ДЕКОМПОЗИЦИЯ БУЛЕВЫХ ФУНКЦИЙ И ПРИЛОЖЕНИЯ

П.Г. Емельянов
emelyanov@iis.nsk.su

ИСИ СО РАН

STEP • Новосибирск • 20 декабря 2024

Композиция и декомпозиция

Композиционное построение новых объектов — мощное математическое средство. Обращение композиции — декомпозиция — сложных объектов/систем является важнейшим методологическим приёмом. Выявление посредством декомпозиции компонент, составляющих систему, позволяет снизить сложность их анализа и преобразований, найти более компактные способы представления, эксплицировать внутреннюю структуру системы. Понимание сложности задачи декомпозиции систем демонстрирует алгоритмические перспективы их исследования. Разработка эффективных методов декомпозиции открывает новые возможности работы с такими объектами/системами.

Одна из самых фундаментальных видов композиции — декартово произведение. Некоторое обобщение декартова произведения, возникающее в рамках алгебры семейств множеств (Family Algebra), известно как произведение семейств множеств (П.С.М., Family Product).

Алгебра семейств множеств (Family Algebra)

Произведение Семейств Множеств (Family Product)

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

ПРИМЕР:

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

$$\begin{aligned} A \bowtie B \quad = \quad & \{\{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}, \{a,b,x\}, \{a,b,y\}, \\ & \{a,c,x\}, \{a,c,y\}, \{b,c,x\}, \{b,c,y\}, \{a,b,c,x\}, \\ & \{a,b,c,y\}\} \end{aligned}$$

ЗАДАЧА: Эффективное обращение Произведения Семейств Множеств.

Δ–Разложимость/Декомпозируемость

Задача была предложена Д.К. Пономарёвым в 2010 году в виде задачи конъюнктивного разложения позитивной ДНФ с непересекающимися по переменным компонентами разложения. В контексте исследования булевы функции и логические формулы взаимозаменяемые понятия, ЛФ рассматривается как один из способов представления БФ.

Определение

Булева функция F называется конъюнктивно разложимой относительно (возможно пустого) подмножества её переменных $\Delta \subseteq \mathtt{var}(F)$, если она эквивалентна конъюнкции $F_1 \wedge F_2$ некоторых функций F_1 и F_2:

1. $\mathtt{var}(F_1) \cup \mathtt{var}(F_2) = \mathtt{var}(F)$;
2. $\mathtt{var}(F_1) \cap \mathtt{var}(F_2) \subseteq \Delta$;
3. $\mathtt{var}(F_1) \setminus \Delta \neq \varnothing$, $\mathtt{var}(F_2) \setminus \Delta \neq \varnothing$.

$\mathtt{var}(F)$ называется носителем функции, F_1 и F_2 — компонентами декомпозиции или делителями или факторами.

∅–Декомпозиция как задача факторизации полиномов

Задача была сформулирована как факторизация мультилинейных полиномов над $\mathbb{F}_2$, представленных в виде списка мономов:

$$ab + ac + bc + abc + abx + aby + acx + acy + bcx + bcy + abcx + abcy$$

$$= (ab + ac + bc + abc)(x + y + 1)$$

В 2013 году был разработан детеминированный полиномиальный алгоритм, основанный на отыскании разбиения носителя исходной функции на носители компонент с дальнейшей проекцией для получения собственно компонент.

- Интерполяционные алгоритмы факторизации, они полиномиальные за счет рандомизации, которая основанна на Лемме Шварца–Зиппеля.
- Д.Ю. Григорьев, Зап. научн. сем. ЛОМИ, Т.137, 1984 — факторизация (плотных) полиномов, представленных в виде массивов коэффициентов.
- A. Shpilka, I. Volkovich, Proc. 37^{th} ICALP, LNCS 6198, 2010 — факторизация (разряженных) полиномомов над конечными полями, представленных в виде арифметических схем.

Факторизация мультилинейных полиномов над $\mathbb{F}_2$

Пусть полином F не имеет тривиальных делителей, т.е. переменных x, для которых $F = xF_x^1$, и его носитель содержит не менее 2 переменных.

Теорема (1)

Если F — факторизуем, т.е. $F = G \cdot H$, то при произвольном выборе $x \in \mathtt{var}(F)$ (очевидно, что $F_x^0 \neq 0 \ \wedge \ F_x' \neq 0$):

- $(F_x^0 \cdot F_x')_y' \neq 0$, *если x и y из одного фактора;*
- $(F_x^0 \cdot F_x')_y' = 0$, *если x и y из разных факторов.*

Если F нефакторизуем, для любой пары $x, y \in \mathtt{var}(F)$, $x \neq y$, верно $(F_x^0 \cdot F_x')_y' \neq 0$. Фактор переменной x неразложим, для кофактора дальнейшая факторизация возможна. Факторы отыскиваются проецированием мономов F на перменные кофакторов.

Полиномы представлены в виде списка/множества мономов. Если $F = \sum_{i=1}^{M} m_i$, то M – длина, $|F| = \sum_{i=1}^{M} |m_i| = O(nM)$ – размер полинома.

Алгоритм и оценка сложности ∅–разложения

Явный FD-Алгоритм

1. Взять произвольную переменную x из $\mathtt{var}(F)$.
2. Пусть $\Sigma_{same} := \{x\}, \Sigma_{other} := \varnothing$, и $F_{same} := 0, F_{other} := 0$.
3. Вычислить $S := F_x^0 \cdot F_x'$.
4. Для каждой переменной $y \in \mathtt{var}(F) \setminus \{x\}$:
   если $S_y' = 0$, то $\Sigma_{other} := \Sigma_{other} \cup \{y\}$
   иначе $\Sigma_{same} := \Sigma_{same} \cup \{y\}$.
5. Если $\Sigma_{other} = \varnothing$, тогда $F_{same} := F, F_{other} := 1$ и стоп.
6. Найти проекции $F_{same} = F{\downarrow_{\Sigma_{same}}}$ и $F_{other} = F{\downarrow_{\Sigma_{other}}}$ и стоп.

Следствие (Теоремы 1)

Временная сложность детерминированного алгоритма факторизации мультилинейных полиномов над $\mathbb{F}_2$ с n переменными в худшем равна $O(n^2|F|^2)$.

Есть параллельная MapReduce реализация на Python, основанная на построении сокращённого сортирующего полинома S (1–5% от размера $|F|^2$).

НОД-Алгоритм факторизации

Теорема (2)

Если $F = G \cdot H$, $x \in \mathtt{var}(G)$, *тогда*

$$H = \gcd(F_x^0, F_x').$$

Таким образом, можно последовательно найти все факторы исходного многочлена. Вычисление GCD для полиномов многих переменных известная классическая алгоритмическая задача в алгебре. Если базовая алгструктура не слишком богата (пример - $\mathbb{F}_2$), то при вычислениях часто получается ноль, что существенно влияет на производительность вычислений. Может предположить, что GCD, реализованный в Maple для конечных полей, — алгоритм LINZIP (de Kleine-Monagan-Wittkopf, Proc. ISSAC'2005), модифицированный для использования Henzel lifting. В версиях Maple до 17, при использовании `Gcd()mod 2` и `Factor()mod 2` для некоторых входных данных сообщалось об ошибке "linzip/not implemented yet". Вероятно, в последних версиях используются другие подходы.

Неявное вычисление $(F_x^0 \cdot F'_x)'_y$

Явное вычисление полинома $F_x^0 \cdot F'_x$ непрактично!

Пусть $A = F'_x$, $B = F_x^0$, вычисляя производную $A \cdot B$ по y $(y \neq x)$ имеем $D = (F_x^0)'_y$ и $C = F''_{xy}$. Итого

$$(F_x^0 \cdot F'_x)'_y = 0 \iff AD + BC = 0 \text{ или } AD = BC \quad (\textbf{PIT!}).$$

Оказывается, что проверку этого тождества можно свести к проверке таких же четырех тождеств меньшего рамера. Действуя так рекурсивно, в конце концов мы получим полиномы малых размеров или константы, для которых проверка тривиальна.

В **Неявном ФП–Алгоритме** Шаги 3–4 **Явного ФП–Алгоритма** модифицируются следующим образом:

Пусть $A = F'_x$, $B = F_x^0$.
Для каждой переменной $y \in \mathtt{var}(F) \setminus \{x\}$:
  Пусть $D = B'_y$, $C = A'_y$.
  Если IsEqual(A, D, B, C), тогда $\Sigma_{other} := \Sigma_{other} \cup \{y\}$,
  иначе $\Sigma_{same} := \Sigma_{same} \cup \{y\}$.

