ТРУДЫ ИНСТИТУТА МАТЕМАТИКИ И МЕХАНИКИ УрО РАН

Том 31 № 3 2025

УДК 512.624.2, 519.714.7, 004.85

О ЗАДАЧЕ ВЫЯВЛЕНИЯ ХУНТЫ ДЛЯ ТАБЛИЧНО ЗАДАННЫХ ФУНКЦИЙ

П. Г. Емельянов

(n,k)-хунта — это булева функция от n переменных, которая зависит не более чем от k этих переменных. Задача определения того, является ли данная булева функция k-хунтой, и нахождения этих $n-k$ несущественных переменных является широко распространенной в машинном обучении (сокращение несущественных признаков) и проектировании электронных схем (сокращение фиктивных переменных). Естественным обобщением данного понятия, широко встречающегося в данных и других областях, например в теории игр, является понятие переменной, оказывающей малое влияние на результаты вычисления функции. Отдельный интерес представляет вопрос: рассматриваются ли несущественные или маловлиятельные переменные по отдельности или ансамблем? Вычислительная эффективность решения данных задач чрезвычайно актуальна для приложений. В частности, это зависит от используемого представления булевой функции. В настоящей работе булевы функции задаются полными или частичными таблицами истинности (примерами наборов данных). Здесь представлены два полиномиальной временной сложности алгоритма для поиска хунт, а также обсуждается случай, когда можно, аппроксимируя заданную функцию, сделать маловлиятельные переменные несущественными.

Ключевые слова: машинное обучение, логический дизайн схем, теория голосования, таблично заданные функции, проблема хунты, иррелевантные свойства, несущественные переменные, маловлиятельные переменные, декартово произведение, SQL JOINs, декомпозиция таблиц.

P. G. Emelyanov. On junta problem for table–defined functions.

A (n,k)-junta is a Boolean function of n variables that depends on at most k of these variables. The task of determining whether a given Boolean function is a k-junta and finding these $n-k$ unessential variables is common in machine learning (irrelevant feature reduction) and electronic circuit design (fictitious variable reduction). A natural extension of this idea, which is commonly found in these and other areas, for example, in game theory, is the notion of a variable with little influence on the results of a function's calculation. Of particular interest is the issue of whether insignificant or low-impact variables are considered individually or as an ensemble. The computational efficiency of solving these problems is of significant relevance for practical applications. This depends, in particular, on the representation of the boolean function employed. In this paper Boolean functions are specified by complete or partial truth tables (sample datasets). This paper presents two poly-time algorithms for finding juntas, as well as discussing the case where we approximate a given function making low-impact variables entirely unessential.

Keywords: machine learning, logic design, voting theory, table-defined functions, junta problem, irrelevant features, inessential variables, low-impact variables, cartesian product, SQL JOINs, decomposition of tables.

MSC: 13-04, 68Q32, 68T09, 68W30, 94D10

DOI: 10.21538/0134-4889-2025-31-3-fon-07

Введение

В машинном обучении, логическом синтезе схем и некоторых других областях часто возникает необходимость в работе с функциональными зависимостями, которые не требуют включения всех атрибутов (входных переменных рассматриваемых функций), поскольку некоторые атрибуты могут быть опущены без потери информации, так как они являются *фиктивными* (*нерелевантными* или *несущественными*) по отношению к интересующей нас задаче [1]. Кроме того, переменная может быть малосущественна для решаемой задачи и ее исключение, приводящее к приближению исходной функции, дает какие-то преимущества, например в затратах на представление или вычисление функции [2]. Сам факт малой/большой существенности переменных является предметом пристального внимания в разделе теории игр, посвященном процедурам голосований и социальному выбору [3], и в теории надежности [4].

Булева функция представляет собой отображение $F : \{0,1\}^n \to \{0,1\}$, где $n \geq 1$ — *местность* F. Переменная x_i является существенной переменной F, если F зависит от своего i-го аргумента, т. е. если существуют значения $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \{0,1\}$ такие, что унарная функция $F(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, a_n)$ является неконстантной. В противном случае x_i — фиктивная переменной. (n,k)-*хунта* — это булева функция от n переменных, которая зависит от $k < n$ переменных. Задачи определения того, является ли данная булева функция (n,k)-хунтой, и нахождение этих $n-k$ фиктивных переменных возникают в различных ситуациях. Один из примеров — упрощение таблиц принятия решений за счет уменьшения количества атрибутов, включенных в условия. Сузив набор атрибутов, можно получить более простой набор правил принятия решения. Удаление необязательных атрибутов позволяет сэкономить место в памяти или вычислительное время. Цель сокращения атрибутов наборов в контексте таблиц принятия решений состоит в том, чтобы получить минимальное подмножество атрибутов с той же силой классификации, что и весь набор атрибутов [5]. Аналогичная задача возникает при проверке корректности логических баз знаний [6]. Другой пример необходимости проверки того, что функция является хунтой, — когда данная информация необходима для применения методов обучения, ориентированных на такой класс функций [7].

Аналогичная проблема актуальна при логическом синтезе схем — удаление фиктивных переменных приводит к схемам с более качественными параметрами [8; 9]. Сокращенный набор входных переменных затем может быть использован для более качественной оптимизации при проектировании. При этом схемы могут быть специфицированы различными способами. Разнообразие форм представления булевых функций в процессе разработки создает значительные вычислительные проблемы, поскольку это оказывает сильное влияние на последующие алгоритмические решения. В частности, зачастую нецелесообразно выполнять преобразование между различными формами из-за потребления ресурсов и проектировщики вынуждены работать с имеющимися представлениями булевых функций.

В некоторых случаях определение (не)существенных переменных булевой функции становится простой задачей. Например, если легко можно построить ее алгебраическую нормальную форму (АНФ), также известную как полином Жегалкина или форма положительной полярности Рида — Мюллера, то эта форма будет содержать только существенные переменные. Однако для дизъюнктивной нормальной формы это верно только в том случае, если она сокращенная; в общем случае это не верно. Важной формой спецификации булевых функций, используемых при проектировании схем, является таблица истинности (далее предполагается лексикографический порядок входных кортежей). Эта спецификация может быть как полной, так и частичной. Использование только столбца-результата полной таблицы истинности дает представление в виде вектора (результирующих) значений. Частично заданные булевы функции, т. е. функции, в которых не все входные кортежи таблицы истинности присутствуют (по англ. don't-care inputs), широко распространены как в логическом дизайне, так и в машинном обучении. Они предоставляются в виде наборов образцов данных, а именно, наборов пар $\langle x_i, F(x_i)\rangle_{i=1}^{S}$ (x_i — это двоичные кортежи длиной n, S может быть меньше 2^n). Представления булевых функций в виде черного ящика, а также их СДНФ/СКНФ или характеристические векторы множеств единиц/нулей также могут рассматриваться как полные или частичные таблицы истинности.

Имеет место естественное обобщение задачи выявления несущественных переменных. Рассматриваемая переменная может и не быть несущественной, но ее влияние мало. Один из подходов к формализации этого понятия [2] — переменная маловлиятельная, если мала вероятность изменения значения функции при изменении значения этой переменной (в целом по всему набору входов). Для многомерных моделей в машинном обучении может оказаться полезным изучить возможность удаления из модели переменной, оказывающей незначительное влияние, чтобы упростить параметры модели и повысить в целом ее устойчивость. Ситуация усложняется, если рассматривать ансамбль переменных: на данный момент есть несколько различающихся определений влиятельности группы переменных [10]. В [11] авторы состави-

ли список различных определений значимости переменных и обсудили единый подход для суждений об относительном влиянии входных переменных на результаты булевых функций. Они вводят общую аксиоматическую систему, которая определяет класс функций значений важности (importance value functions, IVFs), и демонстрируют, что многие хорошо известные концепции значимости переменных удовлетворяют аксиомам IVF. В этом контексте логично сформулировать следующую задачу: путем удаления некоторых входных значений найти такую аппроксимацию булевой функции, при которой малозначимая переменная (или несколько переменных) аппроксимируемой функции становится несущественной переменной аппроксимирующей функции и, следовательно, может быть устранена.

