части (детерминанты), но совпадающие области определения. Результат применения такого подхода продемонстрирован в примере 1. После формирования отношений указанным способом возникает вопрос об установлении между ними условий ссылочной целостности. Рассмотренный далее материал позволяет дать ответ на поставленный вопрос.

3. Основы формальной теории обобщенных зависимостей включения

Типизированные зависимости включения, удовлетворяющие определению 1, не учитывают неопределенные значения. Для сопоставления неопределенных значений рассмотрим условие соответствия кортежей [25], которое является аналогом частичного порядка.

Определение 4. Кортеж $t_i[X]$ *соответствует* кортежу $t_j[X]$ на множестве атрибутов X ($t_j[X] \preccurlyeq t_i[X]$), если $t_i[A_l] \neq Null$, тогда $t_j[A_l] = t_i[A_l]$ или $t_j[A_l] = Null$; если $t_i[A_l] = Null$, тогда $t_j[A_l] = Null$ для любого атрибута $A_l \in X$.

Свойство соответствия, заданное в определении 4, является транзитивным. А это означает, что справедливо следующее утверждение: если $t_j[X] \preccurlyeq t_i[X]$ и $t_i[X] \preccurlyeq t_m[X]$, тогда $t_j[X] \preccurlyeq t_m[X]$. На основе данного свойства в работе [25] дано определение типизированной зависимости включения.

Определение 5. Типизированная зависимость включения $\sigma = R_j[X] \subseteq R_i[X]$ между главным отношением $R_i[X]$ и внешним отношением $R_j[X]$ по атрибутам X выполнена, если для любого кортежа $t_j[X] \in R_j[X]$ имеется *соответствующий* кортеж $t_i[X]$ в отношении $R_i[X]$.

Рассмотрим обобщение зависимостей включения, представленных в примере 1. Для этого будем использовать запись $W[X] \subseteq V$, где W — множество внешних отношений зависимости, V — множество главных отношений зависимости. Справа и слева зависимости подразумевается один и тот же набор атрибутов X, поэтому для краткости будем указывать его только слева.

Определение 6. Обобщенная типизированная зависимость включения $\sigma = W[X] \subseteq V$ между главными отношениями V и внешними отношениями W по атрибутам X выполнена, если для каждого отношения $R_i \in W$ и каждого кортежа $t_i \in R_i$ существует отношение $R_j \in V$ и существует *соответствующий* кортеж $t_j \in R_j$, то есть выполнено условие: $t_i[X] \preccurlyeq t_j[X]$.

С учетом введенных обозначений и определений зависимость в примере 1 будет иметь вид: $\{R_3, R_4\}[X] \subseteq \{R_1, R_2\}$, где X — атрибут «№ клиента». Очевидно, что обычные (бинарные) зависимости включения являются частным случаем зависимостей в определении 6, когда справа и слева присутствует только по одному отношению.

Пусть Σ — множество обобщенных типизированных зависимостей включения, определенных на схеме БД, а σ — произвольная обобщенная типизированная зависимость включения (далее будем использовать краткий термин «зависимость»). Возможно, что σ является элементом множества Σ.

Определение 7. Зависимость σ является логическим следствием множества зависимостей σ ($\Sigma \vDash \sigma$), если все допустимые реализации БД, удовлетворяющие зависимостям Σ, также удовлетворяют зависимости σ. В этом случае зависимость σ будем называть *выполнимой.*

Заметим, что в соответствии с определением 7 все зависимости в Σ являются выполнимыми.

Исследование свойств зависимостей позволили сформировать следующую систему аксиом:

Аксиома 1. (Рефлексия.) Если $W \subseteq V$, тогда $W[X] \subsetneq V$.

Аксиома 2. (Проекция.) Если $W[X] \subsetneq V$ и $Y \subseteq X$, тогда $W[Y] \subsetneq V$.

Аксиома 3. (Объединение.) Если $W[X] \subsetneq V$ и $S[X] \subsetneq V$, тогда $\{W \cup S\}[X] \subsetneq V$.

Аксиома 4. (Транзитивность.) Если $W[X] \subsetneq S$ и $S[X] \subsetneq V$, тогда $W[X] \subsetneq V$.

Здесь V, W, S — множества отношений (таблиц) БД; X, Y — множества атрибутов.

Аксиомы 1–4 могут быть использованы в качестве правил вывода. Так, из системы аксиом следует правило, которое далее будет использоваться:

$$\text{если } W[X] \subsetneq V \text{ и } S \subseteq W, \text{ тогда } S[X] \subsetneq V. \tag{1}$$

Действительно, из условия $S \subseteq W$ по аксиоме 1 следует зависимость $S[X] \subsetneq W$. Затем по аксиоме транзитивности 4 будет получена зависимость $S[X] \subsetneq V$. Правило (1) далее будет называться правилом декомпозиции.

Определение 8. Зависимость σ выводима из множества зависимостей Σ за счет применения аксиом 1–4, если за конечное число шагов будет получена зависимость σ. Далее будем писать $\Sigma \vdash \sigma$.

Рассмотрим обоснование надежности (непротиворечивости) представленной системы аксиом. Для этого необходимо показать выполнимость зависимости σ, если она выводима из множества Σ с использованием системы аксиом: $\Sigma \vdash \sigma \Rightarrow \Sigma \vDash \sigma$.

Теорема 1. *(Надежность.) Система аксиом 1–4 надежна.*

Доказательство. Покажем надежность каждой из аксиом по отдельности.

Аксиома 1. Рассмотрим произвольный кортеж t, принадлежащий, по крайней мере, одному из отношений R_i множества W: $t \in R_i$. Поскольку $W \subseteq V$, то $R_i \in V$. По определениям 4 и 5 кортеж $t[X]$ соответствует самому себе для произвольного множества атрибутов $X \subseteq [R_i]$, что доказывает выполнимость зависимости, полученной за счет рефлексии.

Аксиома 2. Пусть отношение $R_i \in W$ и кортеж $t_i \in R_i$. Поскольку выполнена зависимость $W[X] \subsetneq V$, то существует отношение $R_j \in V$ и кортеж $t_j \in R_j$ такой, что $t_i[X] \preccurlyeq t_j[X]$. Поскольку $Y \subseteq X$, то $t_i[Y] \preccurlyeq t_j[Y]$, что доказывает выполнимость $W[Y] \subsetneq V$. Следовательно, аксиома 2 надежна.

Аксиома 3. Пусть отношение $R_i \in W$ и произвольный кортеж $t_i \in R_i$. Поскольку выполнена зависимость $W[X] \subsetneq V$, то существует отношение $R_j \in V$ и кортеж $t_i \in R_j$. Пусть отношение $R_l \in S$ и кортеж $t_l \in R_l$. Поскольку имеет место зависимость $S[X] \subsetneq V$, то существует отношение $R_p \in V$ и кортеж $t_l \in R_p$. После объединения множеств W и S условия на кортежи t_i и t_l останутся без изменений и не будут противоречить зависимости $\{W \cup S\}[X] \subsetneq V$, что доказывает надежность аксиомы 3.

Аксиома 4. Рассмотрим произвольный кортеж $t_i \in R_i$, где отношение R_i принадлежит множеству W. Тогда во множестве S должно присутствовать отношение R_j и кортеж $t_j \in R_j$, для которого выполнено: $t_i[X] \preccurlyeq t_j[X]$. Поскольку выполнена зависимость $S[X] \subsetneq V$, то существует отношение R_l, принадлежащее множеству V, и существует кортеж $t_l \in R_l$: $t_j[X] \preccurlyeq t_l[X]$. Поскольку операция $\preccurlyeq$ удовлетворяет условию транзитивности, то $t_i[X] \preccurlyeq t_l[X]$ и, следовательно, аксиома 4 надежна. □

Замечание. Для обоснования корректности последующих построений необходимо обсудить свойства, которые не выполнены для обобщенных зависимостей включения. Так, объединение двух (и более) правых частей зависимостей, даже если они имеют непустое пересечение, но не совпадают друг с другом, не является эквивалентным преобразованием:

$$W[X] \subsetneq S \text{ и } W[X] \subsetneq V \text{ не эквивалентно } W[X] \subsetneq \{S \cup V\},$$

если $S \not\subseteq V$ и $V \not\subseteq S$. Допустим, что $R_i \in S$ и $R_i \notin V$. Тогда в зависимости $W[X] \subsetneq \{S \cup V\}$ отсутствует необходимость принадлежности кортежей W одному из отношений V, что противоречит зависимости $W[X] \subsetneq V$. Аналогично доказывается утверждение:

$$W[X] \subsetneq S \text{ не следует } W[X] \subsetneq V \text{ , если } V \subset S.$$

Из этого можно сделать вывод, что правая часть обобщенной зависимости включения не допускает удаления отношений.

Докажем полноту системы аксиом 1–4 на основе ранее полученных свойств.

Теорема 2. *(Полнота.) Система аксиом 1–4 полна.*

Доказательство. При доказательстве полноты требуется показать выводимость $\Sigma \vdash \sigma$ зависимости $\sigma = W[X] \subsetneq V$, если зависимость σ выполнима: $\Sigma \vDash \sigma$.

Предварительно преобразуем множество зависимостей Σ к виду с различными правыми частями: зависимости $W_i[X] \subsetneq V_i$ и $W_j[X] \subsetneq V_j$ заменим зависимостью $\{W_i \cup W_j\}[X] \subsetneq V_i$, если $V_i = V_j$. Из правила декомпозиции (1) следует, что полученное множество зависимостей будет эквивалентно исходному множеству Σ.

Прежде чем рассматривать выводимость зависимости σ, надо найти необходимое и достаточное условие ее выполнимости. Рассмотрим последовательность зависимостей:

$$\begin{cases} W[Y_0] \subsetneq V_0 \\ W_1[Y_1] \subsetneq V_1 \\ \quad\vdots \\ W_p[Y_p] \subsetneq V \end{cases} \tag{2}$$

где $X \subseteq Y_i$, $i = \overline{0, p}$, и для любого V_i, где $i = \overline{0, p-1}$, выполнено условие $V_i \subseteq W_{i+1}$.

Покажем достаточность наличия последовательности (2) для выполнимости σ. Пусть кортеж t принадлежит, по крайней мере, одному из отношений множества W. Зависимость $W[Y_0] \subsetneq V_0$ гарантирует наличие кортежа t_0 в некотором отношении множества V_0, для которого выполнено: $t[Y_0] \preccurlyeq t_0[Y_0]$. Из условия $X \subseteq Y_0$ следует, что $t[X] \preccurlyeq t_0[X]$. Поскольку $V_0 \subseteq W_1$, то кортеж t_0 принадлежит некоторому отношению множества W_1. Аналогичные рассуждения для остальных зависимостей последовательности гарантируют наличие кортежа t_p в отношении множества V, для которого, с учетом свойства транзитивности операции $\preccurlyeq$, выполнено $t[X] \preccurlyeq t_p[X]$. Полученное соотношение доказывает выполнимость зависимости σ.

Для обоснования необходимости (2) предположим, что существует кортеж $t \in W$ и не существует кортеж $u \in V$, для которых было бы выполнено условие $t[X] \preccurlyeq u[X]$. Нарушение частичного порядка может произойти в какой-либо зависимости последовательности, либо при переходе от одной зависимости к другой. Рассмотрим текущую зависимость $W_i[Y_i] \subsetneq V_i$. При этом возможны два варианта:

A) $X \not\subseteq Y_i$. В этом случае атрибуты $X \setminus Y_i$ в кортежах отношений множества V_i могут нарушить условие частичного порядка $\preccurlyeq$, которое будет повторяться в последующих зависимостях последовательности (2). Таким образом, будет найдена реализация БД, в которой выполнены зависимости Σ и не выполнена σ.

В) Не выполнено условие $V_i \subseteq W_{i+1}$. Тогда существует отношение $R_j \in V_i$ и $R_j \notin W_{i+1}$. Отношение R_j может содержать кортеж t_j, для которого выполнено $t[X] \preccurlyeq t_j[X]$, но в последующих зависимостях допускается отсутствие кортежей t_l, удовлетворяющих частичному порядку $t[X] \preccurlyeq t_l[X]$, поскольку $R_j \notin W_{i+1}$. Следовательно, найдена реализация БД, в которой выполнены зависимости Σ и не выполнена σ.

Рассмотренные варианты доказывают необходимость наличия последовательности (2).

Рассмотрим выводимость зависимости σ. После объединения зависимостей с совпадающей правой частью будут получены выводимые зависимости, что гарантируется аксиомой 3. Зависимость $W_i[X] \subsetneq V_i$ выводима из зависимости $W_i[Y_i] \subsetneq V_i$ по аксиоме 2, если $X \subseteq Y_i$, $i = \overline{0,p}$. Поскольку выполнено условие $V_i \subseteq W_{i+1}$, $i = \overline{0,p-1}$, то по аксиоме 1 выполнено $V_i[X] \subsetneq W_{i+1}$. Заметим, что все отношения, участвующие в зависимости, определены на атрибутах X. Далее, используя аксиому транзитивности 4, получаем выводимость $W_i[X] \subsetneq V_{i+1}$. Так как это верно для $i = \overline{0,p-1}$, то последовательное применение аксиомы 4 дает результат $W[X] \subsetneq V$. □

Аксиомы 1–4 одновременно являются правилами вывода для обобщенных зависимостей включения. Использование их для поиска выводимых зависимостей позволит сократить количество выполнимых зависимостей, для поддержания которых на практике требуется дополнительная память и время.

Заключение

В работе рассмотрено обобщение зависимостей включения, которое позволяет объединять одним ограничением целостности сразу несколько отношений БД. Возможно расширение ограничений за счет использования не типизированных зависимостей. Однако, анализ использования таких ограничений [25] показывает, что они являются следствием нерешенных семантических проблем на этапе проектирования БД. Действительно, ссылочная целостность, установленная между неоднородными атрибутами, формально соответствует нетривиальной функциональной зависимости. Такие зависимости должны быть использованы при проектировании структуры логических записей на схеме БД, а не в виде ссылочной целостности. Кроме того, типизированные зависимости включения формально соответствуют тривиальным функциональным зависимостям, которые не используются при формировании схемы БД. Следовательно, не появляется проблема взаимодействия зависимостей включения и функциональных зависимостей.

Рассмотренная в данной работе система аксиом не ограничивается условием ацикличности и количеством атрибутов, входящих в зависимости. Это делает предложенный подход универсальным. Далее предполагается исследовать проблему поиска избыточных зависимостей за счет выводимости.

References

[1] J. Ullman, *Principles of Database Systems*. Stanford University: Computer Science Press, 1980, 484 pp.

[2] D. Maier, *The Theory of Relational Databases*. Rockville: Computer Science Press, 1983, 637 pp.

[3] M. Casanova, R. Fagin, and C. Papadimitriou, “Inclusion dependencies and their interaction with functional dependencies”, *Journal of Computer and System Sciences*, vol. 28, no. 1, pp. 29–59, 1984.

[4] A. K. Chandra and M. Y. Vardi, “The implication problem for functional and inclusion dependencies is undecidable”, *SIAM Journal on Computing*, vol. 14, no. 3, pp. 671–677, 1985.

[5] R. Fagin and M. Y. Vardi, “Armstrong databases for functional and inclusion dependencies”, *Information Processing Letters*, vol. 16, no. 1, pp. 13–19, 1983.

[6] P. M. Kanellakis, R. Cosmadakis, and M. Y. Vardi, “Unary inclusion dependencies have polynomial time inference problems”, in *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, 1983, pp. 264–277.

[7] S. S. Cosmadakis, P. C. Kanellakis, and M. Y. Vardi, “Polynomial-time implication problems for unary inclusion dependencies”, *Association for Computing Machinery*, vol. 37, no. 1, pp. 15–46, 1990.

[8] M. Levene and V. M. W., "Justification for inclusion dependency normal form", *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 2, pp. 281–291, 2000.

[9] C. Beeri, R. Fagin, D. Maier, and Y. M., "On the desirability of acyclic database schemes", *Association for Computing Machinery*, vol. 30, no. 3, pp. 479–513, 1990.

[10] R. Missaoui and R. Godin, "The implication problem for inclusion dependencies: A graph approach", *ACM SIGMOD Record*, vol. 19, no. 1, pp. 36–40, 1990.

[11] M. Hannula and S. Link, "On the interaction of functional and inclusion dependencies with independence atoms", in *Database Systems for Advanced Applications*, J. Pei, Y. Manolopoulos, S. Sadiq, and J. Li, Eds., Springer International Publishing, 2018, pp. 353–369.

[12] J. Biskup and P. Dublish, "Objects in relational database schemes with functional, inclusion and exclusion dependencies", in *3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems*, Springer Berlin Heidelberg, 1991, pp. 276–290.

[13] D. S. Johnson and A. Klug, "Testing containment of conjunctive queries under functional and inclusion dependencies", *Journal Computer and System Sciences*, vol. 28, no. 1, pp. 167–189, 1984.

[14] F. De Marchi, S. Lopes, and J.-M. Petit, "Efficient algorithms for mining inclusion dependencies", in *Advances in Database Technology — EDBT 2002*, C. S. Jensen, S. Šaltenis, K. G. Jeffery, J. Pokorny, E. Bertino, K. Böhn, and M. Jarke, Eds., Springer Berlin Heidelberg, 2002, pp. 464–476.

[15] J. Bauckmann, Z. Abedjan, H. Müller, and F. Naumann, "Discovering conditional inclusion dependencies", in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, Association for Computing Machinery, 2012, pp. 2094–2098.

[16] M. T. Gómez-López, R. M. Gasca, and J. M. Pérez-Álvarez, "Compliance validation and diagnosis of business data constraints in business processes", *Information Systems*, vol. 48, pp. 26–43, 2015.

[17] S. Ma, W. Fan, and L. Bravo, "Extending inclusion dependencies with conditions", *Theoretical Computer Science*, vol. 515, pp. 64–95, 2014.

[18] J. Visser, "Coupled transformation of schemas, documents, queries, and constraints", *Electronic Notes in Theoretical Computer Science*, vol. 200, no. 3, pp. 3–23, 2008.

[19] J. Garmany, J. Walker, and T. Clark, *Logical Database Design Principles*. Auerbach Publications, 1980, 200 pp.

[20] S. Lopes, J.-M. Petit, and F. Toumani, "Discovering interesting inclusion dependencies: Application to logical database tuning", *Information Systems*, vol. 27, no. 1, pp. 1–19, 2002.

[21] Y. Kaminsky, E. Pena, and F. Naumann, "Discovering similarity inclusion dependencies", *Proceedings of the ACM on Management of Data*, vol. 1, no. 1, pp. 1–24, 2023.

[22] M. Levene and G. Loizou, "Null inclusion dependencies in relational databases", *Information and Computation*, vol. 136, no. 2, pp. 67–108, 1997.

[23] M. Levene and G. Loizou, "The additivity problem for data dependencies in incomplete relational databases", in *Semantics in Databases*, vol. 1358, Springer, 1998, pp. 136–169.

[24] H. Köhler and S. Link, "Inclusion dependencies reloaded", in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, New York, NY, USA: Association for Computing Machinery, 2015, pp. 1361–1370.

[25] V. S. Zykin and S. V. Zykin, "Analysis of typed inclusion dependences with null values", *Automatic Control and Computer Sciences*, vol. 52, no. 7, pp. 638–646, 2018.

[26] S. Zykin, "Domains of functional dependences in databases", *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, vol. 22, no. 3, pp. 117–129, 2016, in Russian.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

THEORY OF DATA

Text classification by CEFR levels using machine learning methods and BERT language model

N. S. Lagutina[1], K. V. Lagutina[1], A. M. Brederman[1], N. N. Kasatkina[1]

DOI: 10.18255/1818-1015-2023-3-202-213

[1]P.G. Demidov Yaroslavl State University, 14 Sovetskaya str., Yaroslavl 150003, Russia.

MSC2020: 93A30, 68Q60
Research article
Full text in Russian

Received August 14, 2023
After revision August 25, 2023
Accepted August 30, 2023

This paper presents a study of the problem of automatic classification of short coherent texts (essays) in English according to the levels of the international CEFR scale. Determining the level of text in natural language is an important component of assessing students knowledge, including checking open tasks in e-learning systems. To solve this problem, vector text models were considered based on stylometric numerical features of the character, word, sentence structure levels. The classification of the obtained vectors was carried out by standard machine learning classifiers. The article presents the results of the three most successful ones: Support Vector Classifier, Stochastic Gradient Descent Classifier, LogisticRegression. Precision, recall and F-score served as quality measures. Two open text corpora, CEFR Levelled English Texts and BEA-2019, were chosen for the experiments. The best classification results for six CEFR levels and sublevels from A1 to C2 were shown by the Support Vector Classifier with F-score 67 % for the CEFR Levelled English Texts. This approach was compared with the application of the BERT language model (six different variants). The best model, bert-base-cased, provided the F-score value of 69 %. The analysis of classification errors showed that most of them are between neighboring levels, which is quite understandable from the point of view of the domain. In addition, the quality of classification strongly depended on the text corpus, that demonstrated a significant difference in F-scores during application of the same text models for different corpora. In general, the obtained results showed the effectiveness of automatic text level detection and the possibility of its practical application.

Keywords: natural language processing; text classification; CEFR; BERT

INFORMATION ABOUT THE AUTHORS

Nadezhda S. Lagutina	orcid.org/0000-0002-6137-8643. E-mail: lagutinans@gmail.com PhD, associate professor.
Ksenia V. Lagutina corresponding author	orcid.org/0000-0002-1742-3240. E-mail: lagutinakv@mail.ru PhD, associate professor.
Anastasya M. Brederman	orcid.org/0009-0003-1741-0604. E-mail: anastasyabrederman@mail.ru student.
Natalia N. Kasatkina	orcid.org/0000-0002-6757-9622. E-mail: ninet75@mail.ru PhD, associate professor.

Funding: This study was supported by YarSU Development Program until 2030, project No. GM-2023-123061600058-4 “Development of an automated system for the development of mediative competence in language education”.

For citation: N. S. Lagutina, K. V. Lagutina, A. M. Brederman, and N. N. Kasatkina, “Text classification by CEFR levels using machine learning methods and BERT language model”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 202-213, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

THEORY OF DATA

Классификация текстов по уровням CEFR с использованием методов машинного обучения и языковой модели BERT

Н. С. Лагутина[1], К. В. Лагутина[1], А. М. Бредерман[1], Н. Н. Касаткина[1]

DOI: 10.18255/1818-1015-2023-3-202-213

[1]Ярославский государственный университет им. П.Г. Демидова, ул. Советская, д. 14, г. Ярославль, 150003 Россия.

УДК 004.912
Научная статья
Полный текст на русском языке

Получена 14 августа 2023 г.
После доработки 25 августа 2023 г.
Принята к публикации 30 августа 2023 г.

В данной работе представлено исследование задачи автоматической классификации коротких связных текстов (эссе) на английском языке по уровням международной шкалы CEFR. Определение уровня текста на естественном языке является важной составляющей оценки знаний учащихся, в том числе для проверки открытых заданий в системах электронного обучения. Для решения этой задачи были рассмотрены векторные модели текста на основе стилометрических числовых характеристик уровня символов, слов, структуры предложения. Классификация полученных векторов осуществлялась стандартными классификаторами машинного обучения. В статье приведены результаты трёх наиболее успешных: Support Vector Classifier, Stochastic Gradient Descent Classifier, LogisticRegression. Оценкой качества послужили точность, полнота и F-мера. Для экспериментов были выбраны два открытых корпуса текстов CEFR Levelled English Texts и BEA-2019. Лучшие результаты классификации по шести уровням и подуровням CEFR от A1 до C2 показал Support Vector Classifier с F-мерой 67 % для корпуса CEFR Levelled English Texts. Этот подход сравнивался с применением языковой модели BERT (шесть различных вариантов). Лучшая модель bert-base-cased обеспечила значение F-меры 69 %. Анализ ошибок классификации показал, что большая их часть допущена между соседними уровнями, что вполне объяснимо с точки зрения предметной области. Кроме того, качество классификации сильно зависело от корпуса текстов, что продемонстрировало существенное различие F-меры в ходе применения одинаковых моделей текста для разных корпусов. В целом, полученные результаты показали эффективность автоматического определения уровня текста и возможность его практического применения.

Ключевые слова: автоматическая обработка текста; классификация текста; CEFR; BERT

ИНФОРМАЦИЯ ОБ АВТОРАХ

Надежда Станиславовна Лагутина	orcid.org/0000-0002-6137-8643. E-mail: lagutinans@gmail.com канд. физ.-мат. наук, доцент.
Ксения Владимировна Лагутина автор для корреспонденции	orcid.org/0000-0002-1742-3240. E-mail: lagutinakv@mail.ru канд. тех. наук, доцент.
Анастасия Михайловна Бредерман	orcid.org/0009-0003-1741-0604. E-mail: anastasyabrederman@mail.ru студент.
Наталья Николаевна Касаткина	orcid.org/0000-0002-6757-9622. E-mail: ninet75@mail.ru канд. пед. наук, доцент.

Финансирование: Исследование выполнено за счет средств Программы развития ЯрГУ до 2030 года, проект № GM-2023-123061600058-4 «Разработка автоматизированной системы развития медиативной компетенции в языковом образовании».

Для цитирования: N. S. Lagutina, K. V. Lagutina, A. M. Brederman, and N. N. Kasatkina, “Text classification by CEFR levels using machine learning methods and BERT language model”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 202-213, 2023.

Введение

Автоматизированная оценка эссе (automated essay scoring, AES) — это способ моделирования работы человека-эксперта в области языкознания и педагогики. Определение качества текста на естественном языке является важной составляющей оценки знаний учащихся, в том числе для проверки открытых заданий в системах электронного обучения [1, 2]. Кроме того, текст является одной из основ для коммуникации людей, что порождает необходимость выявления сложности текстов, их качества и возможности понимания целевой аудиторией [3].

Развитие систем AES началось с 60-х годов прошлого века и опиралось в первую очередь на правила грамматики. По мере развития информационных технологий в области компьютерной лингвистики исследователи добавляли анализ стиля текста, его структуры, связности на основе параметров текста различной степени сложности. Большинство работ в этой области используют статистические признаки, такие как функции «мешка слов» (Bag of Words, BoW), количество предложений и т. п. Однако качество текста во многом определяется его связностью, анализом семантики, но даже использование контекстных параметров типа word2vec не решает эту проблему и не повышает качество AES до должного уровня [4]. Новые возможности предоставляет развитие современных языковых моделей [5], в частности модель BERT [6]. Обзор современных работ показывает, что AES растущая область исследований с большим набором потенциально применимых методов, но все еще не зрелая, особенно в сфере практического применения [1].

Отдельным вопросом AES является способ и шкала оценки текстов. Исследователи используют системы на основе подсчета баллов [7], формулируют наборы проверяемых критериев [8], применяют существующие стандарты, описывающие уровни владения языком (общеевропейские компетенции владения иностранным языком CEFR, стандарты преподавания иностранных языков ACTFL, канадские критерии оценки уровня языка CLB, межведомственный круглый стол по вопросам языковой подготовки ILR). Значительная часть работ в образовательной сфере и области AES делит эссе по уровням CEFR [5, 9].

Авторы данной статьи рассмотрели задачу AES как задачу классификации текстов по трём уровням CEFR (A — начальный, B — средний и C — высокий) и шести подуровням (A1, A2, B1, B2, C1, C2) и поставили цель систематизировать и проанализировать результаты классификации с использованием стилометрических параметров текста и стандартных классификаторов машинного обучения и сравнить их с результатами применения языковых моделей BERT.

1. Аналогичные работы

Методы классификации и анализа текстов на естественном языке бурно развиваются в последнее десятилетие, поэтому при выборе инструментов для исследования и сравнения результатов авторы статьи сосредоточились на достижениях последних лет.

Авторы работы [10] использовали модель bert-base-uncased для классификации Кембриджской базы данных открытого языка (EFCAMDAT) и Кембриджского учебного корпуса по английскому языку (CLC-FCE) на пять уровней шкалы CEFR: от A1 до C1. Точность классификации (accuracy) достигла 61.7 ± 1.8 %. Исследователи отметили, что качество сильно зависит от размера корпуса для обучения.

В статье [11] учёные классифицировали по уровням CEFR отдельные предложения. Для сбора и разметки собственного корпуса из 17 000 английских предложений были привлечены высококвалифицированные преподаватели, что обеспечило качество данных. Каждому предложению ставился в соответствие числовой вектор на основе частот слов уровней CEFR, классификация по шести уровням осуществлялась с помощью косинусного сходства между векторами. Предложенный подход показал F-меру со средним значением 84.5 ± 0.7 %. Авторы сравнили этот результат с моделью

BERT, F-мера оказалась 82.5 ± 0.9 % и моделью «мешок слов» с классификатором SVM, где F-мера чуть больше 52 %.

Зависимость качества классификации от качества используемого корпуса данных отмечается в работе [12]. Авторы указывают на разнородность и несбалансированность текстов, собираемых среди изучающих английский язык как второй. Для преодоления этих проблем для моделирования текстов они предложили использовать набор параметров, ориентированных на уровень владения языком, а не на текстовые функции. Эксперименты были проведены в Международной корпусной сети азиатских изучающих английский язык (ICNALE) на наборе данных, включающем эссе от 2 800 авторов. Исследователи провели классификацию по уровням CEFR с помощью многослойного персептрона и линейной регрессии и получили среднюю F-меру 63 % и 43 % соответственно. Аналогичный эксперимент с корпусом EFCAMDAT показал значение F-меры 96 % и 83 %.

Параметры на основе частоты встречаемости различных языковых конструкций были использованы для моделирования текста в исследовании [13]. Эти конструкции, предложенные экспертами преподавателями и лингвистами, авторы назвали микросистемами. Для экспериментов использовались корпуса EFCAMDAT и CEFR-ASAG. Классификация полиномиальной логистической регрессией на шесть уровней CEFR показала среднюю точность (accuracy) 75 % и 95 % при отделении уровня начинающих A от продвинутых B, что может быть полезно для автоматического разделения учащихся на группы.

57 числовых характеристик использовались в статье [14]. Они разделены на четыре группы: меры синтаксической сложности, на основе анализа синтаксических деревьев; меры лексического богатства; n-граммы слов, для n от двух до пяти; колмогоровская сложность как мера количества информации в строке. Классификатором служила рекуррентная нейронная сеть, корпус данных EFCAMDAT. Качество классификации оказалось выше для начального и среднего уровней владения языком CEFR (от A1 до B2) с F-мерой в диапазоне от 73 % до 81 % по сравнению с уровнями C1 и C2, где F-мера упала до 61 % для уровня C1 и 42 % для C2. Матрица ошибок показала, что их большинство возникло в основном в смежных категориях.

Таким образом, для классификации уровней CEFR применяются разнообразные лексические параметры, отражающие специфику предметной области. Однако, авторы данной статьи не увидели системных исследований стилометрических характеристик текста для этой цели, хотя они вполне успешно используются для AES в целом [15]. Кроме того, очень перспективным методом решения задачи выглядит работа с разными моделями BERT. В пользу этого говорят результаты успешного применения BERT для задач оценки эссе [16] и сложности текста [17].

2. Метод автоматического определения уровня владения языком

2.1. Корпуса текстов

Для исследования уровня владения английским языком были взяты два открытых корпуса, тексты в которых размечены по международному стандарту CEFR.

Корпус CEFR Levelled English Texts содержит 1 494 текста из открытых источников The British Council, ESLFast и корпуса cnn-dailymail. Он опубликован на Kaggle (https://www.kaggle.com/datasets/amontgomerie/cefr-levelled-english-texts).

Второй корпус появился в рамках соревнования BEA-2019 [18], посвящённого поиску грамматических ошибок. Открытая часть корпуса содержит 3 350 текстов, 3 300 из которых размечены по шкале CEFR. Тексты были агрегированы из корпусов Cambridge English Write & Improve и LOCNESS.

Дополнительно для экспериментов корпуса объединялись в один набор текстов. Статистические данные о категориях текстов в корпусах представлены в таблице 1.

Оба корпуса имеют разные принципы и источники сборки, поэтому в качестве первичной обработки данных были рассчитаны базовые статистические параметры: среднее, модальное, ме-

Table 1. Number of texts of different CEFR levels in corpora

Таблица 1. Количество текстов различных уровней CEFR в корпусах

Корпус	Всего	A	A1	A2	B	B1	B2	C	C1	C2
CEFR Levelled English Texts	1 494	560	288	272	491	205	286	443	241	202
BEA-2019	3 300	1 430	585	845	1 100	631	469	770	483	287
Объединение корпусов	4 794	1 990	873	1 117	1 591	836	755	1 213	724	489

Table 2. Statistical indicators of the numbers of words in corpora

Таблица 2. Статистические показатели количества слов в корпусах

Корпус	Уровень	Среднее значение	Мода	Медиана	Макс.	Мин.
CEFR Levelled English Texts	**Всего**	**424.82**	**125**	**313**	**2 292**	**37**
CEFR Levelled English Texts	A1	97.94	123	106	307	37
CEFR Levelled English Texts	A2	232.63	107	291	1 222	70
CEFR Levelled English Texts	B1	415.34	309	310	1 628	100
CEFR Levelled English Texts	B2	509.83	299	342	2 668	97
CEFR Levelled English Texts	C1	701.42	610	706	1 669	153
CEFR Levelled English Texts	C2	708.93	518	678	2 292	115
BEA-2019	**Всего**	**188.09**	**194**	**176**	**1 612**	**31**
BEA-2019	A1	85.00	107	78	364	31
BEA-2019	A2	155.76	155	148	686	33
BEA-2019	B1	203.64	202	188	1 046	46
BEA-2019	B2	226.58	151	199	1 612	73
BEA-2019	C1	262.50	199	228	1 407	44
BEA-2019	C2	271.09	209	229	1 093	47

дианное, максимальное и минимальное количества слов (таблица 2), и количество предложений в корпусах (таблица 3).

На основе вышеуказанных данных можно отметить, что показатели корпуса BEA-2019, во-первых, в среднем выше соответствующих показателей корпуса CEFR Levelled English Texts, а во-вторых, характеризуются меньшей вариацией одноимённых индикаторов. В дальнейшем эта закономерность могла стать фактором влияния на результаты классификации.

2.2. Модели текста

Тексты моделировались как вектора числовых характеристик на основе нескольких моделей:

- характеристики уровня символов;
- характеристики уровня слов;
- характеристики уровня структуры;
- эмбеддинги на основе BERT.

Первые три модели основаны на статистических и лингвистических характеристиках текстов. Алгоритмы для их вычисления представлены в предыдущих работах авторов [19, 20].

В группу *характеристик уровня символов* вошли следующие шесть стилометрических характеристик:

Table 3. Number of sentences in texts of different CEFR levels in corpora

Таблица 3. Количество предложений в текстах различных уровней CEFR в корпусах

Корпус	Всего	A1	A2	B1	B2	C1	C2
CEFR Levelled English Texts	36 235	4 572	6 023	5 381	7 552	7 344	5 363
BEA-2019	35 171	3 109	7 366	7 288	6 050	7 044	4 314
Объединение корпусов	71 406	7 681	13 389	12 669	13 602	14 388	9 677

- общее количество символов в тексте;
- частоты букв латинского алфавита;
- частоты появления самых часто встречающихся знаков препинания (точка, занятая, двоеточие, точка с занятой, вопросительный знак, восклицательный знак, кавычки, скобки, тире);
- отношение общего количества букв латинского алфавита к общему количеству символов;
- отношение общего количества знаков препинания без учета точки к общему количеству символов;
- отношение общего количества цифр к общему числу символов.