Рекурсивный шаг: тождеств больше, полиномы меньше

Пусть A, D, B, C проверяемые полиномы и для $Q \in \{A, D, B, C\}$ производную по z и означивание $z = 0$ как Q_1 and Q_2, соответственно.

$$AD = BC \quad \text{iff} \quad (A_1 z + A_2)(D_1 z + D_2) = (B_1 z + B_2)(C_1 z + C_2),$$

Тождество верно, если верна следующая формула:

$$\begin{array}{rcll} A_1 D_1 & = & B_1 C_1 & (1) \\ & \wedge & & \\ A_2 D_2 & = & B_2 C_2 & (2) \quad (*) \\ & \wedge & & \\ A_1 D_2 + A_2 D_1 & = & B_1 C_2 + B_2 C_1 & (3) \end{array}$$

После некоторых манипуляций, (3) можно свести к проверке $(A_1 B_2 + A_2 B_1)(A_1 C_2 + A_2 C_1) = 0$ или в виде формулы

$$A_1 B_2 = A_2 B_1 \ \vee \ A_1 C_2 = A_2 C_1.$$

Множители для преобразования Условия (3)

Преобразование $A_1 D_2 + A_2 D_1 = B_1 C_2 + B_2 C_1$ к форме $A_1 B_2 = A_2 B_1 \ \vee \ A_1 C_2 = A_2 C_1$ опирается на следующие рассуждения. Если хотя бы одно из тождеств (1), (2) не выполняется, то $AD \neq BC$ и проверка завершается. Иначе этот факт используется для проверки (3). Выбирая подходящим образом z, можно обеспечить $A_1, A_2 \neq 0$. Умножая (3) на $A_1 A_2$ имеем

$$A_1^2 A_2 D_2 + A_1 A_2^2 D_1 = A_1 A_2 B_1 C_2 + A_1 A_2 B_2 C_1.$$

Далее используя (1) и (2),

$$A_1^2 B_2 C_2 + A_1 A_2 B_2 C_1 = A_2^2 B_1 C_1 + A_1 A_2 B_1 C_2,$$
$$A_1 B_2 (A_1 C_2 + A_2 C_1) = A_2 B_1 (A_2 C_1 + A_1 C_2).$$

таким образом,

$$(A_1 B_2 + A_2 B_1)(A_1 C_2 + A_2 C_1) = 0.$$

Другие варианты множителей и получаемых условий

$$
\begin{array}{lcl}
A_1A_2 & \to & A_1C_2 = A_2C_1 \ \vee \ A_1B_2 = A_2B_1 \\
A_1B_2 & \to & A_1D_2 = B_2C_1 \ \vee \ A_1B_2 = A_2B_1 \\
A_1C_2 & \to & A_1D_2 = B_1C_2 \ \vee \ A_1C_2 = A_2C_1 \\
A_1D_2 & \to & A_1D_2 = B_2C_1 \ \vee \ A_1D_2 = B_1C_2 \\
A_2B_1 & \to & A_2D_1 = B_1C_2 \ \vee \ A_1B_2 = A_2B_1 \\
A_2C_1 & \to & A_2D_1 = B_2C_1 \ \vee \ A_1C_2 = A_2C_1 \\
A_2D_1 & \to & A_2D_1 = B_2C_1 \ \vee \ A_2D_1 = B_1C_2 \\
B_1B_2 & \to & B_1D_2 = B_2D_1 \ \vee \ A_1B_2 = A_2B_1 \\
B_1C_2 & \to & A_2D_1 = B_1C_2 \ \vee \ A_1D_2 = B_1C_2 \\
B_1D_2 & \to & B_1D_2 = B_2D_1 \ \vee \ A_1D_2 = B_1C_2 \\
B_2C_1 & \to & A_2D_1 = B_2C_1 \ \vee \ A_1D_2 = B_2C_1 \\
B_2D_1 & \to & B_1D_2 = B_2D_1 \ \vee \ A_2D_1 = B_2C_1 \\
C_1C_2 & \to & C_1D_2 = C_2D_1 \ \vee \ A_1C_2 = A_2C_1 \\
C_1D_2 & \to & C_1D_2 = C_2D_1 \ \vee \ A_1D_2 = B_2C_1 \\
C_2D_1 & \to & C_1D_2 = C_2D_1 \ \vee \ A_2D_1 = B_1C_2 \\
D_1D_2 & \to & C_1D_2 = C_2D_1 \ \vee \ B_1D_2 = B_2D_1
\end{array}
$$

Возможны различные стратегии выбора!

ЗАМЕЧАНИЕ: замена P'_x на $P^1_x = P'_x + P^0_x$

При проверке условия

$$\begin{array}{rcll} A_1 D_1 & = & B_1 C_1 & (1) \\ & \wedge & & \\ A_2 D_2 & = & B_2 C_2 & (2) \\ & \wedge & & \\ (A_1 B_2 = A_2 B_1 & \vee & A_1 C_2 = A_2 C_1) & (3) \end{array}$$

можно заменить вхождение всех производных Q_1 на выражения $Q_1 + Q_2$ (легко показать эквивалентность формул), что по сути является использованием позитивного кофактора вместо производной.

Это имеет большое практическое значение!

Фунуция IsEqual **Неявного ФП-Алгоритма**

Define IsEqual(A, D, B, C) returning Boolean.
    Проверка условий для равенства полиномов константе 0.
    Удалить все z : $z|A$ или $z|D$ или $z|B$ или $z|C$.
    Проверка условий для равенства полиномов константе 1.
    Если $\exists z$: $z|AD \not\equiv z|BC$, тогда вернуть FALSE.
    Если $\exists z$: $z \notin \mathrm{var}(AD) \cap \mathrm{var}(BC)$, тогда вернуть FALSE.
    Выбрать переменную z.
    Если not IsEqual$(A_z^0, D_z^0, B_z^0, C_z^0)$, тогда FALSE.
    Если not IsEqual(A_z', D_z', B_z', C_z'), тогда FALSE.
    Если IsEqual$(A_z', B_z^0, A_z^0, B_z')$, тогда TRUE.
    Вернуть IsEqual$(A_z', C_z^0, A_z^0, C_z')$.
End Definition.

— Сложность **Неявного ФП-Алгоритма** $O(n^2|F|^{2.22})$.
— Вычисление IsEqual параллельно для разных y, мемоизация.

Неявный ФП-Алгоритм, основанный на таймаутах

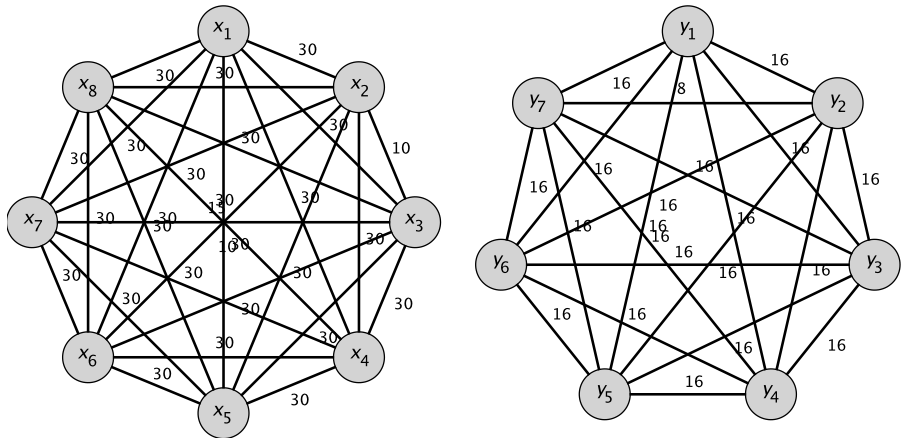
Define ImpliciteFDAlgorithm(F) returning pair of polynomials:
  Let $\Sigma_{same} := \varnothing, \Sigma_{other} := \varnothing$, $F_{same} := 0, F_{other} := 0$.
  Pick x from $\mathrm{var}(F)$.
  Let $A := F'_x$, $B := F^0_x$, $Timeout := 0$.
  For Each Variable $y \in \mathrm{var}(F) \setminus \{x\}$ Do
    Let $D := B'_y$, $C := A'_y$.
    Try assuming $Timeout$.
      If IsEqual(A, D, B, C) Then $\Sigma_{other} := \Sigma_{other} \cup \{y\}$.
                                Else $\Sigma_{same} := \Sigma_{same} \cup \{y\}$. End If.
      If $(Timeout = 0) \wedge (|\Sigma_{same}| > 0) \wedge (|\Sigma_{other}| > 0)$ Then
        Calculate $Timeout$.
      End If.
    Catch $Timeout$ elapsed.
      $\Sigma_{other} := \Sigma_{other} \cup \{y\}$.
    End Try.
  End For.
  Return $F{\downarrow_{\Sigma_{same} \cup \{x\}}}$ and $F{\downarrow_{\Sigma_{other}}}$.
End Define.

Как расчитывать таймаут

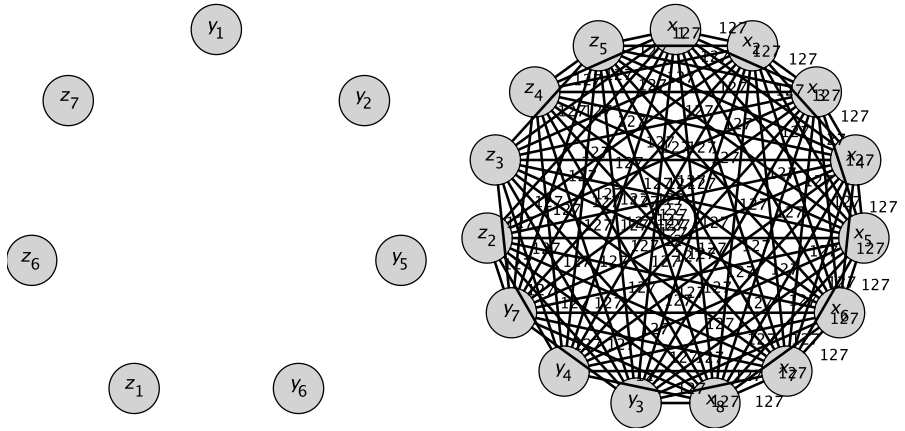
Для расчёта порогового значения было решено использовать информацию о времени классификации по крайней мере для одной переменной из каждого фактора. Как только классификация для обоих случаев выполнена впервые, порог вычисляется как среднее меньшего и большего значения. Однако можно получить несколько значений времени для переменных из фактора, в котором находится исходное значение, прежде чем появится информация о переменной из противоположного фактора. Это может дать более полезное пороговое значение. Порог вида “максимальное+3 стандартных отклонения” давал для полиномов с 10^4 мономами 86% успешной классификации, и время в 2-8 раз меньше времени классификации с использованием исчерпывающего поиска. Увеличение размера многочленов приводит к более точной идентификации переменных, количество сбоев значительно уменьшается. Для полиномов с 10^5 мономами не было зафиксировано ни одного сбоя. Неравенство Чебышева оценивает вероятность ошибочной классификации переменных для “3σ-правила” около 0.11 (“10σ-правило” даёт 0.01) и она не зависит от типа распределения.