Цель и предмет нашего исследования — представление булевых функций в виде таблиц истинности и алгоритмика для них; это вызывает у нас интерес к вычислительным моделям на основе таблиц, в частности к тем, которые используются в реляционных базах данных. Анализ наборов данных различного происхождения — это актуальная проблема. Методы декомпозиции являются мощными аналитическими инструментами в области интеллектуального анализа данных, а также во многих других областях, по сути дела, обеспечивая анализ зависимостей в таблицах. В этой работе рассматривается специальный вид зависимостей. Определение "декартовости" набора данных, т. е. того, может ли он быть представлен в виде неупорядоченного декартова произведения двух (или более) наборов данных, имеет важное значение, по крайней мере в четырех из шести категорий задач анализа данных, а именно: в обнаружении аномалий, моделировании зависимостей, выявлении скрытых закономерностей в данных и создании более лаконичного представления данных. Декартова декомпозиция формирует основу для определений основных типов зависимостей, охватываемых теорией реляционных баз данных, поскольку многочисленные концепции зависимостей основаны на операции `JOIN`, которая является обратной декартовой декомпозиции.

В настоящей статье рассматривается подход к выявлению (не)существенных переменных для таблично заданных булевых функций, который основан на анализе декомпозируемости/разложимости таблиц применением некоторых полиномов, индуцируемых этими таблицами. В разд. 1 представлены используемые в данной работе определения и основные результаты о декомпозиции таблиц, которые были получены в [12–15]. В подразд. 2.1 на основе декартового разложения таблиц истинности сформулировано утверждение о свойствах (не)существенных входов, которое далее используются для разработки метода вычисления переменных хунты. Алгоритм аппроксимации булевой функции (путем удаления малого количества входных данных), приводящий к появлению в аппроксимирующей функции несущественных переменных, рассматривается в подразд. 2.2. В подразд. 2.3 исходя из свойств разложения частичных таблиц истинности относительно атрибута-результата показано, что существуют функции, наборы переменных которых могут быть разделены на две части, являющиеся либо существенными, либо несущественными в зависимости от нашего выбора. Можно сказать, что эти функции — двойственные хунты. Это означает, что после выполнения проекции на любую часть разбиения получается функция, которая равна исходной функции на соответствующих входных данных, причем другая часть содержит несущественные переменные, и наоборот. В худшем случае это позволяет нам сократить как минимум половину исходных переменных. Дальнейшие направления исследований изложены в заключении.

1. Основные факты о декартовой декомпозиции таблиц

Настоящая работа лежит в русле исследований, проводимых нами в течение последних лет, в которых разрабатывается единый подход к некоторым задачам декомпозиции объектов/систем, имеющих схожее происхождение. Обобщение декартова произведения, так называемое *произведение семейств множеств*, определяется следующим образом (базовые мно-

жества $\mathcal{A}$ и $\mathcal{B}$ конечны; $\wp$ обозначает множество всех подмножеств заданного множества):

$$\text{for}\ \ \mathcal{A}\cap\mathcal{B}=\varnothing,\ A\subseteq\wp(\mathcal{A}),\ B\subseteq\wp(\mathcal{B}),\quad A\bowtie B\stackrel{def}{=}\{\alpha\cup\beta\mid\alpha\in A\wedge\beta\in B\}.$$

Пример:

$$A=\{\varnothing,\{x\},\{y\}\}\qquad B=\{\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$$

$$A\bowtie B=\big\{\{a,b\},\{a,c\},\{b,c\},\{a,b,c\},\{a,b,x\},\{a,b,y\},\{a,c,x\},\{a,c,y\},\{b,c,x\},\{b,c,y\},\{a,b,c,x\},\{a,b,c,y\}\big\}.$$

Это одна из операций алгебры семейств множеств [16], она является важным инструментом для композиционного построения объектов и систем. Ее естественным обобщением является допущение об общих элементах между компонентами произведения. Точно так же, как произведение — важный инструмент для композиции, его обращение не менее важно в качестве инструмента для декомпозиции. Оно возникает в различных прикладных областях: в проектировании логических схем, машинном обучении, интеллектуальном анализе данных и баз данных, теории кодов с исправлением ошибок и криптографии, теории (гипер)графов, комбинаторной оптимизации, теории игр, структурной теории надежности, в онтологическом моделировании и так далее.

Легко установить связь между задачей обращения произведения семейств множеств с непересекающимися базовыми множествами и задачей факторизации мультилинейных полиномов. Для предыдущего примера:

$$ab+ac+bc+abc+abx+aby+acx+acy+bcx+bcy+abcx+abcy=(ab+ac+bc+abc)(x+y+1).\quad(1.1)$$

Задача факторизации полиномов является классической задачей алгебры, и разработка эффективных алгоритмов факторизации имеет важное значение в различных областях, в том числе в упомянутых выше. В статье [12] был предложен детерминированный полиномиальной временной сложности алгоритм факторизации мультилинейных полиномов над конечным полем GF(2) (см. также [15]). Ниже приведены основные обозначения и определения.

• *Мультилинейными полиномами* называются полиномы, в которых показатели степени всех переменных во всех мономах равны 1. В работе рассматриваются мультилинейные полиномы из $\mathrm{GF}(2)[x_1,\ldots,x_N]$. Примеры мультилинейных полиномов представлены в (1.1).

• Пусть для полинома P множество $\mathtt{vars}(P)=\{x_1,\ldots,x_N\}$, $|\mathtt{vars}(P)|=N$, — это *множество его переменных*. Отметим, что для мультилинейных полиномов множества переменных его факторов (если они существуют) не пересекаются. Это следствие того факта, что эти полиномы образуют область однозначной факторизации. Неприводимые факторы индуцируют однозначное разбиение $\mathtt{vars}(P)$.

• *Проекцией* полинома P на множество переменных $V\subseteq\mathtt{vars}(P)$, которая обозначается через $P\downarrow_V$ (известная в теории булевых функций как “экзистенциальная абстракция”), — это новый полином, образованный по следующим правилам: во всех мономах исходного полинома переменные $\mathtt{vars}(P)\setminus V$ означиваются в 1, а затем все повторяющиеся мономы, включая константу 1, отбрасываются и из оставшихся формируется полином-результат. Пример: $(xu+xv+xy)\downarrow_{\{x,y\}}=x+xy$, в частности, $P\downarrow_{\mathtt{vars}(P)}=P$ и $P\downarrow_\varnothing=1$.

• *Длина* полинома P — это количество его ненулевых мономов $m_1,\ldots,m_M$: $|P|=M$.

• *Размер* полинома — это величина $\sum_{k=1}^{M}|m_k|=O(NM)$, где для неконстантных мономов $|m_k|=|\mathtt{vars}(m_k)|$. Она описывает общие расходы памяти для представления полинома.

• В целях краткости нотации для *означивания* полинома по переменной будет использовано $P^a_x=P|_{x=a}$. При этом P^{ab}_{xy} означает последовательное означивание $P|_{x=a,y=b}$.