Группа *характеристик уровня слов* была сформирована из 17 характеристик:

- количество слов в тексте;
- средняя длина слов в тексте;
- лексическое разнообразие — отношение числа разных лексем к общему числу слов;
- автосемантичность — отношение числа значащих слов (кроме служебных слов и местоимений) к общему числу слов;
- аналитичность — отношение числа служебных слов к общему числу слов;
- негация — отношение суммы отрицательных частиц к общему числу слов;
- субстантивность — отношение числа существительных к общему числу слов;
- отношение числа существительных во множественном числе к общему числу слов;
- глагольность — отношение числа всех глагольных форм к общему числу слов;
- отношение числа глаголов в прошедшем времени к общему числу слов;
- отношение числа глаголов в настоящем времени к общему числу слов;
- отношение числа модальных глаголов к общему числу слов;
- местоимённость — отношение местоимённых слов к общему числу слов;
- отношение числа личных местоимений к общему числу слов;
- адъектинность — отношение числа прилагательных к общему числу слов;
- отношение числа наречий к общему числу слов;
- отношение числа прилагательных и наречий в сравнительной форме к общему числу слов.

В группу *характеристик уровня структуры* документа вошли ещё четыре характеристики:

- количество предложений в тексте;
- средняя длина предложений;
- количество абзацев в тексте;
- частоты появления предложений, состоящих из определённого числа слов (от 1 до 39, 40-й признак включает в себя все остальные предложения).

Также вектора характеристик данных уровней конкатенировались, чтобы получить дополнительные модели-комбинации двух и трёх уровней.

Эмбеддинги на основе BERT строились на основе следующих моделей, открыто опубликованных на https://huggingface.co/:

- bert-base-cased [21] — классическая языковая модель BERT для английского языка, учитывающая регистр;
- bert-base-uncased — классическая языковая модель BERT для английского языка, не учитывающая регистр;
- bert-large-cased — увеличенная языковая модель BERT для английского языка, учитывающая регистр;
- distilbert-base-cased [22] — языковая модель, обученная на классической модели BERT и имеющая меньший размер;
- DeepPavlov/bert-base-cased-conversational — языковая модель на основе классической модели BERT, дообученная на текстах из социальных сетей и блогах;

- Intel/bert-base-uncased-mrpc — языковая модель на основе классической модели BERT, дообученная на корпусе GLUE MRPC.

BERT-эмбеддинги были выбраны, поскольку именно эта языковая модель в настоящее время достигает лучших результатов в различных задачах обработки текстов на естественном языке. Первые четыре модели из списка выше представляют собой вариации классического BERT, а пятая и шестая — BERT, дообученный на современных англоязычных интернет-текстах.

Таким образом, в исследовании сравнивались несколько векторных моделей текста: стилометрические трёх уровней вместе с их комбинациями и модели на основе шести типов эмбеддингов. Каждый текст представлялся как вектор числовых характеристик, вычисленных на основе одной из моделей.

2.3. Классификация текстов по уровню

После моделирования корпус текстов представлял собой матрицу векторов числовых характеристик. Каждому тексту (вектору) сопоставлялась категория, соответствующая одному из уровней владения языком. Классификация этих данных проводилась с помощью обучаемых методов двух типов: классификаторы машинного обучения и нейросетевые классификаторы.

Перед классификацией корпус текстов был разделён на обучающую, валидационную и тестовую выборки в пропорции 80 %/10 %/10 %. Валидационная выборка использовалась для подбора гиперпараметров.

Классификация векторов стилометрических характеристик проводилась на основе библиотеки машинного обучения scikit-learn (https://scikit-learn.org/stable/index.html). Базовые алгоритмы машинного обучения опираются исключительно на предоставленные данные из корпусов, из-за чего могут снабжать противоречивыми показателями и не давать стопроцентной точности предсказаний. Вместе с тем наиболее успешными в работе стали:

- SVC (Support Vector Classifier): классификатор, функционирующий на основе метода опорных векторов. Среди гиперпараметров выбраны: параметр регуляции — 1, параметр ядра — 0.1, функция ядра для преобразования пространства абзацев — линейное ядро (осуществляется линейная комбинация признаков).
- SGDClassifier (Stochastic Gradient Descent Classifier): классификатор, основанный на стохастическом градиентном спуске. Гиперпараметры: начальное состояние генератора случайных чисел — 42, параметр регуляции — 0.001, коэффициент смешивания регуляторов L1 и L2 — 0.75, максимальное число итераций градиентного спуска — 10 000.
- LogisticRegression: классификатор, который использует логистическую функцию для моделирования вероятности принадлежности точки к определенному классу. Базовыми аргументами стали: обратный коэффициент регуляции — 1, максимальное количество итераций — 100, функция регуляции — гребневая регрессия, алгоритм оптимизации — метод Ньютона-Кона.

Для классификации результатов BERT-моделей использовался однослойный перцептрон, объединённый в общую нейронную сеть с трансформером для языковой модели. Гиперпараметры были выбраны следующие: функция активации в выходном слое — линейная, оптимизатор — Adam, размер батча — 5.

Предсказания классификаторов оценивались при помощи общепринятых метрик: точности, полноты и F-меры.

3. Эксперименты

Тексты классифицировались несколькими способами: на все шесть уровней CEFR (A1, A2, B1, B2, C1, C2), на три уровня CEFR (A, B и C) и на пары подуровней A1 и A2, B1 и B2, C1 и C2. Первые два варианта позволяют оценить эффективность моделей в решении основной задачи исследования, а третий нужен для того, чтобы проанализировать особенности распознавания отдельных уровней.

Table 4. Text classification by 6 levels using machine learning

Таблица 4. Классификация текстов на 6 уровней с помощью машинного обучения

Корпус	Классификатор	Точность	Полнота	F-мера
CEFR Levelled English Texts	Support Vector Classifier	**68**	**68**	**67**
CEFR Levelled English Texts	Stochastic Gradient Descent Classifier	64	66	64
CEFR Levelled English Texts	Logistic Regression	68	68	67
BEA-2019	Support Vector Classifier	40	40	39
BEA-2019	Stochastic Gradient Descent Classifier	38	40	37
BEA-2019	Logistic Regression	**40**	**41**	**39**
Объединение корпусов	Support Vector Classifier	47	47	46
Объединение корпусов	Stochastic Gradient Descent Classifier	42	45	42
Объединение корпусов	Logistic Regression	46	46	45

Table 5. Text classification by 6 levels with BERT

Таблица 5. Классификация текстов на 6 уровней с BERT

Корпус	Модель	Точность	Полнота	F-мера
CEFR Levelled English Texts	bert-base-cased	**69**	**69**	**69**
CEFR Levelled English Texts	bert-base-uncased	67	64	64
CEFR Levelled English Texts	bert-large-cased	68	66	67
CEFR Levelled English Texts	distilbert-base-cased	65	62	62
CEFR Levelled English Texts	bert-base-cased-conversational	69	68	68
CEFR Levelled English Texts	Intel/bert-base-uncased-mrpc	63	61	61
BEA-2019	bert-base-cased	47	42	42
BEA-2019	bert-large-cased	**51**	**48**	**49**
BEA-2019	bert-base-cased-conversational	49	47	47
Объединение корпусов	bert-base-cased	51	50	50
Объединение корпусов	bert-base-cased-conversational	46	47	46

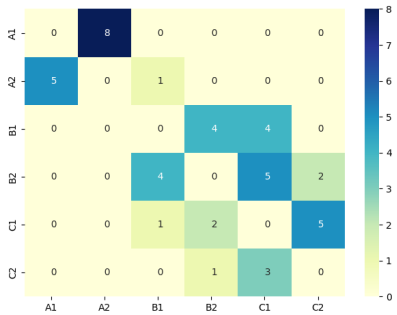
Результаты классификации текстов на шесть уровней владения языком с помощью алгоритмов машинного обучения получились невысокими. Результаты экспериментов приведены в таблице 4. Наиболее успешно с работой справился классификатор логистической регрессии (LogisticRegression). При этом классификация текстов из корпуса CEFR Levelled English Texts проходила на порядок успешнее классификации смежных корпусов: 67 % F-меры по сравнению с 39 и 45 % у корпусов BEA-2019 и объединённого соответственно.

Большую роль в успехе работы алгоритмов играли принципы векторизации данных: чем больше нюансов языка и лингвистических особенностей учитывалось при расчёте числовых параметров, тем выше были итоги классификации. Подобная закономерность указывает на наличие в языке неявных, но формализуемых критериев.

Расширенная классификация текстов при помощи BERT даёт показатели на 1–3 % выше показателей для алгоритмов машинного обучения. Результаты экспериментов BERT приведены в таблице 5. Лучше всего разделяются на категории тексты из корпуса CEFR Levelled English Texts при помощи классической модели BERT: 69 % F-меры. Использование вариаций данной языковой модели, в том числе большей по размеру и дообученных версий не даёт улучшения качества.

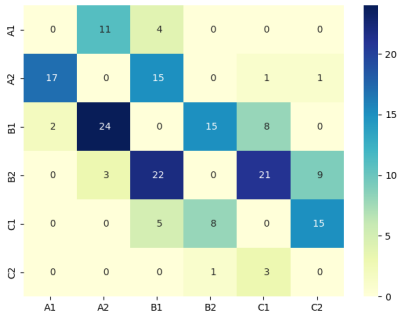
Классификация текстов BEA-2019 выполняется с F-мерой не более 49 %, объединённый корпус также классифицируется в лучшем случае с метриками около 50 %. В таблице 5 приведены BERT-модели, дающие лучшие результаты экспериментов.

На рис. 1 представлены матрицы ошибок для классификации обоих корпусов с помощью bert-base-cased. На рис. 2 представлены соответствующие матрицы ошибок для классификации с по-



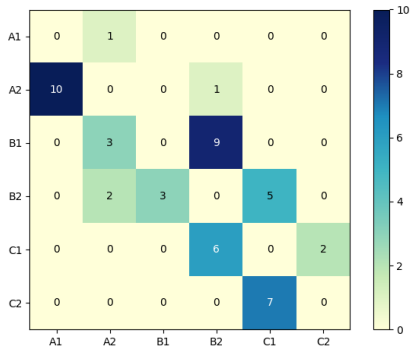
a)

Fig. 1. Number of errors of the model bert-base-cased for the corpus a) CEFR Levelled English Texts, b) BEA-2019



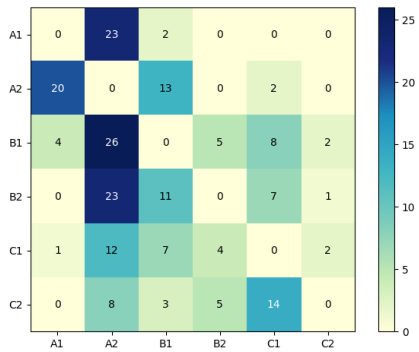
b)

Рис. 1. Количество ошибок модели bert-base-cased для корпуса a) CEFR Levelled English Texts, b) BEA-2019



a)

Fig. 2. Number of errors of the classifier Logistic Regression for the corpus a) CEFR Levelled English Texts, b) BEA-2019



b)

Рис. 2. Количество ошибок классификатора LogisticRegression для корпуса a) CEFR Levelled English Texts, b) BEA-2019

мощью метода опорных векторов. По вертикали указаны корректные уровни владения языком для текстов, а по горизонтали — уровни, за которые эти тексты были приняты ошибочно. На главных диагоналях условно отмечены нули. Видна общая закономерность: классификатор чаще путает между собой соседние уровни, а тексты, сильно отличающиеся по уровню владения языком, отделяются друг от друга хорошо.

При работе с методом опорных векторов меньше всего ошибок оказалось для подуровня A1. Чаще всего с ним путается подуровень A2. В то время как при работе с BERT меньше всего ошибок оказалось для подуровня C2. Подуровень C1 путается чаще всего с C2. Соответствующие матрицы ошибок у рассматриваемых классификаторов оказались обратно симметричны.

Подчеркивает последнее утверждение и закономерность исключений в корпусе CEFR Levelled English Texts: для BERT подуровни A смешиваются с соседними реже, чем подуровни C; для классификаторов, основанных на стилометрических характеристиках, подуровни A смешиваются с соседними чаще, чем подуровни C.

В корпусе CEFR Levelled English Texts ошибки сосредоточены в первую очередь в классификации пар подуровней. Подуровни A1 и A2 смешиваются практически только между собой, C1 и C2 чаще принимаются друг за друга, чем за другие подуровни. Подуровни B1 и B2 тоже путаются между собой, но как и C1, они также часто ошибочно классифицируются.

В корпусе BEA-2019 ошибки тоже концентрируются в области подуровней, однако классификаторы машинного обучения справляются с выделением уровня B и его подуровнями явно хуже. Фактически они не способны к корректному выделению характеристик ступени B.

Table 6. Text classification by 3 levels using machine learning

Таблица 6. Классификация текстов на 3 уровня с помощью машинного обучения

Корпус	Классификатор	Точность	Полнота	F-мера
CEFR Levelled English Texts	Support Vector Classifier	**90**	**90**	**90**
BEA-2019	Logistic Regression	63	64	63
Объединение корпусов	Support Vector Classifier	69	68	68

Table 7. Text classification by 3 levels with BERT

Таблица 7. Классификация текстов на 3 уровня с BERT

Корпус	Модель	Точность	Полнота	F-мера
CEFR Levelled English Texts	bert-base-cased	100	100	100
BEA-2019	bert-base-cased	100	100	100
Объединение корпусов	bert-base-cased	100	100	100

Table 8. Text classification by sublevels using machine learning

Таблица 8. Классификация текстов на подуровни с помощью машинного обучения

Корпус	Подуровни	Точность	Полнота	F-мера
CEFR Levelled English Texts	A1 и A2	73	71	70
CEFR Levelled English Texts	B1 и B2	75	73	72
CEFR Levelled English Texts	C1 и C2	**82**	**82**	**82**
BEA-2019	A1 и A2	**74**	**74**	**74**
BEA-2019	B1 и B2	65	64	59
BEA-2019	C1 и C2	59	58	59
Объединение корпусов	A1 и A2	**81**	**81**	**81**
Объединение корпусов	B1 и B2	69	66	63
Объединение корпусов	C1 и C2	75	71	66

Table 9. Text classification by sublevels with BERT

Таблица 9. Классификация текстов на подуровни с BERT

Корпус	Подуровни	Точность	Полнота	F-мера
CEFR Levelled English Texts	A1 и A2	**79**	**78**	**78**
CEFR Levelled English Texts	B1 и B2	73	73	72
CEFR Levelled English Texts	C1 и C2	73	73	73
BEA-2019	A1 и A2	**76**	**75**	**75**
BEA-2019	B1 и B2	63	61	60
BEA-2019	C1 и C2	55	55	49

Объединение данных на более общие уровни владения языком: A, B и C, сильнее отражает разрыв в качестве работы классификаторов. При работе с алгоритмами машинного обучения максимальный показатель F-меры достигает 90 % для корпуса CEFR Levelled English Texts. Для корпуса BEA-2019 показатели так и остаются низкими: F-мера 63 %. Показатели отражены в таблице 6.

Идентичное обобщение показателей при использовании любой BERT-модели позволяет достичь 100 % качества различения уровней. Это иллюстрируется таблицей 7. Таким образом, BERT-модели определяют уровень языка A, B или C однозначно, что сочетается с результатом анализа ошибок.

Классификация на подуровни моделями машинного обучения представлена в таблице 8. В корпусе CEFR Levelled English Texts лучше всего разделяются уровни C1 и C2: 82 % F-меры, а в корпусе BEA-2019 A1 и A2: 74 % F-меры. Качество деления текстов на подуровни начальной ступени в обоих корпусах практически идентично. Сильное различие в показателях F-меры не даёт конкретной зависимости качества работы алгоритмов от объёма данных.

Классификация каждого из трёх уровней владения языком A, B и C на подуровни 1 и 2 моделью bert-base-cased представлена в таблице 9. Лучше всего разделяются уровни A1 и A2: 75–77 % F-меры. Различение уровней B1 и B2, C1 и C2 по качеству близко к результатам классификации на шесть уровней. Следовательно, ошибки именно в классификации данных категорий приводят к невысокому качеству мультиклассовой классификации из первых экспериментов.

Дополнительно был проведён анализ 20 эссе, написанных студентами ЯрГУ им. П. Г. Демидова. Модель bert-base-cased спрогнозировала уровень A для 19-ти работ и C для одной работы. Верификация результатов экспертом дала формальную оценку F-меры 91.4 %. Этот эксперимент показывает обнадёживающие перспективы практического применения системы автоматического анализа эссе на основе модели BERT.

Заключение

В данной статье рассмотрены векторные модели текста на основе числовых характеристик уровня символов, слов, структуры предложения, а так же эмбеддингов BERT. Систематизация и анализ результатов позволяют сделать несколько выводов. Во-первых, первенство по качеству классификации текстов по уровням CEFR занимает модель BERT. При прогнозировании трёх уровней A, B и C F-мера достигает 100 %, при классификации на шесть от A1 до C2 также наблюдается преимущество перед стандартными методами машинного обучения. Во-вторых, основные ошибки классификации допускаются между соседними уровнями, что вполне объяснимо с точки зрения предметной области. В-третьих, качество классификации сильно зависит от корпуса текстов, что показывает существенное различие F-меры для одинаковых моделей текста, например, модель bert-base-cased обеспечивает 69 % для корпуса CEFR Levelled English Texts и всего 49 % для BEA-2019.

Рассмотренные числовые характеристики не исчерпывают все современные доступные средства моделирования текста. В качестве перспективы исследований авторы хотели бы обратить внимание на разработку дополнительных параметров, отражающих сложные лингвистические параметры структуры текста и его семантики. Эта задача требует привлечения экспертов в области языкознания и педагогики. Так же интересно получить результат классификации для комбинации таких характеристик с более простыми, в том числе с различными эмбеддингами.

References

[1] E. del Gobbo, A. Guarino, B. Cafarelli, L. Grilli, and P. Limone, “Automatic evaluation of open-ended questions for online learning. A systematic mapping”, *Studies in Educational Evaluation*, vol. 77, p. 101 258, 2023.

[2] N. Galichev and P. Shirogorodskaya, “Problema avtomaticheskogo izmereniya slozhnyh konstruktov cherez otkrytye zadaniya”, in *HXI Mezhdunarodnaya nauchno-prakticheskaya konferenciya molodyh issledovatelej obrazovaniya*, in Russian, Novosibirskij gosudarstvennyj pedagogicheskij universitet, 2022, pp. 695–697.

[3] L. E. Adamova, O. Surikova, I. G. Bulatova, and O. O. Varlamov, “Application of the mivar expert system to evaluate the complexity of texts”, *News of the Kabardin-Balkar scientific center of RAS*, no. 2, pp. 11–29, 2021.

[4] D. Ramesh and S. K. Sanampudi, “An automated essay scoring systems: A systematic literature review”, *Artificial Intelligence Review*, vol. 55, no. 3, pp. 2495–2527, 2022.

[5] K. P. Yancey, G. Laflair, A. Verardi, and J. Burstein, “Rating short L2 essays on the CEFR scale with GPT-4”, in *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, 2023, pp. 576–584.

[6] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, “A survey on text classification algorithms: From text to predictions”, *Information*, vol. 13, no. 2, p. 83, 2022.

[7] V. Ramnarain-Seetohul, V. Bassoo, and Y. Rosunally, "Similarity measures in automated essay scoring systems: A ten-year review", *Education and Information Technologies*, vol. 27, no. 4, pp. 5573–5604, 2022.

[8] P. Yang, L. Li, F. Luo, T. Liu, and X. Sun, "Enhancing topic-to-essay generation with external commonsense knowledge", in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 2002–2012.

[9] N. N. Mikheeva and E. V. Shulyndina, "Features of training written Internet communication in a non-linguistic university", *Tambov University Review. Series: Humanities*, vol. 28, no. 2, pp. 405–414, 2023.

[10] V. J. Schmalz and A. Brutti, "Automatic assessment of English CEFR levels using BERT embeddings", in *Proceedings of the Eighth Italian Conference on Computational Linguistics*, 2021.

[11] Y. Arase, S. Uchida, and T. Kajiwara, "CEFR-based sentence difficulty annotation and assessment", in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 6206–6219.

[12] R. Jalota, P. Bourgonje, J. Van Sas, and H. Huang, "Mitigating learnerese effects for CEFR classification", in *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, 2022, pp. 14–21.

[13] T. Gaillat, A. Simpkin, N. Ballier, B. Stearns, A. Sousa, M. Bouyé, and M. Zarrouk, "Predicting CEFR levels in learners of English: The use of microsystem criterial features in a machine learning approach", *ReCALL*, vol. 34, no. 2, pp. 130–146, 2022.

[14] E. Kerz, D. Wiechmann, Y. Qiao, E. Tseng, and M. Ströbel, "Automated classification of written proficiency levels on the CEFR-scale through complexity contours and RNNs", in *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, 2021, pp. 199–209.

[15] Y. Yang and J. Zhong, "Automated essay scoring via example-based learning", in *Web Engineering*, Springer, 2021, pp. 201–208.

[16] E. Mayfield and A. W. Black, "Should you fine-tune BERT for automated essay scoring?", in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2020, pp. 151–162.

[17] J. M. Imperial, "BERT embeddings for automatic readability assessment", in *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, 2021, pp. 611–618.

[18] C. Bryant, M. Felice, Ø. E. Andersen, and T. Briscoe, "The BEA-2019 shared task on grammatical error correction", in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, Association for Computational Linguistics, 2019, pp. 52–75.

[19] K. V. Lagutina and A. M. Manakhova, "Automated search and analysis of the stylometric features that describe the style of the prose of 19th–21st centuries", *Automatic Control and Computer Sciences*, vol. 55, no. 7, pp. 866–876, 2021.

[20] A. M. Manakhova and N. S. Lagutina, "Analysis of the impact of the stylometric characteristics of different levels for the verification of authors of the prose", *Modeling and Analysis of Information Systems*, vol. 28, no. 3, pp. 260–279, 2021, in Russian.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4171–4186.

[22] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*, 2020. arXiv: 1910.01108 [cs.CL].

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Logic for reasoning about bugs in loops over data sequences (IFIL)

D. A. Kondratyev[1]

DOI: 10.18255/1818-1015-2023-3-214-233

[1]A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, 6, Acad. Lavrentjev pr., Novosibirsk 630090, Russia.

MSC2020: 68Q60
Research article
Full text in English

Received May 29, 2023
After revision June 16, 2023
Accepted June 20, 2023

Classic deductive verification is not focused on reasoning about program incorrectness. Reasoning about program incorrectness using formal methods is an important problem nowadays. Special logics such as Incorrectness Logic, Adversarial Logic, Local Completeness Logic, Exact Separation Logic and Outcome Logic have recently been proposed to address it. However, these logics have two disadvantages. One is that they are based on under-approximation approaches, while classic deductive verification is based on the over-approximation approach. One the other hand, the use of the classic approach requires defining loop invariants in a general case. The second disadvantage is that the use of generalized inference rules from these logics results in having to prove too complex formulas in simple cases. Our contribution is a new logic for solving these problems in the case of loops over data sequences. These loops are referred to as finite iterations. We call the proposed logic the Incorrectness Finite Iteration Logic (IFIL). We avoid defining invariants of finite iterations using a symbolic replacement of these loops with recursive functions. Our logic is based on special inference rules for finite iterations. These rules allow generating formulas with recursive functions corresponding to finite iterations. The validity of these formulas may indicate the presence of bugs in the finite iterations. This logic has been implemented in a new version of the C-lightVer system for deductive verification of C programs.

Keywords: deductive verification; Hoare logic; bug localization; program incorrectness; loop invariant; finite iteration; C-lightVer; ACL2

INFORMATION ABOUT THE AUTHORS

Dmitry A. Kondratyev
corresponding author
orcid.org/0000-0002-9387-6735. E-mail: apple-66@mail.ru
researcher, PhD in Computer Science.

For citation: D. A. Kondratyev, "Logic for reasoning about bugs in loops over data sequences (IFIL)", *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 214-233, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Логика для суждений об ошибках в циклах над последовательностями данных (IFIL)

Д. А. Кондратьев[1]

DOI: 10.18255/1818-1015-2023-3-214-233

[1]Институт систем информатики им. А.П. Ершова Сибирского отделения Российской академии наук, 630090, Российская Федерация, г. Новосибирск, проспект Академика Лаврентьева, 6.

УДК 004.052.42
Научная статья
Полный текст на английском языке

Получена 29 мая 2023 г.
После доработки 16 июня 2023 г.
Принята к публикации 20 июня 2023 г.

Классическая дедуктивная верификация не ориентирована на доказательство некорректности программ. Доказательство некорректности программ с помощью формальных методов является актуальной задачей в настоящее время. Специальные логики, такие как Incorrectness Logic, Adversarial Logic, Local Completeness Logic, Exact Separation Logic и Outcome Logic, были недавно предложены для решения данной задачи. Но у данных логик имеется два недостатка. Во-первых, в данных логиках используются подходы, основанные на нижней аппроксимации, тогда как в классической дедуктивной верификации используется подход, основанный на верхней аппроксимации. С другой стороны, использование классического подхода требует в общем случае задания инвариантов циклов. Во-вторых, использование правил вывода для программных конструкций в их самом общем виде приводит к необходимости доказательства сложных формул в простых ситуациях. Нашим результатом, представленным в данной статье, является новая логика для решения данных проблем в случае циклов над последовательностями данных. Такая циклы мы называем финитными итерациями. Предложенную логику мы называем логикой для суждений о некорректности финитных итераций (IFIL). Мы избегаем задания инвариантов финитных итераций с помощью символической замены в условиях корректности переменных таких циклов применениями рекурсивных функций. Наша логика основана на специальных правилах вывода для финитных итераций. Эти правила позволяют выводить формулы с применениями рекурсивных функций, соответствующих финитным итерациям. Истинность этих формул может означать наличие ошибок в финитных итерациях. Данная логика была реализована в новой версии программной системы C-lightVer для дедуктивной верификации программ на языке C.

Ключевые слова: дедуктивная верификация; логика Хоара; локализация ошибок; некорректность программ; инвариант цикла; финитная итерация; C-lightVer; ACL2

ИНФОРМАЦИЯ ОБ АВТОРАХ

Дмитрий Александрович Кондратьев
автор для корреспонденции

orcid.org/0000-0002-9387-6735. E-mail: apple-66@mail.ru
научный сотрудник, кандидат физ.-мат. наук.

Для цитирования: D. A. Kondratyev, “Logic for reasoning about bugs in loops over data sequences (IFIL)”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 214-233, 2023.

Introduction

Deductive verification allows reasoning about program correctness [1]. Classic deductive verification is based on Hoare Logic (HL) [2–4]. Hoare Logic for a particular programming language contains a set of correct inference rules and axioms for all programming constructs. This set is referred to as the axiomatic semantics of programming language. Verification conditions (VC) are the result of the application of inference rules to an annotated program. The validity of the verification conditions means the correctness of the annotated program.

Classic deductive verification is not focused on reasoning about program incorrectness. Reasoning about incorrectness using formal methods is an important task nowadays [5, 6]. Special logics such as Incorrectness Logic (IL) [6–8], Adversarial Logic (AL) [9], Local Completeness Logic (LCL) [10, 11], Exact Separation Logic (ESL) [12], Outcome Logic (OL) [13] and Hyper Hoare Logic (HHL) [14] have recently been proposed to address it. However, these logics have two disadvantages. One is that they are based on the under-approximation approach, while classic deductive verification is based on the over-approximation approach [12]. The disadvantage of the under-approximation approach is the following inference method: for correctness reasoning, you have to forget information as you go along a path, but you must remember all the paths; for incorrectness reasoning, you must remember information as you go along a path, but you have to forget some of the paths [7]. One the other hand, the use of the classic approach requires defining loop invariants in a general case. Let us note that loop invariant problem can be solved in the cases of certain kinds of loops [15–17]. The second disadvantage is that the use of generalized inference rules from these logics results in having to prove too complex formulas in simple cases [6].

We have proposed a new logic to address these problems in the case of loops over data sequences. The development and implementation of our logic is based on the use of the following existing methods and tools:

- **The core of Hoare logic** [2–4]. The inference rules in our logic correspond to the classic form of inference rules proposed in Hoare logic. Our logic may be considered as a special version of Hoare logic for reasoning about the incorrectness of finite iterations.
- **The symbolic method of verification of finite iterations** [17]. This method is applied to a special kind of loop, finite iterations. The core of this method is a symbolic replacement of finite iterations with special recursive functions. This method allows us to avoid defining invariants in the case of finite iterations.
- **The memory model of two subsets of C programming language, C-light and C-kernel** [18, 19]. This semantics uses the memory model based on the *MeM* and *MD* functions. *MeM* maps an object's name to its address, and *MD* maps an object's address to its value. This memory model is insufficient for reasoning about low-level memory operations, but allows proving properties of simple programs with pointers. However, the other part of C-light and C-kernel semantics is excessive for us; for example, so are special functions for modelling types or the inference rule for `goto` statement.
- **The mixed axiomatic semantics method** [20]. The goal of this method is to simplify verification conditions. This method is based on context-based inference rules. For example, many C variables are Pascal-like variables, i. e., the address-of and dereference operators are not applied to them. For the "Pascal" context, semantics based on a simpler schema including only one map of variable names to values is used.
- **The C-lightVer tool** [21, 22]. This is a system for C program deductive verification. The main advantage of this system is the implementation of the symbolic method of verification of finite iterations.
- **The ACL2 theorem prover** [23]. The ACL2 tool is used as a theorem prover in the C-lightVer system. Applicative Common Lisp (ACL) is the input language of the ACL2 system. Thus, C-lightVer system generates verification conditions written in ACL. The advantage of ACL2 system is a special logic based

on computable recursive functions. It allows ACL2 to automatize proving verification conditions with the use of recursive functions corresponding to finite iterations.

- **The algorithm of the generation of recursive functions corresponding to loops** [21, 22, 24]. This algorithm is based on a translation of the loop body to the definition of a recursive function written in the Applicative Common Lisp language. This algorithm was implemented in the C-lightVer system.
- **Strategies for proving properties that may indicate possible errors** [21, 22]. Our logic has been inspired by these strategies. These strategies are checking the validity of certain properties of finite iterations. The validity of these properties may indicate the presence of bugs in the input annotated program. The properties of the loops are generated as formulas with recursive functions corresponding to finite iterations. The proven formulas are added to the underlying theory as lemmas about the loops. Some lemmas may indicate the presence of bugs in the loops. These lemmas may be considered as unsafe properties of loops. The following strategies have been suggested:
  1. Try to prove property that checks whether `break` is always executed at the first loop iteration.
  2. Try to prove the property that checks whether assignments to array elements in the loop body exist and array elements after the loop execution are identical to the array elements before the loop execution. These assignment statements may be never be used in this case.

  These strategies generate implications where the premises are the loop preconditions and the conclusions are the properties of the recursive functions corresponding to finite iterations. However, this approach has two disadvantages:
  1. The user of the verification system should define a loop precondition such that it is sufficient to prove the properties of the finite iterations.
  2. Only two loop properties are checked using these strategies.

  Our logic is focused on the solution of these problems. First, our logic allows us to obtain loop preconditions using inference rules. Second, our logic allows us to define more general properties of the loops.
- **Strongest postcondition calculus** [2]. This approach is used to transform precondition of a given program to generate the strongest postcondition of this program. This approach allows us to obtain the loop precondition in our logic.

Our contribution is a logic for reasoning about bugs in loops over data sequences. This logic was implemented in the new version of the C-lightVer system.

This paper has the following structure. Preliminary information is provided in Section 1. The contribution of this paper is described in Section 2. The experiment demonstrating the application of our logic is described in Section 3.

Related works. The idea of Partial Incorrectness Logic (PIL) has been presented in the paper [25]. Let us note that Partial Incorrectness Logic is based on the same inference method (strongest postcondition calculus) as our logic. However, Partial Incorrectness Logic is applied to nondeterministic programs whereas our logic is applied to deterministic programs. Besides the aforementioned Incorrectness Logic (IL) [6–8], Adversarial Logic (AL) [9], Local Completeness Logic (LCL) [10, 11], Exact Separation Logic (ESL) [12], Outcome Logic (OL) [13] and Hyper Hoare Logic (HHL) [14], there are more practical approach to finding bugs using formal methods. The use of a counterexample generated by an SMT solver for error localization was described in [26]. However, analysis of the counterexample can be fairly complicated, which was demonstrated in [27]. Constrained Horn Clauses (CHCs) [28] allow reasoning about program properties; however, this approach requires defining special proving strategies in the case of real-world programs. Bounded verification described in [29] is based on loop unrolling without using loop invariants. But efficiency of this strategy depends on how many iterations are chosen to be unrolled. The approach [30] based on the deductive verification of the program with a mutation in the conditions of the `if` statements and `while` loops was

implemented in the Frama-C tool [31]. However, this approach requires that loop invariants to be defined. Model checking based on k-induction has been implemented in the ESBMC tool [32]. However, the efficiency of this approach depends on finding inductive invariants. Model checking based on counterexample-guided abstraction refinement (CEGAR) has been implemented in the CPAchecker tool [33]. However, the efficiency of this approach depends on the efficiency of a reachability analysis. The approach based on symbolic execution has been implemented, for example, in the CPA-SymExec tool [34] and in the KLEE tool [35]. However, this approach depends on the efficiency of constraint solvers in reasoning about path feasibility.

1. Existing methods and tools that we use to develop and implement our logic

To develop and implement our logic, we used the following methods and tools: the core of Hoare logic [2–4], the symbolic method of verification of finite iterations [17], the memory model of two subsets of C programming language (C-light and C-kernel) [18, 19], the mixed axiomatic semantics method [20], the C-lightVer tool [21, 22], the ACL2 theorem prover [23], the algorithm of generation of recursive functions corresponding to loops [21, 22], strategies for proving properties that may indicate possible errors [21, 22] and the strongest postcondition calculus [2].

1.1. The core of Hoare logic

Deductive program verification is applied to the Hoare triple. The Hoare triple has the following form:

$$\{P\}\ S\ \{Q\},$$

where

- P is the precondition (logical formula);
- S is the program (sequence of program statements);
- Q is the postcondition (logical formula).

Deductive program verification is an automatic derivation of valid (partially correct) Hoare triples. The partial correctness [2–4] of the Hoare triple means that if the precondition is true before the execution of a program fragment and if its execution terminates, then the postcondition is true upon its completion.

The inference rule has the following structure:

$$\frac{\psi_1, \ldots, \psi_n}{\varphi},$$

where

- ψ_1,…,ψ_n are premises (Hoare triples and logical formulas);
- φ is the conclusion (Hoare triple).

This notation means that φ is derived from ψ_1,…,ψ_n. As an example, let us consider the classic inference rule for the *while* loop:

$$\frac{\{P\}\ \textbf{prog};\ \{I\},\ \{I \wedge B\}\ \textbf{S}\ \{I\},\ I \wedge \neg B \rightarrow Q}{\{P\}\ \textbf{prog};\ \textbf{while B inv I do S}\ \{Q\}},$$

where I is the loop invariant.

It is necessary to use induction to derive the Hoare triple for the *while* loop. The induction statement in this case is called the loop invariant: this statement is true before the loop execution, true after each loop iteration, and ensures the correctness of loop exit. If the loop has a general form, it is necessary to define the loop invariant.

The syntax-driven axiomatic system (i. e. the one that contains inference rules for all syntax constructs of the programming language) is called Hoare logic or axiomatic semantics.

1.2. The symbolic method of verification of finite iterations

Given that $memb(S)$ denotes the multiset of elements of a data sequence S and $empty(S) = true$ if $|memb(S)| = 0$, let us define two functions:

1. $choo(S)$ returns an arbitrary element of $memb(S)$, if $\neg empty(S)$.
2. $rest(S) = S'$, where $memb(S') = memb(S) \setminus \{choo(S)\}$, if $\neg empty(S)$.