Арифметический алгоритм факторизации

- Алгоритм считывает мономы последовательно. Весь полином загружать в память не надо, это позволяет сэкономить на внутреннем представлении полинома.
- Ёмкостная сложность алгоритма $O(n^2)$, временная — $O(n^2M + n^3)$, типичная ситуация $n \ll M$.
- На картинках изображено разбиение переменных полинома:



по факторам для разложимого полинома;



для нефакторизуемого полинома.

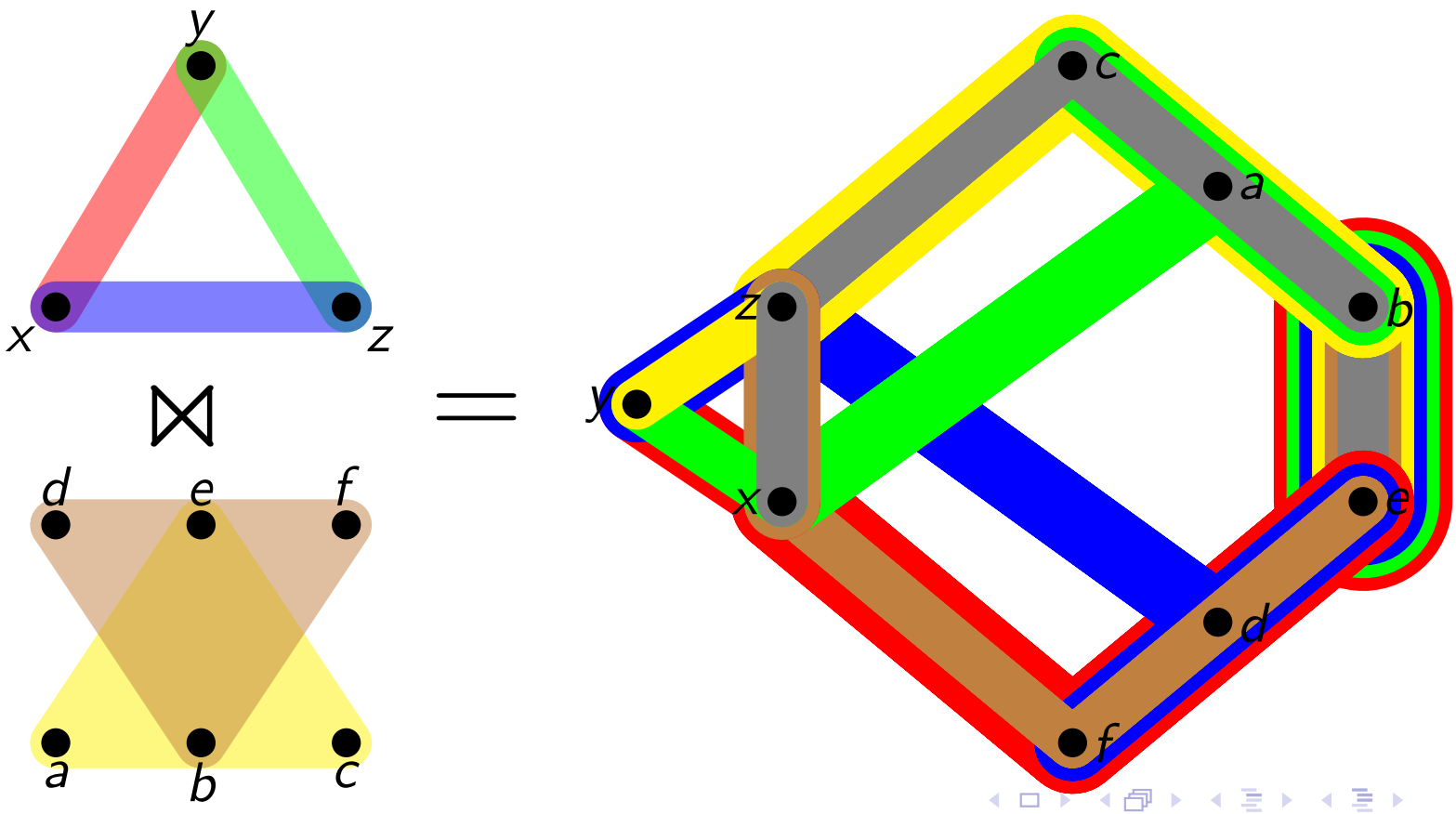
ПРИЛОЖЕНИЯ

- Декомпозиция гиперграфов, комбинаторная оптимизация, теория игр.
- Структурная теория надёжности, теория расписаний.
- Декомпозиция булевых функций.
- Декомпозиция таблиц (в реляционных СУБД и СППР).
- Декомпозиция при онтологическом моделировании.

Декомпозиция гиперграфов

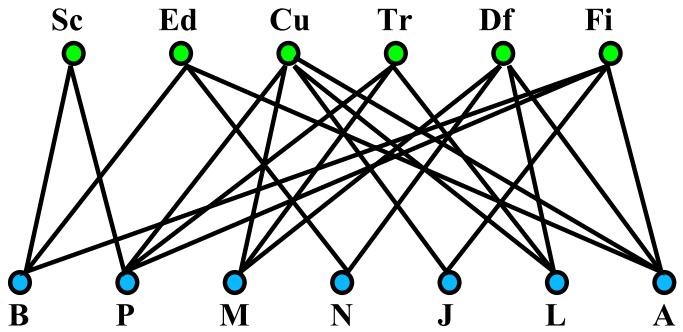
$$A=\{\{x,y\},\{y,z\},\{x,z\}\} \quad B=\{\{a,b,c,e\},\{b,d,e,f\}\}=\{\{b,e\}\bowtie\{\{a,c\},\{d,f\}\}$$

$$A\bowtie B = \{\{a,b,c,e,x,y\},\{a,b,c,e,y,z\},\{a,b,c,e,x,z\},\{b,d,e,f,x,y\},\{b,d,e,f,y,z\},\{b,d,e,f,x,z\}\}$$



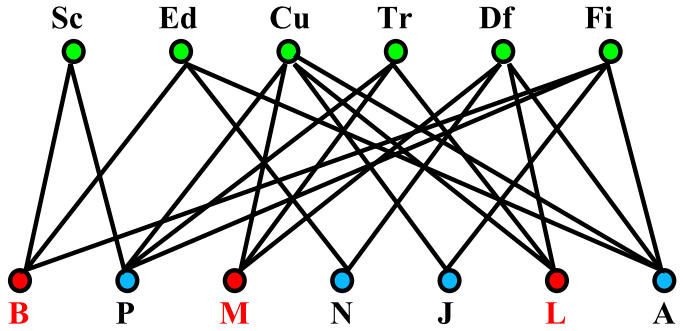
Комитеты–представители (Шепли, начало 60-х годов)

	Вася	Петя	Маша	Коля	Толя	Гоша	Анна
Финансы	1	1	0	0	1	0	1
Оборона	0	0	1	1	0	1	1
Транспорт	0	1	1	0	0	1	0
Культура	0	1	1	0	1	1	1
Образование	1	0	0	1	0	0	1
Наука	1	1	0	0	0	0	0



$G = \langle \{\text{персоны}\}, \{\text{комитеты}\}, \{\text{вхождение}\} \rangle$

Вершины Вася и Маша (или Гоша) “доминируют” над вершинами–комитетами, то есть в каждый комитет входит кто–то из них.

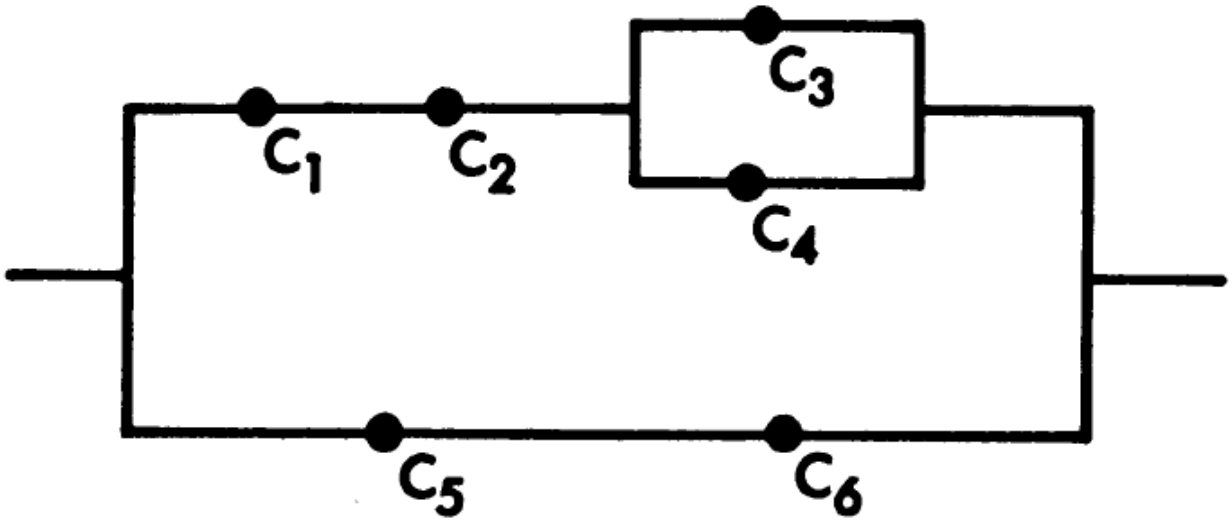


$$\text{ВПТА}+\text{МКГА}+\text{ПМГ}+\text{ПМТГА}+\text{ВКА}+\text{ВП} = (\text{В}+\text{МГ})\times(\text{П}+\text{КА}+\text{ПТА})$$

$$\{\{\text{Вася}\},\{\text{Маша},\text{Гоша}\}\} \bowtie \{\{\text{Петя}\},\{\text{Коля},\text{Анна}\},\{\text{Петя},\text{Толя},\text{Анна}\}\}$$