• (Формальная) *производная* полинома P по переменой x, обозначаемая через P'_x, определяется следующим образом: $P'_x=\big(\sum_{m\in P}(\ldots x^d\ldots)\big)'_x=\sum_{m\in P}\big(d\ldots x^{d-1}\ldots\big)$. Для мультилинейных полиномов $P'_x=P^0_x+P^1_x$.

• Предикаты $x \,|\, P$ и $x \nmid P$ истинны, если переменная x соответственно делит или не делит полином P. Если $x \,|\, P$, переменная x называется *тривиальным фактором* и полином может быть представлен как $P = xP_x^1 = xP'_x$. Будем предполагать, что полиномы не имеют тривиальных факторов.

Следующая теорема, являющаяся основой всех развиваемых методов, приведена впервые в несколько ином виде и доказана в [12, Theorem 4, Lemma 1] (см. также [15, Lemma 18, Theorem 19]). Ниже ее формулировка дана в форме, удобной для алгоритмизации.

Теорема 1. *Если $P = GH$, тогда для любой переменной $x \in \mathtt{vars}(P)$, которая не является тривиальным фактором, верно*:

1. $(P_x^0\, P'_x)'_y \neq 0$, *если x и y принадлежат одному фактору, при этом он неразложим*;
2. $(P_x^0\, P'_x)'_y = 0$, *если x и y принадлежат разным факторам, при этом фактор, содержащий y, может быть разложимым.*

Если P неразложим, то $\forall x, y \in \mathtt{vars}(P)$, $x \neq y$, верно $(P_x^0\, P'_x)'_y \neq 0$.

Переменная x и полином $S(P, x) = S_x = P_x^0 P'_x$, указанные в теореме, называются соответственно классифицирующей/сортирующей переменной и классифицирующим/сортирующим полиномом; процедура проверки обращения в нуль производной сортирующего полинома по некоторой переменной называется процедурой классификации/сортировки этой переменной. Обратите внимание, что S_x может содержать мономы, включающие переменные во второй степени, т. е. не является мультилинейным.

Эта теорема дает простой алгоритм для нахождения разбиения набора переменных по факторам. Дополнив его процедурой проецирования, имеем *алгоритм для факторизации мультилинейных полиномов*.

```
Define ExpliciteFDAlgorithm(P) returning pair of polynomials:
  Pick x from vars(P).
  Let Σ_same := {x},   Σ_other := ∅,   S_x := P_x^0 P'_x.
  For Each Variable y ∈ vars(P) \ {x} Do
    If (S_x)'_y = 0 Then  Σ_other := Σ_other ∪ {y}.
                    Else  Σ_same := Σ_same ∪ {y}. End If.
  End For.
  Return P↓_{Σ_same} and P↓_{Σ_other}.
End Define.
```

Его общая сложность в наихудшем случае равна $O(\mathcal{M}(N, M) + N^2M^2) = O(N^2M^2)$, где $\mathcal{M}$ — сложность умножения двух полиномов в $\mathrm{GF}(2)[x_1, \ldots, x_N]$. Основной проблемой, препятствующей прямому применению этого алгоритма, является необходимость явного вычисления сортирующего полинома $P'_x P_x^0$: для больших полиномов любой выбор сортирующей переменной x приводит к тому, что его длина равна $\Omega(M^2)$ с высокой вероятностью. Поэтому была разработана *модификация* этого алгоритма, уменьшающая количество промежуточных данных в процессе приведения. С помощью характеристических векторов, представляющих мономы, *алгоритм вычисления сортирующего полинома* элегантно записывается в функциональной технике MapReduce, широко используемой в обработке больших данных. С нашим скриптом на языке Python можно ознакомиться по ссылке [17]; другие алгоритмы факторизации обсуждаются в [15]).

В [13] этот метод был применен для демонстрации эффективного обращения SQL-оператора `CROSS JOIN`. Связь между задачами декартовой декомпозиции и факторизации полиномов может быть легко установлена. Каждый кортеж отношения — моном полинома, где значения атрибутов играют роль переменных. Они называются *атрибутивные переменные* и

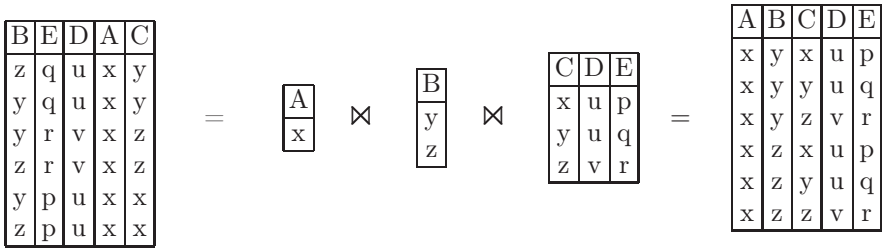

Рис. 1. Обращение декартова произведения таблиц.

различаются для разных значений одного и того же атрибута. Важно отметить, что разные атрибуты одного и того же типа порождают разные переменные. `NULL` также является "типизированным" и в этом смысле не отличается от других переменных, соответствующих атрибуту. Этот полином будем называть *табличным полиномом*. Например, для приведенного на рис. 1 отношения соответствующим табличным полином и его факторизацией являются

$$\begin{aligned} &B_z E_q D_u A_x C_y + B_y E_q D_u A_x C_y + B_y E_r D_v A_x C_z \\ &+ B_z E_r D_v A_x C_z + B_y E_p D_u A_x C_x + B_z E_p D_u A_x C_x \\ &= A_x (B_y + B_z)(E_q D_u C_y + E_r D_v C_z + E_p D_u C_x) \end{aligned} \tag{1.2}$$

Обратите внимание, он однороден — размеры всех мономов равны степени рассматриваемого отношения. Далее будут встречаться таблицы, представляющие таблично заданные функции, где атрибуты интерпретируются как входные данные и значение функции на этих данных. Для них табличный полином определяется точно также (см. примеры в разд. 2 на рис. 3–5).

Однако наборы данных с чисто декартовой структурой произведения встречаются редко. Декартова декомпозиция имеет естественные обобщения, позволяющие нам решать более сложные задачи. Например, показано (см. [15]), что больше полиномов разложимо, если допустить, что компоненты декомпозиции могут совместно использовать переменные из некоторого заданного набора (это также называется *булевой факторизацией*). Эта же идея была использована для разработки алгоритма декомпозиции наборов данных. Предложенный в [14] алгоритм обращения `INNER JOIN ON` (с условием равенства) в отличие от полиномиального аналога не зависит от количества общих переменных (количества значений атрибутов) и, следовательно, остается практичным для больших таблиц. Стоит упомянуть, что при некоторых допущениях возможны и другие типы условий.

На рис. 2 представлен пример обращения `INNER JOIN ON` относительно *связывающего атрибута*, соответствующего x_4. Одним из источников такого рода примеров является область поддержки принятия решений, которая тесно связана с управлением базами данных и имеет множество приложений. С математической точки зрения таблица решений — это отображение, определяемое, иногда частично, явным перечислением аргументов и результатов (набор правил или набор следствий "условия — выводы"). Декомпозиция таблицы решений заключается в нахождении представления отображения $F(X)$ в виде $G(X_1, H(X_2))$, $X = X_1 \cup X_2$, которое может быть не уникальным. Отображение H можно рассматривать как новую ранее неизвестную концепцию, дающее новое знание о данных, в частности, их более компактное представление.