A finite iteration corresponds to the form:

$$\textbf{for x in S do v} := \textbf{body(v, x) end},$$

where

- S is the data sequence;
- x is the variable of type "element of S";
- v is the tuple of the loop variables excluding x;
- $body$ represents the loop body which does not alter x and terminates for every $x \in S$.

Let v_0 denote the initial values of variables from v. Let us define replacement operation $rep(v, S, body)$ for this loop:

1. if $empty(S)$, then $rep(v_0, S, body) = v_0$.
2. if $\neg empty(S)$, then

$$rep(v_0, S, body) = body(rep(v_0, rest(S), body), choo(S)).$$

The following inference rule has been suggested for a finite iteration:

$$\frac{\{P\}\ \textbf{prog};\ \{Q(v \leftarrow rep(v, S, body))\}}{\{P\}\ \textbf{prog};\ \textbf{for x in S do v} := \textbf{body(v, x) end}\ \{Q\}},$$

where $\leftarrow$ denotes a simultaneous substitution.

This method allows us to avoid defining invariants in the case of loops corresponding to finite iterations [17].

1.3. The memory model of two subsets of C programming language, C-light and C-kernel

The C-light language [18] is a representative subset of C. The operational semantics was developed for the C-light language. The memory model based on the *MeM* and *MD* functions is used in this semantics. *MeM* maps an object's name to its address, and *MD* maps an object's address to its value.

The *upd* operation allows us to create a new *MD* map when the memory state changes. Let us define the value of the expression $upd(MD,\ addr,\ val)$, where *MD* is an $address \to value$ map, *addr* is an address and *val* is a value. If *MD* contains an $(adr\ val')$ pair, where val' is some value, then $upd(MD,\ addr,\ val)$ differs from *MD* in that ih has $(adr\ val)$ instead of $(adr\ val')$. If *addr* is not in the range of *MD*, then $upd(MD,\ addr,\ val)$ differs from *MD* in that an $(adr\ val)$ pair is added to it.

Let us consider axioms about *MeM* and *MD*:

1) $MD(NULL) = void$;
2) $MeM(obj) \neq NULL$;
3) $upd(MD,\ NULL,\ val) = MD$;
4) $upd(MeM,\ obj,\ NULL) = MeM$;
5) $delete(MD, NULL) = MD$;
6) $(upd(MD,\ addr,\ val))(addr) = val$;
7) $(upd(MD, adr_1, val))(adr_2) = MD(adr_2)$
   $if\ adr_1 \neq adr_2$;
8) $upd(MD,\ MeM(obj),\ MD(MeM(obj))) = MD$;
9) $upd(MeM,\ obj\ MeM(obj)) = MeM$;

10) $(upd(MeM, obj, addr))(obj) = addr$;
11) $(upd(MeM, obj_1, adr))(obj_2) = MeM(obj_2)$
    $if\ obj_1 \neq obj_2$;
12) $(delete(MD, addr))(addr) = void$;
13) $(delete(MD, adr_1))(adr_2) = MD(adr_2)$
    $if\ adr_1 \neq adr_2$;
14) $(delete(MeM, obj))(obj) = void$;
15) $(delete(MeM, obj_1))(obj_2) = MeM(obj_2)$
    $if\ obj_1 \neq obj_2$;
16) $delete(upd(MD, addr, val), addr) = MD$;
17) $delete(upd(MeM, obj, addr), obj) = MeM$.

Incidentally, because the operational semantics of C-light has an unstructured memory model, this language does not support machine word level operations.

Since the C-kernel language [19] is a subset of the C-light language, its operational semantics is the same as the C-light semantics. Thus, the memory model of C-kernel language is equal to memory of C-light language.

1.4. The C-lightVer tool

The C-lightVer system is based on the classic deductive verification method [21, 22]. C-light is an input language of this system. C-kernel is an intermediate verification language of this tool. The axiomatic semantics has been defined for the C-kernel language.

At the first stage, C-light is translated into an intermediate language, C-kernel. This stage is necessary for elimination of constructs that are complicated for axiomatic semantics. A set of formal rules is used for this translation. For example, increment operators are eliminated by translation into pieces of code with assignments and addition.

At the second stage, verification conditions are generated for the intermediate C-kernel program. This process is based on the axiomatic semantics of C-kernel. Once generated, the verification conditions are passed to the theorem prover.

1.5. The ACL2 theorem prover

ACL2 [23] is used in the C-lightVer system for proving verification conditions with the use of recursive functions corresponding to finite iterations. There are two main advantages of the ACL2 theorem prover:

- If the formula to be proved contains a recursive function, ACL2 can automatically run proving by induction using the definition of this function as the induction schema.
- If the underlying theory contains a theorem that can be considered as a rewriting rule, then ACL2 can automatically apply this rule for rewriting the formula to be proved.

These features allows automatizing proving formulas with *rep* functions in a lot of cases.

1.6. The mixed axiomatic semantics method

The method of mixed axiomatic semantics allows using particular versions of inference rules for particular versions of program constructs [20]. Let us note that there are variables in C programs that are used without referencing and dereferencing operators. A large number of such variables may be used in C programs. A simpler memory model may be used for such variables than the one based on *MeM* and *MD*. Therefore, simpler inference rules may be applied to program constructs with such variables. This allows verification conditions to be simplified.

1.7. The algorithm of the generation of recursive functions corresponding to loops

Let us consider a finite iteration over one-dimensional array:

$$\mathbf{for}\ (\mathbf{i} = \mathbf{i_0};\ \mathbf{i} < \mathbf{n};\ \mathbf{i}++)\ \mathbf{v} := \mathbf{body}(\mathbf{v}, \mathbf{i})\ end,$$

where

- v is the tuple of modifiable variables;
- S ia an one-dimensional array of n elements;
- $S \in v$;
- *body* is the admissible construct.

The admissible construct is one of the following C-kernel operators:

1. An empty operator, including an empty block.
2. The **break**; operator ending the loop.
3. The assignment operator $\mathbf{a} = \mathbf{b};$, where a is a simple type variable or a variable $S[i]$, and b is an expression in C-kernel.
4. The conditional statement **if** $(\mathbf{a})$ **b**, where a is an expression in C-kernel and b is an admissible construct.
5. The conditional statement **if** $(\mathbf{a})$ **b else c**, where a is an expression in C-kernel, and b and c are admissible constructs.
6. The block $\{\mathbf{a_1}\ \mathbf{a_2}\ \ldots\ \mathbf{a_{k-1}}\ \mathbf{a_k}\}$, where a_r is the admissible construct for each r: $1 \le r \le k$.
7. The nested finite iteration
   for $(\mathbf{j} = \mathbf{j_0};\ \mathbf{j} < \mathbf{m};\ \mathbf{j}{+}{+})\ \mathbf{u} := \mathbf{body}(\mathbf{u},\ \mathbf{j})$ **end**.

Since n is not constant in the general case, we can not apply full loop unrolling in this case. We apply the symbolic method of verification of finite iterations in this case. The tuple v includes S and simple type variables that may be changed in the loop body. Let v_0 denote the initial values of variables from v. Let us consider the *rep* definition in this case:

- $rep(v_0, S, body, 0) = v_0$,
- $rep(v_0, S, body, i) = body(rep(v_0, S, body, i-1), S_{i-1})$
  for each $i = 1, 2, \ldots, n$.

If a finite iteration has a `break` statement, we suggest the following solution: when the execution of the loop is terminated by this statement, we assume that the loop iterations continue, but the v values remain unchanged. If `break` was executed at iteration i $(0 < i \le n)$, then for each j $(i \le j \le n)$:

$$rep(v_0, S, body, j) = rep(v_0, S, body, i),$$

The inference rule in this case has the following form:

$$\frac{\{P\}\ \mathbf{prog};\ \{Q(v \leftarrow rep(v, S, body, n))\}}{\{P\}\ \mathbf{prog};\ \mathbf{for}\ (\mathbf{i} = \mathbf{i_0};\ \mathbf{i} < \mathbf{n};\ \mathbf{i}{+}{+})\ \mathbf{v} := \mathbf{body}(\mathbf{v},\ \mathbf{i})\ \mathbf{end}\ \{Q\}},$$

The method for generating the *rep* function body is based on the algorithm that translates loop body constructs to Applicative Common Lisp (the ACL2 language). Let us consider this algorithm [21, 22, 24].

The structure type *frame* is generated. The fields of such structure correspond to loop variables, the *rep* function returns object of type *frame*. All objects of the type *frame* are referred to as *fr*. Each loop instruction can be represented as a creation of new *fr* object with the fields in it appearing as the updated fields of previous *fr* object.

The sequential execution of the statements is translated to $b*$ construct:

$$(b*\ (\ldots(var\ expr)\ldots)\ result),$$

where $(var\ expr)$ means binding *var* to the value of *expr* which may depend on previously bound variables. We use *fr* as such *var* and we use the updates of the *fr* fields as *expr*. To simulate the loop exit, we use the Boolean field *loop-break* of the *frame* object. This field is true only after `break` has been executed.

The following definition of *gen_rep* is the implementation of translation of admissible constructs to Applicative Common Lisp:

- *gen_rep*(**empty statement**) = $(fr\ fr)$
- *gen_rep*(**break**;) = $((when\ t)\ fr)$
- *gen_rep*(**c = b**;) = $(fr\ (change\text{-}frame\ fr\ :c\ b))$
- *gen_rep*(**a[i] = b**;) = $(fr\ (change\text{-}frame\ fr\ :a\ (update\text{-}nth\ i\ b\ fr.a)))$
- *gen_rep*(**if (c) b else d**) =
  $(fr\ (if\ c$
  $(b*\ (gen\_rep(b))\ fr)\ (b*\ (gen\_rep(d))\ fr)))$
  $((when\ fr.loop\text{-}break)\ fr)$
- *gen_rep*($\{\mathbf{a_1}\ \mathbf{a_2}\ \ldots\ \mathbf{a_{k-1}}\ \mathbf{a_k}\}$) =
  $(fr\ (b^*(gen\_rep(a_1)\ \ldots\ gen\_rep(a_k))\ fr))$
  $((when\ fr.loop\text{-}break)\ fr)$

The replacement operation obtained returns not only the tuple v, but also a structure with the Boolean field (*loop-break*). The default value of *loop-break* is false. The *rep* function obtained contains the break condition. Once the `break` statement has been executed, this *rep* will return a structure with the true value of the *loop-break* field. Additionally, this *rep* checks the value of the *loop-break* field from the result of the recursive call. If this value is true, than this *rep* returns the same result as this recursive call. Thus, the values of the loop variables do not change after the execution of the `break` statement in this implementation.

1.8. Strategies for proving properties that may indicate possible errors

Given the user-defined precondition P of the loop, two strategies for proving properties that may indicate possible errors have been suggested [21, 22].

1.8.1. A strategy for finding loops with unused assignments to array elements

Let a loop implementing a finite iteration over an array contain assignments to elements of this array, and let the values of the array elements after the loop execution be equal to the values of these elements before the loop execution.

The strategy for finding such loops checks each loop over the array containing assignments to the array elements [21]. Let this loop be the i-th in the program code. The strategy is based on generating the lemma $P \rightarrow (a = rep_i(a, args).a)$, where

- P is the precondition;
- a is the array over which the finite iteration is performed;
- rep_i is the replacement operation for the finite iteration;
- $args$ are the arguments of rep_i;
- $rep_i(a, args).a$ is the array a after the loop execution,

and checking the validity of this lemma.

If this lemma was proved, than these assignments can appear to be unused statements, which may indicate the presence of an error.

1.8.2. A strategy for checking the execution of the `break` statement at the first loop iteration

Suppose that the loop implementing a finite iteration over an array contains a `break` statement that is always executed at the first loop iteration.

The strategy checks each loop over an array containing a `break` statement [21]. Let the loop be the i-th in the program code. The strategy is based on generating the lemma

$$P \rightarrow ((j_0 = rep_i(a, args).j) \wedge (rep_i(a, args).loop\text{-}break)),$$

where

- P is the precondition;
- a is the array over which the finite iteration is performed;

- j is the counter of the *for* loop which implements the finite iteration;
- *args* are the arguments of rep_i; the value j_0 of the loop variable j before the loop execution;
- $rep_i(a, args).j$ is the value of j after the loop execution;
- *loop-break* is a special field in the returned data structure; its value is true if and only if the `break` statement was executed during the loop execution,

and checking the validity of this lemma.

Thus, the first clause in the lemma conclusion is the assertion that the loop execution did not change the value of the loop counter. The second clause in the conclusion is the assertion that the `break` statement in the loop was executed.

If this lemma was proved, then this case may indicate the presence of an error.

Both strategies have been integrated in our logic.

1.9. Strongest postcondition calculus

The strongest postcondition [2] inference is applied to a program and its precondition. The formula $sp(S, P)$ is the strongest postcondition of a program S with a precondition P iff

- the triple $\{P\}\ S\ \{sp(S, P)\}$ is correct and
- if Q is a formula then validity of the formula $sp(S, P) \rightarrow Q$ implies the correctness of the triple $\{P\}\ S\ \{Q\}$.

This calculus allows defining axiomatic semantics using the following approach: if *stmt* is a program statement, then the inference rule for this statement may be defined as

$$\frac{\{sp(stmt, P)\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{stmt};\ \textbf{prog}\ \{Q\}}.$$

This approach can be considered as forward tracking: moving from the beginning of the program to its end and eliminating the leftmost statement (at the top level) by applying the corresponding inference rule. Our logic was developed using this inference style.

2. Logic for reasoning about bugs in loops over data sequences

The set of inference rules from our logic can be partitioned into the following two subsets: base inference rules and main inference rules. The goal of the base inference rules is to obtain the loop precondition. The goal of the main inference rules is to obtain the properties that may indicate the presence of bugs in the loops.

2.1. Base of logic for reasoning about bugs in loops over data sequences

Let us consider the inference rules for the kernel of the C-kernel-like subset of C language.

The inference rule for an empty program is:

$$\frac{P \rightarrow Q}{\{P\}\ \textbf{emptyProgram}\ \{Q\}}. \tag{1}$$

The strongest postcondition of the empty program has the following form:

$$sp(\textbf{emptyProgram},\ P) = P.$$

The inference rule for the variable declaration is:

$$\frac{\{\exists MeM' P(MeM \leftarrow MeM') \wedge MeM = upd(MeM', var, addr)\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{type var};\ \textbf{prog}\ \{Q\}}, \tag{2}$$

where *addr* is the new address ($addr \notin Dom(MD)$). The strongest postcondition of variable declaration has the following form:

$$sp(\textbf{type var},\ P) = \\ \exists MeM' P(MeM \leftarrow MeM') \wedge MeM = upd(MeM', var, addr).$$

The inference rule for variable assignment is:

$$\frac{\{\exists MD' P(MD \leftarrow MD') \wedge MD = upd(MD', MeM(var), rval)\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{var} = \textbf{rval};\ \textbf{prog}\ \{Q\}}. \tag{3}$$

The strongest postcondition of variable assignment is:

$$sp(\textbf{var} = \textbf{rval},\ P) = \\ \exists MD' P(MD \leftarrow MD') \wedge MD = upd(MD', MeM(var), rval).$$

The mixed axiomatic semantics method allows us to define the following version of this inference rule when neither referencing nor dereferencing operators are used on *var*:

$$\frac{\{\exists var' P(var \leftarrow var')\ \wedge var = rval(var \leftarrow var')\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{var} = \textbf{rval};\ \textbf{prog}\ \{Q\}}. \tag{4}$$

The strongest postcondition of variable assignment in this case is:

$$sp(\textbf{var} = \textbf{rval},\ P) = \\ \exists var' P(var \leftarrow var')\ \wedge var = rval(var \leftarrow var').$$

The inference rule for assignment to an array element has the following form:

$$\frac{\{\exists MD' P(MD \leftarrow MD') \wedge MD = upd(MD', MeM(a, i), rval)\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{a[i]} = \textbf{rval};\ \textbf{prog}\ \{Q\}}. \tag{5}$$

Strongest postcondition of assignment to an array element has the following form:

$$sp(\textbf{a[i]} = \textbf{rval},\ P) = \\ \exists MD' P(MD \leftarrow MD') \wedge MD = upd(MD', MeM(a, i), rval).$$

The mixed axiomatic semantics method allows us to define the following version of this inference rule when neither referencing nor dereferencing operators are used on the array element:

$$\frac{\{\exists a' P(a \leftarrow a') \wedge a = update(a', i, rval)\}\ \textbf{prog};\ \{Q\}}{\{P\}\ \textbf{a[i]} = \textbf{rval};\ \textbf{prog}\ \{Q\}}, \tag{6}$$

where *update* is the array update operation with the following axioms:

1) $(update(a,\ i,\ val))[i] = val$;
2) $(update(a, i_1, val))[i_2] = a(i_2)$
   $if\ i_1 \neq i_2$;
3) $update(a,\ i,\ a[i]) = a$.

The strongest postcondition of assignment to an array element has the following form in this case:

$$sp(\textbf{a[i]} = \textbf{rval},\ P) = \\ \exists a' P(a \leftarrow a') \wedge a = update(a', i, rval).$$

The inference rule for the `if` statement has the following form:

$$\frac{\{P \wedge B\}\ \mathbf{S_1};\ \mathbf{prog}\ \{Q\},\ \{P \wedge \neg B\}\ \mathbf{S_2};\ \mathbf{prog}\ \{Q\}}{\{P\}\ \mathbf{if\ B\ then\ S_1\ else\ S_2};\ \mathbf{prog}\ \{Q\}}. \tag{7}$$

The strongest postcondition of the `if` statement has the following form:

$$sp(\mathbf{if\ B\ then\ S_1\ else\ S_2},\ P) = \\ sp(\mathbf{S_1}, P \wedge B) \vee sp(\mathbf{S_2}, P \wedge \neg B).$$

These inference rules allow obtaining the finite iteration precondition that is used in the main inference rules.

2.2. Main inference rules of the logic for reasoning about bugs in loops over data sequences

Let us consider inference rules for the inference of formulas whose validity may indicate the presence of bugs.

The following inference rule corresponds to the strategy for finding loops with unused assignments to array elements:

$$\frac{P \rightarrow (S = rep(v, S, n).S)}{\{P\}\ \mathbf{for}\ (\mathbf{i}\ =\ \mathbf{i_0};\ \mathbf{i}\ <\ \mathbf{n};\ \mathbf{i}++)\ \mathbf{v} := \mathbf{body(v,\ i)}\ \mathbf{end};\ \mathbf{prog}\ \{Q\}}, \tag{8}$$

where

- P is the precondition;
- S is the array over which the finite iteration is performed;
- rep is the replacement operation for the finite iteration;
- $rep(v, S, n).S$ is the array S after the loop execution.

The following inference rule corresponds to the strategy for checking the execution of the `break` statement at the first loop iteration:

$$\frac{P \rightarrow ((i_0 = rep(v, S, n).i) \wedge (rep(v, S, n).loop{-}break))}{\{P\}\ \mathbf{for}\ (\mathbf{i}\ =\ \mathbf{i_0};\ \mathbf{i} < \mathbf{n};\ \mathbf{i}++)\ \mathbf{v} := \mathbf{body(v,\ i)}\ \mathbf{end};\ \mathbf{prog}\ \{Q\}}, \tag{9}$$

where

- P is the precondition;
- S is the array over which the finite iteration is performed;
- rep is the replacement operation for the finite iteration;
- $rep(v, S, n).i$ is the value of the loop counter i after the loop execution;
- $rep(v, S, n).S$ is the array a after the loop execution.

The next inference rules are based on modifying the definition of the rep function. Let us note that the next inference rules are not strategies from [21, 22] encoded in our logic, they represent a completely new approach. If the rep function contains an `if` statement, than we add to two new fields to the *frame* structure: $if{-}true_k$ and $if{-}false_k$, where k is the number of `if` statement in the finite iteration.

The $if{-}true_k$ field contains a conjunction of the values of the condition of the k-th `if` statement on each iteration. Thus, the true value of this field means that the condition of the k-th `if` statement is true on each iteration. This information may be indicative of an error.

The $if{-}false_k$ field contain a conjunction of negations of the values of the condition of the k-th `if` statement on each iteration. Thus, the true value of this field means that the condition of the k-th `if` statement is false on each iteration. This information may be indicative of an error.

For each k (for each `if` statement in the finite iteration) the following inference rules are applied:

$$\frac{P \rightarrow rep(v, S, n).if{-}true_k}{\{P\}\ \textbf{for}\ (\mathbf{i}\ =\ \mathbf{i_0};\ \mathbf{i}\ <\ \mathbf{n};\ \mathbf{i}++)\ \mathbf{v} := \textbf{body}(\mathbf{v},\ \mathbf{i})\ \textbf{end};\ \textbf{prog}\ \{Q\}}, \tag{10}$$

where

- P is the precondition;
- S is the array over which the finite iteration is performed;
- rep is the replacement operation for the finite iteration;
- $rep(v, S, n).if{-}true_k$ is the value of the $if{-}true_k$ field after the loop execution,

and

$$\frac{P \rightarrow rep(v, S, n).if{-}false_k}{\{P\}\ \textbf{for}\ (\mathbf{i}\ =\ \mathbf{i_0};\ \mathbf{i}\ <\ \mathbf{n};\ \mathbf{i}++)\ \mathbf{v} := \textbf{body}(\mathbf{v},\ \mathbf{i})\ \textbf{end};\ \textbf{prog}\ \{Q\}}, \tag{11}$$

where

- P is the precondition;
- S is the array over which the finite iteration is performed;
- rep is the replacement operation for the finite iteration;
- $rep(v, S, n).if{-}false_k$ is the value of the $if{-}false_k$ field after the loop execution.

We can modify the definition of the *rep* function to calculate the values of arbitrary formulas over finite iteration variables. It allows reasoning about the properties of finite iterations. It is possible to extend our logic with new inference rules for error localization.

Let us note that we may apply several inference rules to a particular finite iteration to reason about several types of bugs. We call the proposed logic the Incorrectness Finite Iteration Logic (IFIL).

2.3. Termination of inference based on the proposed logic

Let us consider the following theorem:

Theorem 1. *If the inference rules from IFIL are applied to an annotated sequence of program statements that can contain only variable declarations, assignments to variables, assignments to elements of arrays, `if` statements and finite iterations then this inference will be terminated.*

Proof. Given

- *statements* is the name of the sequence of the program statements considered in the statement of the theorem,
- *count_of_statements* is the function that returns the count of statements in the sequence of the program statements (including nested statements, for example, statements from branches of *if statements*),
- $count\_of\_statements(statements) = n$,
- *top_level_head* is the function that returns the first element at the top level of the sequence of the program statements (for example, we consider each *if statement* as one statement at the top level of the sequence of the program statements),
- *top_level_tail* is the function that returns the sequence of the program statements without the first element at the top level of the sequence of the program statements (for example, we consider each *if statement* as one statement at the top level of the sequence of the program statements),

let us prove this theorem by induction on n. Thus, the proof consists of two cases:

1. **Induction base.** This case corresponds to $n = 0$. The *statements* has the following form

$$emptyProgram$$

in this case. Thus, $top\_level\_head(statements)$ is equal to

$$emptyProgram$$

and $top\_level\_tail(statements)$ is equal to $emptyProgram$. Let us note that

$$count\_of\_statements(top\_level\_head(statements)) = count\_of\_statements(emptyProgram) = 0$$

Thus,

$$count\_of\_statements(top\_level\_tail(statements)) = \\ n - count\_of\_statements(top\_level\_head(statements)) = 0$$

We should apply inference rule 1 from Section 2.1 in this case. Since the application of inference rule 1 from Section 2.1 will be terminated, the inference process will be terminated, too.

2. **Induction step.** Let us consider all possible values of $top\_level\_head(statements)$:
   (a) *Variable declaration.* The *statements* has the following form

   $$type\ \ var;\ \ prog$$

   in this case. Thus, $top\_level\_head(statements)$ is equal to

   $$type\ \ var$$

   and $top\_level\_tail(statements)$ is equal to *prog*. Let us note that

   $$count\_of\_statements(top\_level\_head(statements)) = count\_of\_statements(type\ var) = 1$$

   Thus,

   $$count\_of\_statements(top\_level\_tail(statements)) = \\ n - count\_of\_statements(top\_level\_head(statements)) = n - 1$$

   We should apply inference rule 2 from Section 2.1 in this case. Since the application of inference rule 2 will be terminated and

   $$count\_of\_statements(prog) = count\_of\_statements(top\_level\_tail(statements)) = n - 1$$

   in this case, we can apply the induction hypothesis. The induction hypothesis means that the application of the inference process to annotated program code *prog* will be terminated in this case. Thus, the inference process will be terminated.
   (b) *Variable assignment.* The *statements* has the following form

   $$var = rval;\ \ prog$$

   in this case. Thus, $top\_level\_head(statements)$ is equal to

   $$var = rval$$

   and $top\_level\_tail(statements)$ is equal to *prog*. Let us note that

   $$count\_of\_statements(top\_level\_head(statements)) = count\_of\_statements(var = rval) = 1$$

   Thus,

   $$count\_of\_statements(top\_level\_tail(statements)) = \\ n - count\_of\_statements(top\_level\_head(statements)) = n - 1$$

   We should apply either inference rule 3 from Section 2.1 or inference rule 4 from Section 2.1 in this case. Since the application of either of the inference rules will be terminated and

   $$count\_of\_statements(prog) = count\_of\_statements(top\_level\_tail(statements)) = n - 1$$

in both cases, we can apply the induction hypothesis. The induction hypothesis means that application of inference process to annotated program code *prog* will be terminated in this case. Thus, the inference process will be terminated.

(c) *Assignment to an array element.* The *statements* has the following form

$$a[i] = rval;\ prog$$

in this case. Thus, $top\_level\_head(statements)$ is equal to

$$a[i] = rval$$

and $top\_level\_tail(statements)$ is equal to *prog*. Let us note that

$$count\_of\_statements(top\_level\_head(statements)) = count\_of\_statements(a[i] = rval) = 1$$

Thus,

$$count\_of\_statements(top\_level\_tail(statements)) = \\ n - count\_of\_statements(top\_level\_head(statements)) = n - 1$$

We should apply either inference rule 5 from Section 2.1 or inference rule 6 from Section 2.1 in this case. Since the application of either of the inference rules will be terminated and

$$count\_of\_statements(prog) = count\_of\_statements(top\_level\_tail(statements)) = n - 1$$

in both cases, we can apply the induction hypothesis. The induction hypothesis means that the application of the inference process to annotated program code *prog* will be terminated in this case. Thus, the inference process will be terminated.

(d) *if statement.* The *statements* has the following form

$$if\ B\ then\ S_1\ else\ S_2;\ prog$$

in this case. Thus, $top\_level\_head(statements)$ is equal to

$$if\ B\ then\ S_1\ else\ S_2$$

and $top\_level\_tail(statements)$ is equal to *prog*. Let us note that

$$count\_of\_statements(top\_level\_head(statements)) = \\ count\_of\_statements(if\ B\ then\ S_1\ else\ S_2) = \\ 1 + count\_of\_statements(S_1) + count\_of\_statements(S_2)$$

Thus,

$$count\_of\_statements(top\_level\_tail(statements)) = \\ n - count\_of\_statements(top\_level\_head(statements)) = \\ n - 1 - count\_of\_statements(S_1) - count\_of\_statements(S_2)$$

Let us note that

$$count\_of\_statements(prog) = count\_of\_statements(top\_level\_tail(statements)) = \\ n - 1 - count\_of\_statements(S_1) - count\_of\_statements(S_2)$$

Consequently,

$$count\_of\_statements(S_1;\ prog) = count\_of\_statements(S_1) + count\_of\_statements(prog) = \\ count\_of\_statements(S_1) + n - 1 - count\_of\_statements(S_1) - count\_of\_statements(S_2) = \\ n - 1 - count\_of\_statements(S_2)$$

and

$$count\_of\_statements(S_2;\ prog) = count\_of\_statements(S_2) + count\_of\_statements(prog) = \\ count\_of\_statements(S_2) + n - 1 - count\_of\_statements(S_1) - count\_of\_statements(S_2) = \\ n - 1 - count\_of\_statements(S_1)$$

Since $count\_of\_statements(S_2) \geq 0$,

$$count\_of\_statements(S_1;\ prog) = n - 1 - count\_of\_statements(S_2) \leq n - 1$$

Since $count\_of\_statements(S_1) \geq 0$,

$$count\_of\_statements(S_2;\ prog) = n - 1 - count\_of\_statements(S_1) \leq n - 1$$

We should apply inference rule 7 from Section 2.1 in this case. Since the application of inference rule 7 will be terminated and

$$count\_of\_statements(S1;\ prog) \leq n - 1$$

and

$$count\_of\_statements(S2;\ prog) \leq n - 1$$

we can apply the induction hypothesis. The induction hypothesis means that the application of the inference process to annotated program code $S_1;\ prog$ will be terminated and the application of the inference process to annotated program code $S_2;\ prog$ will be terminated. Thus, the inference process will be terminated.

(e) *Finite iteration.* The *statements* has the following form

$$for\ (i\ =\ i_0;\ i\ <\ n;\ i++)\ v := body(v,\ i)\ end;\ prog$$

in this case. Thus, $top\_level\_head(statements)$ is equal to

$$for\ (i\ =\ i_0;\ i\ <\ n;\ i++)\ v := body(v,\ i)\ end$$

and $top\_level\_tail(statements)$ is equal to *prog*. We should apply one the following rule:

- inference rule 8 from Section 2.2,
- inference rule 9 from Section 2.2,
- inference rule 10 from Section 2.2,
- inference rule 11 from Section 2.2

in this case. Since the application of all mentioned inference rules from Section 2.2 will be terminated, the inference process will be terminated.

□

Let us note that if the input sequence of the program statements does not contain finite iteration, we infer only the strongest postcondition instead of the formula describing the property of a finite iteration. Proving other properties of our logic is a difficult problem due to the heuristic nature of our logic. In any case, we are planning to discovery and to prove other properties of our logic in a future work.

3. Experiment

Our logic was implemented in the C-lighVer system as an alternative semantics for the C-kernel language. We performed a bug localization experiment on the `negate_first` program from the well-known verification challenge [36]:

```
void negate_first(int n, int* a) {
    int i;
    for (i = 0; i < n; i++) {
        if (a[i] < 0) {a[i] = -a[i]; break;}}}
```

The precondition of this program is

$$(a_0 = a) \wedge (0 < n) \wedge (n \leq length(a_0))$$

The postcondition of this program is

$$\begin{gathered}(\neg found\_negative(n,\ a_0) \rightarrow \\ a = a_0) \\ \wedge \\ (found\_negative(n,\ a_0) \rightarrow \\ a = update(a_0, count\_index(n,\ a_0), -a_0[count\_index(n,\ a_0)]))\end{gathered}$$

where

- *found-negative* predicate checks if there is a negative element in the array;
- *count-index* function calculates the index of the first negative element of the array if the array contains such an element. The value of this function is undefined in other cases.

The main problem of reasoning about this program is the `break` statement in the loop.

Let us consider the `negate_first` program with an introduced error:

```
1. /*@ requires (a0 = a) && (0 < n) && (n <= length(a0));
2. ensures (!found_negative(n, a0) ==> a == a0) &&
3.         (found_negative(n, a0) ==>
              a == update(a0, count_index(n, a0), -a0[count_index(n, a0)]))
4. */
5. void negate_first(int n, int* a) {
6.   int i;
7.   for (i = 0; i < n; i++) \{
8.        if (a[i] < a[i]) {a[i] = -a[i]; break;}}}
```

The specifications of this function were defined using the ACSL language. The error is using `a[i]` instead of `0` in the `if` condition at line 8.

The following formulas obtained have been proved automatically using the ACL2 system:

```
(implies
    (and (integer-listp a) (integer-listp a_0) (equal a a_0)
         (integerp n) (< 0 n) (<= n (length a_0)))
    (equal
        a
        (rep
            (frame-init
                0
                a
                nil
            )
            (envir-init
                n
            )
        ).a
    )
)
```

and

```
(implies
     (and (integer-listp a) (integer-listp a_0) (equal a a_0)
          (integerp n) (< 0 n) (<= n (length a_0)))
     (equal
         t
         (rep
             (frame-init
                 0
                 a
                 nil
             )
             (envir-init
                 n
             )
         ).iffalse1
    )
)
```

where

- `frame-init` is the constructor of the *frame* structure;
- `envir-init` is the constructor of the *envir* structure which stores values that have not been modified by the finite iteration.

The mixed axiomatic semantics method allows using simple inference rules without *MeM* or *MD* to obtain these formulas.

The validity of these formulas means the following:

- assignment in the loop body is never used;
- the condition of `if` statement is always false.

Thus, our logic can help localize bug without using loop invariants.

Conclusion

The new result presented in this paper is a logic for reasoning about bugs in loops over data sequences. This logic is based on special inference rules for finite iterations. These rules allow generating properties that may indicate errors in finite iterations. These properties are generated as formulas with recursive functions corresponding to finite iterations. This logic has been implemented in a new version of the C-lightVer system for a deductive verification of C programs. We have performed reasoning about the incorrectness of an illustrative example to demonstrate how our logic works.

There are three advantages of our approach:

1. Our logic does not require defining invariants of finite iterations.
2. Our logic is based on an over-approximation approach similar to classic deductive verification. It simplifies the development of a unified approach to reasoning about correctness and incorrectness.
3. Our logic is based on the use of special inference rules for finite iterations. These rules allow generating simple formulas to be proved in the case of finite iterations.

We believe that our approach has promise, because we can extend our logic with new special inference rules for finite iterations. Thus, we are planning to extend our logic with new inference rules to handle more types of bugs. For example, we will soon be applying our approach to finite iterations over lists, trees and other dynamic data structures.

References

[1] R. Hähnle and M. Huisman, "Deductive software verification: From pen-and-paper proofs to industrial tools", in *Computing and Software Science*, vol. 10000, Springer, 2019, pp. 345–373.

[2] K. R. Apt and E.-R. Olderog, "Fifty years of Hoare's logic", *Formal Aspects of Computing*, vol. 31, no. 6, pp. 751–807, 2019.

[3] K. R. Apt and E.-R. Olderog, "Assessing the success and impact of Hoare's logic", in *Theories of Programming: The Life and Works of Tony Hoare*, 2021, pp. 41–76.

[4] C. A. R. Hoare, "An axiomatic basis for computer programming", *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[5] B. Möller, P. O'Hearn, and T. Hoare, "On algebra of program correctness and incorrectness", in *Relational and Algebraic Methods in Computer Science*, vol. 13027, Springer, 2021, pp. 325–343.

[6] Q. L. Le, A. Raad, J. Villard, J. Berdine, D. Dreyer, and P. W. O'Hearn, "Finding real bugs in big programs with incorrectness logic", *Proceedings of the ACM on Programming Languages*, vol. 6, no. OOPSLA1, pp. 1–27, 2022.

[7] P. W. O'Hearn, "Incorrectness logic", *Proceedings of the ACM on Programming Languages*, vol. 4, no. POPL, pp. 1–32, 2019.

[8] A. Raad, J. Berdine, H.-H. Dang, D. Dreyer, P. O'Hearn, and J. Villard, "Local reasoning about the presence of bugs: Incorrectness separation logic", in *Computer Aided Verification*, vol. 12225, Springer, 2020, pp. 225–252.

[9] J. Vanegue, "Adversarial logic", in *Static Analysis*, vol. 13790, Springer, 2022, pp. 422–448.

[10] M. Milanese and F. Ranzato, "Local completeness logic on Kleene algebra with tests", in *Static Analysis*, vol. 13790, Springer, 2022, pp. 350–371.