Структурная часть теории надежности (Бирнбаум и Эсэри, J.SIAM, 1965)

MODULES OF COHERENT BINARY SYSTEMS



$$\varphi = x_1x_2x_3 \vee x_1x_2x_4 \vee x_5x_6 = [x_1x_2(x_3 \vee x_4)] \vee x_5x_6$$

Таблицы — это гомогенные мультилинейные полиномы

Декартово произведение в реляционных СУБД реализуется в SQL-оператором

SELECT * FROM table1 CROSS JOIN table2;

A	B
x	y
x	z

$\bowtie$

C	D	E
x	u	p
y	u	q
z	v	r

$=$

A	B	C	D	E
x	y	x	u	p
x	y	y	u	q
x	y	z	v	r
x	z	x	u	p
x	z	y	u	q
x	z	z	v	r

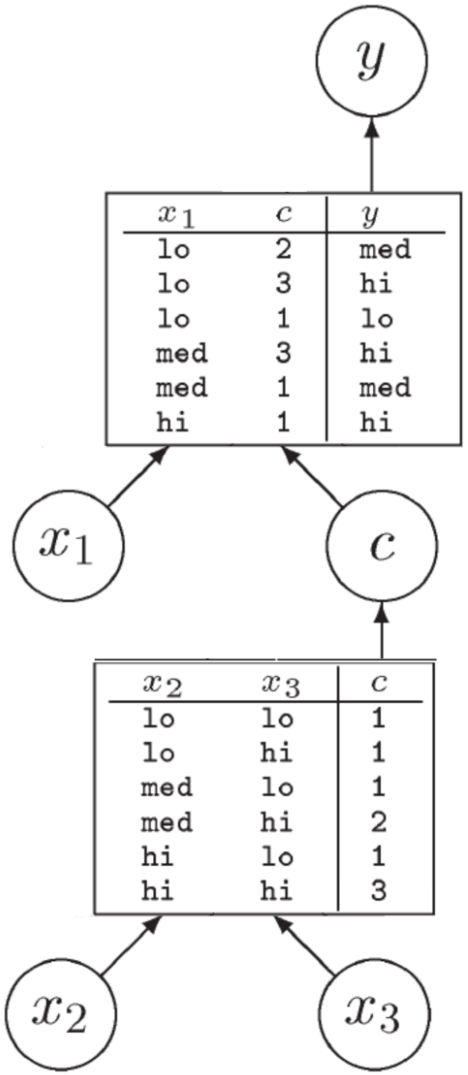
$$z_B qux_A y_C + y_B qux_A y_C + y_B rvx_A z_C + z_B rvx_A z_C + y_B pux_A x_C + z_B pux_A x_C =$$
$$x_A \cdot (y_B + z_B) \cdot (quy_C + rvz_C + pux_C)$$

Важна реализация на SQL (есть на TransactSQL).

— Отыскание функциональных зависимостей в таблицах;
— “Разрезание” таблиц;
— Обращения оператора JOIN ON.

Декомпозиция таблиц принятия решений

x_1	x_2	x_3	y
lo	lo	lo	lo
lo	lo	hi	lo
lo	med	lo	lo
lo	med	hi	med
lo	hi	lo	lo
lo	hi	hi	hi
med	med	lo	med
med	hi	lo	med
med	hi	hi	hi
hi	lo	lo	hi
hi	hi	lo	hi



Оценка вероятности ∅–декомпозируемости

Утверждение

Необходимое условие для декомпозируемости F

$$\forall x \in \mathtt{var}(F) \quad : \quad НОД(|F'_x|, |F|) > 1.$$

Утверждение

Если случайный полином F над $n > 2$ переменными имеет M мономов, то

$$\mathbb{P}[F \text{ } \varnothing\text{–недекомпоз.}] > 1 - \left(1 - \frac{\phi(M)}{M}\right)^n > 1 - \left(1 - \frac{1}{e^\gamma \ln\ln M + \frac{3}{\ln\ln M}}\right)^n,$$

где ϕ и γ — функция Эйлера и его же константа соответственно.

Обобщения задачи факторизации

- декомпозиция с непустым множеством разделяемых переменных Δ:

  — в 2016 построен FP-полиномиальный алгоритм для мультилинейных полиномов над $\mathbb{F}_2$;

  — Д.К. Пономорёв в 2019 построил полиномиальный алгоритм для позитивных ДНФ;

- декомпозиция с дефектом, содержащим предписанные переменные.

Система уравнений для Δ–декомпозиции, $\Delta \neq \varnothing$

Пусть F — булевый полином над переменными $\mathtt{var}(F) = \{x_1, \ldots, x_n\}$ и $\Delta \subseteq \mathtt{var}(F)$, $|\Delta| = k$, — множество разделяемых переменных.

Определение

$$F|_\delta = (F|_{x=0, x\in\Delta\setminus\delta})|'_{y,y\in\delta}$$

Каждый моном $\prod_{x_i\in\delta} x_i$, $\delta \subseteq \Delta$, включая $\varnothing$, имеет коэффициенты–полиномы A_δ и B_δ (все над $\mathtt{var}(F) \setminus \Delta$) в соответствующих компонентах би–декомпозиции. Эти коэффициенты удовлетворяют следующей системе 2^k квадратичных уравнений:

для всех подмножеств $\delta \subseteq \Delta$ $$\sum_{\substack{\forall\alpha,\beta\subseteq\delta \\ \alpha\cup\beta=\delta}} A_\alpha B_\beta = F|_\delta\,.$$

Случай $|\Delta| = 2$

$$\begin{aligned} F &= xyF''_{xy} + xF'^{0}_{xy} + yF^{0\prime}_{xy} + F^{00}_{xy} \\ &= (A_{x,y}xy + A_x x + A_y y + A_\varnothing)(B_{x,y}xy + B_x x + B_y y + B_\varnothing) \end{aligned}$$

$$\begin{cases} A_\varnothing B_\varnothing = F^{00}_{xy} \\ A_x B_x + A_x B_\varnothing + A_\varnothing B_x = F'^{0}_{xy} \\ A_y B_y + A_y B_\varnothing + A_\varnothing B_y = F^{0\prime}_{xy} \\ A_{x,y}B_{x,y} + A_{x,y}(B_x + B_y + B_\varnothing) + B_{x,y}(A_x + A_y + A_\varnothing) + A_x B_y + A_y B_x = F''_{xy}, \end{cases}$$

Решая уравнения последовательно от первого и далее и подставляя полученные на предыдущих шагах варианты A_δ и B_δ на каждом шаге начиная со второго будем иметь квадратные уравнения вида $XY + aX + bY = c$, которые, аналогично диофантовым уравнениям, снова сводятся к факторизации некоторых полиномов.

Так как мы заинтересованы в Δ–декомпозиции, необходимо обеспечить, чтобы множества переменных $\cup_\delta \mathtt{var}(A_\delta)$ и $\cup_\delta \mathtt{var}(B_\delta)$ не пересекались.

Примеры Δ–декомпозиции

- Пример, когда функция не имеет $\varnothing$–декомпозиции, но возможна $\{x\}$–декомпозиция:

$$x+ux+vx+uvx+ust+vst+stx+uvstx=(x+u+v+xuv)(x+st).$$

- Данная функция не имеет $\varnothing$–декомпозиции и декомпозиций с единственной разделяемой переменной, но возможна $\{x,y\}$–декомпозиция

$$ytuv+stuv+suvx+yst+ysx+ytx+stx+yt+sx=(xs+yt+st)(x+y+uv)$$

- $\Delta=\mathtt{var}(F)$

$$\begin{aligned} uvx+uvy+uxy+vxy &= (u+v+x+y)\\ &\quad (uv+ux+uy+vx+vy+xy)\end{aligned}$$

Декомпозиция с предписанным Δ–дефектом

Задача отыскания дефекта D, препятствующего $\varnothing$-декомпозиции, но для которого известно множество переменных Δ:

$$F(X,Y) = F_1(X)F_2(Y) + D(\Delta), \quad \Delta \subset X \cup Y, \quad X \cap Y = \varnothing,$$

Пример:

$$x_1x_2x_3y_1y_2 + x_1x_2x_3y_2y_3 + x_1x_2x_3 + x_1y_1y_2 + x_2y_1y_2 + x_2y_2y_3 + x_3y_1y_2 + x_3y_2y_3 + y_2y_3 + x_2 + x_3 =$$

$$(x_1x_2x_3 + x_1 + x_2 + x_3)(y_1y_2 + y_2y_3 + 1) + (x_1y_2y_3 + y_2y_3 + x_1), \ \Delta = \{x_1, y_2, y_3\}.$$

Отметим, мономы $x_1y_2y_3$ и x_1 не входят в исходный полином.

