Санкт-Петербургский государственный политехнический университет
Физико-механический факультет
Кафедра прикладная математика

Диссертация допущена к защите
Зав. кафедрой
_____ **В.Е. Клавдиев**

«____»_____

ДИССЕРТАЦИЯ
на соискание степени МАГИСТРА

Тема: ***Схемы вычисления полиномов и их приложения***

Направление: 010500 – Прикладная математика и информатика
Магистерская программа: 510209 –
Математическое и программное обеспечение компьютерных систем

Выполнил студент гр. 6057/2 М.П. Кожевников

Руководитель, к.ф.-м.н., с.н.с. Н.Н. Васильев

Консультанты:
по охране труда, к.т.н., доц. В.В. Монашков

Санкт-Петербург
2010

Содержание

Введение

Предметом данной работы являются различные схемы вычисления полиномов и их приложения к

- эффективному *вычислению полиномов с предобработкой*
- редукция[1] систем полиномов относительно полиномиального базиса

Целью работы является сравнительный анализ различных подходов к построению таких схем в контексте указанных приложений, а также их эффективная реализация.

Мы будем в основном рассматривать разреженные полиномы многих переменных.

Синтез вычисляющих программ

Вычисление полиномов с предобработкой обычно относят к *синтезу вычисляющих программ*, несмотря на то, что, зачастую, программой результат предобработки можно назвать лишь в довольно общем смысле – как правило, это некоторая форма описания последовательности действий, которые необходимо произвести над значениями переменных с тем, чтобы получить значение полинома от этих переменных. Разумеется, такая последовательность может быть записана в форме программы на интерпретируемом языке или, чаще, исполняемого модуля компилируемого языка, откуда и происходит термин.

Синтез вычисляющих программ как подход к вычислению значений функций сформировался достаточно давно – в конце пятидесятых годов прошлого века. Его задача заключается в построении эффективной процедуры вычисления значения заданной функции, что было вдвойне актуально с учетом мощности ЭВМ того времени. В большинстве случаев вычисление значения искомой функции так или иначе сводилось к вычислению значения некоторого полинома – например, через разложение в ряд или интерполяцию.

Несмотря на интенсивный рост производительности вычислительных машин, задача актуальна и сейчас – например, для некоторых видов симуляции или численного решения дифференциальных уравнений, а также в тех случаях, где вычисления производятся с высокой точностью и, как следствие, существенно возрастает стоимость каждой арифметической операции.

Для полиномов одной переменной данная проблема изучена достаточно хорошо. В тех случаях, когда полином не очень сложен, но скорость его вычисления критична, обычно применяются методы *с обработкой коэффициентов* [12]. Мы, однако, будем рассматривать преимущественно разреженные полиномы многих переменных. В этом случае применение методов с обработкой коэффициентов не вполне оправдано, так как

[1] В данном контексте можно было бы использовать термин «нормализация», однако он является не вполне корректным, так как относительно произвольного полиномиального базиса полином может не иметь нормальной формы

- обработка коэффициентов может сократить количество операций, необходимых для вычисления полинома, не более чем вдвое [21]

- умножение на коэффициент часто существенно дешевле умножения мономов – в том числе, это верно для задачи редукции, которую мы рассматриваем.

Исходя из этого, мы рассматриваем схемы вычисления полиномов многих переменных без обработки коэффициентов. Данная задача рассматривалась и ранее, например, в [18, 19]. Нас же интересует сравнительный анализ различных подходов к построению схем в терминах их сложности и практической эффективности.

Существует много различных вариантов постановки задачи вычисления значений полиномов. В нашем случае задан набор разреженных полиномов многих переменных, чьи значения требуется эффективно вычислять для произвольных значений переменных.

В такой постановке данная задача возникает, как правило, в достаточно узкоспециализированных областях. Как следствие, программные пакеты, предназначенные для ее решения, создаются с учетом специфики предметной области – это может быть, например, дополнительная информация о структуре полинома или ограничения на значения коэффициентов. Что касается систем компьютерной алгебры общего назначения, нам неизвестно ни одной, обладающей специальной функциональностью для решения таких задач.

Указанные факторы делают затрудняют сравнение нашего подхода с существующими аналогами, поэтому для данного приложения мы ограничимся сравнением эффективности различных схем в рамках рассматриваемого подхода.

Редукция

Схема вычисления полинома может применяться не только для вычисления его значений, но и для выполнения произвольной операции подстановки (см. 1.2.1). В частности, схему можно использовать для более эффективной редукции полинома или системы полиномов относительно полиномиального базиса (см. 3). Последняя же задача представляет интерес как сама по себе, так и в качестве компонента для алгоритма вычисления базиса Грёбнера [24], чья эффективность во много определяется эффективностью алгоритма редукции [23]. Совместная редукция нескольких полиномов может также быть полезна в ряде алгоритмов [20] вычисления базисов Грёбнера.

В данной работе мы реализуем метод редукции полиномов, основанный на схемах вычисления полиномов (РСВП). Данный метод был описан в [16] и впервые реализован в [23]. Он позволяет достичь высокой производительности и обладает высокой гибкостью за счет возможности выбрать подходящую схему в зависимости от специфики задачи. Мы рассмотрим, как зависит его эффективность от используемой схемы, а также опишем некоторые аспекты его реализации.

1 Постановка задачи

Мы будем рассматривать преимущественно полиномы с целыми или рациональными коэффициентами, либо с коэффициентами в некотором конечном поле. Бо̀льшая часть утверждений верна для любого типа коэффициентов. В тех случаях, когда это не так, тип коэффициентов будет оговорен отдельно.

1.1 Основные определения

Введем основные понятия, которые потребуются нам в данной работе.

Определение 1. *Мономом называется произведение неотрицательных целых степеней заданных переменных* $\prod_i x_i^{n_i}$. *Множество всевозможных мономов на заданном наборе переменных образует моноид и обозначается* $\mathbb{M}$. *Сумма показателей степеней* $\sum_i n_i$ *называется полной степенью монома.*

Определение 2. *Линейное пространство над полем*[2] $\mathbb{K}$ *с базисом* $\mathbb{M}$ *образует кольцо полиномов* $\mathbb{K}[x]$. *Членом полинома называется произведение коэффициента и соответствующего монома. Множество базисных элементов, имеющих в полиноме* $p \in \mathbb{K}[x]$ *ненулевые коэффициенты, называется его носителем* $s(p)$.