[11] B. Bruni, R. Giacobazzi, R. Gori, and F. Ranzato, "A correctness and incorrectness program logic", *Journal of the ACM*, vol. 70, no. 2, pp. 1–45, 2023.

[12] P. Maksimović, C. Cronjäger, A. Lööw, J. Sutherland, and P. Gardner, "Exact separation logic: Towards bridging the gap between verification and bug-finding", in *37th European Conference on Object-Oriented Programming (ECOOP 2023)*, vol. 263, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 19:1–19:27.

[13] N. Zilberstein, D. Dreyer, and A. Silva, "Outcome logic: A unifying foundation of correctness and incorrectness reasoning", *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 522–550, 2023.

[14] T. Dardinier and P. Müller, *Hyper hoare logic: (Dis-)Proving program hyperproperties (extended version)*, 2023. arXiv: 2301.10037 [cs.LO].

[15] A. Humenberger, M. Jaroschek, and L. Kovács, "Invariant generation for multi-path loops with polynomial assignments", in *Verification, Model Checking, and Abstract Interpretation*, vol. 10747, Springer, 2018, pp. 226–246.

[16] S. Chakraborty, A. Gupta, and D. Unadkat, "Full-program induction: Verifying array programs sans loop invariants", *International Journal on Software Tools for Technology Transfer*, vol. 24, no. 5, pp. 843–888, 2022.

[17] V. A. Nepomniaschy, "Symbolic method of verification of definite iterations over altered data structures", *Programming and Computer Software*, vol. 31, no. 1, pp. 1–9, 2005.

[18] V. A. Nepomniaschy, I. S. Anureev, I. N. Mikhailov, and A. V. Promskii, "Towards verification of C programs. C-light language and its formal semantics", *Programming and Computer Software*, vol. 28, no. 6, pp. 314–323, 2002.

[19] V. A. Nepomniaschy, I. S. Anureev, and A. V. Promskii, "Towards verification of C programs: Axiomatic semantics of the C-kernel language", *Programming and Computer Software*, vol. 29, no. 6, pp. 338–350, 2003.

[20] I. V. Maryasov, V. A. Nepomniaschy, A. V. Promsky, and D. A. Kondratyev, "Automatic C program verification based on mixed axiomatic semantics", *Automatic Control and Computer Sciences*, vol. 48, no. 7, pp. 407–414, 2014.

[21] D. A. Kondratyev and V. A. Nepomniaschy, "Automation of C program deductive verification without using loop invariants", *Programming and Computer Software*, vol. 48, no. 5, pp. 331–346, 2022.

[22] D. A. Kondratyev and A. V. Promsky, "The complex approach of the C-lightVer system to the automated error localization in C-programs", *Automatic Control and Computer Sciences*, vol. 54, no. 7, pp. 728–739, 2020.

[23] J. S. Moore, "Milestones from the pure lisp theorem prover to ACL2", *Formal Aspects of Computing*, vol. 31, no. 6, pp. 699–732, 2019.

[24] D. A. Kondratyev, I. V. Maryasov, and V. A. Nepomniaschy, "The automation of C program verification by the symbolic method of loop invariant elimination", *Automatic Control and Computer Sciences*, vol. 53, no. 7, pp. 653–662, 2019.

[25] L. Zhang and B. L. Kaminski, "Quantitative strongest post: A calculus for reasoning about the flow of quantitative information", *Proceedings of the ACM on Programming Languages*, vol. 6, no. OOPSLA1, pp. 1–29, 2022.

[26] S. Dailler, D. Hauzar, C. Marché, and Y. Moy, "Instrumenting a weakest precondition calculus for counterexample generation", *Journal of Logical and Algebraic Methods in Programming*, vol. 99, pp. 97–113, 2018.

[27] B. Becker, C. B. Lourenço, and C. Marché, "Explaining counterexamples with giant-step assertion checking", in *Proceedings of the 6th Workshop on Formal Integrated Development Environment*, vol. 338, 2021, pp. 82–88.

[28] Q. L. Le, J. Sun, L. H. Pham, and S. Qin, *S2TD: A separation logic verifier that supports reasoning of the absence and presence of bugs*, 2022. arXiv: 2209.09327 [cs.PL].

[29] T. Dardinier, G. Parthasarathy, and P. Müller, "Verification-preserving inlining in automatic separation logic verifiers", *Proceedings of the ACM on Programming Languages*, vol. 7, no. OOPSLA1, pp. 789–818, 2023.

[30] R. Könighofer, R. Toegl, and R. Bloem, "Automatic error localization for software using deductive verification", in *Hardware and Software: Verification and Testing*, vol. 8855, Springer, 2014, pp. 92–98.

[31] P. Baudin, F. Bobot, D. Bühler, L. Correnson, F. Kirchner, N. Kosmatov, A. Maroneze, V. Perrelle, V. Prevosto, J. Signoles, and N. Williams, "The dogged pursuit of bug-free C programs: The Frama-C software analysis platform", *Communications of the ACM*, vol. 64, no. 8, pp. 56–68, 2021.

[32] M. R. Gadelha, F. Monteiro, L. Cordeiro, and D. Nicole, "ESBMC v6.0: Verifying C programs using *k*-induction and invariant inference", in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 11429, Springer, 2019, pp. 209–213.

[33] S. Löwe, "CPAchecker with explicit-value analysis based on CEGAR and interpolation", in *Tools and Algorithms for the Construction and Analysis of Systems*, vol. 7795, Springer, 2013, pp. 610–612.

[34] D. Beyer and T. Lemberger, "CPA-SymExec: Efficient symbolic execution in CPAchecker", in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 900–903.

[35] C. Cadar and M. Nowack, "KLEE symbolic execution engine in 2019", *International Journal on Software Tools for Technology Transfer*, vol. 23, no. 6, pp. 867–870, 2021.

[36] B. Jacobs, J. Kiniry, and M. Warnier, "Java program verification challenges", in *Formal Methods for Components and Objects*, vol. 2852, Springer, 2003, pp. 202–219.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

The Boltzmann distribution in the problem of rational choice by population of a patch under an imperfect information about its resources

A. N. Kirillov[1], I. V. Danilova[2]

DOI: 10.18255/1818-1015-2023-3-234-245

[1]Karelian Research Centre of the Russian Academy of Sciences, 11, Pushkinskaya street, Petrozavodsk, 185910, Russia.
[2]Petrozavodsk State University, Ave. Lenina, 33, Petrozavodsk, 185910, Russia.

MSC2020: 90B90
Research article
Full text in English

Received June 30, 2023
After revision July 25, 2023
Accepted August 2, 2023

The problem of rational choice by the population of a patch containing energy (nutritive) resources is considered. This problem belongs to the theory of optimal foraging, which, in turn of, studies issues related to the behavior of the population when it leaves the patch or chooses the most suitable one. In order to define the optimal patch choice for population, a variational approach, based on the idea of the Boltzmann distribution is proposed. To construct the probability distribution the utility functions are used, that take into account factors that can influence the patch choice of a population: available information about the quality of patches, the energy utility of patches, the cost of moving to the patch, the cost of information about the quality of patches. The main goal of the paper is to investigate the influence of available information about the amount of resources, contained in patches, on a decision-making process generated by the foragers while a suitable patch choosing. The optimal rationality is determined in the cases taking into account the information cost, the average energy utility of all patches, the rationality depending on the patch. The conditions under which the population, with the lack of information, select the "poor" patch, in sense of its resources, are obtained. The latter provides a theoretical justification of experimental observations, according to which a population can choose a patch with worse quality. The obtained results have a general character and may be used not only in behavioral ecology but when constructing any decision making processes.

Keywords: Boltzmann distribution; rationality of choice; measure of awareness; information cost; utility function

INFORMATION ABOUT THE AUTHORS

Alexander N. Kirillov	orcid.org/0000-0002-3356-1846. E-mail: krllv1812@yandex.ru Leading Researcher of ICT Laboratory, Doctor of Physical and Mathematical Sciences, Associate Professor.
Inna V. Danilova corresponding author	orcid.org/0000-0001-7031-4580. E-mail: danilovainna1987@mail.ru Senior Lecturer, Candidate of Physical and Mathematical Sciences.

Funding: This work was supported by the Russian Science Foundation (project No. 23-21-00092), https://rscf.ru/project/23-21-00092/.

For citation: A. N. Kirillov and I. V. Danilova, "The Boltzmann distribution in the problem of rational choice by population of a patch under an imperfect information about its resources", *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 234-245, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Распределение Больцмана в проблеме рационального выбора популяцией участка при неполной информации о его ресурсах

А. Н. Кириллов[1], И. В. Данилова[2]

DOI: 10.18255/1818-1015-2023-3-234-245

[1]Карельский научный центр Российской академии наук, ул. Пушкинская, д.11, г. Петрозаводск, 185910, Россия.
[2]Петрозаводский государственнй университет, просп. Ленина, д. 33, г. Петрозаводск, 185910, Россия.

УДК 517.977
Научная статья
Полный текст на английском языке

Получена 30 июня 2023 г.
После доработки 25 июля 2023 г.
Принята к публикации 2 августа 2023 г.

Рассматривается задача рационального выбора популяцией участка, содержащего энергетические (пищевые) ресурсы. Рассматриваемая задача относится к теории оптимального фуражирования, которая в свою очередь изучает вопросы, касающиеся поведения популяции, когда она покидает участок или выбирает наиболее подходящий. Для определения оптимального для популяции выбора участка предлагается вариационный подход, основанный на идее распределения Больцмана. Для построения распределения Больцмана вводятся функции полезности, которые учитывают факторы, способные повлиять на выбор популяции: имеющаяся информация о качестве участков, энергетическая полезность участков, затраты на перемещение к участку, стоимость информации о качестве участков. Основная цель статьи – исследовать влияние имеющейся информации о количестве ресурсов, содержащихся в участках, на процесс принятия решений, генерируемых популяцией при выборе подходящего участка. Оптимальная рациональность определяется с учетом стоимости информации, средней энергетической ценности всех участков, рациональности, зависящей от качества участка. Получены условия, при которых популяция при недостатке информации выбирает «бедный» участок в смысле энергетической ценности (ресурсов). Последнее дает теоретическое обоснование экспериментальным наблюдениям, согласно которым, популяция может выбрать участок худшего качества. Полученные результаты носят общий характер и могут быть использованы не только в поведенческой экологии, но и при построении любых процессов принятия решений.

Ключевые слова: распределение Больцмана; рациональность выбора; мера информированности; стоимость информации; функция полезности

ИНФОРМАЦИЯ ОБ АВТОРАХ

Александр Николаевич Кириллов | orcid.org/0000-0002-3356-1846. E-mail: krllv1812@yandex.ru
ведущий научный сотрудник лаборатории ИКТ, доктор физ.-мат. наук, доцент.

Инна Владимировна Данилова
автор для корреспонденции | orcid.org/0000-0001-7031-4580. E-mail: danilovainna1987@mail.ru
старший преподаватель, кандидат физ.-мат. наук.

Финансирование: Исследование выполнено за счет гранта Российского научного фонда № 23-21-00092, https://rscf.ru/project/23-21-00092/.

Для цитирования: A. N. Kirillov and I. V. Danilova, “The Boltzmann distribution in the problem of rational choice by population of a patch under an imperfect information about its resources”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 234-245, 2023.

Introduction

The open problem of rationality and rational choice is one of the crucial challenges in artificial intelligence, reinforcement learning, computational neuroscience, behavioral ecology of animals, economics. It is important to understand the mechanisms of decision-making process under bounded resource, in particular, available information. R. Aumann, who received the nobel prize (2005) for investigations of conflict and cooperation, wrote [1] that the problem of rationality "... is perhaps the most challenging conceptual problem in the area today: to develop a meaningful formal definition of rationality in a situation in which calculation and analysis themselves are costly and/or limited". Moreover, the problem of bounded rationality becomes especially acute in connection with applications mentioned above (see, for example, [2], [3] and the references therein).

One of the applications of the rational choice theory is in the optimal foraging theory, which studies, in particular, the patch selection by population, that is most suitable for consumption of the resources contained in it. As it is generally accepted, the behavior of population aimed at maximizing the amount of consumed energy [4–8].

Within the framework of this theory, the concept of ideal free distribution (IFD) was proposed [9, 10]. According to the IFD, a population has perfect information about patches quality and is distributed between patches so as to maximize an energy consumption rate. But empirical observations show that the IFD model is not adequate to real patch selection processes. A population, under the lack of information about patch resources, may choose poor patches (see, for example, [11] and the references therein).

In order to overcome the disadvantages of the IFD concept, U. Dieckmann proposed an approach [11], based on the Boltzmann distribution and utility functions of patches. If p_i is the probability for a population to select the i-th patch, $i = 1, \dots, n$, then, according to the Boltzmann distribution

$$p_i = \frac{e^{qU_i}}{\sum\limits_{j=1}^{n} e^{qU_j}}, \tag{1}$$

where U_i is the utility function corresponding to the i-th patch, q is non-negative constant, n is the number of patches.

It is worth to note that the Boltzmann distribution is one of the main notions of statistical physics [12]. Namely, consider a physical system which may be in one of the states $1, \dots, n$ with energies $E_1, \dots, E_n$, respectively. Then the probability p_i for the system to be in the i–th state is given by (1), where $U_i = E_i$, $q = -\dfrac{1}{kT}$, T is the temperature of the "large" external heat source, $k > 0$ is the constant.

The methods of statistical physics have become ubiquitous in mathematics and mathematical modeling. Such notions as the entropy, the Gibbs and the Boltzmann distributions, the thermodynamic potential, to name a few, play a significant role in ergodic theory [13], [14], machine learning (particularly, reinforcement learning) [15], [16], decision-making theory [17], information theory [18], etc.

This paper deals with a problem of decision making processes, on the basis of the Boltzmann distribution, in the foraging theory. As it is noted above, the foraging theory, originally, appeared as a branch of behavioral ecology. Its goal is to develop a theoretical base for description and explanation of animal behavior while food searching. Later on, the ideas of optimal foraging theory were used to formalize the processes of choice in different areas.For example, nowadays, we observe the appearance of such fields as information foraging [19] and robot foraging [20]. The latter permits us to use, in this paper, an abstract language without binding to the optimal foraging connected with the behavioral ecology. But, in order to make the presentation more pictorial, describing the decision-making process, along with the term "agent" we use the term "population".

In the paper, the authors continue their researches dealing with mathematical modeling in optimal foraging theory, namely, [21], [22], [23].

The main goal of the paper is to investigate the influence of available information about the amount of resources, contained in patches, on a decision-making process generated by the population while a suitable patch choosing. As one might expect, when decision makers have a limited amount of information, the rationality of their behavior is bounded [2, 17]. On the basis of the Boltzmann distribution, we investigate this problem in optimal foraging, namely, in the selection of suitable patch by population.

The paper is organized as follows. In the Introduction the main applications of the theory of optimal foraging and methods for studying its problems are presented. Section 1 describes the proposed variational principle for determination of the optimal choice. In Section 2 the influence of the imperfect information on the rationality of choice is analyzed. Section 3 deals with the case when the rationality q is not constant but depends on information. In the Conclusion the results of the study are summarized.

1. Optimal rationality

According to [11], the parameter q in (1) has the sense of "controlling the degree of optimality" in choosing a patch. In other words, q is the rationality of decision-making process, assuming $q \geqslant 0$. In what follows, we call the parameter q the rationality of patch selecting, or, more simply - the rationality. Such sense of q is justified by the following proposition.

Proposition 1. *Denote* $U_{i_s} = \max_i \{U_i,\ i = 1, \dots, n\}$, $s = 1, \dots, k$. *Then:*

$p_{i_s} \to \dfrac{1}{k}$ *as* $q \to \infty$;

$p_i \to 0$ *as* $q \to \infty$ *if* $i \in \{1, \dots, n\} \setminus \{i_s,\ s = 1, \dots, k\}$;

$p_i = \dfrac{1}{n}$ *if* $q = 0$.

Proof. We have the following presentation of p_i from (1)

$$p_i = \frac{1}{1 + \sum\limits_{j=1,\ j\neq i}^{n-1} e^{q(U_j - U_i)}}, \tag{2}$$

which implies the proof. □

Remark 1. The special case of Proposition 1, namely for $i = 1$, was formulated in [11].

The sense of Proposition 1 is as follows. The probability of choosing by an agent (population) the patch with the largest utility tends to its maximum value as q tends to infinity. If $k = 1$ (k is the number of maximum utilities, equal each other) then the maximum probability equals 1 and corresponds to the largest utility. Zero rationality, $q = 0$, means equiprobability of patch selection, or absolutely random choice. These facts confirm the sense of q as a measure of rationality.

Denote by V_i the amount of resources (energy) in the i-th patch, $V^* = \max\{V_j,\ j = 1, \dots, n\}$. Obviously, the patch with the maximum utility U_{i_s} may not contain the maximum resource, i. e. $V_{i_s} \neq V^*$, that depends on the form of the utility function.

Let us consider V_i, $i = 1, \dots, n$, as the values of a random variable V, and $p_i = P(V = V_i)$, determined by (1), is the probability for a population to choose the $i-$th patch. Denote by $E = E(V)$ and $D = D(V)$ the mathematical expectation and dispersion, respectively, of V. Let us consider p_i as the functions of q : $p_i = p_i(q)$, i. e. other parameters in p_i are supposed to be fixed. Then

$$E = E(V, q) = \sum_{i=1}^{n} V_i p_i(q) = \frac{1}{\sum\limits_{j=1}^{n} e^{qU_j}} \sum_{i=1}^{n} V_i e^{qU_i}.$$

In what follows, for convenience, $E(V, q)$ will be denoted as $E(q)$.

Remark 2. In what follows we write $E(\infty) = \lim\limits_{q\to\infty} E(q)$. Moreover, if $E(q) < E(\infty)$ for all $q \geqslant 0$, we write that $\max E(q) = E(\infty)$.

It is easy to show that

$$\lim_{q\to\infty} E(q) = \frac{1}{k}\sum_{s=1}^{k} V_{i_s},$$

where i_s is determined in Proposition 1.

If $U_i = V_i$ and $V_i \neq V_j$, $i, j = 1, \ldots, n$, then $\lim\limits_{q\to\infty} E(q) = V^*$. In such case

$$\max_q E(q) = V^*.$$

The latter argument and Proposition 1 motivate the following definition.

Definition 1. The rationality q^* is called optimal if

$$E(q^*) = \max_q E(q).$$

Definition 2. The choice under $q = q_1$ is more rational than that under $q = q_2$ if

$$E(q_1) > E(q_2).$$

Remark 3. Usually, $E(q)$ is considered as an average income when selecting corresponds to some distribution p_i, $i = 1, \ldots, n$. The latter justifies Definition 1 and provides one more confirmation of the interpretation of q as the value of patch selection rationality, given in [11].

Proposition 2.

$$\frac{dE(q)}{dq} = \sum_{i=1}^{n} V_i U_i p_i - \sum_{i=1}^{n} V_i p_i \cdot \sum_{i=1}^{n} U_i p_i, \tag{3}$$

where p_i are given by (1).

Proof. We have

$$\frac{dp_i(q)}{dq} = \frac{1}{\left(\sum\limits_{j=1}^{n} e^{qU_j}\right)^2} \cdot \left(U_i e^{qU_i} \cdot \sum_{j=1}^{n} e^{qU_j} - e^{qU_i} \cdot \sum_{j=1}^{n} U_j e^{qU_j}\right) = U_i p_i - p_i \cdot \sum_{j=1}^{n} U_j p_j,$$

hence

$$\frac{dE(q)}{dq} = \sum_{i=1}^{n} \frac{dp_i(q)}{dq} V_i = \sum_{i=1}^{n} V_i U_i p_i - \sum_{i=1}^{n} p_i V_i \cdot \sum_{i=1}^{n} p_i U_i.$$

□

Corollary 1. *Suppose $U_i = V_i$, $i = 1, \ldots, n$. Then*

$$\frac{dE(q)}{dq} = D(q),$$

where $D(q)$ is a dispersion of V.

Hence, if $U_i = V_i$, $i = 1, \ldots, n$, then $E(q)$ is monotonically increasing with respect to q, and $\max_q E(q) = \lim\limits_{q\to\infty} E(q) = E(\infty) = V^*$. But if there exists k such that $U_k \neq V_k$ then the monotone increasing of $E(q)$ cannot be guaranteed, and $q = \infty$ may not be the optimal rationality.

2. Bounded rationality

Let us consider the influence of imperfect information or the lack of it on the rationality of choice. Such case may be classified as the choice under bounded rationality. In [11] the following utility function U_i, characterizing the i-th patch from the point of view of population, was proposed

$$U_i = I_i V_i + (1 - I_i)\overline{V} - T(d_i), \tag{4}$$

where I_i is the population measure of the patch i awareness, $I_i \in [0, 1]$, V_i is the amount of food (energy) resource containing in the i-th patch, $\overline{V} = \gamma_1 V_1 + \ldots + \gamma_n V_n$ is the average utility of the patches for a population, $\gamma_1 + \ldots + \gamma_n = 1$, $\gamma_i \geqslant 0$, $T(d_i)$ is the cost of moving to the i-th as a function of the distance d_i to it.

As it was noted in Introduction, we consider the general problem of rational choice. Therefore, we do not use the distance parameter d_i. The aim of this paper is to give an approach to modeling the optimal rational choice. Thus we consider the following form of U_i

$$U_i = I_i V_i + (1 - I_i)\overline{V} - C_i(q), \tag{5}$$

where $C_i(q)$ is the cost of information about the patch i. It is worth to note the dependence of the cost on rational parameter q, that models the influence of rational behavior on the cost. Detailed description of C_i will be given below. In what follows, we, on the basis of (5), find the optimal rationality q^* in the sense of proposed approach $q^* = \arg\max_q E(q)$ and investigate the influence of the imperfect information on rationality of choice.

2.1. Free information

Let us consider the following utility functions:

$$U_i = I_i V_i, \ I_i \in [0; 1].$$

Such form of U_i underlines the fact that the utility of the $i-$th patch depends on available information about its resources. The latter implies the possibility of the situation when $V_i > V_j$ but $U_i < U_j$. The following proposition refines this reasoning for two patches.

Proposition 3. *Assume* $n = 2$, $U_i = I_i V_i$, $i = 1, 2$, $V_2 > V_1$.
If $I_2 V_2 > I_1 V_1$ *then* $\max_q E(q) = E(\infty) = V_2$.
If $I_2 V_2 < I_1 V_1$ *then* $\max_q E(q) = E(0) = \dfrac{V_1 + V_2}{2}$.
If $I_2 V_2 = I_1 V_1$ *then* $E(q) = E(0)$ *for any* $q > 0$.

The proof directly follows from (2) and (3). The Proposition 3 shows that the optimal rationality depends on the available information. The lack of information implies that the average income is not the largest: $\dfrac{V_1 + V_2}{2} < V_2$ in the second case.

Remark 4. It is worth to note that in the second case the choice of any patch is equally probable: $p_1 = p_2 = \dfrac{1}{2}$. This result gives an explanation of the fact that in practice a population makes a choice which is not agreeable with IFD conception. Really, the IFD asserts that the choice of population is optimal if $q = \infty$ which implies that $E(\infty) = V_1 < \dfrac{V_1 + V_2}{2}$. But our variational approach gives the value $E(0) > E(\infty)$ for the absolutely random choice confirmed by practice [11].

Remark 5. The result, analogous to Proposition 3, for $n > 2$, may be easily obtained though without such clarity as for $n = 2$.

2.2. Average utility

Let us consider the utility function taking into account the average utility of the patches $\overline{V} = \gamma_1 V_1 + \ldots + \gamma_n V_n$ about patches

$$U_i = I_i V_i + (1 - I_i)(\gamma_1 V_1 + \ldots + \gamma_n V_n), \tag{6}$$

where $\gamma_1 + \ldots + \gamma_n = 1$, $\gamma_1 \geqslant 0$. It follows from (6) that if $I_i = 1$, i. e. an agent (population) has the perfect (full) information about the resources V_i of i-th patch, then the average utility is being removed because of it uselessness. And vice-versa, if $I_i = 0$, i. e. an agent has no information about V_i, utility U_i is being replaced by average utility of patches.

Proposition 4. *Assume $n = 2$, $V_2 > V_1$, U_i, $i = 1, 2$, is determined by (6). Then*

$$\frac{dE(q)}{dq} > 0.$$

Proof. Consider $U_2 - U_1 = (I_2 V_2 + (1 - I_2)\overline{V}) - (I_1 V_1 (1 - I_1)\overline{V}) = (V_2 - \overline{V})I_2 + (\overline{V} - V_1)I_1 > 0$. Then, taking into account that $V_2 > V_1$ and (2), we finish the proof. □

Corollary 2. *Under the assumptions of Proposition 4,* $\max_q E(q) = E(q^*) = E(\infty) = V_2$.

Remark 6. It is interesting to compare the conclusions of Propositions 3 and 4. The addition of average income, $\overline{V}$, (Proposition 4), strengthens the population in its solution to choice a "rich" patch ($V_2 > V_1$): it is chosen always. If $U_i = I_1 V_i$ then the rich patch is being chosen only under the condition $I_2 V_2 > I_1 V_1$ (Proposition 3).

2.3. Information cost

Assume that the utility function of the i-th patch, without information cost, is $U_i = V_i I_i$, where $I_i \in [0, 1]$ is an agent measure of the awareness of the patch i. Suppose an agent has to pay some cost for information about the amount of resources V_i in patches. Denote by $\beta_i \in [0, V_i]$ the price of the unit of information I_i about V_i. Hence, $\beta_i I_i$ is the cost of information I_i.

Let us assume that an agent's rational behavior diminishes the information cost to the value $\beta_i I_i f_i(q)$, where $f_i(q) \in [0, 1]$ for $q \geqslant 0$.

Therefore, the utility function of the i-th patch has the form

$$U_i = I_i V_i - \beta_i I_i f_i(q).$$

Then we can easily obtain that

$$\frac{dE(q)}{dq} = \sum_{i=1}^{n} V_i p_i (U_i + q U_i') - \sum_{i=1}^{n} V_i p_i \cdot \sum_{i=1}^{n} p_i (U_i + q U_i'),$$

where $U_i' = \dfrac{dU_i}{dq} = -\beta_i I_i f_i'(q)$. To make an analysis of $E(q)$ more pictorial, let us introduce the following assumptions on $f_i(q)$ and consider the case $n = 2$. Assume that

$f_1 = f_2 = f$, which means the equality of agent's possibilities to diminish information cost for all patches;

$f(0) = 1$, which means the absence of influence of an "irrational" ($q = 0$) agent on the information cost;

$f(q) \in [0, 1]$ for $q \geqslant 0$.

It is easy to show that

$$E(q) = \frac{V_1 - V_2}{1 + e^{q(U_2 - U_1)}} + V_2, \tag{7}$$

and, hence

$$\frac{dE(q)}{dq} = \frac{(V_2 - V_1)e^{q(U_2-U_1)}}{(1 + e^{q(U_2-U_1)})^2} \cdot (U_2 - U_1 - q(U_2' - U_1')).$$

Denote

$$A = I_2V_2 - I_1V_1, \quad B = I_2\beta_2 - I_1\beta_1.$$

Then, $U_2 - U_1 - q(U_2' - U_1') = A - Bf(q) + qBf'(q) = B\left(\frac{A}{B} - (f(q) - qf'(q))\right)$. Denote $g(q) = f(q) - qf'(q)$. Then

$$\frac{dE(q)}{dq} = \frac{(V_2 - V_1)e^{q(U_2-U_1)}}{(1 + e^{q(U_2-U_1)})^2} \cdot B\left(\frac{A}{B} - g(q)\right). \tag{8}$$

Hence, we can formulate the following obvious proposition.

Proposition 5. *Assume $V_2 > V_1$, $f \in C^1[0, \infty)$. Then*

$\max_q E(q) = E(q^*)$ if $\frac{A}{B} > 1$ and there exists the unique $q^* > 0$ such that $g(q^*) = \frac{A}{B}$, $g(q) < \frac{A}{B}$ for $q \in (q^*, \infty)$;

$\max_q E(q) = E(\infty)$ if $\frac{A}{B} - g(q) > 0$ for $q \geqslant 0$;

$\max_q E(q) = E(0) = \frac{V_1 + V_2}{2}$ if $\frac{A}{B} - g(q) < 0$ for $q \geqslant 0$.

The proof directly follows from (8). Proposition 4 shows the influence of $f(q)$, which may be considered as the rational cost behavior, on the optimal rationality.

Example 1. $f_i(q) = \frac{1}{1 + \alpha_i q}$, $\alpha_i \geqslant 0$, where α_i is the measure of influence of q on the cost of information I_i.

We have

$$\frac{dE(q)}{dq} = \frac{(V_2 - V_1)e^{q(U_2-U_1)}}{\left(1 + e^{q(U_2-U_1)}\right)^2}\left(A - \left(\frac{\beta_2 I_2}{1 + \alpha_2 q} - \frac{\beta_1 I_1}{1 + \alpha_1 q}\right) - q\left(-\frac{\alpha_2\beta_2 I_2}{(1 + \alpha_2 q)^2} + \frac{\alpha_1\beta_1 I_1}{(1 + \alpha_1 q)^2}\right)\right).$$

Assume, for simplicity of transformations, that $\alpha_1 = \alpha_2 = 1$. Denote $x = \frac{1}{1 + q}$. From the equality above, it is easy to obtain

$$\frac{dE(q)}{dq} = \frac{(V_2 - V_1)e^{q(U_2-U_1)}}{\left(1 + e^{q(U_2-U_1)}\right)^2}(A - B \cdot x^2). \tag{9}$$

Thus, we can obtain the following result, a special case of Proposition 4.

Proposition 6. *Assume $V_2 > V_1$. Then*

1. *$\frac{dE(q)}{dq} = 0$ for $q = q^*$ if and only if $\frac{B}{A} > 1$, and with this $q^* = \sqrt{\frac{\beta_2 I_2 - \beta_1 I_1}{I_2 V_2 - I_1 V_1}} - 1$.*
2. *If $A > 0$, $B > 0$, $A < B$, then $E(q^*) = \min_q E(q)$, $\lim\limits_{q\to\infty} E(q) = V_2$.*
3. *If $A < 0$, $B < 0$, $B < A$, then $E(q^*) = \max_q E(q)$, $\lim\limits_{q\to\infty} E(q) = V_1$.*

The proof easily follows from (9), (7) and $U_2 - U_1 = A - \dfrac{B}{1+q}$.

Figures 1 and 2 illustrate cases 2 and 3 of the Proposition 6 respectively. Figures 3 and 4 illustrate behaviour of function $h(q) = A - B\dfrac{1}{(1+q)^2}$ depending on the parameters V_i, I_i, β_i, $i = 1, 2$.

At the same time, figures 1 and 3, as well as 2 and 4, reflect the influence of the function $h(q)$ on the sign of the derivative (9) and, accordingly, it's influence on the optimal value of the mathematical expectation (7).

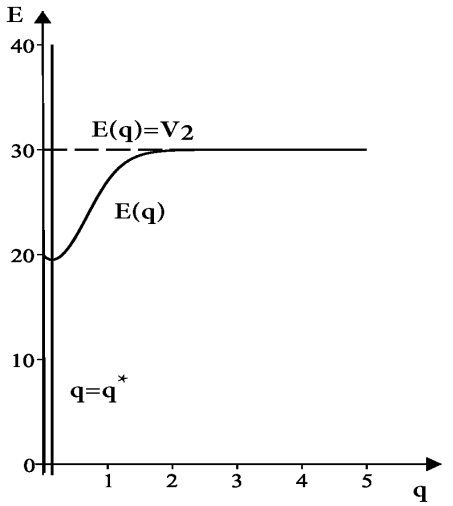


Fig. 1. Expected value $E(q)$ with $V_1 = 10$, $V_2 = 30$, $I_1 = 0,4$, $I_2 = 0,7$, $\beta_1 = 25$, $\beta_1 = 5$.

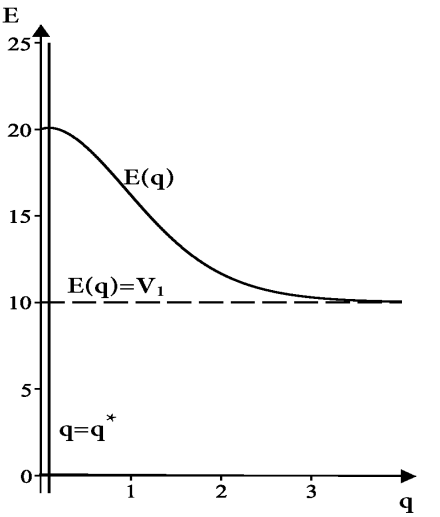


Fig. 2. Expected value $E(q)$ with $V_1 = 10$, $V_2 = 30$, $I_1 = 0,2$, $I_2 = 0,8$, $\beta_1 = 20$, $\beta_1 = 8$.

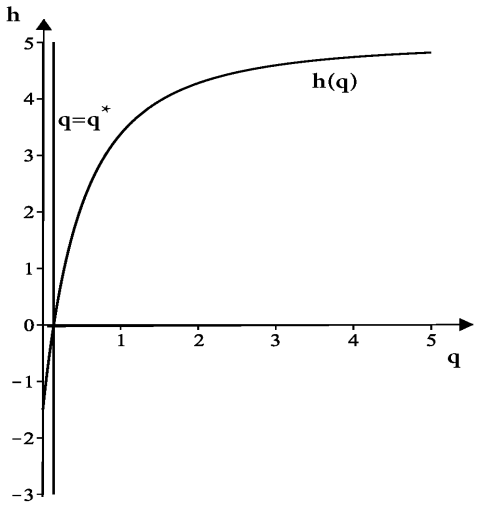


Fig. 3. $h(q)$, $V_1 = 10$, $V_2 = 30$, $I_1 = 0,4$, $I_2 = 0,7$, $\beta_1 = 25$, $\beta_1 = 5$.

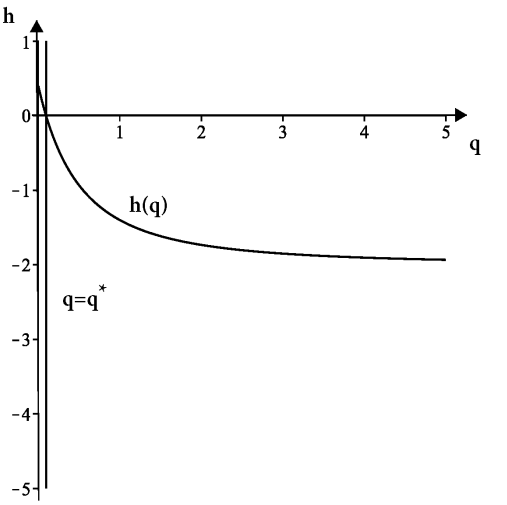


Fig. 4. $h(q)$, $V_1 = 10$, $V_2 = 30$, $I_1 = 0,2$, $I_2 = 0,8$, $\beta_1 = 20$, $\beta_1 = 8$.

3. Variable rationality

1. It is naturally to suppose that rationality of selection is the function of information, available to population, about patches. Hence, in what follows, we set $q = q(I_1, \ldots, I_n)$.

In order not to complicate transformations below, let us consider the case of two patches, $q = q(I_1, I_2)$ and $U_i = I_i V_i$, $i = 1, 2$.

We have

$$p_i = \frac{e^{q(I_1,I_2)U_i}}{\sum\limits_{j=1}^{2} e^{q(I_1,I_2)U_j}},$$

and $E = E(I_1, I_2)$.