Приложения: декомпозиция булфункций в задаче оптимизации микроэлектронных схем (например, для FPGA), задачи анализа данных и извлечения знаний (построение схем реляционных СУБД, нормализация отношений), СППР (анализ и оптимизация таблиц принятия решений).

Декомпозиция с предписанным Δ–дефектом: подходы

Теорема 1 даёт три подхода к решению задачи. Обозначим $H = F_x^0 + D$.

- Метод неопределённых коэффициентов (для D) позволяет решать задачу для полиномов с 20-30 переменными (сводится к решению больших систем линейных уравнений над $\mathbb{F}_2$).
- Задачу можно свести к решению алгебраического однородного линейного уравнения над полиномами из $\mathbb{F}_2[X, Y]$:

$$\frac{F'^0_{xy}}{G} H^1_y + \frac{F'^1_{xy}}{G} H^0_y = 0 \quad \text{где} \quad G = \text{НОД}(F'^0_{xy}, F'^1_{xy}).$$

  Отыскание решения этого уравнения (сизигии), возможно методом, использующим Базисы Грёбнера для $\mathbb{F}_2[x_1, \ldots, x_n]$ и его фактор-кольца $\mathbb{F}_2[x_1, \ldots, x_n]\langle x_1^2 + x_1, \ldots, x_n^2 + x_n\rangle$ булевых полиномов.
- С помощью аналогичных преобразований можно вывести линейное дифференциальное уравнение:

$$F'_x H'_y + F''_{xy} H = 0$$

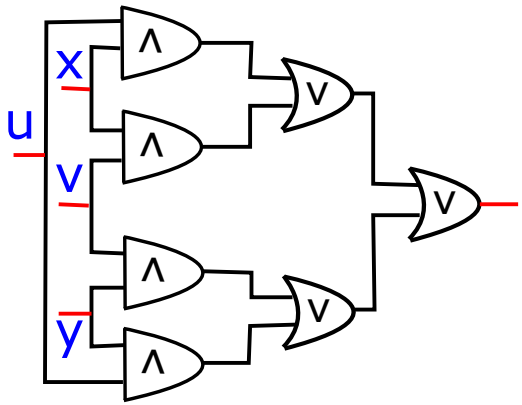
Декомпозиция булевых функций

Неявный ФП-алгоритм факторизации мультилинейных полиномов над $\mathbb{F}_2$ напрямую дает алгоритм конъюнктивной декомопзиции на компоненты с непересекающимися наборами переменных для булевых функций, представленных в виде АНФ (полиномов Жегалкина) и позитивных/негативных ДНФ/КНФ.

Неявный ФП-алгоритм можно рассматривать как обобщенный шаблон алгоритма, который может быть настроен на разные представления булевых функций: СДНФ/хар.мн-ва единиц, СКНФ/хар.мн-ва нулей, табличные представления (в т.ч. частичные), ROBDD, OKFD, AIG.

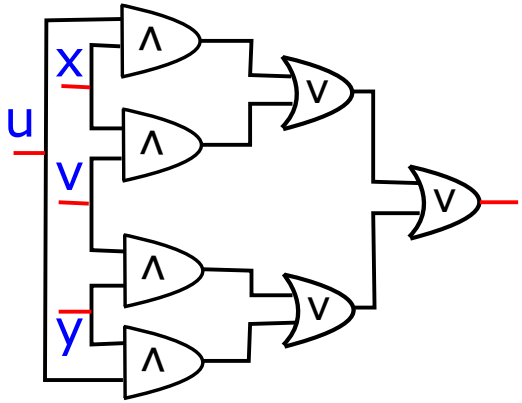
Исследовательская тема: неявный ФП-алгоритм можно модифицировать для одновременного построения AND- и OR-разложениий булевых функций.

Оптимизация логических схем (комбинационная часть)

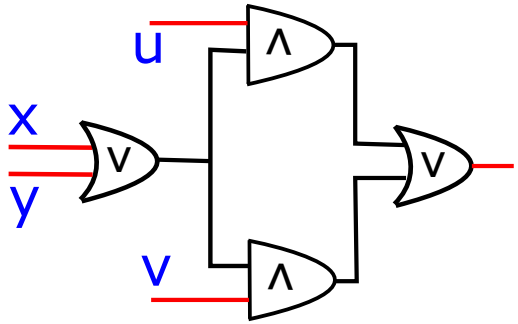


$xu \vee xv \vee yu \vee yv$

Оптимизация логических схем (комбинационная часть)

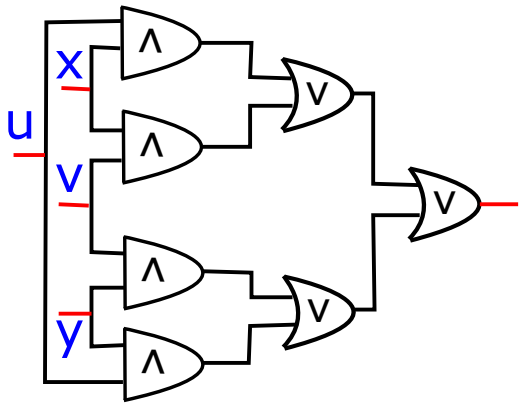


$xu \vee xv \vee yu \vee yv$

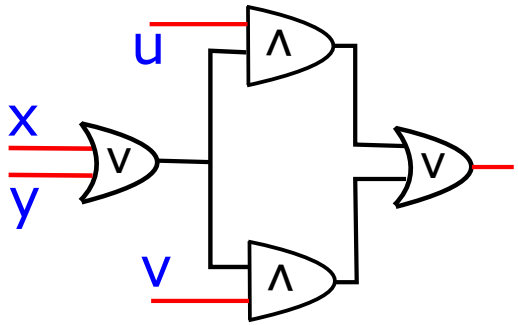


$(x \vee y)u \vee (x \vee y)v$

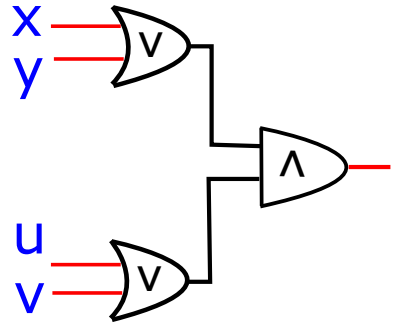
Оптимизация логических схем (комбинационная часть)



$xu \vee xv \vee yu \vee yv$



$(x \vee y)u \vee (x \vee y)v$



$(x \vee y)(u \vee v)$

ВАЖНО: при программировании вычислительных систем на основе архитектуры ПЛИС (FPGA) существенным шагом является отыскание декомпозиции булевых функций специального вида.

Многоуровневая декомпозиция булевых функций

$$F = absu \vee absv \vee absw \vee abtu \vee abtv \vee abtw \vee abxy \vee abxz \vee$$
$$acsu \vee acsv \vee acsw \vee actu \vee actv \vee actw \vee acxy \vee acxz \vee$$
$$desu \vee desv \vee desw \vee detu \vee detv \vee detw \vee dexy \vee dexz$$

"AND–First" стратегия дает:

$$F = (a(b \vee c) \vee de)((s \vee t)(u \vee v \vee w) \vee x(y \vee z)).$$

"OR–First" стратегия оптимизатора ESPRESSO дает:

$$F = x(a(c \vee b) \vee de)(z \vee y) \ \vee \ (a(c \vee b) \vee de)(t \vee s)(w \vee v \vee u)$$

- F имеет 96 символов. Переменные встречаются от 3 до 16 раз.
- "AND–First" дает 13 символов и глубину 4 (read–once форма).
- ESPRESSO дает 18 символов и глубину 5 (не read–once).

Многоуровневая декомпозиция булевых функций

$$F = absu \vee absv \vee absw \vee abtu \vee abtv \vee abtw \vee abxy \vee abxz \vee$$
$$acsu \vee acsv \vee acsw \vee actu \vee actv \vee actw \vee acxy \vee acxz \vee$$
$$desu \vee desv \vee desw \vee detu \vee detv \vee detw \vee dexy \vee dexz$$