Этот вид декомпозиции известен в области логического проектирования как декомпозиция Ашенхерста. Указанный метод может быть использован для декомпозиции булевых функций, заданных в виде характеристических векторов их нулей/нулей (множеств ON/OFF), в комбинацию меньших функций. Пример, приведенный на рис. 2, можно интерпретировать следующим образом. Исходная булева функция, заданная, допустим, с помощью характеристического вектора ее единиц (самая большая таблица T), может быть разложена на совокупность

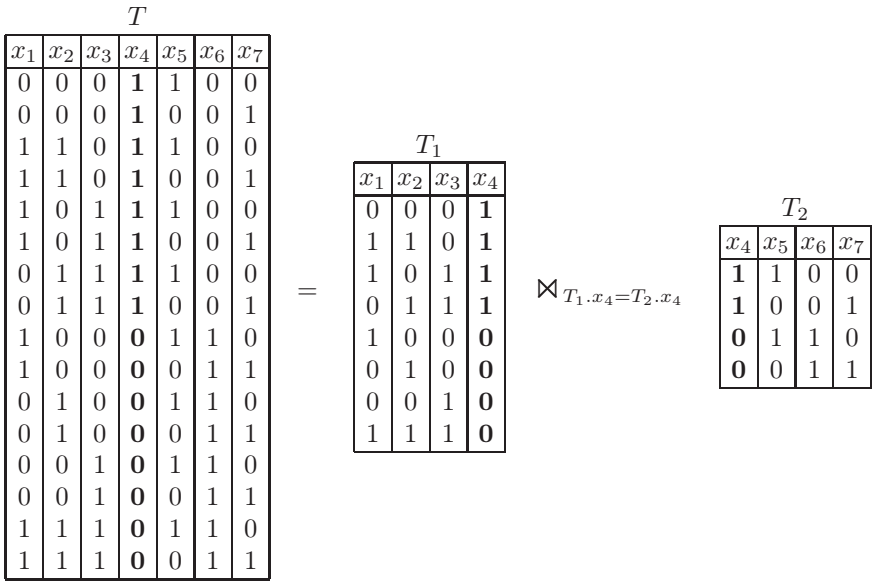

Рис. 2. Обращение `INNER JOIN ON` с равенством.

меньших функций, которые имеют общую переменную x_4 и также представлены в виде своих характеристических векторов (таблиц T_1 и T_2). Описание этой декомпозиции приведено в подразд. 2.3.

Таблицы истинности булевых функций являются хорошо известным примером таблиц принятия решений. Таблица истинности для логической функции с входными переменными n — это булева матрица $m \times n+1$, где m меньше или равно 2^n. Если меньше, то эта булева функция называется *частично заданной* или просто *частичной*. Все строки матрицы уникальны в первых n столбцах, а последний столбец называется *атрибут-результат* или просто *результат*. В области логического проектирования таблицы частичной истинности появляются при наличии *don't-care-inputs* (*DC-inputs*), т. е. входных кортежей, для которых результат вычисления функции неважен.

З а м е ч а н и е 1. В связи с главной темой данной статьи — (не)существенные переменные — хотелось бы отметить следующее обстоятельство. Для пары входов $\langle a_1,\ldots,a_{i-1},0,a_{i+1},\ldots,a_n\rangle$ и $\langle a_1,\ldots,a_{i-1},1,a_{i+1},\ldots,a_n\rangle$ в определении частичной функции возможны случаи: 1) оба входят , 2) оба не входят, 3) входит какой-нибудь один из них. Интерпретация последнего "непарного" случая — считать вход x_i существенным или несущественным — зависит от решаемой задачи (при доопределении на отсутствующем входе возможны оба варианта). В контексте данной статьи рассматриваются частичные функции, для которых верны первые два случая.

В статье рассматриваются только булевы функции, хотя некоторые результаты применимы и к любым таблично заданным функциям. Кроме того, тема векторных булевых функций также выходит за рамки данной статьи.

2. Выявление несущественных входов таблиц истинности

Выявление существенных переменных булевых функций, представленных в виде таблиц истинности, может быть относительно простым процессом. Если атрибуты не являются существенными, удаление соответствующего столбца из таблицы приведет к созданию таблицы, в которой одинаковые строки будут отображаться ровно дважды. Это в точности соответствует

такому свойству необязательной переменной, что производная булевой функции по этой переменной обращается в нуль: $F'_x = F^0_x \oplus F^1_x = 0$. Проверка поочередно всех атрибутов таблицы позволяет выявить все несущественные. Однако целесообразно рассмотреть другой подход, демонстрирующий новые идеи, которые могут быть полезными, например, для переменных с низким уровнем воздействия.

Для большей ясности мы используем следующую терминологию. Название *переменная* относится к входной переменной булевой функции, название *вход* — к входным переменным таблицы истинности, т. е. к переменным, соответствующим первым n атрибутам таблицы, а название *выход* — к переменной, соответствующей последнему атрибуту таблицы. Для переменных табличного полинома мы используем имена *атрибутивная переменная* и *результирующая переменная*. В случае булевых таблиц каждый атрибут (столбец), который содержит два значения, индуцирует две переменные.

2.1. Выявление несущественных входов посредством факторизации табличного полинома

Для краткости будем называть булевы функции, в которых все входные переменные являются существенными, *артелями* (также они называются невырожденными булевыми функциями) в отличие от хунт. Артелями с одной переменной являются две неконстантные булевы функции — тождественная и отрицание. Среди полностью определенных булевых функций существует $A(n) = \sum_{k=0}^{n}(-1)^k C_n^k 2^{2^{n-k}}$ артелей над n входами.

n-местные функции, являющиеся тождественными константами, представляют собой пример тотально вырожденных функций. Отметим, что в части теории игр, посвященной голосованию, $(n,1)$-хунты называются *функциями-диктаторами*. Опишем процесс генерации (n,k)-хунт, используя декартово произведение таблиц. Таблицы истинности (n,k)-хунт могут быть получены в три этапа.

1. Взять всевозможные декартовы произведения булевой таблицы $2^{n-k} \times n-k$ (все строки различны и лексикографически упорядочены) и таблиц истинности всех k-артелей.

2. Породить всевозможные таблицы, полученные перестановками n крайних левых столбцов в каждой таблице, созданной на предыдущем шаге (результат — всегда самый правый столбец).

3. Отсортировать все полученные таблицы по n крайним левым столбцам в лексикографическом порядке.

Другими словами, (n,k)-хунта содержит в качестве своего ядра k-артель, каждая строка которой "размножена" всеми возможными булевыми кортежами степени $n-k$. Таким же образом частичные хунты получаются из частичных булевых функций-артелей. Это предлагает алгоритмический способ определения всех существенных переменных в терминах факторизации табличных полиномов: атрибутные переменные, соответствующие существенным входам, относятся к тому же фактору, что и результирующие переменные.

Предложение 1. *Дана таблица истинности, полная или частичная.*

1. *Табличный полином артели неразложим* (*все входы таблицы истинности являются существенными*).

2. *Если табличный полином разложим, то существенные входы соответствуют атрибутивным переменным, которые принадлежат фактору, содержащему результирующие переменные, а атрибутивные переменные кофактора соответствуют несущественным входам.*

Доказательство. Рассмотрим артель F и предположим противное — табличный полином для F разложим. Напомним, разложение мультилинейных полиномов индуцирует разбиение множества его переменных.

Факты об этом табличном полиноме:

- Так как все его входы существенны, то для каждого входа x ее атрибутивные переменные $x_0, x_1 \in \mathtt{vars}(F)$, и если F разложим, то они обе принадлежат одному фактору.

- Поскольку он не является тождественно константой, обе результирующие переменные $F_0, F_1 \in \mathtt{vars}(F)$.