Определение 3. *Линейное отношение порядка* $>$ *на множестве* $\mathbb{M}$ *называется мономиальным упорядочением. Если, кроме того,* $(\forall u \in \mathbb{M}\, u > 1) \wedge (\forall u, v, w \in \mathbb{M}\, u > v \Rightarrow wu > wv)$, *то упорядочение называется допустимым.*

1.2 Схемы вычисления полиномов

Схемы вычисления полиномов можно рассматривать как алгоритм вычисления значения полинома, записанный в терминах умножения и сложения полиномов, а также умножения полинома на число (коэффициент). В некоторых случаях возведение в степень также рассматривают а качестве элементарной операции.

Такие схемы удобно изображать в виде диаграмм (рис. 1), где корневые узлы отвечают переменным и единице, а промежуточные соответствуют операциям. Входящие дуги при этом указывают источники операндов.

Для большинства полиномов схема вычисления, очевидно, неуникальна. Например, обе схемы на диаграмме 2 соответствуют полиному $xy + xz$. Может возникнуть впечатление, что каждая схема однозначно соответствует некоторой расстановке скобок – на диаграмме 1 это $xy + xz$ слева и $x(y + z)$ справа. Это не совсем верно, так как, например, расстановка скобок не указывает, являются ли два вхождения одного и того же монома ссылками на

[2]Вообще говоря, можно рассматривать и полиномы над кольцами, но распространение на этот случай методов, рассматриваемых в данной работе, требует применения определенных специальных приемов

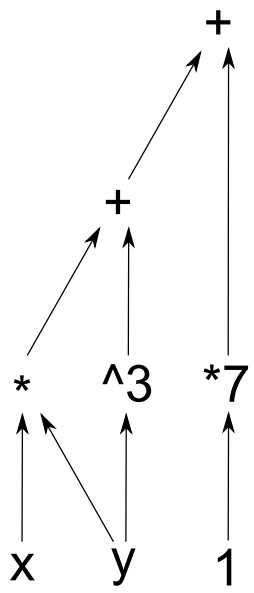

Рис. 1: Схема для полинома $y^3 + xy + 7$

один узел схемы или на два различных узла. Например, диаграмма 3 изображает две схемы вычисления полиномов, соответствующие (в смысле расстановки скобок) выражению $xy(xy + 1)$. Другими словами, выразительность скобочных структур ограничена деревьями, тогда как схема представляет собой граф общего вида.

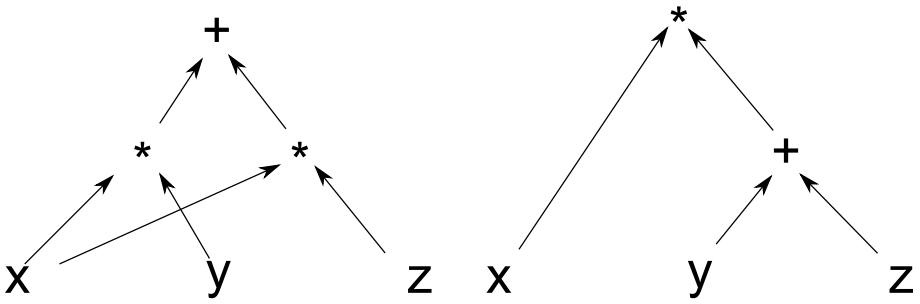

Рис. 2: Две схемы для полинома $xy + xz$

1.2.1 Схема как алгоритм подстановки

В более широком смысле, схема вычисления полинома является схемой подстановки значений в этот полином, причем в качестве значений могут выступать элементы любого кольца, допускающие умножение на коэффициенты рассматриваемого полинома. В частности, в зависимости от типа коэффициентов это могут быть комплексные, вещественные, рациональные или целые числа, либо целые числа по заданному модулю. Кроме того, очевидно, это могут быть полиномы того же типа, что и рассматриваемый.

Таким образом, метки «+» и «*» в узлах схемы соответствуют умножению и сложению в соответствующем кольце. Следует обратить внимание, что умножение на коэффициент не обязательно совпадает с умножением в кольце. В качестве примера можно рассмотреть

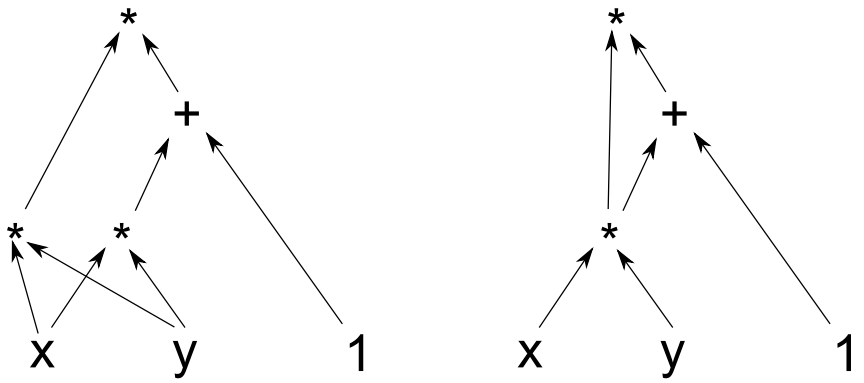

Рис. 3: Две схемы для полинома $xy(xy+1)$

подстановку векторов с покомпонентным умножением и сложением в полином с целыми коэффициентами.

1.2.2 Критерии качества

Нас интересуют схемы, которые позволяют решать определенные задачи более эффективно. Следовательно, естественным критерием качества, как правило, является скорость решения соответствующей задачи. Однако, использование такого критерия на практике часто оказывается неудобным (в силу отсутствия эффективного способа вычисления), поэтому используются различные упрощенные критерии. Для задачи вычисления в качестве упрощенного критерия может выступать общее количество операций, возможно, с весами.