"AND–First" стратегия дает:

$$F = (a(b \vee c) \vee de)((s \vee t)(u \vee v \vee w) \vee x(y \vee z)).$$

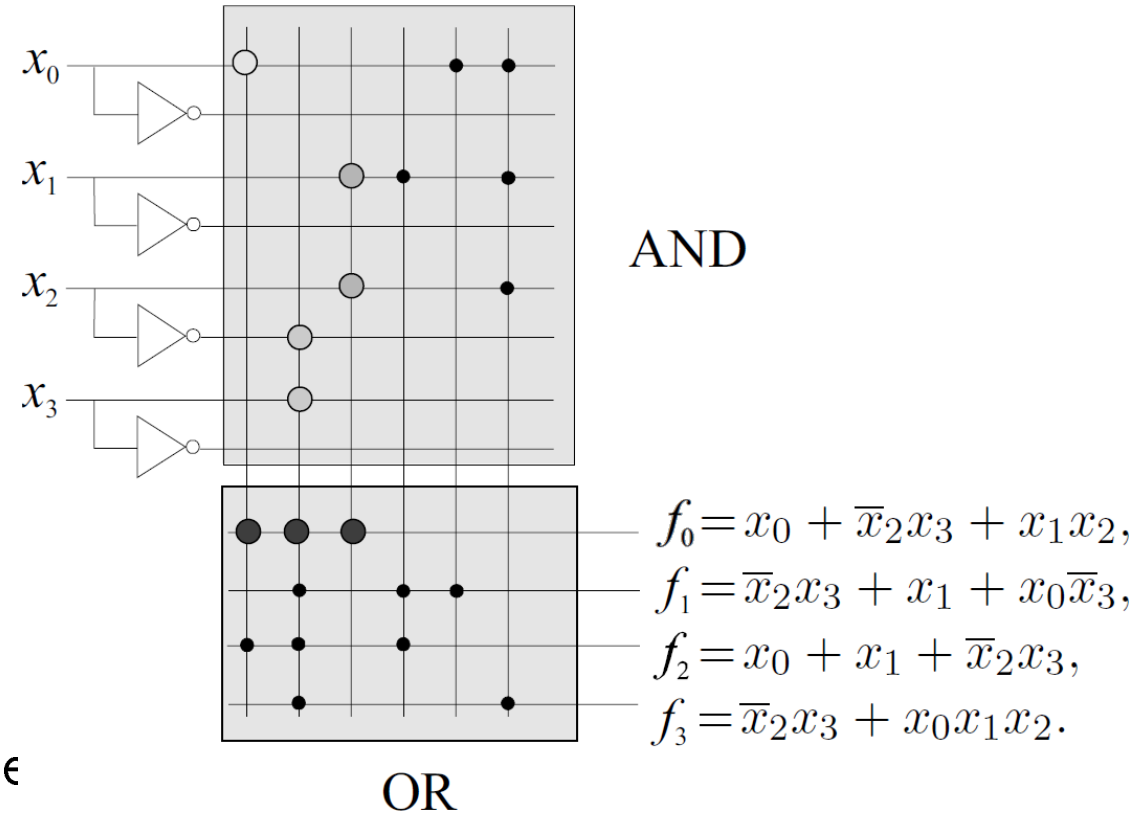
"OR–First" стратегия оптимизатора ESPRESSO дает:

$$F = x(a(c \vee b) \vee de)(z \vee y) \bigvee (a(c \vee b) \vee de)(t \vee s)(w \vee v \vee u)$$

- F имеет 96 символов. Переменные встречаются от 3 до 16 раз.
- "AND–First" дает 13 символов и глубину 4 (read–once форма).
- ESPRESSO дает 18 символов и глубину 5 (не read–once).

Декомпозиция и минимизация булевых функций

- Оптимизация булевых функций, представленных в виде ТИ или ДНФ/КНФ (SoP/PoS) методами минимизации (сокращение числа конъюнктов/дизъюнктов и их длины). Важно в рамках двух-уровневого дизайна.
- Восходят к методам Карно и Квайна. В общем случае NP-сложная задача.
- Если предварительно декомпозировать булеву функцию в более компактные компоненты, то можно получить более компактное представление для всей функции.



Пример: декомпозиция и минимизация булевых функций

- Функция 1: “Snake–in–the–Box” толщиной 7 в 17-мерном кубе (102 вершин змеи). СДНФ: 102 единиц, 1’734 символов.
- Функция 2: Hamming Code в 15-мерном кубе с кодовым расстоянием 7 (32 кодовых слова). СДНФ: 32 единиц, 480 символов.
- **Большая функция**: конъюнкция Функции 1 и Функции 2. СДНФ: 3’264 единиц, 104’448 символов.
- Минимум ДНФ Большой функции: 801 термов, 24’832 символа, время ESPRESSO–минимизации 36 мин.
- Минимум ДНФ Snake-17-7: 51 термов, 816 символов, время минимизации 1 сек. ДНФ Hamming Code не минимизируется.
- Декомпозиция Большой функция занимает 12 минут. Представление Большой функции в форме конъюнкции СДНФ Hamming-15-7 и минимальной ДНФ Snake-17-7 имеет 83 термов + 1 конъюнкция, 1296 символов. Оно меньше полученной напрямую минимальной ДНФ в 10 раз по термам и в 20 раз по символам.

Декомпозиция булевых таблиц

- Произвольные булевы таблицы (все строки различны) — это описания б.ф. с помощью их нулей и/или единиц (характеристического множества нулей/единиц; СДНФ/СКНФ):

  — декомпозиция относительно столбца-аргумента.

- Таблицы истинности (полные или частичные) б.ф.:

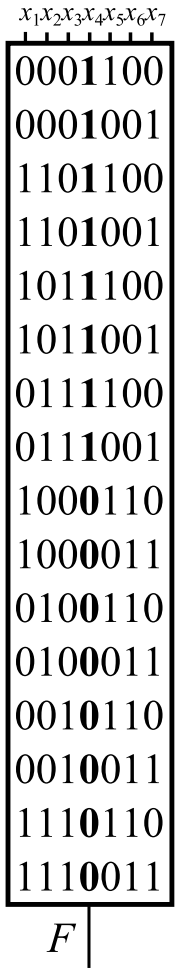
  — декартова декомпозиция: компонента–декомпозиции, которая не содержит результат, описывает несущественные переменные б.ф.

  — декомпозиция относительно столбца-аргумента или столбца результата.

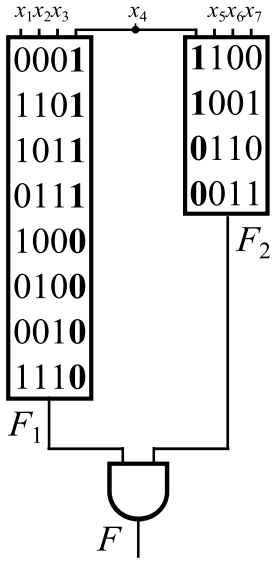
Представляют интерес как методы с использованием таблично-ориентированных операций (например, SQL), так и методы, основанные на манипуляциях с полиномами в различных представлениях.

Декомпозиция таблиц с явным атрибутом-коннектором

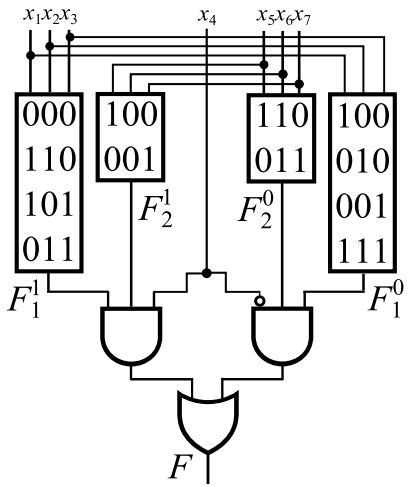
Таблица единиц F



1)



2)



3)

Более того, здесь $F_1^1 = \overline{F_1^0}$

$$F(U,V,x) = \bar{x}F(U,V,0) \vee xF(U,V,1) = \bar{x}F_1^0(U)F_2^0(V) \vee xF_1^1(U)F_2^1(V).$$

Декомпозиция частичных ТИ относительно результата

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

$=$

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

$$F(x_1, x_2, x_3, x_4) = H(F_1(x_1, x_2), F_2(x_3, x_4))$$

H могут быть разные: $\wedge, \vee$, левая или правая проекция. Выбор последних означает, что в качестве F можно взять либо F_1 либо F_2. Существуют функции, где от половины и более переменных могут не использоваться при вычислениях на значимых аргументах:

$$F(X, Y) = H(F_1(X), F_2(Y)) = F_1(X) = F_2(Y).$$

Новое понятие бесполезных переменных
(некоторое подмножество носителя относительно дополнения)!

Тестирование для схем, представленных в ROBDD

Набор тестов	Схема	Тип	Выход	Кол. вход.	Размер BDD	Время (сек.)	Уск.12	Изм.1	Изм.2
EPFL	mem_ctrl_part0	and	po067	113	669	0,0055	6,38	0,1011	0,7549
EPFL	mem_ctrl_part0	or	po223	113	909	0,0094	7,94	0,5868	0,3322
ISCAS	c5315_part1	and	po083	53	326	0,2496	8,67	10^{-6}	0,5552
ISCAS	s13207_part0	or	n388	58	428	0,0006	10,14	0,2819	0,7290
ISCAS	s15850.1_part0	and	n2833_1	67	2173	5757,95	2,05	0,1667	0,4680
ISCAS	s15850.1_part0	and	n488	145	2204	0,0758	8,39	0,0032	0,2001
ISCAS	s5378_part0	or	po37	59	1712	0,0037	8,5	0,2253	0,4796
IWLS2005	ac97_ctrl_part0	or	n9928_1	58	2142210	30,5784	9,14	0,0508	0,9981
IWLS93	i2	or	po0	201	6468	20,6707	3,82	0,0004	0,2191
IWLS93	pair_part0	or	po081	51	3271	1293,8	4,24	0,0008	0,0997
IWLS93	pair_part1	or	po019	53	7103	1246,58	6,27	0,0003	0,0715
LGS91	i10_part0	or	po063	53	3237	0,3468	10,48	0,4992	0,9608
LGS91	i2	or	po0	201	335	103,119	4,59	0,0001	1,0030
LGS91	pair_part0	or	po004	51	9600	210,595	4,91	0,0004	0,0452
Other	oc_wb_dma	or	n2705	54	186	0,0026	4,95	0,0107	1,0054
Other	oc_wb_dma	or	n2800	54	225	0,0027	5,97	0,0107	1,0044
Other	sudoku_check_part2	and	n3454	729	730	12311,4	11,25	0,1111	1,0014

Q&A

СПАСИБО ЗА ВНИМАНИЕ!