- Если он разложим, то не может содержать фактор вида $F_0 + F_1$, потому что, если это было бы так, то исходная таблица не являлась бы таблицей истинности для булевой функции. Действительно, раскрывая факторы $(\ldots + \alpha + \ldots)(F_0 + F_1)$, имеем полином $\ldots + \alpha F_0 + \alpha F_1 + \ldots$, который нельзя свернуть в таблицу истинности ввиду того, что одним и тем же входным значениям соответствуют разные выходные. Также результирующие переменные F_0 и F_1 не могут входить в разные факторы разложения, потому что при обратном действии появились бы мономы вида $\ldots + \alpha F_0 F_1 + \ldots$, а это невозможно.

- Если он разложим, то в каждом факторе все его мономы содержат одну из двух атрибутивных переменных, соответствующих входам, попавшим в этот фактор. Более того, должны встречаться обе переменные, так как в противном случае имеем константный вход, который очевидно избыточен.

Таким образом, если табличный полином артели разложим, то он имеет вид

$$(x_0 G_0 + x_1 G_1)(\ldots + \alpha_k F_{v_k} + \ldots) = \ldots + x_0 \alpha_k G_0 F_{v_k} + x_1 \alpha_k G_1 F_{v_k} + \ldots.$$

Если $G_0 = G_1$, то имеет место быть ситуация, когда изменение значения входа x при остальных фиксированных входах $\alpha_k G_0$ не приводит к изменению результата функции F_{v_k} (и это верно для всех k). Получили противоречие, что все входы существенны. Кроме того, аттрибутивные переменные несущественного входа выделяются из табличного полинома в виде фактора $x_0 + x_1$.

Отметим, что ситуация $G_0 \neq G_1$ невозможна в связи с тем, что может возникнуть только для частичных функций, у которых в определении есть “непарные” входные данные (см. замечание 1). Пример такого табличного полинома (таблица истинности легко восстанавливается) приведен ниже (здесь артель — исключающее ИЛИ, “существенность” атрибутов x, y, z не определена):

$$\begin{aligned} &u_0 v_0 x_0 y_0 z_0 F_0 + u_0 v_0 x_0 y_1 z_1 F_0 + u_0 v_0 x_1 y_0 z_1 F_0 + u_1 v_1 x_0 y_0 z_0 F_0 \\ &+ u_1 v_1 x_0 y_1 z_1 F_0 + u_1 v_1 x_1 y_0 z_1 F_0 + u_0 v_1 x_0 y_0 z_0 F_1 + u_0 v_1 x_0 y_1 z_1 F_1 \\ &+ u_0 v_1 x_1 y_0 z_1 F_1 + u_1 v_0 x_0 y_0 z_0 F_1 + u_1 v_0 x_0 y_1 z_1 F_1 + u_1 v_0 x_1 y_0 z_1 F_1 \\ &= (x_0 y_0 z_0 + x_0 y_1 z_1 + x_1 y_0 z_1)(u_0 v_0 F_0 + u_0 v_1 F_1 + u_1 v_0 F_1 + u_1 v_1 F_0). \end{aligned}$$

Предложение доказано.

Алгоритмическая реализация данного подхода очень проста: надо реализовать проверку условий теоремы 1. Отметим свойства задачи, оптимизирующие вычислительные ресурсы. Во-первых, достаточно вычислить разбиение переменных, сами факторы не нужны, а значит, разумно реализовать алгоритм именно разбиения переменных, а не факторизации (существуют алгоритмы факторизации, строящие напрямую факторы без отыскания разбиения переменных). Во-вторых, аттрибуты таблицы сами по себе индуцируют некоторое разбиение атрибутивных переменных в смысле принадлежности к тому или иному фактору разложения. Таким образом, проверять можно одну атрибутивную переменную для каждого атрибута, вторая классифицируется автоматически. В-третьих, упомянутый алгоритм сортирует переменные по факторам быстрее, если в качестве сортирующей переменной выбрана переменная из

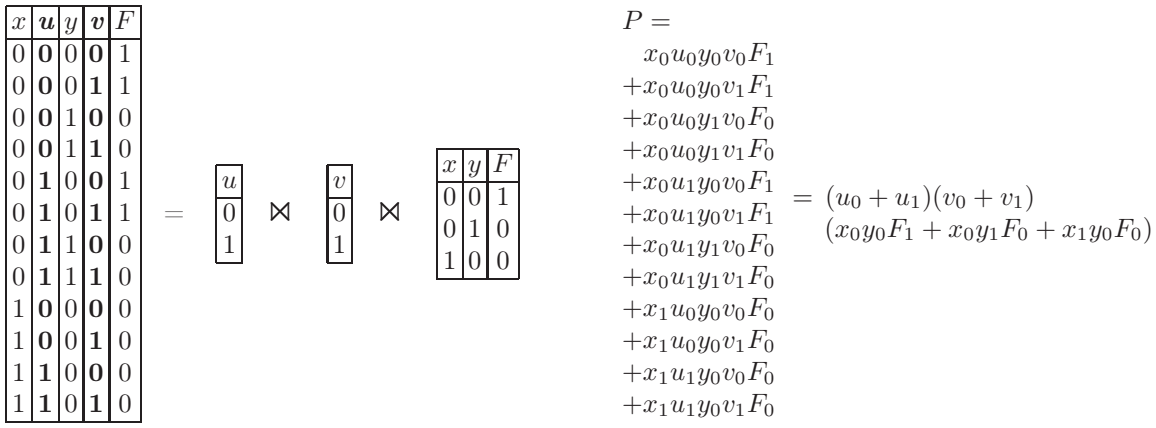

Рис. 3. Разложение (4,2)-хунты (булева функция частичная).

фактора, содержащего большее количество переменных. Можно предположить, что несущественных переменных сравнительно мало. Таким образом, в качестве сортирующей следует выбирать одну из результирующих переменных. В заключение отметим, что подход не требует предположений о том, какие входы могут быть несущественными. Они определяются в результате общего процесса факторизации. Пример приведен на рис. 3.

2.2. Аппроксимация таблиц истинности, порождающая несущественные входы

Основываясь на той же методологии, можно изучить несколько иной подход к определению несущественных переменных. Некоторая переменная может и не является несущественной, однако ее влияние минимально в том смысле, что результат функции зависит от этой переменной только на небольшом количестве входных данных. Возможно, при удалении небольшого количества строк из таблицы истинности можно было бы добиться того, что функция перестанет зависеть от этой переменной (возвращать ли эти строки без этой переменной обратно в определение функции зависит от дополнительного предметного анализа). Еще одна мотивация для этих исследований заключается в следующем. В общем случае проблема идентификации переменных с низким уровнем влияния в различных формах представления булевых функций — это сложная алгоритмическая задача [18; 19]. В данной работе рассматривается новый алгоритмический подход к этой проблеме в случае, когда булевы функции представлены в виде таблиц истинности. Кроме того, мы хотим распространить концепцию слабого воздействия на группу из нескольких переменных, т. е. когда переменные рассматриваются ансамблем, а не по одиночке.

Рассмотрим форму записи полинома, индуцируемую подмножеством его переменных:

$$Coll(P, X, Y) = \sum_i C_i(X) m_i(Y), \quad \texttt{vars}(P) = X \cup Y, \quad X \cap Y = \varnothing,$$

$$C_i(X) \in \mathrm{GF}(2)[X], \quad \text{мономы } m_i(Y) \in \mathrm{GF}(2)[Y].$$

$C_i(X)$ называются *свернутыми относительно X коэффициентами*. Пример:

$$P = ux^2y + u^2xy + x^2y + ux + xy + x,$$

$$Coll(P, \{u\}, \{x, y\}) = (u+1)x^2y + (u^2+1)xy + (u+1)x$$

Идея аппроксимации состоит в нахождении соответствующих мономов табличного полинома (строк таблицы) с помощью теоремы 1 и предложения 1, удаление которых делает рассматриваемый вход несущественным. Это метод еще можно охарактеризовать как метод неопределенных коэффициентов.