Выбор подходящего критерия качества представляет собой чрезвычайно важный фактор при разработке алгоритма построения схем вычисления полиномов. Критерии, используемые для других приложений, мы обсудим позже.

1.2.3 Полиномы одной переменной

Для случая полиномов одной переменной существует два основных способа построения схем. Первый – очевидный – способ состоит в последовательном вычислении всех мономов и сложении результатов, а второй известен как «схема Горнера». Следует обратить внимание, что семантика слова «схема» в данном названии отлична от используемой в нашем контексте, поэтому здесь и далее мы будем говорить о *методе* Горнера.

Для многих задач метод Горнера дает схему, близкую к оптимальной, однако в ряде случаев существуют и более эффективные подходы [12].

1.2.4 Полиномы многих переменных

В случае полиномов многих переменных построение схемы, оптимальной в смысле заданного критерия качества, является нетривиальной проблемой. Для полиномов высоких степеней построение такой схемы может быть слишком сложным с вычислительной точки зрения.

В большинстве случаев можно эффективно построить схемы, близкие к оптимальной. Это оправдано также и тем, что используемый приближенный критерий качества может не вполне соответствовать истинному критерию, и, как следствие, оптимальная схема может не быть наиболее эффективной.

1.2.5 Схемы для нескольких полиномов

Понятие схемы естественным образом обобщается на случай совместного вычисления нескольких полиномов. Это актуально преимущественно в задачах вычисления значений полиномов, однако совместная редукция набора многочленов относительно заданного базиса также может иметь свои приложения – например, такая задача возникает при реализации алгоритма F4 вычисления базиса Грёбнера [20], хотя она и формулируется в матричной форме.

1.3 Вычисление значений полиномов

В тех случаях, когда речь идет о вычислении значений полинома одной переменной, часто применяются *схемы с обработкой коэффициентов*. Описание таких схем можно найти, например, в [11, 22, 27]. Эта тема достаточно хорошо изучена как в теоретическом, так и в практическом плане. Нас же интересуют преимущественно разреженные полиномы многих переменных, что требует применения совершенно иных методов.

Рассматриваемые подходы обладают высокой гибкостью – их можно применять практически независимо от типа коэффициентов полинома. Кроме того, они легко обобщаются на случай совместного вычисления нескольких полиномов, что может быть весьма полезно в некоторых задачах.

Мы указали ранее на тот факт, что сравнение с другими аналогичными системами представляет определенную сложность, в связи с чем мы будем рассматривать лишь сравнительную эффективность различных схем.

1.4 Редукция полиномов

Алгоритм РСВП, который мы рассмотрим подробнее в разделе 3, можно рассматривать как операцию замены переменных в рассматриваемом полиноме, выполняемую в факторалгебре. Это означает, что результатом умножения мономов является *нормальная форма* их произведения. Не зная значений аргументов, сложность данной операции нельзя определить с какой-либо фиксированной точностью – в практических задачах от операции к операции она может варьироваться на несколько порядков 3.2.1.

Кроме того, в задаче вычисления значений схема используется многократно с различными значениями аргументов, тогда как необходимость редукции одного и того же многочлена относительно многих различных наборов полиномов возникает крайне редко.

Как следствие, в первом случае оправдано построение наилучшей возможной схемы, а во втором необходимо соблюдать баланс между качеством схемы и временем ее построения.

Поэтому, хотя обе рассматриваемых нами задачи могут быть выражены в терминах схем вычисления полиномов, природа их весьма различна, и, как следствие, для них лучше приспособлены различные методы построения схем.

2 Методы построения схем

2.1 Структура схемы

Существуют методы, которые строят упрощенные, в некотором смысле, СВП. В корневых узлах такой схемы стоят не первые степени переменных, а некоторые нетривиальные мономы, которые, в свою очередь, могут быть вычислены различными способами. Соответственно, для них можно построить отдельную схему, называемую схемой *совместного вычисления мономов* (ССВМ).

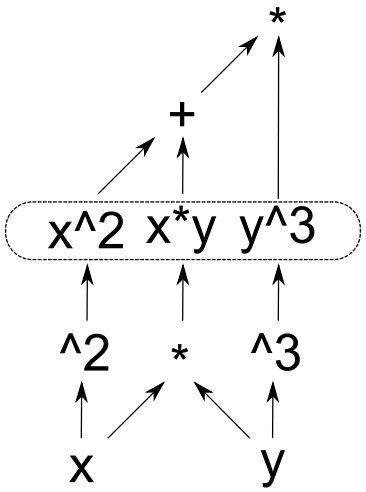

Рис. 4: Схема для $x^2y^3 + xy^4$ как композиция двух схем. Пунктиром выделен переход.

Можно рассматривать это как разбиение схемы на две части (рис. 4). Формально они отличаются лишь тем, что нижняя не содержит операций сложения. Методы построения ССВМ мы рассмотрим в разделе 2.6.

В данной работе мы рассматриваем методы, сохраняющие количество сложений и все операции умножения на коэффициенты. Более того, во всех методах, кроме метода Горнера, домножение на коэффициенты и суммирование производятся в конце. Следовательно, в этих случаях мы можем отбросить эти операции и рассматривать задачу вычисления набора одночленов, то есть задачу построения ССВМ.

Мы будем обозначать носитель рассматриваемого полинома как M, а множество его переменных как V.

2.2 Сложность построения оптимальной схемы

Пусть задан критерий качества $f : s \to h, s \in S, h \in \mathbb{R}$, где S – множество возможных схем. Необходимо для заданного полинома $p(\bar{x}) = \sum_i c_i \bar{x}^{\bar{\alpha}_i}$ построить отвечающую ему схему с наибольшим возможным h.

В разд. 2.6 мы рассмотрим задачу о построении кратчайшей аддитивной последовательности, которая NP-полна и эквивалентна задаче построения оптимальный СВП для

вычисления значений полинома в числах фиксированной длины с плавающей точкой. Нам неизвестно формального доказательства NP-полноты для построения оптимальной СВП для задачи редукции, однако интуитивно можно предполагать, что усложнение критерия качества не приведет к уменьшению сложности построения схемы.