Доказательство Теоремы 1

Пусть F — неконстантный мультилинейный полином над $\mathbb{F}_2$. Если F факторизуем над $\mathbb{F}_2$, он обладает свойствами, на основании которых будет построена полиномиальная процедура для разбиения множества переменных F на непересекающиеся множества Σ_1 и Σ_2, такие, что если F факторизуем, то он имеет множители–полиномы над ними. Далее нужно провести проекцию F на множества Σ_1 и Σ_2.
Отметим, что $F'_x = F^1_x + F^0_x$ и всякий мультилинейный полином может быть записан в виде разложением Рида $F = F'_x \cdot x + F^0_x = (F^1_x + F^0_x)x + F^0_x$. Кроме того, если некоторая переменная x содержится во всех мономах F, тогда либо F не факторизуем (в случае $F = x$), либо тривиально факторизуем, то есть $F = xF^1_x = xF'_x$.
Далее будем полагать, что у F нет тривиальных факторов.
Пусть F — мультилинейный полином над множеством переменных $\{x, x_1, \ldots, x_n\}$.
Если F факторизуем, то он может быть представлен в виде $F = (x \cdot Q + R) \cdot H$, где

- полиномы Q, R и H не содержат x;
- Q и R не имеют общих переменных с H;
- R является непустым полиномом (так как F не факторизуем тривиально);
- $x \cdot Q + R$ — нефакторизуемый полином.

Тогда имеем $F^0_x = R \cdot H$ и $F'_x = Q \cdot H$. Очевидно, что F^0_x и F'_x могут быть вычислены за полиномиальное время.

Доказательство Теоремы 1 — продолжение

Пусть y — переменная из F, отличная от x. Рассмотрим следующее выражение:

$$(Q\cdot R\cdot H^2)'_y = Q'_y\cdot R\cdot H^2 + Q\cdot(R\cdot H^2)'_y = Q'_y\cdot R\cdot H^2 + R'_y\cdot Q\cdot H^2 + 2H'_y\cdot Q\cdot R\cdot H$$

так как в $\mathbb{F}_2$ для всех z верно $2z = z + z = 0$, то

$$= H^2\cdot(Q'_y\cdot R + R'_y\cdot Q) = H^2\cdot(Q\cdot R)'_y.$$

Т.е., если $y \in \mathtt{var(H)}$, то $(Q\cdot R)'_y = 0$ и $(Q\cdot R\cdot H^2)'_y = 0$. Докажем обратное, предположив, что $y \notin \mathtt{var(H)}$ и покажем, что производная не равна нулю. Так как y не принадлежит H, полиномы Q и R в общем случае имеют вид

$$Q = Ay + B, \qquad R = Cy + D,$$

для некоторых полиномов A, B, C, D не содержащих y. Тогда $Q\cdot R = ACy^2 + (AD + BC)y + BD$ и, следовательно, $(Q\cdot R)'_y = AD + BC$. Т.о., необходимо показать, что $AD + BC \neq 0$. Предположим обратное:

$$AD + BC = 0 \quad \text{или, что то же самое,} \quad AD = BC.$$

Доказательство Теоремы 1 — продолжение

Пусть $B = f_1 \cdot \ldots \cdot f_m$ и $C = g_1 \cdot \ldots \cdot g_n$ являются (однозначными) факторизациями B и C в нефакторизуемые полиномы. Тогда $AD = f_1 \cdot \ldots \cdot f_m \cdot g_1 \cdot \ldots \cdot g_n$ и можно полагать, что $A = f_1 \cdot \ldots \cdot f_k \cdot g_1 \cdot \ldots \cdot g_l$ для некоторого $0 \leqslant k \leqslant m$ и $0 \leqslant l \leqslant n$ (если $k = l = 0$, полагается $A = 1$). Полиномы B, C, D могут быть представлены в такой же форме. Обозначим через (U, V) наибольший общий делитель полиномов U и V. Тогда $A = (A, B) \cdot (A, C)$, $B = (A, B) \cdot (D, B)$, аналогично для C и D, и мы имеем

$$\begin{aligned} x \cdot Q + R &= x \cdot (Ay + B) + (Cy + D) \\ &= x \cdot ((A, B)(A, C)y + (A, B)(D, B)) + \\ &\quad ((A, C)(D, C)y + (D, B)(D, C)) = ((A, B)x + (D, C))((A, C)y + (D, B)), \end{aligned}$$

что приводит к противоречию, так как по предположению $x \cdot Q + R$ не факторизуем.

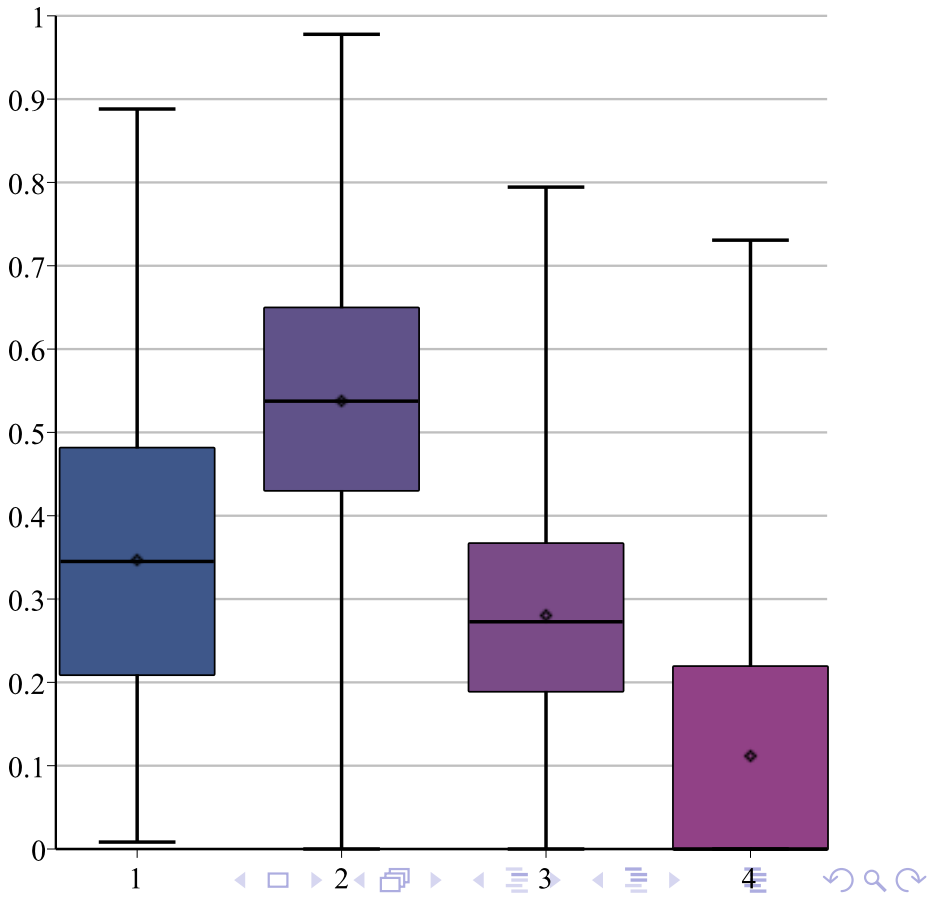
Доказательство Теоремы 1 — продолжение

Таким образом, мы получили процедуру разбиения множества переменных F на непересекающиеся множества Σ_1 и Σ_2 следующим образом. Выбрав некоторую начальную переменную x, полагаем $\Sigma_1 = \{x\}$, $\Sigma_2 = \varnothing$ и вычисляем полином $Q \cdot R \cdot H^2$ (равный $F'_x \cdot F^0_x$). Затем для каждой переменной y из F (отличной от x) вычисляется производная $(Q \cdot R \cdot H^2)'_y$. Если она равна нулю, добавляем y в Σ_2, в противном случае — в Σ_1. Если в конце $\Sigma_2 = \varnothing$, то полином F не факторизуем. Если n — размер исходного полинома, рассматриваемого как последовательность символов, то приведенный алгоритм требует $O(n^2)$ шагов для вычисления полинома $G = Q \cdot R \cdot H^2$ и проверки, равна ли производная G'_y нулю для переменной y (вычисление производной и проверка равенства делается за линейное число шагов для полинома над $\mathbb{F}_2$, заданного как символьная последовательность). Так как это нужно проверить для каждой переменной $y \neq x$, то алгоритм вычисляет разбиение переменных за $O(n^3)$ шагов. *QED*

Экпериментальня/статистическая оценка сложности для случайных ROBDD

- Замечание: выявление неразложимости функции происходит быстрее, чем разложимости. Оценка сложности только для разложимых случаев — это оценка в “худшем” для всех функций.
- 40 декомпозируемых BDDs над 100 переменных, где каждая компонента имеет 50 переменных, узлов – от 2.5 до 5 миллионов. Время – от 10 до 40 минут.
- Для диапазона значений p характ. уравнения от мат.ож. до мат.ож.+ст.откл. сложность алгоритма