Suppose, q is a differentiable function. Denote $g = g(I_1, I_2) = q(I_2V_2 - I_1V_1)$, $q_i = \frac{\partial q}{\partial I_i}$, $g_i = \frac{\partial g}{\partial I_i}$, $i = 1, 2$. Then

$$\frac{\partial E}{\partial I_i} = (V_1 - V_2)\frac{e^g g_i}{(1 + e^g)^2},$$

where

$$g_1 = q_1(I_2V_2 - I_1V_1) - qV_1, \quad g_2 = q_2(I_2V_2 - I_1V_1) + qV_2.$$

If $(I_1,\ I_2)$ is the stationary point of $E(I_1, I_2)$ then

$$\begin{cases} g_1(I_1, I_2) = q_1(I_2V_2 - I_1V_1) - qV_1 &= 0, \\ g_2(I_1, I_2) = q_2(I_2V_2 - I_1V_1) + qV_2 &= 0. \end{cases}$$

The necessary condition for solvability of this system with respect to I_1, I_2 is the following equality

$$V_1\frac{\partial q}{\partial I_2} + V_2\frac{\partial q}{\partial I_1} = 0.$$

Therefore, if $\frac{\partial q}{\partial I_2} \cdot \frac{\partial q}{\partial I_1} > 0$ then $E(I_1, I_2)$ has no points of extremum. It is impossible to obtain interesting results about optimality of E without specification of $q = q(I_1, I_2)$.

2. Now, consider the following case

$$p_i = \frac{e^{q_iU_i}}{\sum\limits_{j=1}^{n} e^{q_jU_j}},$$

i. e. the rationality of choice depends on the patch. For example, an agent uses different methods of acquisition of information about patch resources. To illustrate the influence of the lack of information on the rationality of choice, let us consider a special case: $n = 2$, $q_1 = q(I)$, $q_2 = q(\alpha I)$, $I \in [0, 1]$, $\alpha \in [0, 1)$, $U_1 = I_1V_1 = IV_1$, $U_2 = I_1V_2 = \alpha IV_2$. Hence

$$p_1 = \frac{e^{q(I)IV_1}}{e^{q(I)IV_1} + e^{q(\alpha I)\alpha IV_2}}, \quad p_2 = \frac{e^{q(\alpha I)IV_1}}{e^{q(I)IV_1} + e^{q(\alpha I)\alpha IV_2}}.$$

We consider E as the function of $I : E = E(I)$. Let us assume that $q(0) = 0$. Really, the rationality of choice, obviously, equals zero if an agent has no information about a patch. Moreover, if $q = 0$ then it is equally probable to choose any patch, as it was noted above.

From (7)

$$E(I) = (V_1 - V_2) \cdot \frac{1}{1 + e^{g(I)}} + V_2,$$

where $g(I) = q(\alpha I)\alpha IV_2 - q(I)IV_1$. Then

$$E'(I) = (V_2 - V_1) \cdot \frac{e^{g(I)}}{(1 + e^{g(I)})^2} \cdot g'(I),$$

where $g'(I) = \frac{dq(u)}{du}|_{u=\alpha I}\alpha^2V_2I + q(\alpha I)\alpha V_2 - (q'(I)V_1I + q(I)V_1)$. We have an obvious proposition.

Proposition 7. *Assume* $V_2 > V_1$.
If $g'(I) > 0$, *for any* $I \in (0;1)$, *then* $\max_I E(I) = E(1)$.
If $g'(I) < 0$, *for any* $I \in (0;1)$, *then* $\max_I E(I) = E(0)$.

Let us discuss this result. We see that the maximum of available information, $I = 1$, does not guarantee the maximum value of $E(I)$. In particular, an additional, sufficient, condition is required, namely $g'(I) > 0$. The second case, $g'(I) < 0$, provides the sufficient condition of absolutely irrational behavior. An agent is unable to dispose of available information.

Conclusion

In this paper the problem of rational for population patch choice was considered. The approach based on the idea of the Boltzmann distribution was proposed to define the optimal patch choice. Utility functions were used to construct the Boltzmann distribution. The methods for analysis of the rationality of the patch choice were proposed: analysis of the mathematical expectation as a function depending on the parameter q (Optimal rationality, Bounded rationality) and as a function depending on the measure of the patch awareness I (Variable rationality). The influence of available information about the amount of resources contained in the patch on the decision-making process of patch choice was investigated.

References

[1] R. B. Aumann, "Rationality and bounded rationality", *Games and econimic behavior*, vol. 21, no. 1, pp. 2–14, 1997.

[2] P. A. Ortega, D. A. Braun, J. Dyer, K.-E. Kim, and N. Tishby, *Information-theoretic bounded rationality*, preprint, 2015. arXiv: 1512.06789 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1512.06789.

[3] D. A. Braun and P. A. Ortego, "Information-theoretic bounded rationality and ε-optimality", *Entropy*, vol. 16, pp. 4662–4676, 2014.

[4] M. D. Breed and J. Moore, *Encyclopedia of animal behavior*. Elsevier Ltd., 2019, 889 pp.

[5] E. Kagan and I. Ben-Gal, *Search and foraging individual motion and swarm dynamics*. Taylor and Francis Group, LLC, 2015, 268 pp.

[6] B. Y. Hayden and M. E. Walton, "Neuroscience of foraging", *Frontiers in Neuroscience*, vol. 8, p. 81, 2014.

[7] D. L. Barack, C. S. W., and P. M. L., "Posterior cingulate neurons dynamically signal decisions to disengage during foraging", *Neuron*, vol. 96, no. 2, pp. 339–347, 2017.

[8] J. S. Greene, M. Brown, M. Dobosiewicz, I. G. Ishida, E. Z. Macosko, X. Zhang, R. A. Butcher, D. J. Cline, P. T. McGrath, and C. I. Bargmann, "Balancing selection shapes density-dependent foraging behaviour", *Nature*, vol. 539, pp. 254–258, 2016.

[9] R. Cressman and V. Krivan, "The ideal free distribution as an evolutionarily stable state in density-dependent population games", *Oikos*, vol. 119, no. 8, pp. 1231–1242, 2010.

[10] R. Cressman and V. Krivan, "Two-patch population models with adaptive dispersal: The effects of varying dispersal speeds", *Mathematical Biology*, vol. 67, pp. 329–358, 2013.

[11] M. Shuichi, R. Arlinghaus, and U. Dieckmann, "Foraging on spatially distributed resources with suboptimal movement, imperfect information, and travelling costs: Departures from the ideal free distribution", *Oikos*, vol. 119, no. 9, pp. 1469–1483, 2010.

[12] L. D. Landau and E. M. Lifshitz, *Statistical physics*. Nauka, 1976, 584 pp.

[13] I. P. Kornfeld, Y. G. Sinai, and S. V. Fomin, *Ergodic theory*. Nauka, 1980, 384 pp.

[14] R. Bowen, *Methods of symbolic dynamics*. Mir, 1979, 244 pp.

[15] C. J. C. H. Watkins and P. Dayan, "Technical note Q-Learning", *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[16] A. Kianercy and A. Galstyan, "Dynamics of Boltzmann Q learning in two-player two-action games", *Physical review*, vol. 85, no. 4, p. 041 145, 2012.

[17] P. A. Ortega and D. A. Braun, "Thermodynamics as a theory of decision-making with information-processing costs", *Proceedings of the Royal Society*, vol. 469, no. 2153, p. 20 120 683, 2013.

[18] S. K. Mitter and N. J. Newton, "Information and entropy flow in the Kalman-Bucy filter", *Journal of Statistical Physics*, vol. 118, pp. 145–176, 2005.

[19] P. Pirolli, *Information foraging theory*. Oxford university press, 2007, 221 pp.

[20] K. Lerman and A. Galstyan, "Mathematical model of foraging in a group of robots: Effect of interference", *Autonomous robots*, vol. 13, pp. 127–141, 2002.

[21] A. N. Kirillov and I. V. Danilova, "Dynamics of population patch distribution", *Modeling and Analysis of Information Systems*, vol. 25, no. 3, pp. 268–275, 2018, in Russian.

[22] A. N. Kirillov and I. V. Danilova, "Utility function in the foraging problem with imperfect information", *Information and Control Systems*, vol. 105, no. 2, pp. 71–77, 2020.

[23] I. V. Danilova, A. N. Kirillov, and A. A. Krizhanovsky, "Boltzmann distribution in relation to the problem of population migration", *Proceedings of Voronezh State University. Series: Systems Analysis and Information Technologies*, no. 2, pp. 92–102, 2020, in Russian.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

On a geometric approach to the estimation of interpolation projectors

M. V. Nevskii[1], A. Y. Ukhalov[1]

DOI: 10.18255/1818-1015-2023-3-246-257

[1]P.G. Demidov Yaroslavl State University, 14 Sovetskaya str., Yaroslavl 150003, Russia.

MSC2020: 41A05, 52B55, 52C07
Research article
Full text in Russian

Received July 4, 2023
After revision August 11, 2023
Accepted August 16, 2023

Suppose Ω is a closed bounded subset of $\mathbb{R}^n$, S is an n-dimensional non-degenerate simplex, $\xi(\Omega;S) := \min\{\sigma \geqslant 1 : \Omega \subset \sigma S\}$. Here σS is the result of homothety of S with respect to the center of gravity with coefficient σ. Let $d \geqslant n+1$, $\varphi_1(x), \ldots, \varphi_d(x)$ be linearly independent monomials in n variables, and $\varphi_1(x) \equiv 1$, $\varphi_2(x) = x_1, \ldots, \varphi_{n+1}(x) = x_n$. Put $\Pi := \operatorname{lin}(\varphi_1, \ldots, \varphi_d)$. The interpolation projector $P : C(\Omega) \to \Pi$ with a set of nodes $x^{(1)}, \ldots, x^{(d)} \in \Omega$ is defined by equalities $Pf\left(x^{(j)}\right) = f\left(x^{(j)}\right)$. Denote by $\|P\|_\Omega$ the norm of P as an operator from $C(\Omega)$ to $C(\Omega)$. Consider the mapping $T : \mathbb{R}^n \to \mathbb{R}^{d-1}$ of the form $T(x) := (\varphi_2(x), \ldots, \varphi_d(x))$. We have $\frac{1}{2}\left(1 + \frac{1}{d-1}\right)(\|P\|_\Omega - 1) + 1 \leqslant \xi(T(\Omega);S) \leqslant \frac{d}{2}(\|P\|_\Omega - 1) + 1$, where S is a $(d-1)$-dimensional simplex with vertices $T\left(x^{(j)}\right)$. We discuss this and other relations for polynomial interpolation of functions continuous on a segment. Some results of numerical analysis are presented.

Keywords: polynomial interpolation; projector; norm; absorption coefficient; esimation

INFORMATION ABOUT THE AUTHORS

Mikhail V. Nevskii corresponding author	orcid.org/0000-0002-6392-7618. E-mail: mnevsk55@yandex.ru Head of the Chair, Doctor of Science, Docent.
Alexey Y. Ukhalov	orcid.org/0000-0001-6551-5118. E-mail: alex-uhalov@yandex.ru Assistant Professor, Cand. of Science, Docent.

For citation: M. V. Nevskii and A. Y. Ukhalov, "On a geometric approach to the estimation of interpolation projectors", *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 246-257, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

О геометрическом подходе к оцениванию интерполяционных проекторов

М. В. Невский[1], А. Ю. Ухалов[1]

DOI: 10.18255/1818-1015-2023-3-246-257

[1]Ярославский государственный университет им. П.Г. Демидова, ул. Советская, 14, Ярославль, 150003, Россия.

УДК 514.17, 517.51, 519.6
Научная статья
Полный текст на русском языке

Получена 4 июля 2023 г.
После доработки 11 августа 2023 г.
Принята к публикации 16 августа 2023 г.

Пусть Ω — замкнутое ограниченное подмножество $\mathbb{R}^n$, S — n-мерный невырожденный симплекс, $\xi(\Omega;S) := \min\{\sigma \geqslant 1 : \Omega \subset \sigma S\}$. Здесь σS есть результат гомотетии S относительно центра тяжести с коэффициентом σ. Пусть $d \geqslant n+1$, $\varphi_1(x),\dots,\varphi_d(x)$ — линейно независимые мономы от n переменных, причём $\varphi_1(x) \equiv 1$, $\varphi_2(x) = x_1, \dots, \varphi_{n+1}(x) = x_n$. Положим $\Pi := \operatorname{lin}(\varphi_1,\dots,\varphi_d)$. Интерполяционный проектор $P : C(\Omega) \to \Pi$ по набору узлов $x^{(1)},\dots,x^{(d)} \in \Omega$ определяется с помощью равенств $Pf\left(x^{(j)}\right) = f\left(x^{(j)}\right)$. Обозначим через $\|P\|_\Omega$ норму P как оператора из $C(\Omega)$ в $C(\Omega)$. Рассмотрим отображение $T : \mathbb{R}^n \to \mathbb{R}^{d-1}$, имеющее вид $T(x) := (\varphi_2(x),\dots,\varphi_d(x))$. Справедливы неравенства $\frac{1}{2}\left(1+\frac{1}{d-1}\right)(\|P\|_\Omega - 1) + 1 \leqslant \xi(T(\Omega);S) \leqslant \frac{d}{2}(\|P\|_\Omega - 1) + 1$, где S — $(d-1)$-мерный симплекс с вершинами $T\left(x^{(j)}\right)$. В статье это и другие соотношения обсуждаются для полиномиальной интерполяции функций, непрерывных на отрезке. Приводятся некоторые результаты численного анализа.

Ключевые слова: полиномиальная интерполяция; проектор; норма; коэффициент поглощения; оценивание

ИНФОРМАЦИЯ ОБ АВТОРАХ

Михаил Викторович Невский
автор для корреспонденции
orcid.org/0000-0002-6392-7618. E-mail: mnevsk55@yandex.ru
Заведующий кафедрой, доктор физ.-мат. наук, доцент.

Алексей Юрьевич Ухалов
orcid.org/0000-0001-6551-5118. E-mail: alex-uhalov@yandex.ru
Доцент, кандидат физ.-мат. наук, доцент.

Для цитирования: M. V. Nevskii and A. Y. Ukhalov, “On a geometric approach to the estimation of interpolation projectors”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 246-257, 2023.

1. Основные определения и соотношения

Пусть Ω — замкнутое ограниченное подмножество $\mathbb{R}^n$. Под $C(\Omega)$ понимается пространство непрерывных функций $f : \Omega \to \mathbb{R}$ с равномерной нормой

$$\|f\|_{C(\Omega)} := \max_{x\in\Omega} |f(x)|.$$

Для невырожденного симплекса $S \subset \mathbb{R}^n$ через σS обозначим результат гомотетии S относительно центра тяжести с коэффициентом σ. Положим $\xi(\Omega; S) := \min\{\sigma \geqslant 1 : \Omega \subset \sigma S\}$. Включение $\Omega \subset S$ эквивалентно равенству $\xi(\Omega; S) = 1$. По нашей терминологии величина $\xi(\Omega; S)$ называется *коэффициентом поглощения множества* Ω *симплексом* S. Ниже $\mathrm{ver}(S)$ есть совокупность вершин S.

Пусть $d \in \mathbb{N}$, $d \geqslant n+1$; $\varphi_1(x), \ldots, \varphi_d(x)$ — линейно независимые функции, представляющие собой мономы от n переменных вида $x^\alpha = x_1^{\alpha_1} \ldots x_n^{\alpha_n}$. Здесь $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}_+^n$. Предполагается, что $\varphi_1(x) \equiv 1$, $\varphi_2(x) = x_1$, $\ldots$, $\varphi_{n+1}(x) = x_n$. Под d-мерным пространством многочленов от n переменных будем понимать совокупность $\Pi := \mathrm{lin}(\varphi_1, \ldots, \varphi_d)$. Отметим важные варианты $\Pi = \Pi_k(\mathbb{R}^n)$ — пространство многочленов общей степени $\leqslant k$ $(k \in \mathbb{N})$ и $\Pi = \Pi_\alpha(\mathbb{R}^n)$ — пространство многочленов степени $\leqslant \alpha_i$ по x_i $(\alpha \in \mathbb{N}^n)$.

Совокупность точек $x^{(1)}, \ldots, x^{(d)} \in \Omega$ называется *допустимым набором узлов* для интерполяции функций из $C(\Omega)$ с помощью многочленов из Π, если $\Delta := \det(\mathbf{A}) \neq 0$. Здесь и ниже $\mathbf{A}$ есть $(d \times d)$-матрица

$$\mathbf{A} := \begin{pmatrix} 1 & \varphi_2\left(x^{(1)}\right) & \ldots & \varphi_d\left(x^{(1)}\right) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \varphi_2\left(x^{(d)}\right) & \ldots & \varphi_d\left(x^{(d)}\right) \end{pmatrix}.$$

Интерполяционный проектор $P : C(\Omega) \to \Pi$ по этому набору узлов определяется с помощью равенств $Pf\left(x^{(j)}\right) = f\left(x^{(j)}\right)$, $j = 1, \ldots, d$. Аналогом интерполяционной формулы Лагранжа является представление

$$Pf(x) = \sum_{j=1}^{d} f\left(x^{(j)}\right) \lambda_j(x), \quad \lambda_j(x) := \frac{\Delta_j(x)}{\Delta}, \tag{1}$$

где $\Delta_j(x)$ — определитель, который получается из Δ заменой j-й строки на строку $\left(\varphi_1(x), \ldots, \varphi_d(x)\right)$. Многочлены $\lambda_j \in \Pi$ обладают свойством $\lambda_j\left(x^{(k)}\right) = \delta_j^k$. Их коэффициенты в базисе $\varphi_1, \ldots, \varphi_d$ составляют столбцы матрицы $\mathbf{A}^{-1}$. Мы называем λ_j *базисными многочленами Лагранжа* проектора P.

В случае $\Pi = \Pi_1(\mathbf{R}^n)$, $d = n+1$ и $\varphi_j(x) = x_{j-1}$ $(j = 2, \ldots, d)$ многочлены λ_j также называются *базисными многочленами Лагранжа симплекса* S с вершинами в узлах интерполяции. В этой ситуации, если $\Omega \not\subset S$,

$$\xi(\Omega; S) = (n+1) \max_{1\leqslant k\leqslant n+1} \max_{x\in\Omega} (-\lambda_k(x)) + 1. \tag{2}$$

Для выпуклого Ω равенство (2) доказывается в [1]; в общем случае доказательство проводится по той же схеме. Заметим, что $\xi(\mathrm{conv}(\Omega); S) = \xi(\Omega; S)$.

Ниже рассматриваются лишь допустимые наборы узлов и те множества Ω, каждое из которых содержит такой набор.

Обозначим через $\|P\|_\Omega$ норму P как оператора из $C(\Omega)$ в $C(\Omega)$. Из (1) следует, что

$$\|P\|_\Omega = \max_{x\in\Omega} \sum_{j=1}^{d} |\lambda_j(x)|. \tag{3}$$

Равенства $\lambda_j(x) = \dfrac{\Delta_j(x)}{\Delta}$ эквивалентны матричному соотношению

$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ \varphi_2\left(x^{(1)}\right) & \varphi_2\left(x^{(2)}\right) & \dots & \varphi_2\left(x^{(d)}\right) \\ \vdots & \vdots & & \vdots \\ \varphi_d\left(x^{(1)}\right) & \varphi_d\left(x^{(2)}\right) & \dots & \varphi_d\left(x^{(d)}\right) \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_d \end{pmatrix} = \begin{pmatrix} 1 \\ \varphi_2(x) \\ \vdots \\ \varphi_d(x) \end{pmatrix}. \tag{4}$$

В дальнейшем важную роль будет играть отображение $T : \mathbb{R}^n \to \mathbb{R}^{d-1}$, определяемое равенством

$$y = T(x) := (\varphi_2(x), \dots, \varphi_d(x)) = (x_1, \dots, x_n, \varphi_{n+1}(x), \dots, \varphi_d(x)).$$

Мы будем рассматривать T на множестве Ω. Отмеченный выше выбор первых мономов $\varphi_j(x)$ обеспечивает обратимость T. Обозначим $y^{(j)} := T\left(x^{(j)}\right)$. Соотношение (4) равносильно

$$\lambda_1 y^{(1)} + \dots + \lambda_d y^{(d)} = y, \quad \sum_{j=1}^{d} \lambda_j = 1.$$

Это означает, что числа $\lambda_j(x)$ являются барицентрическими координатами точки $y = T(x)$ относительно $(d-1)$-мерного симплекса с вершинами $y^{(j)}$. Поэтому

$$\|P\|_\Omega = \max_{x \in \Omega} \sum_{j=1}^{d} |\lambda_j(x)| = \max \left\{ \sum_{j=1}^{d} |\beta_j| : \ \sum_{j=1}^{d} \beta_j = 1, \ y = \sum_{j=1}^{d} \beta_j y^{(j)} \in T(\Omega) \right\}. \tag{5}$$

Правое равенство в (5) выражает норму проектора P через барицентрические координаты точек множества $T(\Omega)$ относительно невырожденного $(d-1)$-мерного симплекса с вершинами $y^{(j)}$.

Через $\theta(\Pi; \Omega)$ обозначим минимальную величину нормы проектора $P : C(\Omega) \to \Pi$ при условии, что соответствующие P узлы интерполяции принадлежат Ω :

$$\theta(\Pi; \Omega) := \min_{x^{(j)} \in \Omega} \|P\|_\Omega.$$

Проектор, норма которого равна $\theta(\Pi; \Omega)$, будем называть *минимальным*. Введём в рассмотрение следующую числовую характеристику множества Ω, представляющую собой *минимальный коэффициент поглощения* этого множества невырожденными симплексами с вершинами в Ω:

$$\xi_n(\Omega) := \min \left\{ \xi(\Omega; S) : \ S - n\text{-мерный симплекс}, \operatorname{ver}(S) \subset \Omega, \ \operatorname{vol}(S) \neq 0 \right\}.$$

Пусть $x^{(1)}, \dots, x^{(d)}$ — допустимый набор узлов интерполяции функций из $C(\Omega)$ с помощью многочленов из Π, $P : C(\Omega) \to \Pi$ — соответствующий проектор. Тогда точки $y^{(j)} = T\left(x^{(j)}\right)$ составляют допустимый набор узлов интерполяции функций из $C(T(\Omega))$ с помощью многочленов из $\Pi_1\left(\mathbb{R}^{d-1}\right)$. Рассмотрим интерполяционный проектор $\overline{P} : C(T(\Omega)) \to \Pi_1\left(\mathbb{R}^{d-1}\right)$ по системе узлов $y^{(1)}, \dots, y^{(d)}$. Если $f \in C(\Omega)$, $g \in C(T(\Omega))$ и $g(y) = f(x)$ при $y = T(x)$, то равенствам $Pf\left(x^{(j)}\right) = f_j$ соответствуют равенства $\overline{P}g\left(y^{(j)}\right) = g_j := g\left(y^{(j)}\right)$. Поэтому интерполяционные многочлены $p \in \Pi$ и $q \in \Pi_1\left(\mathbb{R}^{d-1}\right)$ также связаны соотношением $p(x) = q(y)$. Пусть $\lambda_j \in \Pi$ — базисные многочлены Лагранжа проектора P, μ_j — базисные многочлены Лагранжа проектора $\overline{P}$ (или симплекса $S = \operatorname{conv}\left(y^{(1)}, \dots, y^{(d)}\right) \subset \mathbb{R}^{d-1}$), то $\mu(y) = \mu(T(x)) = \lambda_j(x)$. Следовательно,

$$\|\overline{P}\|_{T(\Omega)} = \|P\|_\Omega.$$

В связи с этим при оценивании нормы $\|P\|_\Omega$ оказывается возможным применить геометрические неравенства для нормы проектора $\overline{P}$ при линейной интерполяции на $(d-1)$-мерном множестве $T(\Omega)$. Этот подход был предложен М. В. Невским; полученные им результаты содержатся в статьях [2], [3] и монографии [1]. Приведём нужные нам утверждения.

Для проектора $P : C(\Omega) \to \Pi$ с узлами $x^{(j)}$ справедливо неравенство

$$\frac{1}{2}\left(1+\frac{1}{d-1}\right)(\|P\|_\Omega - 1) + 1 \leqslant \xi(T(\Omega); S) \leqslant \frac{d}{2}(\|P\|_\Omega - 1) + 1, \tag{6}$$

где S — $(d-1)$-мерный симплекс с вершинами $T\left(x^{(j)}\right)$. Справедливы соотношения

$$\frac{1}{2}\left(1+\frac{1}{d-1}\right)(\theta(\Pi;\Omega) - 1) + 1 \leqslant \xi_{d-1}(T(\Omega)) \leqslant \frac{d}{2}(\theta(\Pi;\Omega) - 1) + 1. \tag{7}$$

Если $\|P\|_\Omega \neq 1$, то (6) можно записать в виде

$$\frac{1}{2}\left(1+\frac{1}{d-1}\right) \leqslant \frac{\xi(T(\Omega); S) - 1}{\|P\|_\Omega - 1} \leqslant \frac{d}{2}. \tag{8}$$

При $\theta(\Pi;\Omega) \neq 1$ соотношение (7) равносильно

$$\frac{1}{2}\left(1+\frac{1}{d-1}\right) \leqslant \frac{\xi_{d-1}(T(\Omega)) - 1}{\theta(\Pi;\Omega) - 1} \leqslant \frac{d}{2}. \tag{9}$$

Точку $y = T(x) \in T(\Omega)$ назовём 1-точкой относительно симплекса $S = \operatorname{conv}\left(y^{(1)}, \ldots, y^{(d)}\right)$, если

$$\|P\|_\Omega = \sum_{j=1}^{d} |\lambda_j(x)|$$

и среди чисел $\lambda_j(x)$ имеется ровно одно отрицательное. Если такая точка существует, то правое неравенство в (6) становится равенством. Последнее утверждение доказано в [2] в эквивалентном виде. Понятия 1-вершины куба и 1-точки произвольного множества были введены соответственно в [3] и [4].

Если 1-точка множества $T(\Omega)$ существует для симплекса $S = \operatorname{conv}\left(T\left(x^{(1)}\right), \ldots, T\left(x^{(d)}\right)\right)$ такого, что $\xi(T(\Omega); S) = \xi_{d-1}(T(\Omega))$, то правое соотношение в (7) является равенством и проектор $P : C(\Omega) \to \Pi$ с узлами $x^{(j)}$ является минимальным.

Случай $\Pi = \Pi_1(\mathbb{R}^n)$ подробно исследовался в цикле работ авторов (см., например, [3], [1], [5], [4] и библиографию в этих работах). В этой ситуации $d = \dim \Pi_1(\mathbb{R}^n) = n+1$; отображение T является тождественным. Наиболее интересные результаты были получены в случае, когда Ω — n-мерный куб или n-мерный евклидов шар.

В настоящей статье соотношения (6)–(9) обсуждаются для полиномиальной интерполяции на отрезке. Всюду далее $n = 1$, $\Pi = \Pi_k(\mathbb{R}^1)$, $k \geqslant 1$, $d = \dim \Pi_k(\mathbb{R}^1) = k+1$, $\varphi_j(x) = x^j$ $(1 \leqslant j \leqslant k)$. Возьмём $\Omega = [-1, 1]$, тогда

$$T(\Omega) = T([-1,1]) = \{(x, \ldots, x^k) \in \mathbb{R}^k : -1 \leqslant x \leqslant 1\}.$$

В пунктах 2–4 рассматриваются случаи $k = 2, 3, 4$. В пункте 5 собраны численные оценки величин $\theta\left(\Pi_k\left(\mathbb{R}^1\right); [-1,1]\right)$ и $\xi_k\left(T([-1,1])\right)$ при $1 \leqslant k \leqslant 10$. Наконец, в пунктах 6–7 приводится материал, касающийся равномерных узлов и узлов Чебышёва. В статье отмечаются все найденные случаи, когда справа в (6)–(9) выполняются равенства.

Численные результаты получены А. Ю. Ухаловым, подробные вычислительные данные размещены в базе Mendeley Data [6]. При проведении вычислений применялась система компьютерной математики Wolfram Mathematica (см., например, [7], [8], [9]). Использовались также специально написанные программы на языке C++. Для обращения матриц и решения экстремальных задач использовались функции библиотеки DLIB (см. [10]).

2. Квадратичная интерполяция на отрезке

Простейший случай интерполяции многочленами степени выше первой — квадратичная интерполяция на отрезке. Аналитическое решение задачи о минимальном проекторе при указанном геометрическом подходе дано в [2], компьютерные методы применялись в [11]. Рассмотрим этот случай в качестве иллюстрации.

Известно (см., например, [12]), что минимальная величина нормы интерполяционного проектора в этой ситуации равна $5/4$ и эта величина реализуется для равномерных узлов. Покажем, как отмеченный результат получается с помощью (6)–(7). Дополнительно получается, что минимальных проекторов здесь бесконечно много.

Пусть $\Pi = \Pi_2\left(\mathbb{R}^1\right)$. Тогда $d = \dim \Pi = 3$, т. е. $k = d - 1 = 2$. Отображение T имеет вид $x \longmapsto (x, x^2)$, и множество $T(\Omega) = T([-1,1]) = \{(x, x^2) \in \mathbb{R}^2 : -1 \leqslant x \leqslant 1\}$ есть часть параболы. Для узлов интерполяции $-1 \leqslant r < s < t \leqslant 1$ имеем

$$\mathbf{A} = \begin{pmatrix} 1 & r & r^2 \\ 1 & s & s^2 \\ 1 & t & t^2 \end{pmatrix},$$

симплекс S представляет собой треугольник с вершинами (r, r^2), (s, s^2), (t, t^2), расположенными на этой части параболы. Процесс поглощения таким треугольником параболического сектора изображён на рис. 1.

Из выпуклости функции $\psi(x) = x^2$ следует, что 1-точка множества $T([-1,1])$ относительно симплекса S существует для любых узлов. Значит, справа в (6) имеет место равенство:

$$\xi(T(\Omega); S) = \frac{3}{2}\left(\|P\|_\Omega - 1\right) + 1 = \frac{3\|P\|_\Omega - 1}{2}. \tag{10}$$

Поскольку (10) справедливо для любого проектора $P : C[-1,1] \to \Pi_2(\mathbb{R}^1)$, то имеет место и равенство справа в (7):

$$\xi_2(T(\Omega)) = \frac{3\theta\left(\Pi; \Omega\right) - 1}{2}.$$

Таким образом, в квадратичном случае нахождение минимальной нормы проектора $\theta\left(\Pi; \Omega\right)$ эквивалентно вычислению $\xi_2(T(\Omega))$, т. е. минимального коэффициента поглощения треугольником указанной части параболы. В качестве узлов минимального проектора надо взять первые координаты вершин обнаруженного в итоге треугольника. Техническим путём задача редуцируется к треугольнику S с вершинами $(-r, r^2)$, $(0,0)$, (r, r^2), $0 < r \leqslant 1$. Для него экстремальными точками $y \in T(\Omega)$ могут быть лишь $(\pm 1, 1)$, $\left(\pm\dfrac{r}{2}, \dfrac{r^2}{4}\right)$. Две последние точки определяются тем, что в каждой из них касательная к параболе параллельна боковой стороне S. Вычисления дают

$$\xi(T(\Omega); S) = \max\left(\frac{11}{8}, \frac{3}{r^2} - 2\right), \quad \|P\|_\Omega = \max\left(\frac{5}{4}, \frac{2}{r^2} - 1\right).$$

Интересно, что при

$$\frac{2\sqrt{2}}{3} = 0.942809\ldots \leqslant r \leqslant 1$$

эти величины не зависят от r и равны соответственно $11/8$ и $5/4$, причём это минимальные возможные значения. Значит,

$$\xi_2(T(\Omega)) = \frac{11}{8}, \quad \theta\left(\Pi; \Omega\right) = \frac{5}{4}.$$

Минимальным является любой проектор с узлами $-r$, 0, r при $r \in \left[\dfrac{2\sqrt{2}}{3}, 1\right]$. Других минимальных проекторов в этом случае нет. Заметим, что соотношения (7) и (9) записываются соответственно как

$$\frac{11}{16} < \frac{11}{8} = \frac{11}{8}, \quad \frac{3}{4} < \frac{3}{2} = \frac{3}{2}.$$

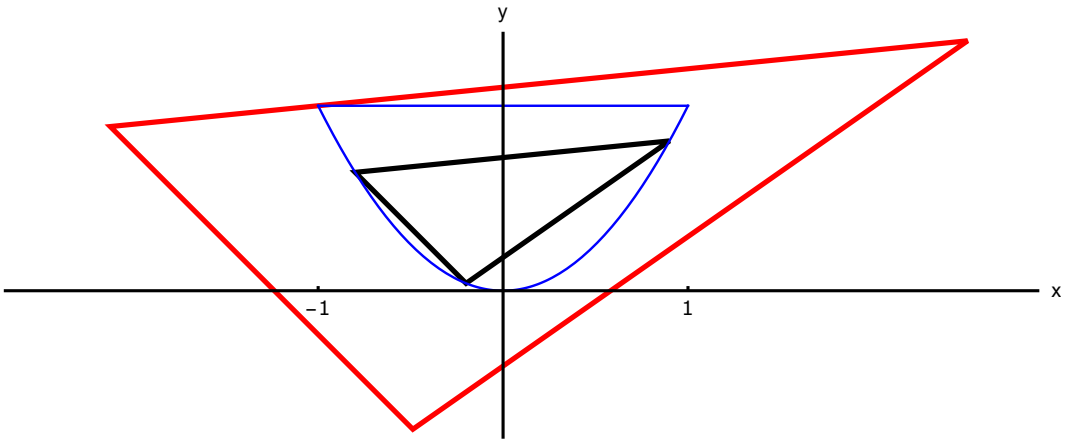


Fig. 1. The absorption of the parabolic sector by a triangle

Рис. 1. Поглощение параболического сектора треугольником

3. Кубическая интерполяция на отрезке

В случае $\Pi = \Pi_3\left(\mathbb{R}^1\right)$ верно $d = \dim \Pi = 4$, $k = d - 1 = 3$. Отображение T имеет вид $x \longmapsto (x, x^2, x^3)$, поэтому множество $T(\Omega) = T([-1, 1]) = \{(x, x^2, x^3) \in \mathbb{R}^3 : -1 \leqslant x \leqslant 1\}$. Это трёхмерная линия с концами в точках $(-1, 1, -1)$ и $(1, 1, 1)$, проекции которой на координатные плоскости конгруэнтны кривым $Y = X^2$, $Y = X^3$ и $X(t) = t^2$, $Y(t) = t^3$; последняя имеет нулевой угол в точке $X = Y = 0$.

Для узлов $-1 \leqslant r < s < t < u \leqslant 1$

$$\mathbf{A} = \begin{pmatrix} 1 & r & r^2 & r^3 \\ 1 & s & s^2 & s^3 \\ 1 & t & t^2 & t^3 \\ 1 & u & u^2 & u^3 \end{pmatrix},$$

симплекс S представляет собой тетраэдр с вершинами

$$(r, r^2, r^3), \quad (s, s^2, s^3), \quad (t, t^2, t^3), \quad (u, u^2, u^3), \tag{11}$$

принадлежащими $T(\Omega)$. Поглощение множества $T(\Omega)$ тетраэдром S иллюстрируется на рис. 2.

Минимальные значения $\|P\|_\Omega$ и $\xi(T(\Omega); S)$ были найдены с помощью компьютера. Минимум $\|P\|_\Omega$, равный $1.422919\ldots$, достигается на симметричных узлах

$$-1\,, \quad -0.417791\ldots\,, \quad 0.417791\ldots\,, \quad 1\,. \tag{12}$$

Минимум $\xi(T(\Omega); S)$, равный $1.635778\ldots$, доставляет тетраэдр, вершины которого получаются после применения преобразования T к точкам $-1\,,\ -0.481618\ldots\,,\ 0.481618\ldots\,,\ 1\,$.