2.3 Тривиальная схема

Существует схема, которая не требует построения – она зафиксирована в форме записи полинома. Согласно этой схемы, мы должны сначала вычислить все мономы, встречающиеся в полиноме, а затем просуммировать результаты. Мы будем называть эту схему тривиальной.

2.4 Обобщенный метод Горнера

Обобщенный (на случай полиномов многих переменных) метод Горнера (ОМГ) прост в реализации и позволяет получать достаточно хорошие схемы. Алгоритм построения 1 является рекурсивным. Фактически, он представляет исходный полином t в виде $t = pd + q$, причем ни один член q не делится на d. Затем операция повторяется для p и q, до тех пор, пока на месте t не окажется одночленом.

Легко видеть, что число рекурсивных вызовов не превышает число членов в исходном полиноме. Сложность каждого вызова составляет $O(n)$ операций вычисления наибольшего общего делителя двух одночленов GCD. Всего не более $O(n^2)$ операций GCD. Сложность одной такой операции можно считать постоянной, если ограничить величину коэффициентов некоторой константой и зафиксировать число переменных.

Например, для полинома $x^2yz + xyz + x^3 + xyz + yz^2 + z^3 + yz + y$ метод Горнера даст схему, соответствующую $x(yz(x + z + 1) + x * (x + 1)) + z(y(z + 1) + z^2) + y$, что дает 7 умножений и 8 сложений для задачи вычисления значений, вместо 17 умножений и 8 сложений, если использовать тривиальную схему.

2.5 Метод минимального покрывающего дерева

Рассмотрим орграф по отношению делимости на множестве $M' = M \cup V$ с весами на ребрах, равными некоторой функции частного, например, его полной степени. Нам необходимо выбрать некоторое подмножество ребер этого графа, которое будет отвечать схеме вычисления рассматриваемого набора мономов. Подмножество должно быть таким, чтобы каждый узел из $M \setminus V$ имел ровно одно входящее ребро, с тем чтобы не вычислять один и тот же моном дважды. Заметим, что такое подмножество всегда существует, так как каждый элемент $M \setminus V$ делится на хотя бы один элемент V. Пример построения можно видеть на рисунке 5. Здесь для краткости опущены все дуги, кроме соединяющих мономы на соседних уровнях.

Algorithm 1: HORNER(T,S)

Input: список T членов полинома в порядке убывания, контейнер схемы S

$P \leftarrow \{\}$

$d \leftarrow T[0]$

if *Length(T)*$= 1$ **then**

  return $AddNode(S, T[0])$

end

foreach $t \in T$ **do**

  $g \leftarrow GCD(t, d)$

  if $TotalDegree(g) > 0$ **then**

    $P \leftarrow P \cup \{t\}$

    $d \leftarrow g$

  end

end

$Q \leftarrow T \setminus P$

$P' \leftarrow \{\}$

foreach $t \in P$ **do**

  $P' \leftarrow P' \cup \{t/d\}$

end

$N_p \leftarrow Horner(P', S)$ `// recursive call`

$N_d \leftarrow AddNode(S, d)$

$N_p \leftarrow AddNode(S, \langle *, N_d, N_p \rangle)$

if $Q \neq \emptyset$ **then**

  $N_q \leftarrow Horner(Q, S)$

  $N_p \leftarrow AddNode(S, \langle +, N_p, N_q \rangle)$

end

return p

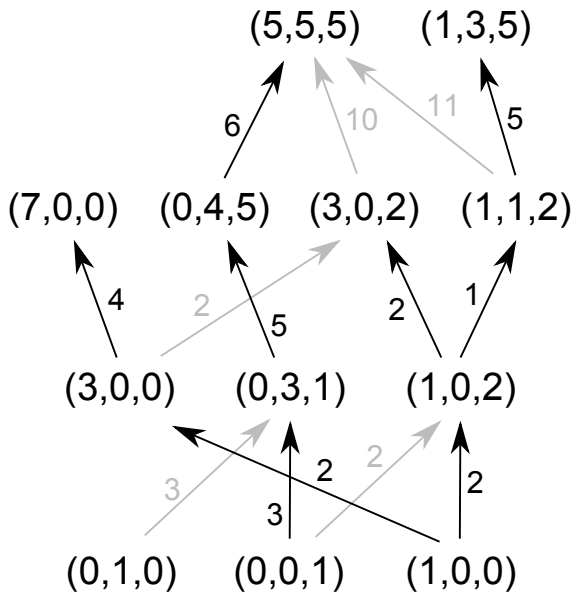

Рис. 5: Граф делимости (неполный). Черные дуги порождают схему.

Среди всех возможных подмножеств, отвечающих данным критериям, мы выберем одно, обладающее наименьшим весом, и обозначим соответствующий подграф G_{min}. Заметим, что название метода не отвечает структуре, которую мы построили – последняя представляет собой не дерево, а лес из $|V|$ деревьев, причем каждое из деревьев может не являться минимальным покрывающим деревом соответствующего подграфа, в силу ограничения на количество входящих ребер. Тем не менее, аналогия с задачей о минимальном покрывающем дереве кажется нам достаточно ясной.

Преобразуем, далее, выбранное подмножество в схему. Схема будет содержать по одной операции умножения на каждое ребро G_{min}. Ясно, что множители, отвечающие каждому ребру (частное мономов в соседних узлах), входят во множество корневых мономов. Также туда входят первые степени всех переменных, за исключением тех, которые оказались изолированными в G_{min}, как, например, моном $(0, 1, 0)$ на рис. 5. Пример получающейся схемы изображен на рис. 6

После того, как подмножество наименьшего веса выбрано, нам остается добавить операции, соответствующие умножению на коэффициенты и суммированию, а также построить схему для вычисления корневых мономов. Очевидно, в данном случае разделение схемы на две части выглядит несколько нелогично. Мы обсудим этот момент и его влияние на эффективность схемы в разд. 2.6.