Алгоритм отыскания апроксимации функции удалением маловлиятельных входов.

Вход алгоритма:

Таблица истинности булевой функции и вход(ы), относительно которых отыскивается аппроксимация.

Шаги алгоритма:

1. Преобразуем табличный полином $P(X)$ следующим образом: каждый моном полинома домножается на уникальную булеву переменную. Эта величина указывает, что этот моном будет включен или исключен из итогового табличного полинома. Переменные, соответствующие данным величинам $C = \{c_i\}_{i=1}^{M}$, будут называться *управляющими*. Далее работа ведется с полиномом $P^c = P(C, X)$.

2. С использованием (любой) результирующей переменной в качестве сортирующей строим сортирующий полином для $S(P^c)$.

3. Вход несущественен, если его атрибутивные переменные и результирующая переменная относятся к разным факторам. Как уже известно, это подразумевает равенство нулю производной сортировочного полинома. Данный полином равен нулю тогда и только тогда, когда все свернутые относительно C коэффициенты полинома $Coll(S(P^c), C, X)$ равны нулю. Легко убедиться, что эти свернутые коэффициенты представляют собой квадратичные полиномы от управляющих переменных. Следует отметить, что группа входов может быть обработана аналогичным образом: для всех входов в этой группе нужно найти соответствующие производные и пополнить общую систему. В таком случае удаляемых строк таблицы может оказаться меньше, чем в случае, если рассматривать переменные последовательно по одной.

4. Необходимо отслеживать, чтобы удаление строк таблицы не привело к тому, что конечная булева функция станет константной. Это может быть выражено средствами алгебры логики, но по причине снижения в таком случае общей производительности ситуацию проще отследить программно во время проверки решения.

5. Это порождает систему квадратных уравнений над M переменными. Ее решения предоставляют варианты сокращения строк таблицы истинности. Посколько хочется как можно лучше аппроксимировать булеву функцию, то решение следует выбирать так, чтобы управляющие переменные как можно реже принимали значение “нуль”.

Выход алгоритма:

Варианты удаления строк (мономов табличного полинома), которые индуцируют варианты аппроксимации исходной булевой функции, которые являются артелью. □

Временная сложность данного метода пока неизвестна. Хотя решение систем квадратных уравнений над GF(2) в общем случае является сложной задачей, в данном конкретном случае это делается довольно эффективно, поскольку в большинстве случаев полиномы системы являются просто мономами. На рис. 4 приведен пример такой аппроксимации. Отметим, что система уравнений может иметь много решений. Так, если для приведенного примера было бы

u	v	$\boldsymbol{x}$	y	F
0	0	**1**	0	1
0	0	**1**	1	0
0	**1**	**0**	**1**	**0**
0	1	**1**	1	0
1	**0**	**0**	**1**	**0**
1	0	**1**	0	0
1	0	**1**	1	1
1	1	**1**	0	0
1	1	**1**	1	0

$$
\begin{aligned}
P^c = {} & c_1u_0v_0x_1y_0F_1 \\
& +c_2u_0v_0x_1y_1F_0 \\
& +c_3u_0v_1x_0y_1F_0 \\
& +c_4u_0v_1x_1y_1F_0 \\
& +c_5u_1v_0x_0y_1F_0 \\
& +c_6u_1v_0x_1y_0F_0 \\
& +c_7u_1v_0x_1y_1F_1 \\
& +c_8u_1v_1x_1y_0F_0 \\
& +c_9u_1v_1x_1y_1F_0
\end{aligned}
$$

Система:

$$
\begin{cases}
c_1c_3 = 0 \\
c_1c_5 = 0 \\
c_3c_7 = 0 \\
c_5c_7 = 0
\end{cases}
$$

Решение:

$c_1 = 1, c_3 = 0$

$c_5 = 0, c_7 = 1$

u	v	y	$\widetilde{F}$
0	0	0	1
0	0	1	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Рис. 4. Аппроксимация булевой функции, порождающая несущественные входы.

выбрано другое решение, где $c_1 = c_7 = 0$, то результатом аппроксимации была бы константная функция, что, возможно, не является целью в данном случае.

2.3. Выявление относительных (не)существенных переменных

В разд. 1 приведена декомпозиция для оператора `JOIN ON`, когда атрибут, рассматриваемый как предполагаемое соединение, соответствует входам. Еще один интересный вариант появляется, когда рассматривается декомпозиция частичной таблицы истинности (DC-входы отсутствуют в таблице) по отношению к атрибуту-результату. Это позволяет выявить новое понятие относительной (не)существенности части входов.

Пример на рис. 5 объясняет эту идею. В итоге исходная функция представима с использованием функции-комбинатора H следующим образом: $F(x_1, x_2, x_3, x_4) = H(F(x_1, x_2), F(x_3, x_4))$.

Доопределяя комбинатор H для DC-входов, можно получить различные разложения. Существуют четыре возможных результата для доопределения H. Например, если доопределить H до дизъюнкции/конъюнкции (OR/AND), то получится OR/AND-декомпозиция булевой функции с непересекающимися по переменным компонентами.

Однако наиболее примечательный случай возникает, когда функция H доопределяется до получения так называемых *левой* или *правой проекций*. Их векторами значений являются (0101) и (0011) соответственно. Проекция в данном случае означает, что результат первого или второго операнда передается в качестве результата всей функции, а другой операнд игнорируется. Таким образом, имеем “относительную” (не)несущественность переменных: все входные переменные разбиваются на две части; любую из них можно выбрать в качестве существенных переменных, в то время как другая будет содержать несущественные. При этом отдельно взятая переменная может не являться несущественной (как в примере на рис. 5). Это позволяет сократить количество входных переменных как минимум наполовину, поскольку самая большая часть разбиения содержит самое меньшее половину всех входных переменных исходной функции. Следовательно, существуют эффективно вычислимые “двойственные” хунты.

Предложение 2. *Существуют частичные булевы функции $F^{(n)}(X)$, X — множество входных переменных, $|X| = n > 1$, и булевы функции $F_1^{(n_1)}(X_1)$ и $F_2^{(n_2)}(X_2)$, $X = X_1 \cup X_2$, $X_1 \cap X_2 = \varnothing$, $|X_1| = n_1 > 0$, $|X_2| = n_2 > 0$, $n_1 + n_2 = n$, такие что*

$$F^{(n)}(X) = F_1^{(n_1)}(X_1) = F_2^{(n_2)}(X_2)$$

на соответствующих подмножествах аргументов, где изначально определена $F^{(n)}(X)$. Проверка этого свойства для $F^{(n)}(X)$ и построение функции $F_1^{(n_1)}(X_1)$ и $F_2^{(n_2)}(X_2)$ выполнимы за полиномиальное время.

x_1	x_2	x_3	x_4	F
1	0	0	0	0
0	0	1	0	1
0	0	0	1	1
0	1	1	0	1
0	1	0	1	1
1	1	1	0	1
1	1	0	1	1
1	0	1	1	0

(a)

x_1	x_2	F_1
0	0	1
1	0	0
0	1	1
1	1	1

$\bowtie_{F_1=F_2}$

x_3	x_4	F_2
0	0	0
1	0	1
0	1	1
1	1	0

x	y	H
0	0	0
1	1	1
DC		DC

(b)

Рис. 5. (a) Пример декомпозиции. (b) Функция-комбинатор.