Итак, компьютерные вычисления дают $\theta\,(\Pi; \Omega) = 1.422919\ldots$, $\xi_3(T(\Omega)) = 1.635778\ldots$. При таких значениях оба неравенства в (7) являются строгими и имеют вид $1.28194\ldots < 1.63577\ldots < 1.84583\ldots$. Соотношение (9) записывается как $\frac{2}{3} < 1.503307\ldots < 2$.

Как отмечалось в пункте 2, при квадратичной интерполяции 1-точка множества $T(\Omega)$ существует для любого проектора. В рассматриваемом случае это не так. Например, пусть

$$r = -\frac{\sqrt{2+\sqrt{2}}}{2}, \quad s = -\frac{\sqrt{2-\sqrt{2}}}{2}, \quad t = \frac{\sqrt{2-\sqrt{2}}}{2}, \quad u = \frac{\sqrt{2+\sqrt{2}}}{2}.$$

Эти точки являются корнями многочлена Чебышёва четвёртой степени $8x^4 - 8x^2 + 1$, поэтому называются *узлами Чебышёва.* Тогда $\|P\|_\Omega = \sqrt{2+\sqrt{2}} = 1.847759\ldots$, $\xi(T(\Omega); S) = 2.496605\ldots$, и неравенство (6) имеет вид $1.5651727\ldots < 2.496605\ldots < 2.695518\ldots$. Поскольку правое соотношение в (6) равенством не является, 1-точки множества $T(\Omega)$ относительно симплекса S с вершинами (11) в данном случае не существует.

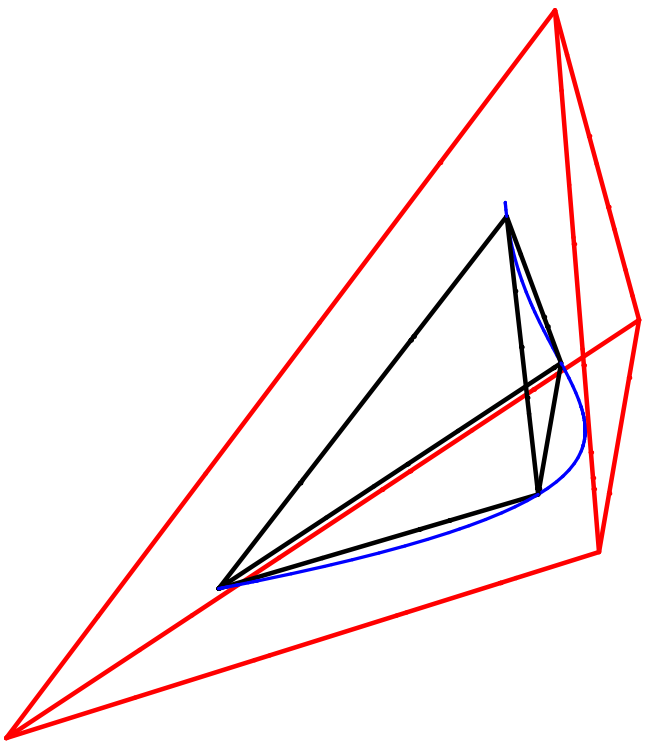


Fig. 2. The absorption of the set $T([-1,1])$ by a tetrahedron

Рис. 2. Поглощение множества $T([-1,1])$ тетраэдром

Тем не менее, как показывают наши вычисления, для некоторых наборов узлов правое соотношение в (6) обращается в равенство (с точностью не менее 10^{-11}). Указанным свойством обладают равномерные узлы, а также узлы (12) минимального проектора.

Для равномерных узлов имеем $\|P\|_\Omega = 1.63113030\ldots$, $\xi(T(\Omega); S) = 2.26226061\ldots$, соотношение (6) принимает вид $1.420753\ldots < 2.262260\ldots = 2.262260\ldots$ Отметим, что вычисление $\|P\|_\Omega$ и $\xi(T(\Omega); S)$ производилось с помощью двух различных программ. Однако последнее равенство выполняется с точностью не менее чем 10^{-18}.

Для узлов (12) норма проектора минимальна: $\|P\|_\Omega = \theta(\Pi;\Omega) = 1.422919\ldots$, коэффициент поглощения $\xi(T(\Omega);S) = 1.845839\ldots$, а соотношение (6) записывается как $1.281946\ldots < 1.845839\ldots = 1.845839\ldots$

Разумеется, приближённые вычисления даже с большой точностью не гарантируют наличия справа в (6) равенства. Строгое доказательство этого равенства даёт использование подхода с применением понятия 1-точки (см. пункт 1). Приведём данные для двух рассматриваемых случаев.

Для равномерных узлов

$$x^{(1)} = -1, \quad x^{(2)} = -\frac{1}{3}, \quad x^{(3)} = \frac{1}{3}, \quad x^{(4)} = 1$$

и точки $x^* = -0.699055\ldots$ выполняются равенства $\|P\|_\Omega = 1.631130\ldots = \sum |\lambda_j(x^*)|$, причём

$$\lambda_1(x^*) = 0.890801\ldots, \quad \lambda_2(x^*) = -0.315565\ldots, \quad \lambda_3(x^*) = 0.360848\ldots, \quad \lambda_4(x^*) = 0.063915\ldots.$$

Следовательно, $y^* = T(x^*)$ является 1-точкой множества $T(\Omega)$ относительно тетраэдра с вершинами $T\left(x^{(j)}\right)$. В соответствии со сказанным в пункте 1 правое соотношение в (6) обращается в равенство. Заметим, что $\det(\mathbf{A}) = 1.053497\ldots$.

Для узлов

$$x^{(1)} = -1, \quad x^{(2)} = -0.417791\ldots, \quad x^{(3)} = 0.417791\ldots, \quad x^{(4)} = 1$$

при $x^{**} = -0.733172\ldots$ имеем $\|P\|_\Omega = 1.422919\ldots = \sum |\lambda_j(x^{**})|$. Так как

$$\lambda_1(x^{**}) = -0.211459\ldots, \quad \lambda_2(x^{**}) = 0.771708\ldots, \quad \lambda_3(x^{**}) = 0.381082\ldots, \quad \lambda_4(x^{**}) = 0.058668\ldots,$$

то $y^{**} = T(x^{**})$ является 1-точкой множества $T(\Omega)$ относительно тетраэдра с вершинами $T\left(x^{(j)}\right)$. Наличие 1-точки означает, что и в этом случае справа в (6) имеет место равенство. Здесь $\det(\mathbf{A}) = -1.138679\ldots$.

4. Интерполяция многочленами из $\Pi_4\left(\mathbb{R}^1\right)$

При $\Pi = \Pi_4\left(\mathbb{R}^1\right)$ имеем $d = \dim\Pi = 5$, $k = d-1 = 4$. Отображение T имеет вид $x \longmapsto (x, x^2, x^3, x^4)$, множество $T(\Omega) = T([-1,1]) = \{(x,x^2,x^3,x^4) \in \mathbb{R}^4 : -1 \leqslant x \leqslant 1\}$. Для узлов $-1 \leqslant r < s < t < u < v \leqslant 1$

$$\mathbf{A} = \begin{pmatrix} 1 & r & r^2 & r^3 & r^4 \\ 1 & s & s^2 & s^3 & s^4 \\ 1 & t & t^2 & t^3 & t^4 \\ 1 & u & u^2 & u^3 & u^4 \\ 1 & v & v^2 & v^3 & v^4 \end{pmatrix}.$$

Координаты вершин симплекса $S \subset \mathbb{R}^4$ записаны в матрицу $\mathbf{A}$ построчно, начиная со второго столбца.

Узлы Чебышёва, т. е. корни многочлена Чебышёва пятой степени $16x^5 - 20x^3 + 5x$, имеют вид

$$r = -\frac{\sqrt{5+\sqrt{5}}}{2\sqrt{2}}, \quad s = -\frac{\sqrt{5-\sqrt{5}}}{2\sqrt{2}}, \quad t = 0, \quad u = \frac{\sqrt{5-\sqrt{5}}}{2\sqrt{2}}, \quad v = \frac{\sqrt{5+\sqrt{5}}}{2\sqrt{2}}.$$

В этом случае $\|P\|_\Omega = \dfrac{1+4\sqrt{5}}{5} = 1.988854\ldots$, но это не наименьшее возможное значение.

Приведём результаты компьютерных вычислений. Минимум $\|P\|_\Omega$, равный $1.559490\ldots$, достигается на симметричных узлах -1, $-0.620911\ldots$, 0, $0.620911\ldots$, 1. Минимум $\xi(T(\Omega);S)$, равный $1.981193\ldots$, доставляет симплекс, вершины которого получаются после применения оператора T к точкам -1, $-0.650738\ldots$, 0, $0.650738\ldots$, 1.

Таким образом, компьютер даёт $\theta(\Pi;\Omega) = 1.559490\ldots$, $\xi_4(T(\Omega)) = 1.981193\ldots$. С этими значениями неравенства (7) являются строгими и имеют вид $1.349681\ldots < 1.981193\ldots < 2.398725\ldots$.

5. Оценки величин $\theta\left(\Pi_k\left(\mathbb{R}^1\right);[-1,1]\right)$ и $\xi_k\left(T([-1,1])\right)$ при $1 \leqslant k \leqslant 10$

В этом пункте приводятся полученные численным путём оценки минимальных норм интерполяционных проекторов, действующих из $C[-1,1]$ на пространства $\Pi_k\left(\mathbb{R}^1\right)$ при $1 \leqslant k \leqslant 10$, и минимальных коэффициентов поглощения в тех же случаях. Для краткости мы обозначили $\theta_k := \theta\left(\Pi_k\left(\mathbb{R}^1\right);[-1,1]\right)$, $\xi_k := \xi_k\left(T([-1,1])\right)$.

Table 1. Minimal norms of projectors and corresponding absorption coefficients

Таблица 1. Минимальные нормы проекторов и соответствующие коэффициенты поглощения

k	$\theta_k \leqslant$	$\xi(S)$
1	1	1
2	1.25	1.375
3	1.422919...	1.845839...
4	1.559490...	2.224196...
5	1.672210...	2.574785...
6	1.768134...	2.911143...
7	1.851599...	3.239031...
8	1.925457...	3.561425...
9	1.991685...	3.880036...
10	2.051705...	4.195926...

Table 2. Minimal absorption coefficients and corresponding norms of projectors

Таблица 2. Минимальные коэффициенты поглощения и соответствующие нормы проекторов

k	$\xi_k \leqslant$	$\|\|P\|\|$
1	1	1
2	1.375000...	1.250000...
3	1.635778...	1.604018...
4	1.981193...	1.626067...
5	2.210535...	1.782786...
6	2.455130...	1.858521...
7	2.678509...	1.962845...
8	2.907301...	2.029565...
9	3.128316...	2.108072...
10	3.351866...	2.164915...

В таблице 1 даются значения k (степень интерполяционного многочлена), верхняя оценка θ_k и величина $\xi(S) := \xi(T([-1,1]);S)$ коэффициента поглощения множества $T([-1,1])$ симплексом с вершинами в точках $T\left(x^{(j)}\right)$, где $x^{(j)}$ — узлы интерполяционного проектора, на котором получена приведённая оценка θ_k. Значения $\theta_1 = 1$ и $\theta_2 = 1.25$ являются точными. Равенство справа в (6) выполняется при $k = 1, 2, 3$.

В таблице 2 приводятся верхние оценки минимальных коэффициентов поглощения ξ_k и нормы $\|P\| := \|P\|_{[-1,1]}$ тех проекторов, узлы которых восстанавливаются из вершин экстремальных симплексов. Значения $\xi_1 = 1$ и $\xi_2 = 1.375$ являются точными. Правое равенство в (6) имеет место при $k = 1, 2$.

Найденные численно узлы минимальных проекторов и параметрические координаты вершин экстремальных симплексов приведены в [6]. Там же можно найти значения оценок для θ_k и ξ_k с большим числом знаков после запятой.

6. Равномерные узлы

Компьютерные вычисления коэффициентов поглощения и норм проекторов для равномерных узлов (см. таблицу 3) осложняются быстрым уменьшением модуля определителя матрицы $\mathbf{A}$. Например, для $k = 11$ имеем $\det(\mathbf{A}) = 3.63581 \cdot 10^{-14}$. Далее, с ростом k, абсолютное значение определителя быстро убывает. При вычислениях на системах со стандартным представлением чисел с плавающей точкой трудно гарантировать точность результатов. По этой причине мы приводим значения рассматриваемых величин только для $1 \leqslant k \leqslant 10$. Заметим, что правое равенство в (6) выполняется при $k = 1, 2, 3$.

Table 3. Absorption coefficients and norms of projectors for regular nodes

Таблица 3. Коэффициенты поглощения и нормы проекторов для равномерных узлов

k	$\xi(S)$	$\|\|P\|\|$
1	1	1
2	1.375	1.25
3	2.262260...	1.631130...
4	3.812500...	2.207824...
5	6.167317...	3.106301...
6	9.461457...	4.549341...
7	13.824447...	6.929739...
8	21.876588...	10.945645...
9	41.283675...	17.848612...
10	72.576233...	29.899955...

Table 4. Absorption coefficients and norms of projectors for Chebyshev nodes

Таблица 4. Коэффициенты поглощения и нормы проекторов для чебышёвских узлов

k	$\xi(S)$	$\|\|P\|\|$
1	1.414213...	1.414213...
2	2.000000...	1.666666...
3	2.496605...	1.847759...
4	2.962610...	1.988854...
5	3.414213...	2.104397...
6	3.857835...	2.202214...
7	4.296558...	2.287016...
8	4.732050...	2.361856...
9	5.165299...	2.428829...
10	5.596925...	2.489430...
11	6.027339...	2.544766...
12	6.456823...	2.595678...

7. Узлы Чебышёва

В таблице 4 приводятся результаты вычислений коэффициентов поглощения и норм проекторов для узлов Чебышёва, то есть узлов, совпадающих с нулями многочлена Чебышёва нужной степени. Правое равенство в (6) выполняется при $k = 1, 2$.

Как и в случае равномерных узлов, за точность вычислений коэффициентов поглощения на компьютере можно ручаться только при небольших k. Мы приводим значения только для $1 \leqslant k \leqslant 12$. Уже для $k = 12$ имеем $\det(\mathbf{A}) = 3.68529 \cdot 10^{-15}$. При возрастании k определители становятся еще меньше по модулю, что может приводить к снижению точности при использовании переменных типа double языка C++. Нормы проектора для чебышёвских узлов вычислялись нами по точной формуле (см., например, [12]), поэтому на значения норм данное замечание не распространяется.

Отметим, что используемая нами для вычисления норм проекторов программа на C++ даёт удовлетворительное совпадение значений с точной формулой по крайней мере при $k \leqslant 30$.

References

[1] M. V. Nevskii, *Geometricheskie Ocenki v Polinomial'noj Interpolyacii*. P. G. Demidov Yaroslavl State University, 2012, 218 pp., in Russian.

[2] M. V. Nevskii, "Inequalities for the norms of interpolation projectors", *Modeling and Analysis of Information Systems*, vol. 15, no. 3, pp. 28–37, 2008.

[3] M. V. Nevskii, "On a certain relation for the minimal norm of an interpolation projector", *Modeling and Analysis of Information Systems*, vol. 16, no. 1, pp. 24–43, 2009.

[4] M. V. Nevskii and A. Y. Ukhalov, "Linear interpolation on a Euclidean ball in $\mathbb{R}^n$", *Modeling and Analysis of Information Systems*, vol. 26, no. 2, pp. 279–296, 2019.

[5] M. V. Nevskii and A. Y. Ukhalov, "On optimal interpolation by linear functions on an n-dimensional cube", *Modeling and Analysis of Information Systems*, vol. 25, no. 3, pp. 291–311, 2018.

[6] A. Ukhalov, "Supplementary materials for the article "On a geometric approach to the estimation of interpolation projectors"", *Mendeley Data, V1*, 2023. DOI: 10.17632/snh5m99yxr.1.

[7] P. Wellin, *Essentials of Programming in Mathematica*. Cambridge University Press, 2016, 436 pp.

[8] S. Mangano, *Mathematica Cookbook: Building Blocks for Science, Engineering, Finance, Music, and More*. O'Reilly Media Inc., 2010, 830 pp.

[9] S. Wolfram, *An Elementary Introduction to the Wolfram Language*. Wolfram Media, Inc., 2017, 340 pp.

[10] D. E. King, "Dlib-ml: A machine learning toolkit", *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[11] N. S. Bogomolova, "Kvadratichnaya interpolyaciya i zadacha o pogloshchenii treugol'nikom parabolicheskogo sektora", in *Put' v Nauku. Matematika. Tezisy Dokladov Vserossijskoy Molodezhnoi Konferencii*, in Russian, 2022, pp. 39–41.

[12] S. Pashkovskij, *Vychislitel'nye Primeneniya Mnogochlenov i Ryadov Chebysheva*. Nauka, 1983, 384 pp., in Russian.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Application of the algorithm for finding the outer median of a graph in the problems of determining the reliability of technical systems

V. B. Tikhonov[1], Y. A. Plaksa[1], S. A. Kurochkina[1], N. A. Prusova[1] DOI: 10.18255/1818-1015-2023-3-258-263

[1]Yaroslavl Higher Military School of Air Defense, 150001, Yaroslavl, Moskovsky prospect, building 28.

MSC2020: 05C35
Research article
Full text in Russian

Received July 5, 2023
After revision August 7, 2023
Accepted August 16, 2023

The problem of locating a service center for technical systems with known values of failure flows is considered. This problem was solved using the minisum algorithm of graph theory. The dependence of the system availability factor on the average time between failures and the average recovery time of the system elements is obtained. It is shown that the optimal location of the maintenance point is the median of the graph located at one of its vertices.

Keywords: undirected weighted graph; minisum algorithm; graph vertex; graph median; gear ratio; maintenance center; reliability index.

INFORMATION ABOUT THE AUTHORS

Vladimir B. Tikhonov	orcid.org/0000-0002-5524-282X. E-mail: kaktus38@yandex.ru Professor of the Department of Physics, Doctor of Technical Sciences, Professor.
Yuri A. Plaksa	orcid.org/0000-0003-0785-8272. E-mail: plaksa-06@mail.ru Associate Professor of the Department of Automation, Candidate of Technical Sciences, Associate Professor.
Svetlana A. Kurochkina corresponding author	orcid.org/0000-0001-7261-0439. E-mail: svetlana_k_621@mail.ru Associate Professor of the Department of Mathematics, Candidate of Physical and Mathematical Sciences.
Nataliya A. Prusova	orcid.org/0000-0002-2805-5228. E-mail: natali_pet@mail.ru Associate Professor of the Department of Mathematics, Candidate of Pedagogical Sciences.

For citation: V. B. Tikhonov, Y. A. Plaksa, S. A. Kurochkina, and N. A. Prusova, “Application of the algorithm for finding the outer median of a graph in the problems of determining the reliability of technical systems”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 258-263, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

DISCRETE MATHEMATICS IN RELATION TO COMPUTER SCIENCE

Применение алгоритма поиска внешней медианы графа в задачах определения надежности технических систем

В. Б. Тихонов[1], Ю. А. Плакса[1], С. А. Курочкина[1], Н. А. Прусова[1]

DOI: 10.18255/1818-1015-2023-3-258-263

[1]Ярославское высшее военное училище противовоздушной обороны, 150001, г. Ярославль, Московский проспект, дом 28.

УДК 519.17
Научная статья
Полный текст на русском языке

Получена 5 июля 2023 г.
После доработки 7 августа 2023 г.
Принята к публикации 16 августа 2023 г.

Рассмотрена задача о размещении центра обслуживания технических систем при известных значениях потоков отказов. Даная задача решалась с помощью минисуммного алгоритма теории графов. Получена зависимость коэффициента готовности системы от среднего времени наработки между отказами и среднего времени восстановления элементов системы. Показано, что оптимальным местом расположения пункта технического обслуживания является медиана графа, расположенная в одной из его вершин.

Ключевые слова: неориентированный взвешенный граф; минисуммный алгоритм; вершина графа; медиана графа; передаточное число; центр технического обслуживания; показатель надежности.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Владимир Борисович Тихонов	orcid.org/0000-0002-5524-282X. E-mail: kaktus38@yandex.ru профессор кафедры физики, доктор техн. наук, профессор.
Юрий Андреевич Плакса	orcid.org/0000-0003-0785-8272. E-mail: plaksa-06@mail.ru доцент кафедры автоматики, кандидат техн. наук, доцент .
Светлана Алексеевна Курочкина автор для корреспонденции	orcid.org/0000-0001-7261-0439. E-mail: svetlana_k_621@mail.ru доцент кафедры математики, кандидат физ.-мат. наук.
Наталия Александровна Прусова	orcid.org/0000-0002-2805-5228. E-mail: natali_pet@mail.ru доцент кафедры математики, кандидат пед. наук.

Для цитирования: V. B. Tikhonov, Y. A. Plaksa, S. A. Kurochkina, and N. A. Prusova, “Application of the algorithm for finding the outer median of a graph in the problems of determining the reliability of technical systems”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 258-263, 2023.

Обычно в теории надежности графы используются для иллюстрации алгоритмов восстановления технических систем [1]. Однако есть ряд задач, использующих графы для определения оптимального или рационального размещения центров технического обслуживания, решение которых позволяет обеспечить более высокую надежность этих систем.

В практических приложениях часто возникает необходимость решения задачи планирования и расположения центров обслуживания. Впервые задача о размещении объектов была сформулирована в XVII веке, получившая впоследствии название задачи Вебера [2]. Геометрическое решение данной задачи для треугольника было представлено Э. Торричелли [3]. Во второй половине XX века задача решена численно методом наименьших квадратов [3]. В настоящее время проблема поиска оптимального места размещения объектов остается актуальной: для определения места расположения баз снабжения, коммутаторов в телефонной сети, подстанций в электросетях и т. д. Современные задачи оптимального размещения можно разделить на два типа: задачи о размещении взаимосвязанных объектов [4—6] и задачи размещения-распределения. Ко второму случаю относятся, в частности, задачи о p-медиане и размещения с предпочтениями клиентов. Для описания моделей задач размещения используется различный математический аппарат, методы решения определяются характеристиками модели. Так для поиска решения задачи о p-медиане в [7—9] применяется метод целочисленного линейного программирования, формулировка сводится к задаче о паре матриц распределения. Такой подход к решению задачи размещения распространенный, но предполагает математическую определенность переменных величин и их ограниченное количество. Как и в случае других NP-полных задач, при оптимальном поиске прибегают к приближенным инструментам, например, в [10] рассматривается алгоритм наискорейшего спуска для нахождения локального оптимума. Но приближенные методы имеют ряд недостатков, один их них — отсутствие эффективной точной процедуры решения. Полиномиальные приближенные алгоритмы решения представлены и в [11], в работе проведен также вероятностный анализ задачи размещения, а расстояния между вершинами графов определяются как случайные величины с одинаковой функцией распределения. Параметры задачи могут быть описаны не только количественными характеристиками, но и качественными. В [12] рассмотрена однокритериальная минисуммная задача размещения центра обслуживания в сети дорог, где переменная является лингвистической, то есть может иметь как количественную оценку, так и описывать качественные понятия. В том числе различную смысловую нагрузку может нести и термин «объект». Таким образом, задача о размещении пунктов обслуживания, в которых требуется расположить пункт обслуживания на графе так, чтобы сумма кратчайших расстояний от этого пункта до вершин графа была бы минимально возможной, имеет обширную сферу практического применения, в том числе и военно-прикладную.

С точки зрения эксплуатации изделий, как функциональной, так и технической, любую сложную техническую систему по своей структуре можно рассматривать как распределенную иерархическую систему, которая состоит из пространственно-разнесенных между собой элементов, размещение на местности и функционирование которых подчинено достижению одной общей цели.

Комплексным показателем надежности технических систем является коэффициент готовности, который показывает вероятность того, что объект окажется в работоспособном состоянии в произвольный момент времени, кроме планируемых периодов, в течение которых эксплуатация системы не предусмотрена [13]. Он определяется по формуле:

$$K_{\Gamma} = \frac{T_o}{T_o + T_{\text{в}}}, \tag{1}$$

где T_o — среднее время наработки между отказами системы, $T_{\text{в}}$ — среднее время восстановления системы.

Формула (1) показывает, что требуемое значение коэффициента готовности напрямую зависит от работоспособности изделий. Для ее обеспечения могут быть использованы центры технического обслуживания, размещение которых существенно влияет на среднее время восстановления. Снижение времени доставки неисправных элементов в ремонтный орган и обратно или выезда сервисной бригады к отказавшему элементу технической системы позволяет повысить значение

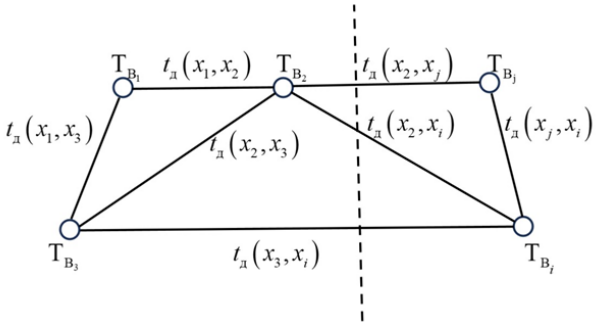


Fig. 1. Loading scheme of vertices and edges of a graph

Рис. 1. Схема нагружения вершин и ребер графа

коэффициента готовности. Поэтому актуальной является задача размещения центра технического обслуживания таким образом, чтобы обеспечить максимальное значение коэффициента готовности технической системы, и, следовательно, максимальную эффективность ее применения.

Выше было отмечено, что для решения задач подобного рода может быть использована теория графов [14]. Наиболее близким к предлагаемому решению является минисуммный алгоритм размещения, обеспечивающий поиск медианы графа — вершины, сумма взвешенных расстояний от которой до остальных вершин минимальна. Суть метода заключается в следующем [14].

Пусть дан неориентированный граф $G = (X, R)$, где X — множество вершин, а R — множество ребер. Для каждой i-й вершины графа вводится понятие передаточного числа:

$$\sigma(x_i) = \sum_{x_j \in X} v_j \, d(x_i, x_j) = \sum_{x_j \in X} v_j \, d(x_j, x_i), \tag{2}$$

где v_j — вес j-й вершины, $d(x_i, x_j)$ — кратчайшее расстояние между i-й и j-й вершинами. Вершина x_0, для которой $\sigma(x_0) = \min[\sigma(x_i)]$, где $x_i \in X$, является медианой графа.

Если рассматривать территориально распределенную техническую систему в качестве множества элементов, неделимых на данном уровне иерархии, то такое множество можно представить в виде неориентированного взвешенного графа. Тогда в качестве множества вершин будет выступать множество элементов технической системы, размещенных на местности, а в качестве множества ребер можно рассматривать маршруты транспортной сети для перемещения необходимых ресурсов (элементов замены, сервисных бригад) для восстановления изделий между пунктами их размещения. Вес ребер будет определяться минимальным временем доставки этих ресурсов по данному маршруту $t_д(x_j, x_i)$.

На практике, чаще всего, элементы технической системы неравнозначны, объединены в подсистемы и вносят различный вклад в формирование ее надежности. Этот вклад определяется критериями отказа, как самих элементов и подсистем, так и технической системы в целом. При этом, в соответствии с выражением (1), надежность каждого элемента технической системы характеризуется двумя величинами: T_o и $T_в$. Поэтому каждая вершина графа данной системы, вообще говоря, должна быть взвешена двумя весами — T_o и $T_в$. Но T_o не зависит от времени транспортировки элемента замены между вершинами графа, а $T_в$ зависит. Следовательно, при решении задачи об оптимальном размещении центра технического обслуживания вершины достаточно нагрузить весами, равными $T_в$. Схема нагружения вершин и ребер графа представлена на рис. 1.

Величина $T_в$ для каждой вершины складывается из $t_в$ — времени выполнения мероприятий восстановления элемента системы, находящегося в j-й вершине, которое не зависит от места расположения центра технического обслуживания (время диагностирования, время замены блока и т.п.); $t_д$ — времени доставки отказавшего элемента замены из j-й вершины в i-ю вершину, которое определяется местоположением вершин, и равного ему времени доставки отремонтированного элемента

замены из i-й вершины в j-ю вершину (время обратного пути); $t_{\text{р}}$ — времени ремонта отказавшего элемента замены j-го изделия в центре технического обслуживания. В результате получим:

$$T_{\text{в}_j} = t_{\text{в}_j} + 2t_{\text{д}}(x_j, x_i) + t_{p_j}, \tag{3}$$

Поскольку и $t_{\text{в}}$, и t_p инвариантны к размещению пункта обслуживания, то выражение (3) может быть записано как:

$$T_{\text{в}_j} = f(t_{\text{д}}(x_j, x_i)), \tag{4}$$

где $f(t_{\text{д}}(x_j, x_i))$ — некоторая функция минимального времени доставки, определяемая условиями конкретной задачи.

В результате критерий оптимальности размещения центра технического обслуживания можно представить, как некоторую функцию местоположения вершин. Тогда обобщенное передаточное число для i-й вершины графа можно представить в следующем виде:

$$\sigma(x_i) = \sum_{x_j \in X} F[t_{\text{д}}(x_j, x_i)]. \tag{5}$$

В формуле (5) предполагается, что каждый элемент суммы, образующей передаточное число, в соответствии с формулой (2) равный произведению весов вершин $T_{\text{в}}$ и времен доставки ресурсов $t_{\text{д}}(x_j, x_i)$ будет в конечном счете определятся минимальным временем доставки $t_{\text{д}}(x_j, x_i)$ и фактически являться некоторой функцией F этой величины.

Таким образом, для решения задачи необходимо найти такую вершину на графе, для которой обобщенное передаточное число примет минимальное значение: $\sigma(x_0) = \min[\sigma(x_j)]$, где $x_j \in X$. Как показано в методе внешней медианы графа [14] такая точка будет находиться в одной из вершин графа.

Пусть имеем техническую систему, состоящую из N элементов. Если при отказе любого одного элемента из N происходит отказ системы, то систему можно представить в виде системы с основным соединением [15]. В некоторых системах допускается отказ m элементов из N, где $m \geq 1$. В этом случае можно говорить о системе с нагруженным резервом и неограниченным восстановлением. Показатели надежности таких систем рассчитываются по формулам [15]:

$$T_o = \frac{1 + \sum\limits_{k=1}^{m} \sum\limits_{|B|=k} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}}}{\sum\limits_{|B|=m} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}} \sum\limits_{l \in A} \frac{1}{T_{o_j}}}, \quad T_B = \frac{\sum\limits_{|B|=m+1} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}}}{\sum\limits_{|B|=m} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}} \sum\limits_{l \in A} \frac{1}{T_{o_j}}}, \tag{6}$$

где m — количество избыточных элементов в системе;
B — множество номеров отказавших (восстанавливаемых) элементов системы;
A — множество номеров работоспособных элементов системы;
$|B|$ — мощность (число элементов) множества B, $|B| + |A| = N$;
$\sum_{k=1}^{m}$ — сумма, включающая k-е уровни графа состояний системы, $k = \overline{1, m}$;
$\sum_{|B|=k}$, $\sum_{|B|=m}$, $\sum_{|B|=m+1}$ — сумма, включающая все состояния k-го, m-го, $(m+1)$-го уровня графа состояний системы соответственно.

С учетом выражений (1) и (6) получим формулу для расчета коэффициента готовности m/N системы:

$$K_{\Gamma} = \frac{1 + \sum\limits_{k=1}^{m} \sum\limits_{|B|=k} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}}}{1 + \sum\limits_{k=1}^{m} \sum\limits_{|B|=k} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}} + \sum\limits_{|B|=m+1} \prod\limits_{j \in B} \frac{T_{\text{в}_j}}{T_{o_j}}}. \tag{7}$$

Обозначим элементы в выражении (7) следующим образом:

$$A_1 = 1 + \sum_{k=1}^{m} \sum_{|B|=k} \prod_{j \in B} \frac{T_{\text{в}j}}{T_{o_j}}, \quad A_2 = \sum_{|B|=m+1} \prod_{j \in B} \frac{T_{\text{в}j}}{T_{o_j}}. \tag{8}$$

Тогда $K_\Gamma = A_1/(A_1 + A_2)$.

Исходя из вышеизложенного, можно сказать, что коэффициент готовности технической системы обратно пропорционален выражению A_2. Выражение для A_2 (см. выражение (8)) представляет собой сумму произведений взвешенных средних времен восстановления элементов системы, которые определяются весами ребер графа (минимальными временами доставки).

Следовательно, в этом случае коэффициент готовности технической системы будет обратно пропорционален сумме взвешенных времен доставки из i-го элемента в центр технического обслуживания и медиана графа будет находиться в одной из его вершин. Таким образом, для определения местоположения центра технического обслуживания можно применить минисуммный алгоритм на графах, заменив передаточное число (2) на обобщенное передаточное число (5).

Данный метод был использован для определения местоположения центра технического обслуживания группировки средств противовоздушной обороны.

References

[1] A. Oleinik, E. A. Lukashev, S. P. Poserenin, and M. E. Stavrovskiy, “Graph method in reliability theory and practice of technical service”, *Izvestiya MGTU MAMI*, vol. 4, no. 2, pp. 236–247, 2010, in Russian.

[2] H. J. Miser, *Handbook of Operations Research: foundations and fundamentals*. Van Nostrand Reinhold, 1978, 622 pp.

[3] M. Aoki, *Introduction to optimization techniques. Fundamentals and applications of nonlinear programming*. Macmillan, 1971, 335 pp.

[4] G. G. Zabudsky and N. S. Veremchuk, “Reshenie zadachi vebera na ploskosti s minimaksnym kriteriem i zapreshchennymi zonami”, *Izvestiya Irkutskogo gosudarstvennogo universiteta. Seriya: Matematika*, vol. 9, pp. 10–25, 2014, in Russian.

[5] V. Beresnev and A. Mel’nikov, “Approximate algorithms for the competitive facility location problem”, *Journal of Applied and Industrial Mathematics*, vol. 5, pp. 180–190, 2011.

[6] V. Demidenko, “Generalizing strong feasibility conditions for the quadratic assignment problem with anti-Monge and Toeplitz matrices”, in *Doklady Natsionalnoi Akademii Nauk Belarusi*, vol. 47, 2003, pp. 15–18.

[7] A. A. Kolokolov, T. V. Levanova, and M. A. Loresh, “Algoritmy murav’inoj kolonii dlja zadach optimal’nogo razmeshhenija predprijatij”, *Omskij nauchnyj vestnik*, vol. 38, no. 4, pp. 62–67, 2006, in Russian.

[8] I. L. Vasiliev, K. B. Klimentova, and Y. A. Kochetov, “Novye nizhnie otsenki dlya zadachi razmeshcheniya s predpochteniyami klientov”, *Zhurnal vychislitel’noi matematiki i matematicheskoi fiziki*, vol. 49, no. 6, pp. 1055–1066, 2009, in Russian.

[9] E. V. Alekseeva and Y. A. Kochetov, “Geneticheskii lokal’nyi poisk dlya zadachi o p-mediane s predpochteniyami klientov”, *Diskretnyi analiz i issledovanie operatsii*, vol. 14, no. 1, pp. 3–31, 2007, in Russian.

[10] Y. A. Kochetov, M. G. Pashchenko, and A. Plyasunov, “O slozhnosti lokal’nogo poiska v zadache o p-mediane”, *Diskretnyi analiz i issledovanie operatsii*, vol. 12, no. 2, pp. 44–71, 2005, in Russian.