Рассмотрим пример. Пусть задан полином $x^2yzw + xy^2zw + xyz^2w + xyzw^2$. ММПД в этом случае построит схему вида $x(xyzw + y^2zw + yz^2w + yzw^2)$, тогда как интуитивно оптимальной будет $xyzw(x + y + z + w)$. Проблема заключается в неспособности данного метода добавлять промежуточные мономы. Если добавить такую возможность, то мы получим интересный метод, предположительно уступающий по эффективности методу аддитивных цепочек, но превосходящий и метод Горнера, и ММПД. В рамках данной

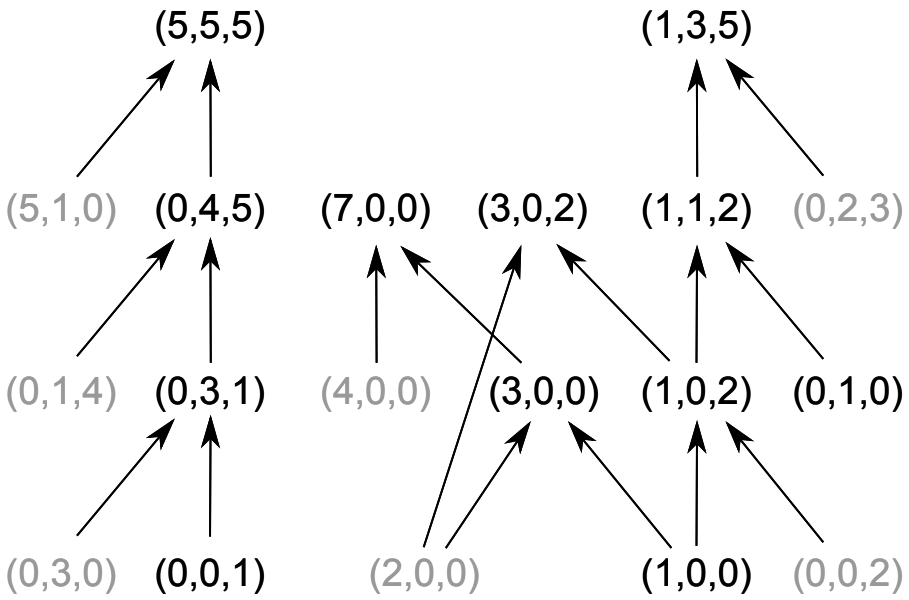

Рис. 6: Схема, построенная по графу на рис. 5. Серым выделены добавочные корневые мономы.

работы мы не будем рассматривать его подробнее.

2.6 CCBM и аддитивные цепочки

Схемы совместного вычисления мономов, как было указано выше, является частным случаем схем вычисления полиномов. В отличие от последних, они содержат лишь операции умножения мономов. Исторически, однако, задача совместного вычисления мономов рассматривалась независимо.

Задача совместного вычисления мономов часто формулируется в терминах *аддитивных цепочек*.

Аддитивной цепочкой (addition chain) в n-мерном пространстве называется последовательность a_i целочисленных векторов, первыми n членами которой являются элементы канонического базиса, а каждый из следующих элементов представляет собой сумму каких-либо двух предыдущих:

$$a_{ij} = \delta_{ij}, \quad i = 1..n, \; j = 1..n$$

$$a_i = a_j + a_k, \quad j \leq k < i, \quad i > n$$

Целочисленные векторы здесь можно рассматривать как мультииндексы мономов. Тогда сложение векторов будет соответствовать умножению мономов.

Лучше всего на сегодняшний день изучены одномерные аддитивные цепочки – они применяются в ряде алгоритмов, а том числе в криптографии, для эффективного возведения в степень (*потенцирования*). Бинарный алгоритм возведения в степень является частным случаем алгоритма на основе аддитивных цепочек. Вопрос о длине $l(m)$ кратчайшей аддитивной цепочки, содержащей число m, достаточно подробно описан в [11].

Для $l(m)$ известна асимптотическая оценка [6]: $l(m) \sim log(m)$ (здесь и далее логарифмы по основанию 2). Заметим, что бинарный алгоритм дает цепочку длины $2log(m)$ в худшем случае.

Задача о кратчайшей одномерной аддитивной цепочке также часто обобщается следующим образом – вместо одного числа, предлагается найти кратчайшую цепочку, содержащую множество чисел [11]. В англоязычной литературе эта обобщенная постановка называется addition *sequence* problem, и мы будем использовать термин *аддитивная последовательность*, так как более подходящего термина нам неизвестно. Длину такой цепочки обозначают $l(m_1, m_2, ..., m_k)$. Длину кратчайшей k-мерной аддитивной цепочки обозначают $l([m_1, m_2, ..., m_k])$. Интересный результат получен в [14] – показана «эквивалентность» двух последних задач. Именно, для любого k и $\{m_i\}_{i=1}^{k}$, $l([m_1, m_2, ..., m_k]) = l(m_1, m_2, ..., m_k) + k - 1$.

Также установлено [26], что

$$\forall k \, L(\bar{m}_1, \bar{m}_2, ..., \bar{m}_k) \sim log\left(max_{i=1..k}(\bar{m}_i)\right),$$

где $L(\bar{m}_1, \bar{m}_2, ..., \bar{m}_k) = max_{\{m_i < \bar{m}_i\}} l(\{m_i\})$.

В некоторых алгоритмах, таких как DSA, используются многомерные аддитивные цепочки. Также в ряде задач используются аддитивные цепочки с вычитанием – они могут быть короче обычных аддитивных цепочек. Однако, это оправдано лишь в тех случаях, где такая операция допускается целевым доменом и имеет в нем разумную вычислительную сложность.

Также подробно изучен вопрос о сложности построения кратчайшей аддитивной цепочки – эта задача является NP-полной [9]. Существуют и достаточно эффективные приближенные алгоритмы [5, 10, 7].

Заметим, тем не менее, что и в теории аддитивных цепочек, и в вопросах их практическиого применения остается множество открытых вопросов.