A	B	C	D	E
a	p	u	x	1
b	q	u	x	1
a	p	u	y	2
a	q	v	y	2
b	p	u	x	3
b	p	v	y	3

$$P = [\{\{A,B\},\{C\},\{D\}\}, \{\{A\},\{B,C\},\{D\}\}, \{\{A\},\{B\},\{C,D\}\}]$$

Рис. 6. Пример неразложимой таблицы с разложимыми относительно атрибута E подтаблицами.

Эта декомпозиция может быть реализована следующим образом (изложение для общего случая k-значной функции).

I. Исходную таблицу разделить на k подтаблиц таким образом, чтобы все строки в каждой подтаблице содержали одинаковое значение атрибута-результата. Затем этот атрибут следует исключить из дальнейших манипуляций.

II. Для каждой подтаблицы выполнить полную декартову декомпозицию (все компоненты далее неразложимы). Проекции на полученные разбиения не требуются. Итого имеется набор разбиений $P = [p_1, \ldots, p_k]$ для атрибутов таблицы $\mathcal{A}$, где одна часть p_k соответствует декартовой декомпозиции одной подтаблицы.

III. Использовать напрямую проекцию на разбиения атрибутов нельзя, так как все подтаблицы могут быть разложимы, в то время как вся таблица в целом не может. Пример этого представлен на рис. 6. Таблица будет разложима, если во всех подтаблицах существует "согласованное" разбиение атрибутов. Такое разбиение может быть построено посредством некоторой процедуры замыкания. Это шаг замыкания выполняется, когда части разбиений для разных подтаблиц имеют общий атрибут. Они объединяются до тех пор, пока данный процесс не стабилизируется. Декомпозиция существует, если замыкание не совпадет со всем набором атрибутов таблицы. Замыкание реализуется следующим образом:

1. Выбрать произвольное множество π любого разбиения P.
2. Инициализировать результирующее множество R посредством π.
   //когда алгоритм остановится, R содержит части "согласованного" разбиения атрибутов.
3. Инициализировать активное множество A посредством элементов π.
   //оно будет пополняться атрибутами, используемыми в замыкании.
4. Пока $A \neq \varnothing$ делать:
5. Взять произвольный атрибут a из A; удалить его из A.
6. Для каждой $p \in P$ делать:
7. Выбрать из p атрибутное множество π, содержащее a.
8. $A := A \cup (\pi \setminus R)$.
9. $R := R \cup \pi$.
10. Если $R = \mathcal{A}$, то таблица неразложима; иначе — разложима.
11. Если таблица разложима, то R и $\mathcal{A} \setminus R$ являются атрибутными множествами компонент декомпозиции.
12. Для получения результата выполнить проекции на эти атрибутные множества.

Процедуру замыкания можно выполнить за $O(k \cdot |\mathcal{A}|^2)$ временных шагов, где k — это количество частей разбиения, а $|\mathcal{A}|$ — общее количество атрибутов в таблице.

Заключение

Рассмотрены два алгоритма полиномиальной временной сложности для нахождения хунт и случай, когда переменные с низким уровнем влияния в исходной функции становятся несущественными в ее аппроксимации, которая определена через удаление некоторых входных значений. Дальнейшие направления исследований:

• Везде, где есть возможность, распространить эти методы на небулевы таблично определенные функции.

• Исследовать оптимальные стратегии и сложность решения систем булевых квадратных уравнений, возникающих в методе неопределенных коэффициентов, который используется для аппроксимации функций.

• Применить метод неопределенных коэффициентов для аппроксимации табличных функций в контексте декомпозиции на основе атрибута-результата.

• Как уже отмечалось во введении, конверсия между различными представлениями не всегда целесообразна. Следует разработать реализацию этих методов с использованием SQL и реляционных баз данных, так как в этом случае кажется естественным использовать табличноориентированную модель вычислений для достижения максимальной производительности.

• В [11] авторы не рассматривают таблицы истинности и основанных на них понятие влиятельности переменных в контексте IFVs. Соответствие этого метода аксиоматической системе IFV является предметом дальнейших исследований.

СПИСОК ЛИТЕРАТУРЫ

1. **Rong M., Gong D., Gao X.** Feature selection and its use in big data: challenges, methods, and trends // IEEE Access. 2019. Vol. 7. P. 19709–19725. https://doi.org/10.1109/ACCESS.2019.2894366
2. **O'Donnell R.** Analysis of Boolean functions. Cambridge: Cambridge Univ. Press, 2014. 417 p. ISBN: 9781139952507.
3. **Mossel E.** Probabilistic view of voting, paradoxes, and manipulation // Bull. Amer. Math. Soc. New Series. 2022. Vol. 59, no. 3. P. 297–330. https://doi.org/10.1090/bull/1751
4. **Birnbaum Z.W. and Esary J.D.** Modules of coherent binary systems // J. Soc. Indust. Appl. Math. 1965. Vol. 13, no. 2. P. 444–462. https://https://doi.org/10.1137/0113027
5. **Borowik G., Łuba T., Zydek D.** Features reduction using logic minimization techniques // Inter. J. Electron. Telecom. 2012. Vol. 58, no. 1. P. 71–76. https://doi.org/10.2478/v10177-012-0010-x
6. **Amgoud L., Doder D.** Compilation of logical arguments // Proc. Twenty-Eighth Inter. Joint Conf. Artificial Intelligence (IJCAI-19). 2019. P. 1502–1508. https://doi.org/10.24963/ijcai.2019/208
7. **Blum A. L., Langley P.** Selection of relevant features and examples in machine learning // Artific. Intellig. 1997. Vol. 97, no. 1-2. P. 245–271. https://doi.org/10.1016/S0004-3702(97)00063-5
8. Boolean models and methods in mathematics, computer science, and engineering / eds. Y. Crama, P.L. Hammer. Cambridge: Cambridge Univ. Press, 2011. 687 p. (Ser. Encyclopedia Math. Appl.; vol. 134). ISBN: 9780521847520.
9. **Tolo S., Andrews J.** fault tree analysis including component dependencies // IEEE Transactions on Reliability. 2024. Vol. 73, no. 1. P. 413–421. https://doi.org/10.1109/TR.2023.3264943
10. **Biswas A., Sarkar P.** Influence of a set of variables on a Boolean function // SIAM J. Discr. Math. 2023. Vol. 37, no. 3. P. 2148–2171. https://doi.org/10.1137/22M1503531
11. **Harder H., Jantsch S., Baier C., Dubslaff C.** A unifying formal approach to importance values in Boolean functions // Proc. Thirty-Second Inter. Joint Conf. Artificial Intelligence (IJCAI-23). 2023. P. 2728–2737. https://doi.org/10.24963/ijcai.2023/304
12. **Emelyanov P., Ponomaryov D.** On tractability of disjoint AND-decomposition of Boolean formulas // Perspectives of system informatics (PSI 2014) / eds. A. Voronkov, I. Virbitskaite. Cham : Springer, 2015. P. 92–101. (Ser. Lecture Notes in Comp. Sci.) https://doi.org/10.1007/978-3-662-46823-4_8
13. **Emelyanov P., Ponomaryov D.** Cartesian decomposition in data analysis // Proc. Siberian Symp. Data Science and Engineering (SSDSE 2017). 2017. P. 55–60. https://doi.org/10.1109/SSDSE.2017.8071964