[11] E. K. Gimadi, “O veroyatnostnom analize priblizhennogo algoritma resheniya zadachi o p-mediane”, *Diskretnyi analiz i issledovanie operatsii*, vol. 17, no. 3, pp. 19–31, 2010, in Russian.

[12] I. N. Rosenberg, “Odnokriterial’naya minisummnaya zadacha razmeshcheniya tsentra obsluzhivaniya s lingvisticheskimi peremennymi”, *Izvestiya Yuzhnogo federal’nogo universiteta. Tekhnicheskie nauki*, vol. 31, no. 2, pp. 56–63, 2003, in Russian.

[13] *GOST 27.002-2015: Dependability in technics. terms and definitions*, in Russian, 2015.

[14] N. Christofides, *Graph theory: An algorithmic approach*. Academic Press, Inc., 1975, 400 pp.

[15] A. Polovko and S. V. Gurov, *Osnovy teorii nadezhnosti*. BHV, 2006, 704 pp., in Russian.

MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 30, NO. 3, 2023
journal homepage: www.mais-journal.ru

ALGORITHMS

The algorithms for the Eulerian cycle and Eulerian trail problems for a multiple graph

A. V. Smirnov[1]

DOI: 10.18255/1818-1015-2023-3-264-282

[1]P.G. Demidov Yaroslavl State University, Sovetskaya str., 14, Yaroslavl, 150003, Russia.

MSC2020: 05C45, 05C65
Research article
Full text in Russian

Received August 13, 2023
After revision August 26, 2023
Accepted August 30, 2023

In this paper, we study undirected multiple graphs of any natural multiplicity $k > 1$. There are edges of three types: ordinary edges, multiple edges and multi-edges. Each edge of the last two types is a union of k linked edges, which connect 2 or $(k+1)$ vertices, correspondingly. The linked edges should be used simultaneously. If a vertex is incident to a multiple edge, it can be also incident to other multiple edges and it can be the common end of k linked edges of some multi-edge. If a vertex is the common end of some multi-edge, it cannot be the common end of another multi-edge.

We set the problem of finding the eulerian walk (the cycle or the trail) in a multiple graph, which generalizes the classical problem for an ordinary graph. We formulate the necessary conditions for existence of an eulerian walk in a multiple graph and show that these conditions are not sufficient. Besides that, we show that the necessary conditions of existence of an eulerian cycle and eulerian trail are not mutually exclusive for an arbitrary multiple graph, that is why it is possible to construct a multiple graph where two types of eulerian walks exist simultaneously. Any multiple graph can be juxtaposed to the ordinary graph with quasi-vertices, which represents the structure of the initial graph in a simpler form. In particular, each eulerian walk in the multiple graph corresponds to the eulerian walk in the graph with quasi-vertices. The algorithm for getting such a graph is formulated. Also, the auxiliary problem of finding the covering trails with given endpoints in an ordinary graph is studied. Two algorithms are obtained for this problem. We elaborate the algorithm for finding the eulerian walk in a multiple graph, which has the exponential complexity. We suggest the polynomial algorithm for the special case of a multiple graph and show that the necessary conditions are sufficient for existence of an eulerian walk in this special case.

Keywords: multiple graph; multiple path; divisible graph; reachability set; covering trails; eulerian trail; eulerian cycle; graph with quasi-vertices

INFORMATION ABOUT THE AUTHORS

Alexander V. Smirnov corresponding author	orcid.org/0000-0002-0980-2507. E-mail: alexander_sm@mail.ru PhD, Associate Professor, Department of Theoretical Computer Science.

Funding: Yaroslavl State University (project VIP-016).

For citation: A. V. Smirnov, "The algorithms for the Eulerian cycle and Eulerian trail problems for a multiple graph", *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 264-282, 2023.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 30, № 3, 2023
сайт журнала: www.mais-journal.ru

ALGORITHMS

Алгоритмы для задач об эйлеровом цикле и эйлеровой цепи в кратном графе

А. В. Смирнов[1]

DOI: 10.18255/1818-1015-2023-3-264-282

[1]Ярославский государственный университет им. П.Г. Демидова, ул. Советская, 14, Ярославль, 150003, Россия.

УДК 519.17
Научная статья
Полный текст на русском языке

Получена 13 августа 2023 г.
После доработки 26 августа 2023 г.
Принята к публикации 30 августа 2023 г.

В статье рассматриваются неориентированные кратные графы произвольной натуральной кратности $k > 1$. Кратный граф содержит ребра трех типов: обычные, кратные и мультиребра. Ребра последних двух типов представляют собой объединение k связанных ребер, которые соединяют 2 или $(k+1)$ вершину соответственно. Связанные ребра могут использоваться только согласованно. Если вершина инцидентна кратному ребру, то она может быть инцидентна другим кратным ребрам, а также она может быть общим концом k связанных ребер мультиребра. Если вершина является общим концом мультиребра, то она не может быть общим концом никакого другого мультиребра.

Ставится задача об эйлеровом маршруте (цикле или цепи) в кратном графе, которая обобщает классическую задачу для обычного графа. Сформулированы необходимые условия существования эйлерова маршрута в кратном графе, показано, что эти условия не являются достаточными. Кроме того, показано, что для произвольного кратного графа необходимые условия существования эйлерова цикла и эйлеровой цепи не являются взаимоисключающими, поэтому можно построить кратный граф, в котором одновременно существуют два вида эйлеровых маршрутов. Кратному графу сопоставляется обычный граф с квазивершинами, в упрощенном виде представляющий структуру исходного графа. В частности, каждому эйлерову маршруту в кратном графе соответствует эйлеров маршрут в графе с квазивершинами. Формулируется алгоритм построения такого графа. Также рассмотрена вспомогательная задача о покрывающих цепях с заданными концами в обычном графе, получены два алгоритма ее решения. Разработан алгоритм поиска эйлерова маршрута в кратном графе экспоненциальной трудоемкости. Для частного случая кратного графа предложен полиномиальный алгоритм, показано, что в этом частном случае необходимые условия существования эйлерова маршрута являются достаточными.

Ключевые слова: кратный граф; кратный путь; делимый граф; множество достижимости; покрывающие цепи; эйлерова цепь; эйлеров цикл; граф с квазивершинами

ИНФОРМАЦИЯ ОБ АВТОРАХ

Александр Валерьевич Смирнов
автор для корреспонденции
orcid.org/0000-0002-0980-2507. E-mail: alexander_sm@mail.ru
канд. физ.-мат. наук, доцент, кафедра теоретической информатики.

Финансирование: ЯрГУ (проект VIP-016).

Для цитирования: A. V. Smirnov, “The algorithms for the Eulerian cycle and Eulerian trail problems for a multiple graph”, *Modeling and analysis of information systems*, vol. 30, no. 3, pp. 264-282, 2023.

Введение

В данной статье мы рассмотрим задачу об *эйлеровом маршруте* (цикле или цепи) в кратном графе. Кратные графы содержат три типа ребер (обычные, кратные и мультиребра) и являются обобщением обычных графов — по сути, обычный граф имеет кратность $k = 1$. Определения кратного графа кратности $k > 1$ и делимого кратного графа были сформулированы в статье [1].

Среди других известных обобщений графов наиболее близкими нам концепциями являются мультиграфы, гиперграфы (см., например, [2, 3]), а также метаграфы (см. [4, 5]). Действительно, как и в мультиграфах, в кратных графах допускается наличие нескольких ребер между парой вершин (набор таких ребер мы будем в дальнейшем называть *кратным ребром*), однако в случае кратного графа количество таких ребер должно быть строго равным k. В кратных графах присутствуют *мультиребра*, соединяющие между собой $(k + 1)$ вершину. Но в отличие от гиперребер гиперграфа, мультиребро представляется в виде k связанных ребер, имеющих один общий конец, причем все эти k ребер должны использоваться согласованно. По сути, понятие мультиребра близко понятию ребра между вершиной и метавершиной в метаграфе. При этом в метаграфе, напомним, метапуть между двумя метавершинами фактически моделирует причинно-следственные связи в некоторой предметной области. Однако в кратном графе используется принципиально иной подход к определению пути: *кратный путь* должен состоять ровно из k обычных путей, проходящих по обычным ребрам, а также по связанным ребрам кратных и мультиребер; при этом пути должны быть согласованы (одинаковы) на кратных и мультиребрах. Поэтому кратный граф нельзя считать частным случаем метаграфа.

Отметим также, что частным случаем кратного графа является кратная сеть (см. [6, 7]). Задача о наибольшем потоке в кратной сети обобщает классическую задачу (см. [8]) и имеет ряд приложений в сфере экономики, управления, финансов. В частности, кратные сети и потоки используются для поиска решения NP-трудной задачи целочисленного сбалансирования трех- и четырехмерной матрицы (см., например, [9, 10]).

Ранее мы обобщили для случая кратных графов задачи о кратчайшем пути между двумя вершинами и о минимальном остовном дереве (см., например, [1, 11]). При этом задача о кратчайшем кратном пути полиномиальна (см. [12]), а задача о минимальном остовном дереве является NP-трудной, по крайней мере, для графов кратности $k \geqslant 3$ (см. [13]).

В данной статье мы рассмотрим обобщение для кратных графов еще одной классической задачи теории графов — задачи об эйлеровом маршруте (см. [14]). Будут приведены необходимые условия существования такого маршрута в кратном графе. Отметим, что для обычного графа задача полиномиальна и разработан ряд полиномиальных алгоритмов (см., например, [15, 16]), однако для кратного графа мы получим алгоритм нахождения эйлерова маршрута, который в общем случае потребует экспоненциального количества шагов. Тем не менее, особенности построения кратного графа и эйлерова маршрута в нем позволяют предположить NP-трудность задачи. Также мы рассмотрим подкласс кратных графов, для которого можно построить полиномиальный алгоритм поиска эйлерова маршрута.

1. Кратные графы и деревья: необходимые определения

Напомним несколько определений, связанных с кратными графами и путями, которые ранее были сформулированы в статьях [1, 11].

Определение 1. *Кратный граф* G произвольной натуральной кратности $k > 1$ — это граф, вершины которого могут соединяться ребрами одного из 3 видов:

1. *Обычное ребро* e^o; множество обычных ребер обозначим через E^o.

2. *Кратное ребро* e^k между двумя вершинами, которое состоит из k одинаковых связанных ребер; связанные ребра кратного ребра могут использоваться только согласованно; множество кратных ребер обозначим через E^k.
3. *Связанное ребро* e между двумя вершинами, имеющее один общий конец с другим $(k-1)$ ребром (у любых двух из k связанных ребер только один конец является общим); множество связанных общей вершиной ребер будем называть *мультиребром* e^m; связанные ребра мультиребра могут использоваться только согласованно; множество мультиребер обозначим через E^m.

Если вершина инцидентна какому-либо кратному ребру, то она может быть инцидентна другим кратным ребрам, а также она может быть общим концом какого-либо мультиребра.

Если вершина является общим концом какого-либо мультиребра, то она не может быть общим концом никакого другого мультиребра.

Если вершина является отдельным концом мультиребра или инцидентна обычному ребру, то она не может быть общим концом мультиребра и не может быть инцидентна кратному ребру.

Множества вершин и ребер графа G обозначим через V и E соответственно. Заметим, что $E = E^o \cup E^k \cup E^m$.

В данной статье рассматриваются только неориентированные кратные графы.

Рис. 1 и 2 иллюстрируют определение 1. В левой части рис. 1 кратное ребро представлено в виде объединения k одинаковых ребер между двумя вершинами, что показано штрихами. Равенство (или согласованность) связанных ребер предполагает, что все характеристики этих ребер (например, длина) одинаковы, и эти ребра могут использоваться только одновременно. Так, если осуществляется проход в определенном направлении по одному из связанных ребер, то одновременно с этим все остальные ребра проходятся в том же самом направлении. Кратное ребро может включаться в какие-либо новые структуры только целиком. В дальнейшем мы будем обозначать кратные ребра жирными линиями, как в правой части рис. 1.

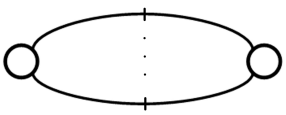


Fig. 1. Multiple edge

Рис. 1. Кратное ребро

В левой части рис. 2 мультиребро $\{x_0, \{x_1, \ldots, x_k\}\}$ представлено в виде объединения k одинаковых ребер, связывающих общую вершину x_0 с k разными вершинами $x_1, \ldots, x_k$. Как и на рис. 1, равенство ребер показано штрихами. Согласованность связанных ребер имеет тот же смысл, что и для кратных ребер. В дальнейшем мультиребра мы будем изображать при помощи расщепляющихся на k частей линий, как в правой части рис. 2.

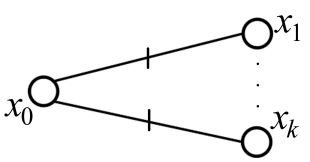


Fig. 2. Multi-edge

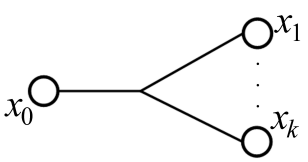


Рис. 2. Мультиребро

Определение 2. *Обычной вершиной* назовем вершину, которая инцидентна обычному ребру или является отдельным концом мультиребра.

Кратной вершиной назовем вершину, которая инцидентна кратному ребру или является общим концом мультиребра.

Из определения 1 следует, что множества обычных и кратных вершин не пересекаются. При этом кратная вершина может быть соединена с обычными только посредством мультиребра.

Определение 3. *Делимым кратным графом* назовем такой граф, в котором между двумя концами одного мультиребра не существует пути, проходящего только по обычным ребрам.

При удалении всех мультиребер делимый граф распадется на n компонент связности (связность здесь понимается в том же смысле, что и для обычных графов), каждая из которых содержит только кратные ребра либо только обычные ребра. При этом связанные ребра каждого мультиребра можно пронумеровать от 1 до k таким образом, что каждой компоненте связности, содержащей только обычные ребра, будут инцидентны связанные ребра мультиребер с одинаковыми номерами.

Определение 4. *Частью* G_i $(i \in \overline{1,k})$ делимого графа $G(V,E)$ назовем подграф, содержащий связанные ребра с номером i всех кратных и мультиребер, а также компоненты связности, состоящие из обычных ребер и инцидентные i-ым связанным ребрам всех мультиребер.

Каждая часть G_i является обычным графом. При этом возможность выделения частей G_i является особенностью делимых графов. В общем случае получить части G_i не удастся.

Пример 1. Рассмотрим представленный на рис. 3 кратный граф G кратности 2 со следующими множествами обычных, кратных и мультиребер (вершины будем обозначать их номерами):

$$E^k = \big\{\{1,3\},\ \{1,4\},\ \{2,3\},\ \{2,4\},\ \{3,4\},\ \{15,16\}\big\};$$

$$E^m = \big\{\{3,\{5,13\}\},\ \{4,\{8,10\}\},\ \{15,\{6,14\}\},\ \{16,\{9,11\}\}\big\};$$

$$E^o = \big\{\{5,6\},\ \{5,7\},\ \{5,8\},\ \{6,7\},\ \{6,9\},\ \{7,8\},\ \{7,9\},\ \{8,9\},\ \{10,11\},$$
$$\{10,12\},\ \{10,13\},\ \{11,12\},\ \{11,14\},\ \{12,13\},\ \{12,14\},\ \{13,14\}\big\}.$$

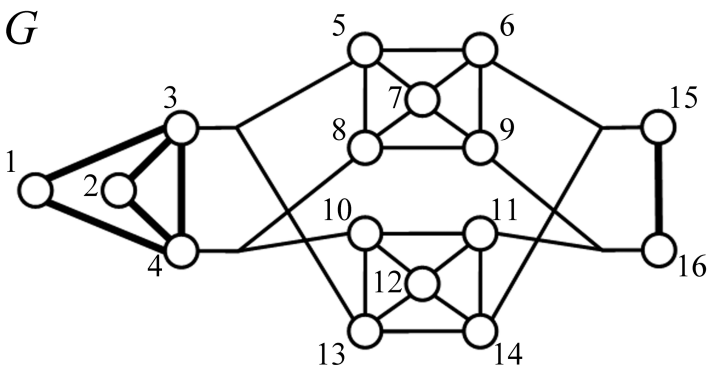


Fig. 3. Divisible graph of multiplicity 2

Рис. 3. Делимый граф кратности 2

Граф G является делимым. Части G_1 и G_2 этого графа показаны на рис. 4, связанные ребра всех кратных и мультиребер изображены пунктирными линиями.

Заметим, что граф перестанет быть делимым, если добавить в него обычное ребро между любой парой вершин из множеств $\{5,6,7,8,9\}$ и $\{10,11,12,13,14\}$.

Определим теперь путь в кратном графе.

Определение 5. $S(x,y) = \cup_{i=1}^{k} S^i(x,y)$ является *кратным путем из вершины x в вершину y* в графе $G(V,E)$, если выполнены следующие условия:

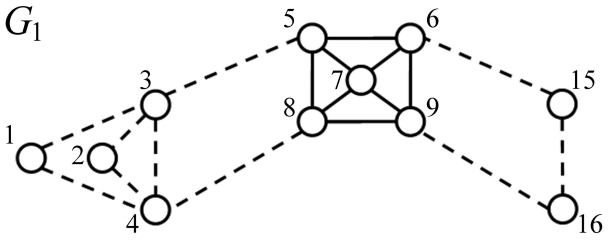


Fig. 4. Partition of a divisible graph

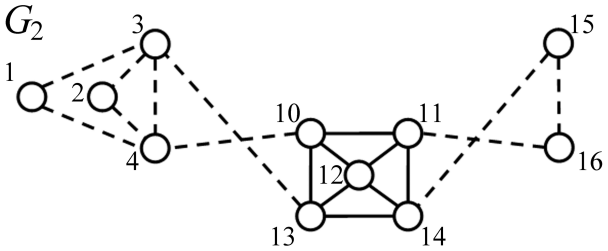


Рис. 4. Части делимого графа

1. $S^i(x,y) = \Big(\{x, v_1^i\}, \{v_1^i, v_2^i\}, \ldots, \{v_{l_i-1}^i, v_{l_i}^i\}, \{v_{l_i}^i, y\}\Big)$, где $l_i \geqslant 0$, — последовательность ребер, представляющая собой обычный (некратный) путь из x в y, где каждое ребро $\{a, b\}$ является либо обычным ребром в графе $G(V, E)$, либо i-ым связанным ребром кратного или мультиребра. Значения l_i и l_j $(i \neq j)$ не согласовываются и могут быть как равными, так и различными. Если в путь $S(x, y)$ не входит ни одного кратного или мультребра, то $S^2(x, y) = S^3(x, y) = \ldots = S^k(x, y) = \varnothing$.
2. Любая обычная вершина может встретиться в $S^i(x, y)$ несколько раз, то есть $S^i(x, y)$ может содержать циклы.
3. Никакая кратная вершина не может встретиться в $S^i(x, y)$ дважды.
4. Любое обычное ребро может встречаться в $S^i(x, y)$ несколько раз, причем направления, в которых оно проходится в разных вхождениях, могут не совпадать.
5. Обычное ребро, входящее в $S^i(x, y)$, может также входить в любой $S^j(x, y)$, $j \neq i$.
6. Все пути $S^i(x, y)$ согласованы (одинаковы) на общей части. Это условие означает, что если связанное ребро какого-то кратного или мультиребра входит в некоторый путь $S^i(x, y)$, то остальные связанные ребра должны входить во все $S^j(x, y)$, $j \neq i$ (по одному связанному ребру в каждый $S^j(x, y)$). При этом порядок вхождения всех кратных и мультиребер во все $S^i(x, y)$ одинаков.

   Фактически это значит, что если e_1 и e_2 — это два ребра пути $S(x, y)$, каждое из которых либо кратное, либо мультиребро, и в проекции $S^i(x, y)$ связанное ребро из e_1 проходится раньше связанного ребра из e_2, то во всех остальных проекциях $S^j(x, y)$ связанные ребра из e_2 могут проходиться только после связанных ребер из e_1.
7. Если $S(x, y)$ содержит мультиребро $\{x_0, \{x_1, \ldots, x_k\}\}$, проходимое в направлении от общего конца, то он не может содержать никакого другого мультиребра $\{y_0, \{x_1, \ldots, x_k\}\}$, проходимого в том же направлении. Аналогичное условие должно выполняться и в случае движения к общему концу.

Определение 6. Кратный путь $S(x, y)$ является *кратным циклом*, если $x = y$ и $S(x, y) \neq \varnothing$.

Пример 2. Проиллюстрируем определение кратного пути. Для этого рассмотрим граф, показанный на рис. 3, и построим в нем один из многочисленных возможных кратных путей $S(1, 2)$ из вершины 1 в вершину 2. Он состоит из двух обычных путей:

$$S^1(1,2) = \big(\underline{\{1,3\}},\ \underline{\{3,5\}},\ \underline{\underline{\{5,6\}}},\ \{6,9\},\underline{\{9,16\}},\underline{\{16,15\}},\ \underline{\{15,6\}},\ \underline{\underline{\{6,5\}}},\ \{5,7\},\ \{7,8\},\ \underline{\{8,4\}},\ \underline{\{4,2\}}\big);$$

$$S^2(1,2) = \big(\underline{\{1,3\}},\ \underline{\{3,13\}},\ \{13,14\},\ \underline{\underline{\{14,11\}}},\ \underline{\{11,16\}},$$
$$\underline{\{16,15\}},\ \underline{\{15,14\}},\ \underline{\underline{\{14,11\}}},\ \{11,12\},\ \{12,13\},\ \{13,10\},\ \underline{\{10,4\}},\ \underline{\{4,2\}}\big).$$

Связанные ребра кратных и мультиребер отмечены подчеркиванием. Двойным подчеркиванием в пути $S^1(1,2)$ отмечено обычное ребро $\{5,6\}$, которое в этом пути проходится дважды, но в противоположных направлениях. Соответственно, в пути $S^2(1,2)$ двойным подчеркиванием отмечено обычное ребро $\{14,11\}$, которое также проходится дважды, но в одном и том же направлении. Таким образом, в пути $S^1(1,2)$ содержится обычный цикл $(6,9,16,15,6)$, а в пути $S^2(1,2)$ содержатся обычные циклы $(14,11,16,15,14)$ и $(13,14,11,16,15,14,11,12,13)$, а вершины 5, 6, 11, 13, 14 проходятся дважды. Однако кратный путь $S(1,2)$ не содержит в себе кратных циклов, как и должно быть (ни одна кратная вершина не проходится дважды). Полученный кратный путь $S(1,2)$ показан на рис. 5. Части $S^1(1,2)$ и $S^2(1,2)$ этого пути представлены на рис. 6.

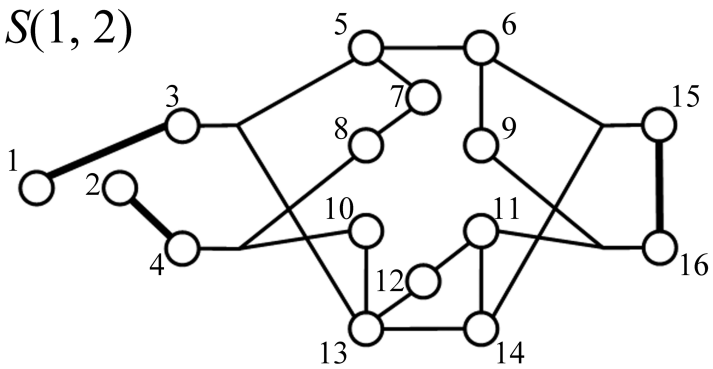
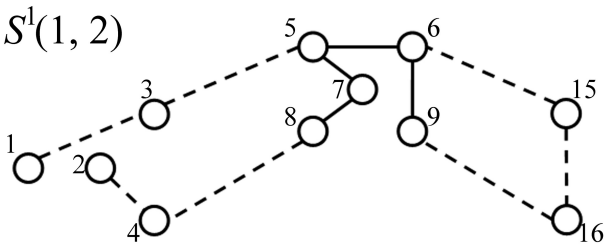


Fig. 5. Multiple path in the multiple graph

Рис. 5. Кратный путь в делимом графе



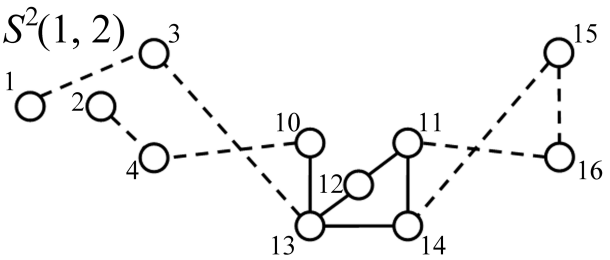


Fig. 6. Partition of the multiple path

Рис. 6. Части кратного пути

Отметим, что при замене в кратном пути $S(1,2)$ ребра $\{4,2\}$ на ребро $\{4,1\}$ мы получим кратный цикл.

Определение 7. Кратный граф $G(V,E)$ является *связным*, если одновременно выполнены два условия:

1. Кратный путь $S(x,y)$ существует для любых двух кратных вершин $x \in V$, $y \in V$.
2. Невозможно выделить такой подграф $G' \subset G$, который будет содержать только обычные ребра, и при этом подграфы G' и $G \setminus G'$ не будут соединены ни одним ребром (обычным ребром или связанным ребром мультиребра).

В отличие от обычных графов, связность кратного графа не предполагает наличие кратных путей из каждой вершины в каждую. Фактически в связном кратном графе между каждой парой вершин должен существовать обычный (некратный) путь, использующий связанные ребра кратных и мультиребер несогласованно, а кратные пути обязательно должны существовать только для пар кратных вершин.

Для делимого кратного графа определение связности может быть переписано в более простой форме, что обусловлено структурой графа.

Определение 8. Делимый кратный граф $G(V, E)$ является *связным*, если одновременно выполнены два условия:

1. Кратный путь $S(x, y)$ существует для любых двух кратных вершин $x \in V$, $y \in V$.
2. Каждая из частей G_i является связным (некратным) графом.

Определение 9. *Множеством достижимости по кратным ребрам* для некоторой кратной вершины x назовем множество R_x^k всех вершин y таких, что существует путь из x в y, проходящий только по кратным ребрам.

Будем обозначать через G_x^k подграф, образованный всеми вершинами из R_x^k и всеми кратными ребрами $e \in E^k$, которые соединяют вершины из R_x^k.

Определение 10. *Множеством достижимости по обычным ребрам* для некоторой обычной вершины x назовем множество R_x^o всех вершин y таких, что существует путь из x в y, проходящий только по обычным ребрам.

Будем обозначать через G_x^o подграф, образованный всеми вершинами из R_x^o и всеми обычными ребрами $e \in E^o$, которые соединяют вершины из R_x^o.

Очевидно, что $x \in R_x^k$, $x \in R_x^o$. Если $y \in R_x^k$, то $R_y^k = R_x^k$. Если $y \in R_x^o$, то $R_y^o = R_x^o$.

Отметим, что все множества достижимости R_x^k и R_x^o могут быть найдены с помощью быстрых полиномиальных алгоритмов, приведенных в статье [1]. Там же сформулированы полиномиальные алгоритмы проверки связности кратного графа.

Определение 11. Для любой вершины $x \in V$ определена ее *степень* $\deg x$ — количество обычных или связанных ребер инцидентных x.

Таким образом, каждое обычное ребро вида $\{x, y\}$ добавляет 1 к $\deg x$, каждое кратное ребро вида $\{x, y\}$ добавляет k к $\deg x$, а каждое мультиребро вида $\{x, \{y_1, \ldots, y_k\}\}$ добавляет k к $\deg x$ и 1 к каждому из $\deg y_i$. Очевидно, что степень любой кратной вершины будет кратна k.

2. Постановка задачи об эйлеровом маршруте в кратном графе

Пусть имеется связный кратный граф $G(V, E)$ кратности $k > 1$. Так же, как и для обычного графа, для него можно определить понятие эйлерова маршрута.

Определение 12. *Эйлеровым маршрутом* μ в кратном графе $G(V, E)$ назовем такой обход графа $G(V, E)$, в котором каждое ребро из E встречается ровно один раз, а связанные ребра каждого кратного и мультиребра из $E^k \cup E^m$ используются только согласованно (одновременно).

Как и в определении 5, для эйлерова маршрута выполнено: $\mu = \cup_{i=1}^{k} \mu_i$, то есть каждый эйлеров маршрут μ в кратном графе представляет собой объединение k обычных маршрутов μ_i ($i \in \overline{1, k}$), в каждом из которых присутствует ровно одно связанное ребро каждого кратного и мультиребра, причем порядок обхода связанных ребер одинаков во всех μ_i.

Определение 13. Замкнутый эйлеров маршрут в кратном графе (начальная вершина равна конечной) называется *эйлеровым циклом*.

Незамкнутый эйлеров маршрут в кратном графе называется *эйлеровой цепью*.

Кратный граф назовем *эйлеровым*, если в нем существует эйлеров маршрут (цикл или цепь).

Если сравнить определение кратного пути и эйлеровой цепи в кратном графе, можно заметить, что эйлерова цепь — это, по сути, кратный путь, в котором допускается использование нескольких

мультиребер с одинаковыми множествами конечных вершин, проходимых в одном направлении, а также повторное использование кратных вершин, однако не допускается повтор встречавшихся ранее обычных ребер. Точно те же допущения относятся к эйлерову циклу в кратном графе в сравнении с кратным циклом.

Построение эйлерова цикла всегда можно начинать с кратной вершины. Эйлерову цепь в большинстве случаев также можно начинать строить с кратной вершины, однако если граф не будет делимым, эйлерова цепь может начинаться и в обычной вершине. Для этого необходимо, чтобы в графе было мультиребро $e^m = \{x_0, \{x_1, \ldots, x_k\}\}$, все концы которого принадлежат одному множеству достижимости: $x_i \in R^o_{x_1}$ $(i \in \overline{1,k})$.

Пример 3. Снова рассмотрим кратный граф из примера 1 (рис. 3). Нетрудно убедиться, что в этом графе существует эйлеров цикл C, представленный на рис. 7.

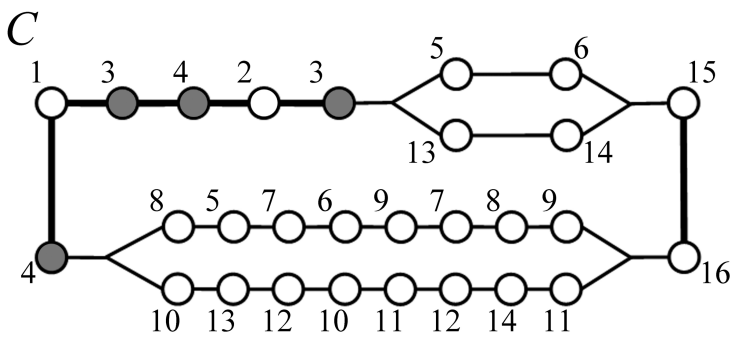


Fig. 7. Eulerian cycle in the multiple graph

Рис. 7. Эйлеров цикл в кратном графе

Следует отметить, что в эйлеровом цикле C две кратные вершины 3 и 4 проходятся дважды (на рисунке отмечены серым), поэтому цикл C является эйлеровым маршрутом, но не является кратным циклом согласно определению 6.

Задача 1 (эйлеров маршрут в кратном графе)**.** *Для данного кратного графа $G(V,E)$ установить, существует ли в нем эйлеров маршрут (цикл или цепь), и найти этот маршрут.*

3. Необходимые условия существования эйлерова маршрута в кратном графе

Сформулируем необходимые условия существования эйлерова маршрута в кратном графе, которые будут во многом похожи на аналогичные условия для обычных графов (но с поправкой на особенности построения кратных графов). Однако, в отличие от последних, эти условия не будут достаточными. Напомним, что в обычном графе эйлеров цикл существует тогда и только тогда, когда все вершины графа имеют четную степень, а эйлерова цепь существует тогда и только тогда, когда в графе есть ровно две вершины нечетной степени (см., например, [17]).

Теорема 1. *Если в кратном графе $G(V,E)$ существует эйлеров цикл, то все его обычные вершины имеют четную степень, а степень каждой кратной вершины v может быть представлена как* $\deg v = k \cdot a_v$*, где k — это кратность графа, a_v — четное число.*

Справедливость утверждения теоремы следует из того, что, обходя эйлеров цикл, мы зайдем в каждую вершину столько же раз, сколько и выйдем, при этом для кратных вершин каждая операция захода или выхода будет выполняться одновременно для k связанных ребер кратного или мультиребра.

Теорема 2. *Если в делимом кратном графе $G(V,E)$ существует эйлерова цепь, то одновременно выполнены следующие условия:*

- *все обычные вершины имеют четную степень;*
- *степень двух кратных вершин x, y представляется в виде* $\deg x = k \cdot a$, $\deg y = k \cdot b$, *где k — это кратность графа, a и b — нечетные числа;*
- *степень всех остальных кратных вершин v представляется в виде* $\deg v = k \cdot c_v$, *где k — это кратность графа, c_v — четное число.*

Теорема 3. *Если в неделимом кратном графе $G(V, E)$ существует эйлерова цепь, то найдутся две вершины x, y, для которых выполняется одно из условий:*

1. *x, y — кратные вершины;* $\deg x = k \cdot a$, $\deg y = k \cdot b$, *где k — это кратность графа, a и b — нечетные числа.*
2. *x — кратная вершина, y — обычная вершина и найдется мультиребро $\{y_0, \{y_1, \ldots, y_k\}\} \in E^m$ такое, что $y_i \in R^o_y$ ($i \in \overline{1,k}$);* $\deg x = k \cdot a$, $\deg y = k + b$, *где k — это кратность графа, a — нечетное число, b — четное неотрицательное число.*
3. *x, y — обычные вершины и найдутся два мультиребра $\{x_0, \{x_1, \ldots, x_k\}\}, \{y_0, \{y_1, \ldots, y_k\}\} \in E^m$ такие, что $x_i \in R^o_x$, $y_i \in R^o_y$ ($i \in \overline{1,k}$);* $\deg x = k + a$, $\deg y = k + b$, *где k — это кратность графа, a и b — четные неотрицательные числа.*

Степень всех остальных обычных вершин четна, а степень всех остальных кратных вершин v представляется в виде $\deg v = k \cdot c_v$, *где k — это кратность графа, c_v — четное число.*

Суммарное количество операций захода и выхода для начальной и конечной вершины эйлеровой цепи при ее обходе будет нечетно, для остальных же вершин это значение четно (как в теореме 1), отсюда следует справедливость теорем 2 и 3.

Более сложное условие теоремы 3 обусловлено тем, что в делимом графе кратная цепь обязательно начинается и заканчивается в кратной вершине, а вот для неделимого графа возможна ситуация, когда кратная цепь начинается или заканчивается в обычной вершине, причем в такой ситуации мы должны взять одновременно k ребер инцидентных этой вершине.

Следствие 1. *Если делимый кратный граф $G(V, E)$ является эйлеровым, то эйлеровой является и каждая его часть G_i ($i \in \overline{1,k}$).*

Сведем кратный граф $G(V, E)$ к обычному графу $G_{ord}(V_{ord}, E_{ord})$, структура которого будет отражать структуру исходного графа. Часть вершин будут соответствовать множествам подграфов G^o_x. Такие вершины мы будем называть *квазивершинами*, а граф G_{ord} — *графом с квазивершинами.*

Алгоритм 1 (построение графа с квазивершинами).