Схему совместного вычисления мономов, очевидно, можно рассматривать как многомерную аддитивную последовательность, однако построение оптимальной схемы сводится к вычислению кратчайшей аддитивной последовательности лишь в случае постоянного критерия качества, например, при вычислении значений мономов в действительных числах с плавающей точкой фиксированной длины или в конечном поле. Если вычисления производятся в целых числах, кратчайшая аддитивная последовательность уже не соответствует, строго говоря, оптимальной схеме вычисления. В таких задачах, как возведение полинома в степень, неоптимальность таких схем была также подтверждена экспериментально (этот факт упоминается в [11]). Логично предположить, что для задачи редукции полинома, где сложность варьируется еще сильнее, схема, соответствующая кратчайшей аддитивной последовательности, будет далека от оптимальной. Тем не менее, такая схема содержит ценную информацию о структуре рассматриваемого набора мономов, и опреде-

ленно заслуживает рассмотрения, возможно, с некоторыми модификациями.

2.7 Схемы специального вида

Для некоторых полиномов специального вида можно построить более эффективные схемы, используя информацию об их структуре. В качестве примера, рассмотрим задачу совместного вычисления мономов для носителя некоторого плотного полинома.

Определение 4. *Полином p называется плотным, если*

$$(\exists x \in V\, x \in s(p)) \wedge (\forall \alpha \in s(p)\, \exists \beta \in s(p), x \in V\, \alpha = x\beta)\,.$$

Значение плотного полинома с n членами может быть вычислено за не более чем $n-1$ операций умножения и $n-1$ операций сложения. Построение соответствующей схемы также может быть реализовано весьма эффективно. Легко видеть, однако, что такую же схему можно построить и методом минимального покрывающего дерева. Мы не будем рассматривать данный алгоритм подробнее в рамках данной работы, так как класс приложений, для которых преимущество в скорости построения схемы вычисления для плотных полиномов дает существенный выигрыш в производительности, достаточно узок.

3 Эффективная редукция разреженных полиномов

Редукция зачастую осуществляется относительно набора полиномов, не являющегося базисом Грёбнера. В этом случае результат не единственен и может зависеть от используемого алгоритма редукции. Однако, можно говорить о нормальной форме полинома в более общем смысле, если зафиксировать метод редукции. В данном разделе мы будем использовать термин «нормальная форма» в этом нестрогом смысле, имея ввиду некоторый полином, полученный редукцией данного относительно заданного набора и нередуцируемый относительно него

Также следует обратить внимание на тот факт, что мы используем два различных метода редукции, первый из которых может быть выбран относительно произвольно, а второй – (РСВП) – строится на основе первого.

3.1 Задача редукции

Задача редукции полинома p относительно набора полиномов H суть задача поиска остатка от деления этого полинома на них. Простейший подход [24] к его нахождению заключается в последовательном вычитании $h \in H$ из p с некоторыми множителями. Чтобы гарантировать, что редукция в какой-то момент завершится, обычно фиксируют некоторое *допустимое мономиальное упорядочение* и используют на каждом шаге один из полиномов, чей старший член (в смысле указанного упорядочения) делит старший член рассматриваемого полинома, а в качестве множителя используется их частное.

Когда же вычитание выполнить нельзя, старший член полинома перемещается в остаток, и процедура повторяется до тех пор, пока p не окажется равным нулю.

На практике, низкая эффективность алгоритма редукции часто связана с ростом количества промежуточных мономов. В процессе редукции количество членов полинома может интенсивно расти. Рассмотрим, в качестве примера, набор

$$G = \{x_1^2 - (1 + \sum_{i=2}^{m} x_i)\} \cup \{x_i | i = 2..m\}.$$

Теперь редуцируем x^{2n} относительно G. Если применять первое правило, пока это возможно, мы получим полином $(1 + \sum_{i=2}^{m} x_i)^n$, содержащий C_n^{n+m-2} членов, который затем будет редуцирован к 0. Данный пример является искусственным и на практике число промежуточных членов обычно не растет настолько интенсивно, однако существенное влияние этого фактора на эффективность алгоритма редукции установлено.

РСВП направлен на сокращение количества промежуточных членов, и в большинстве случаев он позволяет этого достичь, однако можно привести примеры, когда использование обычной редукции более эффективно.

Данный подход наиболее эффективен в тех случаях, когда идеал, порожденных полиномами, относительно которых производится редукция, является нульмерным. Кроме

того, данный подход демонстрирует высокую эффективность по сравнению с обычным алгоритмом редукции в тех случаях, когда носитель редуцируемого монома содержит много мономов и сильно разрежен.

3.2 Описание метода

РСВП основан на двух свойствах нормальных форм (1, 2). Первое из них – линейность оператора нормализации. Второе заключается в том, что нормальная форма произведения двух полиномов равна нормальной форме произведения их нормальных форм.

$$N(\lambda p + \mu q) = \lambda N(p) + \mu N(q) \tag{1}$$

$$N(p \cdot q) = N(N(p) \cdot N(q)) \tag{2}$$

Используя эти правила, можно определить пространство нормальных форм по заданному базису. Легко видеть, что оно изоморфно факторалгебре по идеалу, заданному рассматриваемым базисом.

$$\lambda p +_n \mu q = \lambda p + \mu q \tag{3}$$

$$p \cdot_n q = N(p \cdot q) \tag{4}$$

Итак, РСВП представляет собой операцию подстановки – или *замены переменных* – в факторалгебре. Данную подстановку можно рассматривать как тождественную – как правило, не изменяются даже имена переменных, – но не следует забывать, что она выполняется в другом пространстве.

Заметим также, что в определении операции умножения в пространстве нормальных форм присутствует оператор нормализации. Ему соответствует некоторый алгоритм редукции, не совпадающий с РСВП, так как в противном случае определение последнего было бы некорректным. Мы обсудим этот оператор подробнее в разделе 4.4.

Чтобы редуцировать полином при помощи данного метода, необходимо

- вычислить нормальные формы мономов нижнего уровня,
- выполнить операции согласно схемы, используя (3, 4),
- вернуть результат последней операции в качестве ответа.