14. **Emelyanov P.** On two kinds of dataset decomposition // Proc. 18^{th} Inter. Conf. Computational Science (ICCS 2018). Part II. / eds. Y. Shi et al. Cham : Springer, 2018. (Ser. Lecture Notes in Computer Science; vol. 10861). P. 171–183. https://doi.org/10.1007/978-3-319-93701-4_13
15. **Emelyanov P., Ponomaryov D.** The complexity of AND–decomposition of Boolean functions // Discrete Appl. Math. 2020. Vol. 280. P. 113–132. https://doi.org/10.1016/j.dam.2019.07.005
16. **Knuth D.E.** The art of computer programming. 12 ed. Boston, MA : Addison–Wesley, 2009. Vol. 4, Fascicle 1. 260 p. ISBN: 9780321580504.
17. **Емельянов П.Г.** Факторизация мультилинейных полиномов над GF(2) посредством редуцированного сортирующего полинома: v. 1.1, Python (2025). https://disk.yandex.ru/d/Urv4xOTCcPAqXQ (проверено 2025-11-05).
18. **Aziz R.A., Chu G., Muise C., Stuckey P.** #∃SAT: Projected Model Counting // Proc. 18^{th} Inter. Conf. Theory and Applications of Satisfiability Testing (SAT 2015). Cham : Springer, 2015. P. 121–137. (Lecture Notes in Computer Science; vol. 11077). https://doi.org/10.1007/978-3-319-24318-4_10
19. **Liu Zh., Chen X., Servedio R.A., Sheng Y., and Xie J.** Distribution-free junta testing // ACM Transactions on Algorithms. 2019. Vol. 15, no. 1. Article 1, p. 1–23. https://doi.org/10.1145/3264434

Поступила 11.05.2025
После доработки 20.06.2025
Принята к публикации 27.06.2025
Опубликована онлайн 30.06.2025

Емельянов Павел Геннадьевич
канд. физ.-мат. наук
старший науч. сотрудник
Институт систем информатики им. А. П. Ершова СО РАН
г. Новосибирск
e-mail: emelyanov@iis.nsk.su

REFERENCES

1. Rong M., Gong D., Gao X. Feature selection and its use in big data: challenges, methods, and trends. *IEEE Access*, 2019, vol. 7, pp. 19709–19725. https://doi.org/10.1109/ACCESS.2019.2894366
2. O'Donnell R. *Analysis of Boolean functions*. Cambridge, Cambridge Univ. Press, 2014, 417 p. ISBN: 9781139952507 .
3. Mossel E. Probabilistic view of voting, paradoxes, and manipulation. *Bull. Amer. Math. Soc.* (*New Series*). 2022, vol. 59, no. 3, pp. 297–330. https://doi.org/10.1090/bull/1751
4. Birnbaum Z.W., Esary J.D. Modules of coherent binary systems. *J. Soc. Indust. Appl. Math.*, 1965, vol. 13, no. 2, pp. 444–462. https://doi.org/10.1137/0113027
5. Borowik G., Łuba T., Zydek D. Features reduction using logic minimization techniques. *Inter. J. Electron. Telecom.*, 2012, vol. 58, no. 1, pp. 71–76. https://doi.org/10.2478/v10177-012-0010-x
6. Amgoud L., Doder D. Compilation of logical arguments. In: *Proc. Twenty-Eighth Inter. Joint Conf. Artificial Intelligence* (*IJCAI-19*), 2019, pp. 1502–1508.https://doi.org/10.24963/ijcai.2019/208
7. Blum A.L., Langley P. Selection of relevant features and examples in machine learning. *Artific. Intellig.*, 1997, vol. 97, no. 1–2, pp. 245–271. https://doi.org/10.1016/S0004-3702(97)00063-5
8. Crama Y., Hammer P.L. *Boolean models and methods in mathematics, computer science, and engineering*. Ser. Encyclopedia of mathematics and its applications, vol. 134. Cambridge, Cambridge Univ. Press, 2011, pp. 687. ISBN-13: 9780521847520 .
9. Tolo S., Andrews J. Fault tree analysis including component dependencies. *IEEE Transactions on Reliability*. 2024, vol. 73, no. 1, pp. 413–421.https://doi.org/10.1109/TR.2023.3264943
10. Biswas A., Sarkar P. Influence of a set of variables on a Boolean function. *SIAM J. Discr. Math.*, 2023, vol. 37, no. 3, pp. 2148–2171. https://doi.org/10.1137/22M1503531
11. Harder H., Jantsch S., Baier C., Dubslaff C. A unifying formal approach to importance values in Boolean functions. In: *Proc. Thirty-Second Inter. Joint Conf. Artificial Intelligence* (*IJCAI-23*), 2023, pp. 2728–2737. https://doi.org/10.24963/ijcai.2023/304
12. Emelyanov P., Ponomaryov D. On tractability of disjoint AND-decomposition of Boolean formulas. In: Voronkov A., Virbitskaite I. (eds.) *Perspectives of system informatics* (*PSI 2014*). Lecture notes in computer science. Berlin, Heidelberg, Springer, 2015, vol. 8974, pp. 92–101. https://doi.org/10.1007/978-3-662-46823-4_8

13. Emelyanov P., Ponomaryov D. Cartesian decomposition in data analysis. In: *Proc. Siberian Symp. Data Science and Engineering* (*SSDSE 2017*), 2017, pp. 55–60. https://doi.org/10.1109/SSDSE.2017.8071964
14. Emelyanov P. On two kinds of dataset decomposition. In: Shi Y., et al. *Proc. 18th Inter. Conf. Computational Science* (ICCS 2018). Lecture notes in computer science, part II. Cham, Springer, 2018, vol. 10861, pp. 171–183. https://doi.org/10.1007/978-3-319-93701-4_13
15. Emelyanov P., Ponomaryov D. The complexity of AND-decomposition of Boolean functions. *Discrete Appl. Math.*, 2020, vol. 280, pp. 113–132. https://doi.org/10.1016/j.dam.2019.07.005
16. Knuth D.E. The art of computer programming. Boston, AddisonWesley Prof., 2009, vol. 4, Fascicle 1, 260 p. ISBN-10: 0321580508 .
17. Emelyanov P. Factorization of multilinear polynomials over GF(2) by the reduced sorting polynomial: v. 1.1, Python (2025). `https://disk.yandex.ru/d/Urv4xOTCcPAqXQ` (accessed 2025-11-05).
18. Aziz R.A., Chu G., Muise C., Stuckey P. #∃SAT: Projected model counting. In: *Proc. 18th Inter. Conf. Theory and Applications of Satisfiability Testing* (*SAT 2015*). Lecture notes in computer science, Cham, Springer, 2015, vol. 11077, pp. 121–137. https://doi.org/10.1007/978-3-319-24318-4_10
19. Liu Zh., Chen X., Servedio R.A., Sheng Y., Xie J. Distribution-free Junta testing. *ACM Transactions on Algorithms*, 2019, vol. 15, no. 1, art. 1, p. 1–23. https://doi.org/10.1145/3264434

Received May 11, 2025
Revised June 20, 2025
Accepted June 27, 2025
Published online June 30, 2025

Pavel Gennadievich Emelyanov, Cand. Sci. (Phys.-Math.), A.P. Ershov Institute of Informatics Systems of the Siberia Branch of the Russian Academy of Sciences, Novosibirsk, 630090 Russia, e-mail: emelyanov@iis.nsk.su .

Cite this article as: P. G. Emelyanov. On junta problem for table-defined functions. *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, 2025, vol. 31, no. 3, pp. 105–120 .