1. Для каждой кратной вершины $x \in V$ создадим обычную вершину $x \in V_{ord}$.
2. Для каждого кратного ребра $\{x, y\} \in E^k$ создадим обычное ребро $\{x, y\} \in E_{ord}$.
3. Найдем все множества достижимости R^o_x, в качестве x будем указывать минимальный номер вершины из этого множества.
4. Рассмотрим поочередно все мультиребра $e^m = \{x_0, \{x_1, \ldots, x_k\}\} \in E^m$. Для каждой обычной вершины x_i ($i \in \overline{1,k}$) определим множество достижимости $R^o_{y_i}$, которому она принадлежит: $x_i \in R^o_{y_i}$. Создадим *квазивершину* $q_{\{y_1,\ldots,y_k\}} \in V_{ord}$ (если таковая не была создана раньше) и добавим обычное ребро $\{x_0, q_{\{y_1,\ldots,y_k\}}\} \in E_{ord}$.

Следствие 2. *Если кратный граф $G(V, E)$ является эйлеровым, то эйлеровым является и граф с квазивершинами $G_{ord}(V_{ord}, E_{ord})$, причем каждому эйлерову маршруту в исходном кратном графе соответствует эйлеров маршрут в графе с квазивершинами.*

Обратное, вообще говоря, неверно: граф с квазивершинами может быть эйлеровым, тогда как исходный граф не эйлеров (см. пример 5 ниже); также в графе с квазивершинами может существовать эйлеров цикл или цепь, которым не соответствует эйлеров цикл или цепь в кратном графе.

Отметим, что следствия 1 и 2 останутся верными, если в их условиях мы заменим требование эйлеровости кратного графа $G(V, E)$ на требование выполнения необходимых условий его эйлеровости.

Пример 4. Рассмотрим пример построения графа с квазивершинами согласно алгоритму 1. На рис. 8 слева представлен кратный делимый граф G кратности 2. Заменим кратные вершины и ребра на обычные. Далее определим множества достижимости по обычным ребрам:

$$R^o_7 = \{7, 8, 9, 10\}, \qquad R^o_{11} = \{11\}, \qquad R^o_{12} = \{12, 13\}.$$

Нетрудно заметить, что в кратном графе G есть четыре мультиребра с концами в R^o_7 и R^o_{11}, а также два мультиребра с концами в R^o_{11} и R^o_{12}. Поэтому нужно создать две квазивершины — $q_{7,11}$ и $q_{11,12}$. Остается соединить эти квазивершины с обычными вершинами ребрами $\{1, q_{7,11}\}$, $\{2, q_{7,11}\}$, $\{14, q_{7,11}\}$, $\{15, q_{7,11}\}$, $\{5, q_{11,12}\}$ и $\{6, q_{11,12}\}$. В итоге получим граф с квазивершинами G_{ord}, представленный на рис. 8 справа.

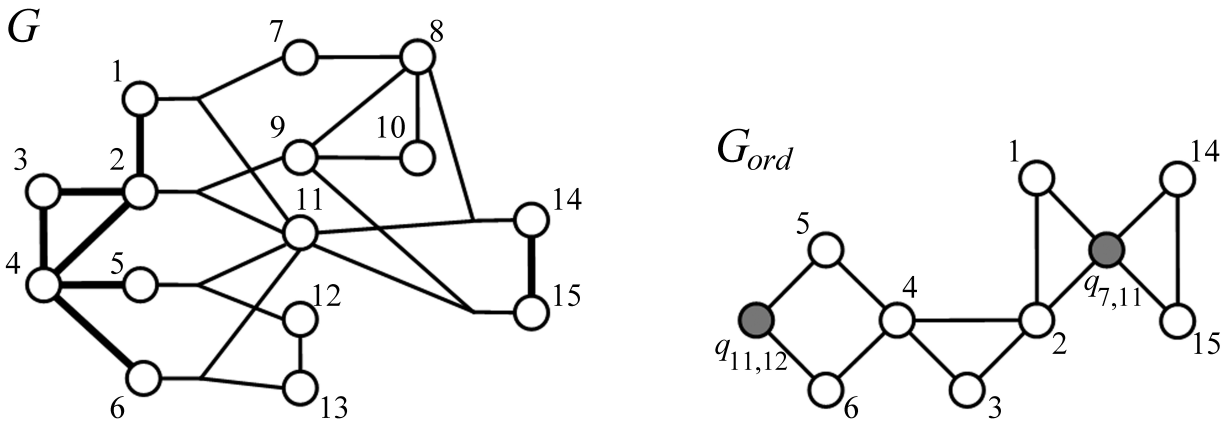


Fig. 8. Multiple graph G and the corresponding ordinary graph G_{ord} with quasi-vertices

Рис. 8. Кратный граф G и соответствующий ему обычный граф G_{ord} с квазивершинами

Заметим, что все степени обычных вершин графа G четны, а степени кратных вершин кратны 4. Поскольку граф имеет кратность 2, мы видим, что для него выполнены необходимые условия существования эйлерова цикла. Более того, эйлеров цикл C легко найти. $C = C^1 \cup C^2$, где C^1 и C^2 — обычные циклы следующей структуры:

$$C^1 = (1, 7, 8, 14, 15, 9, 10, 8, 9, 2, 3, 4, 5, 12, 13, 6, 4, 2, 1), \qquad C^2 = (1, 11, 14, 15, 11, 2, 3, 4, 5, 11, 6, 4, 2, 1).$$

Этому циклу соответствует эйлеров цикл C_{ord} в графе с квазивершинами G_{ord}:

$$C_{ord} = (1, q_{7,11}, 14, 15, q_{7,11}, 2, 3, 4, 5, q_{11,12}, 6, 4, 2, 1).$$

Пример 5. Покажем теперь, что необходимые условия существования эйлерового маршрута в кратном графе не всегда являются достаточными даже в случае делимого графа.

Рассмотрим делимый граф G, представленный в левой части рис. 9. Кратность графа равна 2, степень всех обычных вершин четна, а степень всех кратных вершин равна 4. Следовательно, для этого графа выполняется необходимое условие существования эйлерова цикла.

Нетрудно убедиться, что в каждой из проекций G_1 и G_2 этого графа существует эйлеров цикл (центральная и правая часть рис. 9), однако эти циклы невозможно согласовать на связанных ребрах кратных и мультиребер. Действительно, часть G_1 представляет собой простой цикл $C^1 = (1, 7, 2, 3, 8, 4, 5, 9, 6, 1)$, значит, соответствующий цикл в части G_2 должен быть представлен в форме $C^2 = (1, 10, \ldots, 13, 2, 3, 11, \ldots, 14, 4, 5, 12, \ldots, 15, 6, 1)$, но это невозможно сделать, не пройдя трижды по ребру $\{16, 17\}$, что противоречит определению эйлерова цикла.

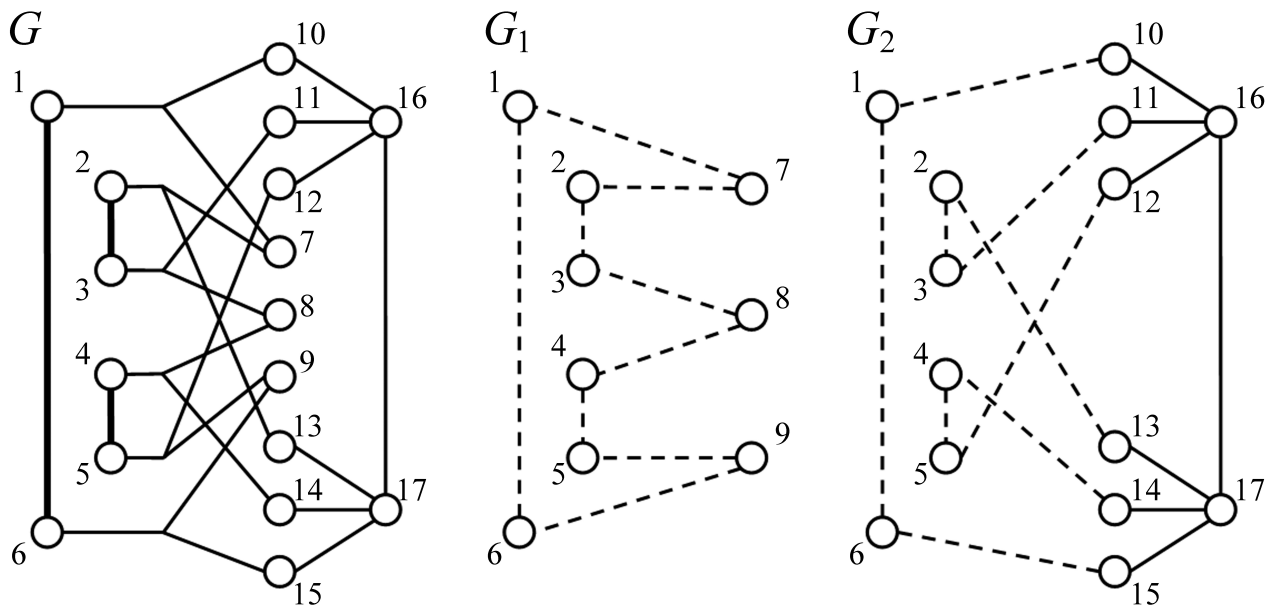


Fig. 9. Non-eulerian divisible multiple graph

Рис. 9. Делимый кратный граф, не являющийся эйлеровым

Отметим, что граф с квазивершинами G_{ord}, построенный для рассматриваемого графа G, также будет представлять собой простой цикл $C_{ord} = (1, q_{7,10}, 2, 3, q_{8,10}, 4, 5, q_{9,10}, 6, 1)$.

Пример 6. Рассмотрим еще одну особенность кратных графов: если граф не является делимым, в нем может одновременно существовать и эйлеров цикл, и эйлерова цепь благодаря тому, что необходимые условия существования эйлерова цикла и цепи не являются взаимоисключающими для таких графов. Действительно, пусть имеется граф G, показанный на рис. 10.

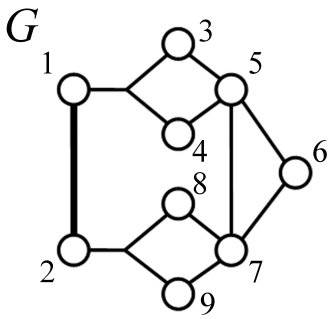


Fig. 10. Non-divisible multiple graph

Рис. 10. Неделимый кратный граф

Этот граф имеет кратность 2 и не является делимым. При этом все обычные вершины имеют четную степень, а степень кратных вершин равна 4. Следовательно, выполнены необходимые условия существования эйлерова цикла. Пример такого цикла C показан на рис. 11 слева.

В то же время, для графа выполнено и необходимое условие существования эйлеровой цепи: $\deg 5 = k + 2$, $\deg 7 = k + 2$ и есть два мультиребра $\{1, \{3, 4\}\}$ и $\{2, \{8, 9\}\}$ таких, что $3, 4 \in R^o_5$ и $8, 9 \in R^o_7$. Следовательно, можно попытаться построить эйлерову цепь $T(5, 7)$. Эта цепь существует (справа на рис. 11).

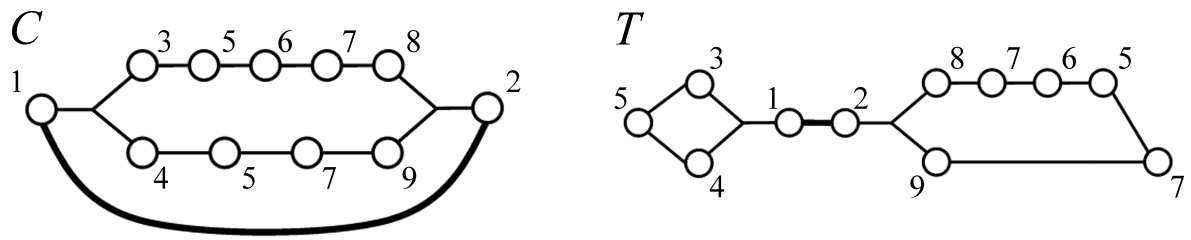


Fig. 11. Eulerian cycle and eulerian trail in the multiple graph G

Рис. 11. Эйлеров цикл и эйлерова цепь в кратном графе G

Отметим еще два любопытных факта. Во-первых, обоим эйлеровым маршрутам C и T будет соответствовать один и тот же эйлеров цикл $C_{ord} = (q_{3,3}, 1, 2, q_{3,3})$ в графе с квазивершинами G_{ord}. Во-вторых, вместо вершины 5 или 7 мы могли бы взять в качестве конца эйлеровой цепи вершину 6 ($\deg 6 = k + 0$), и в этом случае построение эйлеровой цепи тоже было бы возможно.

4. Покрывающие цепи в обычном графе

Рассмотренные в предыдущем разделе примеры приводят к следующей идее алгоритма построения эйлерова маршрута в кратном графе G:

1. Находим эйлеров маршрут C_{ord} в графе с квазивершинами G_{ord}.
2. В каждом подграфе G^o_x находим множество из s покрывающих цепей, где число s, а также начальная и конечная вершина каждой цепи определяется по мультиребрам, соответствующим ребрам цикла C_{ord}.

Под *покрывающими цепями* в обычном графе мы понимаем множество цепей, не пересекающихся по ребрам и не содержащих повторяющихся ребер; при этом множество покрывающих цепей содержит все ребра графа.

Известно, что если начала и концы таких цепей не фиксированы, для наличия s покрывающих цепей достаточно, чтобы ровно $2s$ вершин графа имели нечетную степень. Чтобы найти эти цепи, нужно соединить каждую из $2s$ вершин нечетной степени с дополнительной вершиной v^*, а затем найти эйлеров цикл в получившемся графе (см., например, [17]).

В нашем случае может потребоваться начать или закончить сразу несколько из покрывающих цепей в одной и той же вершине. Отмеченный выше результат останется справедливым: для построения покрывающих цепей нам достаточно будет соединить каждую такую вершину сразу несколькими ребрами с дополнительной вершиной v^* и построить эйлеров цикл в получившемся мультиграфе.

Таким образом, если для кратного графа выполнены необходимые условия существования эйлерова маршрута, в каждом подграфе G^o_x существует s покрывающих цепей, где $2s$ — это количество концов мультиребер в R^o_x. Если какая-то вершина является концом нескольких мультиребер, она учитывается несколько раз. Если граф неделимый и для эйлеровой цепи начальная или конечная вершина y находится в R^o_x, эта вершина k раз учитывается в множестве указанных $2s$ вершин.

Однако для построения эйлерова маршрута в кратном графе недостаточно просто существования s покрывающих цепей в G^o_x — эти цепи должны соединять определенные пары вершин, чтобы

обеспечить согласованный порядок прохода связанных ребер каждого мультиребра в эйлеровом маршруте. Указанное обстоятельство приводит к следующей задаче.

Задача 2 (покрывающие цепи с заданными концами в обычном графе)**.** *В обычном графе* $G_x^o(R_x^o, E_x^o)$ $s \geqslant 1$ *вершин* $x_1, \ldots, x_s$ *отмечены как начальные и* s *вершин* $y_1, \ldots, y_s$ *отмечены как конечные (множества вершин* x_i *и* y_j *могут содержать повторяющиеся вершины и могут пересекаться между собой).*

Требуется найти s *покрывающих цепей* $\mu_1(x_1, y_1), \ldots, \mu_s(x_s, y_s)$ *с заданными начальными и конечными вершинами.*

Отметим, что при $s = 1$ задача 2 сводится к построению эйлеровой цепи в графе G_x^o и разрешима, если для исходного кратного графа выполнены необходимые условия существования эйлерова маршрута. Если же $s > 1$, решения может не быть, даже если условия существования эйлерова маршрута выполнены в исходном кратном графе.

Пример 7. Рассмотрим граф, представленный на рис. 12. В этом графе 10 вершин нечетной степени и 2 — четной, следовательно, в нем можно построить множество из 5 покрывающих цепей.

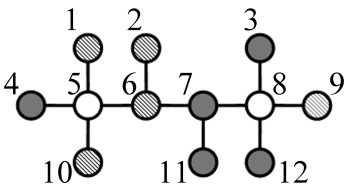


Fig. 12. The problem of finding the covering trails with given endpoints

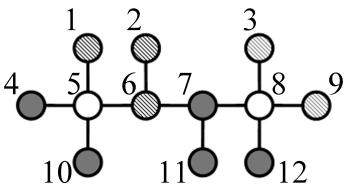


Рис. 12. Проблема получения покрывающих цепей с заданными концами

Однако при фиксации концов этих цепей мы не всегда сможем их получить. Такая ситуация показана слева на рис. 12 (начальные вершины закрашены серым, конечные вершины заштрихованы). При этом не важно, какие именно пары вершин из множеств $\{3, 4, 7, 11, 12\}$ и $\{1, 2, 6, 10, 12\}$ нужно соединить между собой, в любом случае ребро $\{6, 7\}$ придется использовать несколько раз.

Справа на рис. 12 показана ситуация, когда вершина 10 становится начальной для какой-то цепи, а вершина 3 — конечной. В этом случае можно построить, например, такое множество из 5 покрывающих цепей: $\mu_1(4, 1)$, $\mu_2(10, 6)$, $\mu_3(11, 2)$, $\mu_4(7, 9)$, $\mu_5(12, 3)$.

Тем не менее, при другой фиксации концов покрывающие цепи построить не удастся, например: $\mu_1(4, 1)$, $\mu_2(10, 6)$, $\mu_3(11, 3)$, $\mu_4(7, 9)$, $\mu_5(12, 2)$.

Рассмотрим два алгоритма решения задачи 2. Оба они будут в худшем случае выполняться за экспоненциальное число шагов, однако можно предположить, что рассматриваемая задача является NP-трудной.

Идея первого алгоритма состоит в том, что мы соединим каждую из вершин x_i, y_i $(i \in \overline{1, s})$ с дополнительной вершиной v^* ребром (если какая-то вершина встречается в наборе несколько раз, для нее будет создано несколько ребер). В получившемся мультиграфе будем искать эйлеров цикл, в котором ребра инцидентные v^* будут проходиться в заданном порядке и в нужном нам направлении (таком, чтобы при удалении вершины v^* из цикла у нас остались искомые s цепей).

Отметим, что задача построения эйлерова цикла, в котором некоторые ребра проходятся в заданном порядке и в заданном направлении рассматривалась в статьях [18, 19]. В [18] доказано, в частности, что если требуется пройти m ребер и в заданном порядке, и в заданном направлении, задача будет гарантированно иметь решение, если граф $2m$-связен, и в этом случае решение можно найти с помощью полиномиального алгоритма, описанного в [19]. Однако в нашем графе $m = 2s$ и граф в лучшем случае m-связен, поэтому воспользоваться указанным алгоритмом не удастся (более того, в примере 7 мы рассмотрели ситуацию, когда решения может не быть вовсе).

В нашем алгоритме мы будем использовать описанную в работе [20] операцию *каппа-трансформации* (*$\varkappa$-трансформации*) эйлерова цикла. Данная операция состоит в изменении на противоположное направления обхода ребер на выбранном участке эйлерова маршрута, заключенном между двумя вхождениями одной вершины (фактически меняется направление обхода какого-то подцикла). В указанной статье доказано, что с помощью последовательного применения операций $\varkappa$-трансформации можно перебрать все возможные для данного графа эйлеровы маршруты.

Алгоритм 2 (поиск покрывающих цепей с заданными концами перебором эйлеровых циклов).

1. Соединим каждую из вершин x_i, y_i ($i \in \overline{1,s}$) с дополнительной вершиной v^* ребром (если какая-то вершина встречается в наборе несколько раз, для нее будет создано несколько ребер).

2. С помощью полиномиального алгоритма Хиргольцера (см. [15]) найдем эйлеров цикл C в получившемся мультиграфе.

3. Если удалить из цикла C вершины v^* вместе с инцидентными ребрами, у нас останется ровно s покрывающих цепей. Если при этом указанные цепи будут иметь вид: $\mu_1(x_1, y_1), \ldots, \mu_s(x_s, y_s)$, эти цепи являются искомыми, выход. Иначе переходим на шаг 4.

4. Рекурсивно перестраиваем цикл C, используя все возможные $\varkappa$-трансформации. После применения каждой отдельной операции переходим на шаг 3 для проверки. Если вариантов $\varkappa$-трансформации не осталось, задача 2 для данного графа не имеет решения, выход.

Альтернативный подход к решению задачи 2 состоит в том, что мы будем последовательно находить цепи $\mu_i(x_i, y_i)$, включая в них ребра, не являющиеся *перешейками* (*мостами*) и исключая эти ребра из графа. Похожим образом работает алгоритм Флёри поиска эйлерова маршрута (см. [16]), однако нам нужно будет предусмотреть возможность возврата из-за того, что требуется найти сразу s цепей, которые не должны пересекаться по ребрам.

Понятие перешейка также придется трактовать иначе, чем в классическом алгоритме.

Определение 14. Пусть на текущем шаге выполняется построение цепи $\mu_i(x_i, y_i)$ ($i \in \overline{1,s}$), и уже получена цепь от x_i до v. Ребро $\{v, w\}$ является *перешейком*, если после его удаления будет выполнено одно из условий:

- не существует цепи $\mu(w, y_i)$ (перешеек *первого типа*);
- не существует цепи $\mu(x_j, y_j)$ ($j > i$) (перешеек *второго типа*);
- в графе появляется компонента связности, не содержащая ни x_j, ни y_j ($j > i$), ни y_i (перешеек *третьего типа*).

Отметим, что одно и то же ребро может одновременно быть перешейком и первого, и второго типа, но перешеек третьего типа не может одновременно иметь первый или второй тип. Если какое-то ребро $\{v, w\}$ — перешеек третьего типа, остальные ребра инцидентные v не являются перешейками (v будет частью компоненты связности, «отсекаемой» этим перешейком).

Алгоритм 3 (поиск с возвратами покрывающих цепей с заданными концами).

1. $i = 1$.

2. Если $i < s$, переходим на шаг 3. Иначе $i = s$, ищем эйлерову цепь $\mu_s(x_s, y_s)$ с помощью алгоритма Флёри, выход.

3. $v = x_i$.

4. Если $v = y_i$, полагаем $i = i + 1$ и переходим на шаг 2. Иначе переходим на шаг 5.

5. Если какое-то ребро $\{v, w\}$ является перешейком третьего типа, определяем компоненту связности, «отсекаемую» этим перешейком, и находим в ней эйлеров цикл с помощью алгоритма Флёри. Присоединяем найденный цикл к $\mu_i(x_i, y_i)$ и удаляем его из графа. Включаем ребро $\{v, w\}$

в $\mu_i(x_i, y_i)$ и удаляем его из графа вместе с вершиной v. Полагаем $v = w$ и переходим на шаг 4. Иначе переходим на шаг 6.

6. Поочередно рассматриваем все ребра $\{v, w\}$ (вершины w перебираем в порядке возрастания номеров).

Если ребро $\{v, w\}$ не является перешейком первого или второго типа, включаем его в $\mu_i(x_i, y_i)$ и удаляем из графа (если для вершины v это ребро последнее, она также удаляется из графа). Полагаем $v = w$ и переходим на шаг 4.

Если все ребра инцидентные v являются перешейками, переходим на шаг 7.

7 (возврат). Отменяем действия предыдущих шагов до тех пор, пока не дойдем до шага 6, где для какой-то вершины v осталось нерассмотренным хотя бы одно ребро $\{v, w\}$. После этого возобновляем шаг 6 с рассмотрением указанного ребра. Если в процессе возврата мы дошли до вершины x_1 и для нее не осталось нерассмотренных ребер, задача 2 не имеет решения для данного графа, выход.

Пример 8. Проиллюстрируем работу алгоритмов 2 и 3 на примере графа, представленного на рис. 13. Здесь требуется найти покрывающие цепи с фиксированными концами вида: $\mu_1(1, 6), \mu_2(2, 5), \mu_3(3, 4)$.

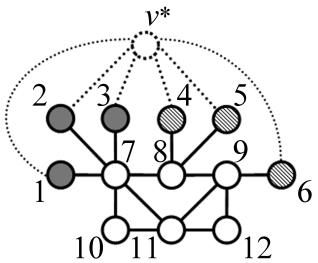


Fig. 13. Search for the covering trails with given endpoints

Рис. 13. Поиск покрывающих цепей с заданными концами

Сначала применим алгоритм 2. Добавим в граф дополнительную вершину v^* и ребра, соединяющие ее с вершинами 1, 2, 3, 4, 5, 6 (показаны на рис. 13 пунктиром). Найдем эйлеров цикл C в графе:

$$C = (\underline{v^*, 1, 7, 2, v^*}, 3, 7, 8, 4, v^*, 5, 8, 9, 11, 7, 10, 11, 12, 9, 6, v^*).$$

Этот цикл не удовлетворяет условию задачи, поэтому будем рекурсивно выполнять $\varkappa$-трансформацию цикла (трансформируемые подциклы отмечены подчеркиванием):

$$C = (v^*, 2, \underline{7, 1, v^*, 3, 7}, 8, 4, v^*, 5, 8, 9, 11, 7, 10, 11, 12, 9, 6, v^*),$$

$$C = (v^*, 2, 7, 3, v^*, 1, 7, \underline{8, 4, v^*, 5, 8}, 9, 11, 7, 10, 11, 12, 9, 6, v^*), \quad \ldots,$$

пока не придем к искомому циклу:

$$C = (v^*, 2, 7, 8, 5, v^*, 1, 7, 11, 9, 6, v^*, 3, 7, 10, 11, 12, 9, 8, 4, v^*).$$

Теперь продемонстрируем работу алгоритма 3 для того же графа (но без дополнительной вершины v^*).

Будем строить цепь $\mu_1(1, 6)$. Сначала включим в нее ребро $\{1, 7\}$. На следующем шаге ребра $\{7, 2\}$ и $\{7, 3\}$ будут перешейками одновременно и первого и второго типа, поэтому добавим в цепь следующее по номеру ребро $\{7, 8\}$. Дальше все три ребра оказываются перешейками: $\{8, 9\}$ — второго типа, а $\{8, 4\}$ и $\{8, 5\}$ — одновременно и первого и второго типа. Поэтому выполняем возврат,

исключая из μ_1 ребро $\{7, 8\}$ и добавляя следующее по номеру ребро $\{7, 10\}$. Продолжая действовать подобным образом, мы получим цепь $\mu_1(1, 6) = (1, 7, 10, 11, 9, 6)$, а затем цепи $\mu_2(2, 5) = (2, 7, 8, 5)$ и $\mu_3(3, 4) = (3, 7, 11, 12, 9, 8, 4)$.

В заключение данного раздела отметим, что задачу 2 можно рассмотреть и для подграфов G_x^k кратного графа $G(V, E)$, если кратные ребра заменить на обычные, а в качестве начальных и конечных вершин покрывающих цепей рассматривать кратные вершины, являющиеся началами мультиребер.

5. Алгоритм поиска эйлерова маршрута в кратном графе

Сформулируем теперь алгоритм поиска эйлерова маршрута в делимом кратном графе $G(V, E)$, который будет использовать алгоритмы 1–3 на отдельных своих шагах.

Алгоритм 4 (поиск эйлерова маршрута в делимом кратном графе).

1. Проверяем выполнение необходимых условий существования эйлерового цикла или цепи (теоремы 1–2). Если они не выполнены, эйлерова маршрута не существует, выход.

2. Находим все R_x^o, с помощью алгоритма 1 строим граф с квазивершинами G_{ord}.

3. Находим эйлеров маршрут C_{ord} в графе G_{ord} с помощью алгоритма Хиргольцера.

4. По очередности и направлению прохождения в маршруте C_{ord} ребер инцидентных квазивершинам определяем для каждого G_x^o множества начальных и конечных вершин и их индексы.

5. В каждом из G_x^o с помощью алгоритма 2 или алгоритма 3 ищем покрывающие цепи с заданными концами. Если для всех G_x^o эта задача решена успешно, переходим на шаг 6, иначе — на шаг 7.

6. Находим искомый эйлеров маршрут C в кратном графе $G(V, E)$, перенося в кратный граф маршрут C_{ord} и объединяя его со всеми найденными покрывающими цепями в нужном порядке. Выход.

7. Рекурсивно перестраиваем цикл C_{ord}, используя все возможные $\varkappa$-трансформации. После применения каждой отдельной операции переходим на шаг 4. Если вариантов $\varkappa$-трансформации не осталось, задача 1 для данного графа не имеет решения, выход.

Отметим, что алгоритм 4 выполнится в худшем случае за экспоненциальное количество шагов, однако можно предположить, что задача 1 является NP-трудной (из-за необходимости согласования всех частей маршрута на связанных ребрах кратных и мультиребер).

Алгоритм 4 сформулирован для делимых графов. Однако его можно перенести и на случай неделимого графа с некоторыми дополнениями (в общем случае увеличивающими перебор). При этом на шаге 1 мы будем проверять условия теорем 1 и 3.

Так, для квазивершин, у которых совпадают два и более индексов, набор начальных и конечных вершин при поиске покрывающих цепей в G_x^o может определяться неоднозначно. Тогда в худшем случае придется перебирать все возможные варианты сопоставления. К примеру, в эйлеровом маршруте сначала проходится мультиребро $\{x_0, \{x_1, x_2\}\}$ в направлении от общего конца, а потом — мультиребро $\{y_0, \{y_1, y_2\}\}$ в направлении к общему концу, и все вершины $x_1, x_2, y_1, y_2 \in R_x^o$. Тогда при поиске покрывающих цепей надо сначала будет рассмотреть вариант $\mu_1(x_1, y_1)$, $\mu_2(x_2, y_2)$, а если таких цепей найти не удастся, то вариант $\mu_1(x_1, y_2)$, $\mu_2(x_2, y_1)$.

Дополнительная сложность появляется, если для графа выполняется условие 2 или 3 из теоремы 3. В этом случае у нас может возникнуть несколько вариантов выбора начальной или конечной обычной вершины эйлеровой цепи (как в примере 6), и в худшем случае придется просмотреть их все. Кроме того, при построении покрывающих цепей нужно будет k раз включить выбранную обычную вершину в множество начальных или конечных вершин для какого-то G_x^o.

6. Полиномиальный алгоритм для частного случая

Пусть теперь $G(V, E)$ — делимый кратный граф. Пусть в графе с квазивершинами G_{ord} степень всех квазивершин $\deg q_{\{i_1,\dots,i_k\}} = 2$ и никакие два множества индексов квазивершин не пересекаются. По сути, это означает, что каждый подграф G^o_x инцидентен ровно двум связанным ребрам разных мультиребер.

Тогда эйлеров маршрут в этом графе можно найти с помощью следующего алгоритма.

Алгоритм 5 (поиск эйлерова маршрута, частный случай делимого кратного графа).

1. Проверяем выполнение необходимых условий существования эйлерового цикла или цепи (теоремы 1–2). Если они не выполнены, эйлерова маршрута не существует, выход.
2. Находим все R^o_x, с помощью алгоритма 1 строим граф с квазивершинами G_{ord}.
3. Находим эйлеров маршрут C_{ord} в графе G_{ord} с помощью алгоритма Хиргольцера.
4. В каждом G^o_x находим эйлерову цепь с помощью алгоритма Хиргольцера. В силу вышесказанного, эта цепь будет соединять два конца связанных ребер мультиребер (степень всех обычных вершин кратного графа четна; убирая два мультиребра, мы получаем две вершины нечетной степени в соответствующем G^o_x).
5. Находим искомый эйлеров маршрут C в кратном графе $G(V, E)$, перенося в кратный граф маршрут C_{ord} и объединяя его со всеми найденными в подграфах G^o_x эйлеровыми цепями (при необходимости обращая эти цепи).

Теорема 4. *В делимом кратном графе с указанными выше ограничениями на квазивершины необходимые условия существования эйлерова маршрута являются достаточными; эйлеров маршрут может быть найден с помощью алгоритма 5 за полиномиальное время.*

Доказательство. Достаточность условий и применимость алгоритма 5 следуют из вышеизложенных рассуждений.

Обоснуем полиномиальность алгоритма 5:

- проверка условий на шаге 1 представляет собой простую проверку степеней вершин;
- множества R^o_x на шаге 2 ищутся с помощью полиномиальных алгоритмов из статьи [1]; построение графа с квазивершинами линейно относительно количества вершин и ребер в исходном кратном графе;
- на шаге 3 применяется полиномиальный алгоритм Хиргольцера;
- на шаге 4 этот же алгоритм запускается $|E^m|/2$ раз;
- восстановление искомого маршрута на шаге 5 линейно относительно количества ребер кратного графа.

□

Отметим, что если мы ослабим условие и допустим пересечение множеств индексов квазивершин, то необходимые условия существования эйлерова маршрута в кратном графе перестанут быть достаточными (такому ослабленному условию соответствует граф из примера 5).

References

[1] A. V. Smirnov, “The shortest path problem for a multiple graph”, *Automatic Control and Computer Sciences*, vol. 52, no. 7, pp. 625–633, 2018.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd. The MIT Press, McGraw-Hill Book Company, 2009, 1292 pp.

[3] C. Berge, *Graphs and Hypergraphs*. North-Holland Publishing Company, 1973, 528 pp.

[4] A. Basu and R. W. Blanning, “Metagraphs in workflow support systems”, *Decision Support Systems*, vol. 25, no. 3, pp. 199–208, 1999.

[5] A. Basu and R. W. Blanning, *Metagraphs and Their Applications*, ser. Integrated Series in Information Systems. Springer US, 2007, vol. 15, 172 pp.

[6] V. S. Rublev and A. V. Smirnov, “Flows in multiple networks”, *Yaroslavsky Pedagogichesky Vestnik*, vol. 3, no. 2, pp. 60–68, 2011, in Russian.

[7] A. V. Smirnov, “The problem of finding the maximum multiple flow in the divisible network and its special cases”, *Automatic Control and Computer Sciences*, vol. 50, no. 7, pp. 527–535, 2016.

[8] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 1962, 210 pp.

[9] V. S. Roublev and A. V. Smirnov, “The problem of integer-valued balancing of a three-dimensional matrix and algorithms of its solution”, *Modeling and Analysis of Information Systems*, vol. 17, no. 2, pp. 72–98, 2010, in Russian.

[10] A. V. Smirnov, “Network model for the problem of integer balancing of a four-dimensional matrix”, *Automatic Control and Computer Sciences*, vol. 51, no. 7, pp. 558–566, 2017.

[11] A. V. Smirnov, “Spanning tree of a multiple graph”, *Journal of Combinatorial Optimization*, vol. 43, no. 4, pp. 850–869, 2022.

[12] A. V. Smirnov, “The optimized algorithm of finding the shortest path in a multiple graph”, *Modeling and Analysis of Information Systems*, vol. 30, no. 1, pp. 6–15, 2023, in Russian.

[13] A. V. Smirnov, “NP-completeness of the minimum spanning tree problem of a multiple graph of multiplicity $k \geqslant 3$”, *Automatic Control and Computer Sciences*, vol. 56, no. 7, pp. 788–799, 2022.

[14] L. Euler, “Solutio problematis ad geometriam situs pertinentis”, *Commentarii Academiae Petropolitanae*, vol. 8, pp. 128–140, 1741, in Latin.

[15] C. Hierholzer, “Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren”, *Mathematische Annalen*, vol. 6, pp. 30–32, 1873, in German.

[16] M. Fleury, “Deux problèmes de géométrie de situation”, *Journal de mathématiques élémentaires*, vol. 2, pp. 257–261, 1883, in French.

[17] F. Harary, *Graph theory*. Addison-Wesley Pub. Co., 1969, 274 pp.

[18] M.-C. Cai and H. Fleischner, “An eulerian trail traversing specified edges in given order”, *Journal of Graph Theory*, vol. 19, no. 2, pp. 137–144, 1995.

[19] M.-C. Cai, “An algorithm for an eulerian trail traversing specified edges in given order”, *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 233–239, 1994.

[20] J. Abrham and A. Kotzig, “Transformations of euler tours”, *Annals of Discrete Mathematics*, vol. 8, pp. 65–69, 1980.