3.2.1 Сложность операций умножения

Очевидно, что эффективность метода существенно зависит от используемой схемы, причем аппроксимировать критерий качества в данном случае весьма проблематично – сложность редукции двух мономов, сигнатуры которых отличаются на единицу, может различаться в произвольное число раз.

В двумерном случае множество мономов можно изобразить графически (рис. 7). Множество редуцируемых мономов, которые можно получить из нередуцируемых мономов путем умножения их на первую степень одной из переменных, будем называть *достаточным множеством редуцируемых мономов* (ДМ), так как зная нормальные формы все мономов в нем, мы можем вычислить нормальную форму любого полинома относительно данного базиса, не выполняя более никаких редукций. ДМ можно рассматривать как своего рода таблицу умножения.

Однако, множество нередуцируемых мономов (и, следовательно, ДМ) конечны лишь для нуль-мерных идеалов, и даже в этом случае вычислять все нормальные формы заранее нет необходимости (это может быть слишком дорого) – мы можем редуцировать те мономы, которые появляются в процессе вычислений. Это реализуется посредством механизма кеширования нормальных форм.

Если потребовать, чтобы нормальная форма каждого монома вычислялась не более чем единожды, то суммарная сложность всех редукций в случае нульмерного идеала будет ограничена. Следовательно, поиск нормальной формы любого полинома можно будет выполнить за полиномиальное время от его длины и степени.

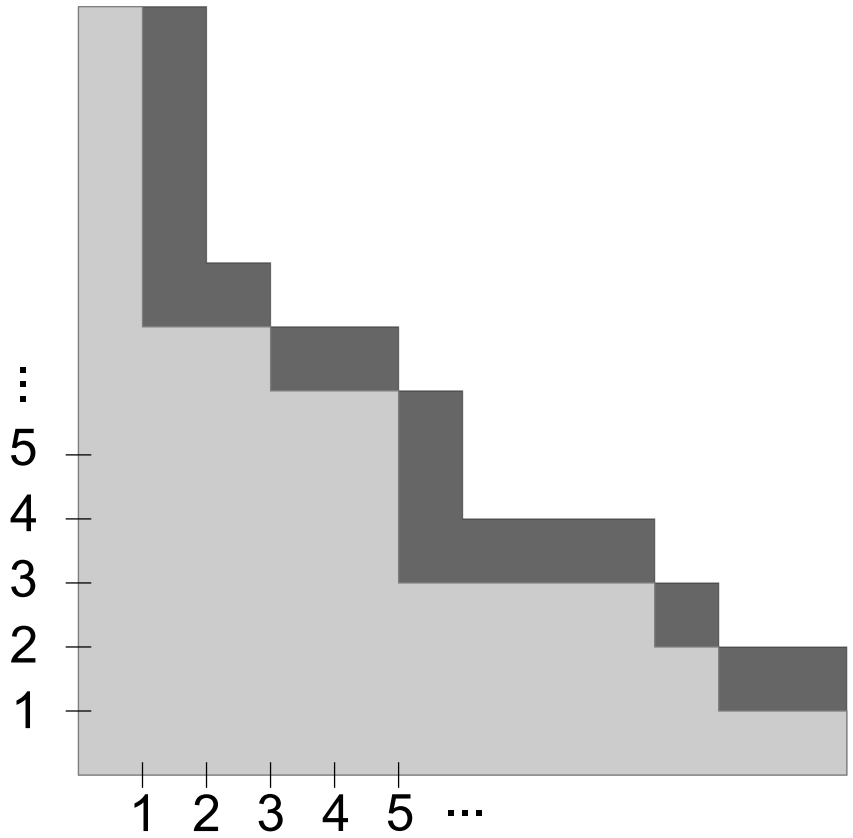

Рис. 7: Светло-серым изображена область нередуцируемых мономов, темно-серым – ДМ.

В данной работе мы реализуем такую схему кеширования (см. 4.4), отвечающую указанному требованию, и оцениваем ее эффективность. Следует заметить, что данная идея не нова – она используется, в частности, в [23]. Есть также определенные основания по-

лагать, что некоторая схожая схема кеширования реализована в системе компьютерной алгебры Singular.

Что касается операций сложения и умножения на число, их стоимость, как правило, мала по сравнению со стоимостью операции умножения. Кроме того, как мы указали выше, количество таких операций отстается постоянным во всех схемах, генерируемых рассматриваемыми методами.

3.3 Наращивание схемы

РСВП демонстрирует высокую эффективность в наших экспериментах. Однако, с тем, чтобы узнать его сильные и слабые стороны, следует рассматривать фактическую сложность выполнения отдельных операций. Мы полагаем, это позволит как улучшить алгоритм вычисления в соответствии с заданной схемой, так и построить более точный приближенный критерий сложности, с тем чтобы генерировать более эффективные схемы.

В разделе 5 мы обсудим некоторые эксперименты с данным алгоритмом. Наблюдая работу РСВП, мы пришли к выводу, что, вероятно, модификация схемы в процессе выполнения редукции может дать существенный прирост производительности. Дело в том, что при редукции могут возникать новые мономы, которых не было в исходном полиноме, и их редукция может потребовать значительного времени. В то же время, велика вероятность, что мы уже знаем нормальную форму одного или нескольких мономов, которые делят рассматриваемый моном. Используя эти мономы, мы можем «достроить» фрагмент схемы, с тем чтобы включить в нее новый моном.

Развивая эту идею, мы неизбежно приходим к заключению, что строить схему лучше прямо в процессе ее выполнения, так как мы обладаем в этот момент дополнительной информацией, которая не была доступна ранее. Например, дополнительные мономы, добавленные в схему в процессе редукции, могут позволить нам достичь целевых мономов более коротким путем.

Схему можно наращивать двумя способами – «в ширину» или «в глубину». Какой из подходов предпочтителен – неясно, – это требует экспериментальной проверки. Кроме того, не вполне ясно, достаточно ли планировать схему на каждом шаге для всех оставшихся мономов, какого-то одного из них, или некоего подмножества, а также, насколько критично качество промежуточной схемы в этом случае.