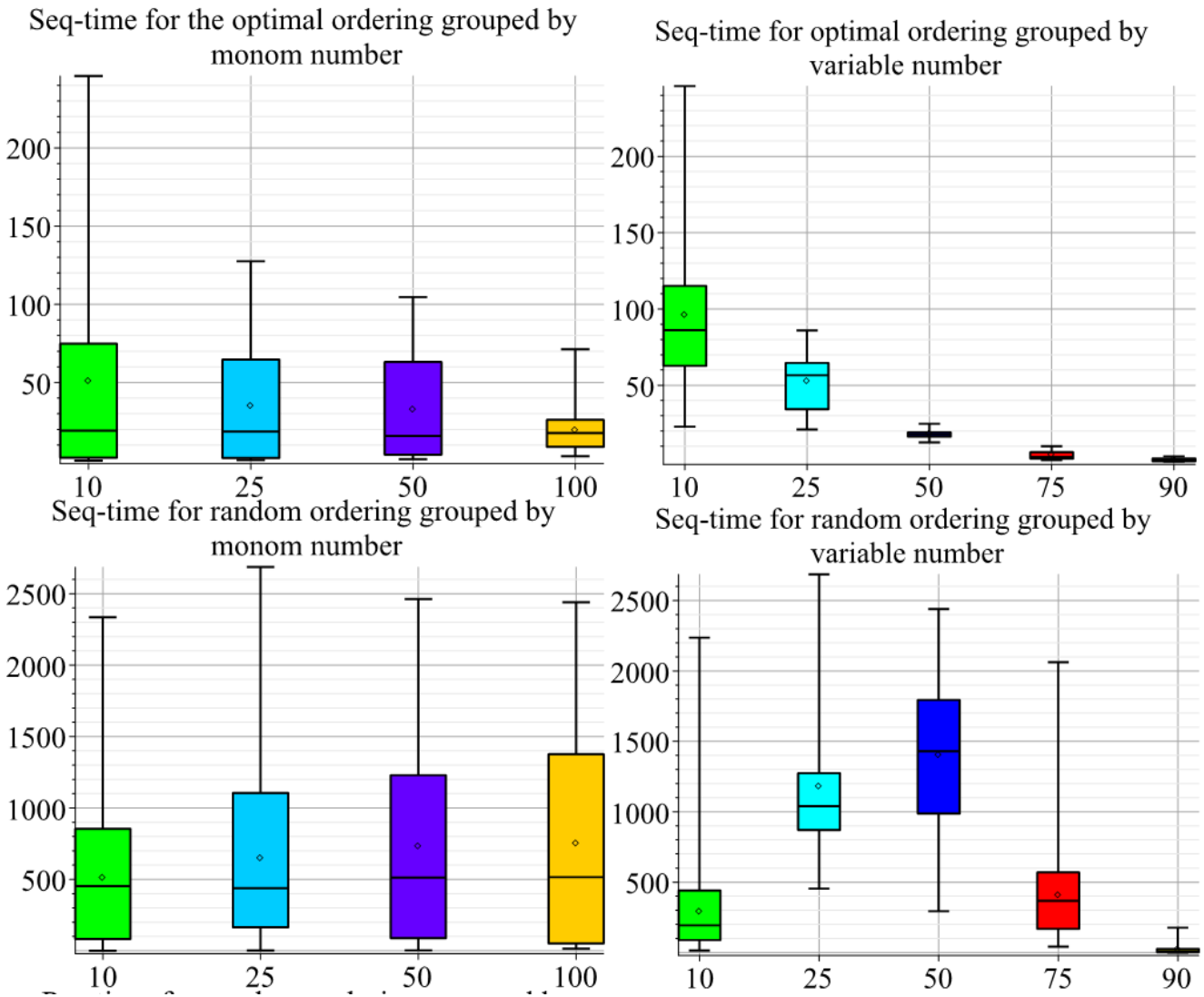
$$T = O(nF^2).$$



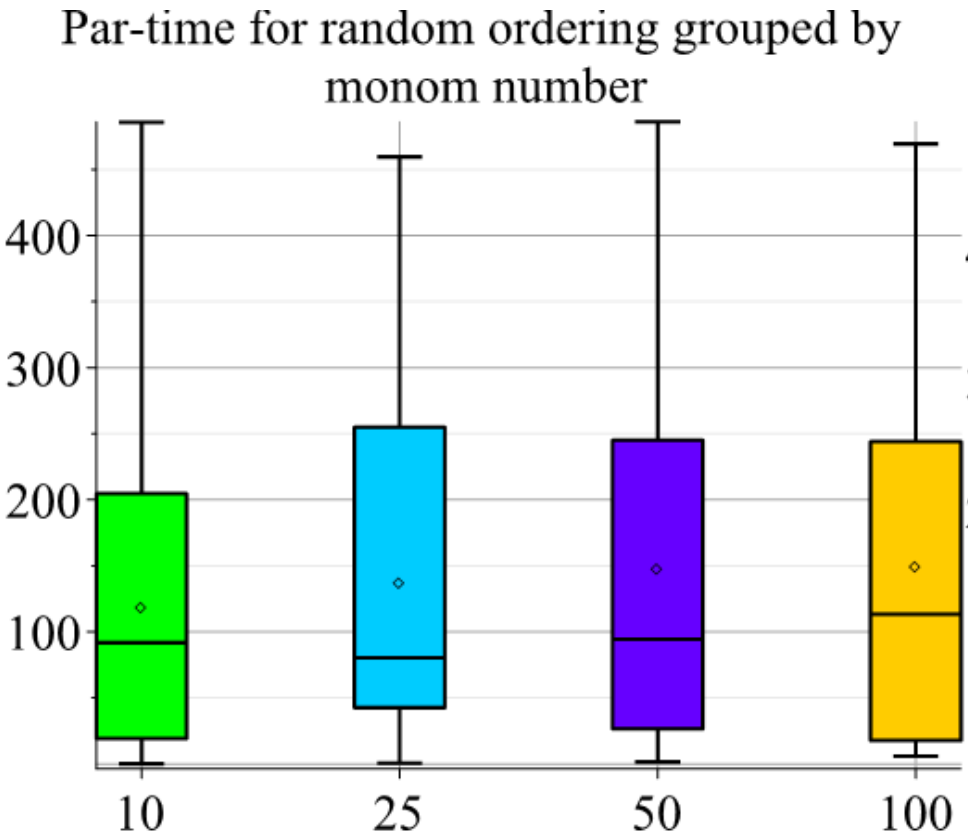
Тестирование факторизации полиномов в форме ZDD

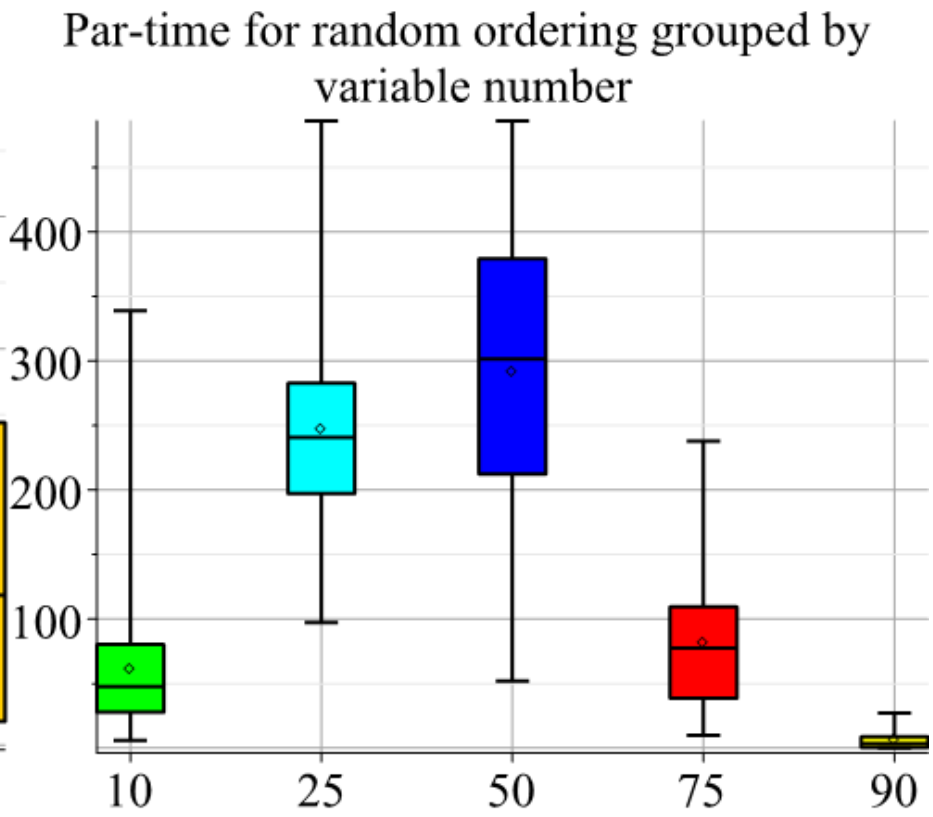
Для реализации была использована библиотека CUDD 3.0, с помощью которой полиномы представлялись в виде ZDD. Были сгенерированы два набора случайных полиномов, являющихся произведением двух компонент с непересекающимися носителями, со следующими характеристиками. Все содержат 100 переменных, которые разбиваются по компонентам 10 на 90, 25 на 75, 50 на 50, 75 на 25, 90 на 10. Один набор — это полиномы, имеющие 10^4 мономов (по компонентам 10 на 1000, 25 на 400, 50 на 200, 100 на 100), и второй — 10^5 мономов (100 на 1000, 200 на 400, 316 на 316). Результаты тестирования первого набора (в вариантах оптимального и случайного упорядочивания переменных и однопоточного и 8-поточного исполнения) представлены на бокс-диаграммах ниже. В представленных диаграммах ускорение означает отношение времени исполнения в 8-поточном режиме ко времени исполнения в однопоточном режиме. В общем и целом, результаты тестирования подтверждают гипотезы о сложностных характеристиках алгоритма.

Измерения вычислений в последовательном режиме



Измерения вычислений в параллельном режиме





Измерения ускорений