К сожалению, мы не имеем возможности вполне исследовать данный вопрос в рамках данной работы.

4 Детали реализации

В рамках данной работы была реализована библиотека для работы с полиномами над полем рациональных чисел или конечным полем с учетом специфики рассматриваемой задачи на языке `C++`. При помощи данной библиотеки был реализован ряд описанных ранее алгоритмов с целью сравнения их эффективности между собой и с системой компьютерной алгебры Maple 14 [31], а также выявления потенциальных путей повышения эффективности рассматриваемых алгоритмов.

При реализации библиотеки использованы некоторые базовые контейнеры, механизм ввода-вывода и средства анализа эффективности кода из библиотеки Indigo API [30], распространяемой под лицензией GPL.

4.1 Прототипирование

На стадии прототипирования описанные методы построения схем – метод Горнера и ММПД, – а также алгоритм вычисления значений полинома по заданной схеме были реализованы на языке Python. Позднее они были также перенесены на C++ с целью повышения эффективности, а также во избежание дублирования отдельных процедур работы с полиномами.

4.2 Представление мономов

Представление мономов в программных пакетах компьютерной алгебры изучено достаточно хорошо. Анализ эффективности различных представлений можно найти в [1]. Мы используем представление в виде вектора степеней с полной степенью.

Анализа состава набора мономов, используемых в процессе редукции полиномов, показал, что многие мономы низких степеней используются повторно. Чтобы снизить потребление памяти и повысить эффективность библиотеки в целом, мы храним все уникальные мономы, используемые в данный момент времени в специальной структуре данных на основе хэш-таблицы с подсчетом ссылок. Таким образом, мономам присваиваются уникальные номера, что существенно упрощает реализацию механизма кеширования нормальных форм.

На основе анализа работы алгоритма редукции для полиномов высоких степеней мы также пришли к выводу, что удаление мономов при обнулении счетчика ссылок не является необходимым, так как количество используемых мономов обычно растет практически в течение всего времени работы алгоритма, после чего резко падает. Мы не выполняем формального сравнения такого подхода с традиционным, так как используемые в данной работе алгоритмы в любом случае требуют наличия у мономов уникальных ключей, т.е. накладных расходов на поддержание структуры с уникальными мономами не избежать. Однако, мы допускаем, что такой подход может быть плохо приспособлен для систем

компьютерной алгебры общего назначения.

Поддерживаются лексикографическое и градуированное обратное лексикографическое упорядочения.

4.3 Представление полиномов

Полиномы представлены в виде списков с общим пулом элементов. Это означает, что при создании нового полинома выделяется лишь небольшой объем памяти на стеке, а память, выделенная в общем пуле для членов полинома, может быть использована повторно после уничтожения данного объекта – это особенно полезно при реализации алгоритма редукции, где возникает большое количество промежуточных полиномов.

В качестве коэффициентов могут выступать элементы конечного поля, чей модуль не превышает 2^{31}, либо рациональные числа произвольной точности. Последние реализованы посредством библиотеки GMP [29], которая представляет собой де-факто стандарт вычислений произвольной точности и используется в таких системах компьютерной алгебры как Mathematica, Maple и Singular, а также во множестве других программных пакетов и приложений.

4.4 Кеширование нормальных форм мономов

Как было сказано выше, используемый способ представления мономов автоматически дает нам уникальные идентификаторы для них, поэтому реализация кеша нормальных форм мономов не представляет сложности. Внимания заслуживает вспомогательный алгоритм редукции, используемый в нашей реализации РСВП. Нашей целью было гарантировать, что нормальная форма любого нужного нам монома будет вычислена лишь однажды (если только мы не удалим ее из кеша явно, например, с целью высвобождения памяти).

Итак, вспомогательный алгоритм редукции получает на вход моном m и должен вернуть в качестве результата его нормальную форму, также записав ее в кеш. Нормальные формы всех промежуточных мономов, потребовавшихся нам в процессе редукции, также должны попасть в кеш.

Соответствие предложенного алгоритма (2) указанным условиям легко проверить. Ясно, что нормальная форма всякого монома, вычисленная данным алгоритмом, будет помещена в кеш. Остается удостовериться, что редукция одного и того же монома не будет выполнена дважды. Это возможно лишь в случае, если данный моном окажется промежуточным продуктом редукции самого себя. Однако все мономы p строго меньше m в смысле заданного упорядочения, и, по индукции, все мономы, могущие встретиться в процессе редукции мономов из p также строго меньше m. Следовательно, указанная ситуация невозможна и алгоритм удовлетворяет указанным требованиям.

Заметим, что практическая реализация данного алгоритма не является рекурсивной, так как для полиномов высоких степеней это могло бы привести к переполнению стека в

Algorithm 2: AuxReduce(m,G,T)

Input: моном m, базис G, кеш нормальных форм T

Output: нормальная форма монома m

if *HasKey(T,m)* **then**

  return $T[m]$ `// monomial cached`

end

foreach $g \in G$ **do**

  $lm \leftarrow LeadingMonomial(g)$

  if lm *divides* m **then**

    $p \leftarrow m - m \cdot g/lm$

    $q \leftarrow 0$

    foreach *term* $t \in p$ **do**

      $q \leftarrow q + Coefficient(t) \cdot AuxReduce(Monomial(t), G, T)$ `// recursive call`

    end

    $T[m] \leftarrow q$

    return $T[m]$

  end

end

$T[m] \leftarrow m$ `// monomial is irreducible`

return $T[m]$

процессе работы алгоритма.

4.5 Реализация СВП

Представление СВП в программе практически в точности повторяет формальное определение – это последовательность операций вида:

- вычислить нормальные формы мономов $m_1, m_2, ..., m_k$ и записать в первые k ячеек памяти
- вычислить сумму или произведение полиномов в ячейках i и j и записать в ячейку r
- вычислить произведение полинома в ячейке i и коэффициента f и записать в ячейку r

Классы, реализующие вычисление значений полинома, РСВП и подсчет количества операций в схеме, представляют собой интерпретаторы такой программы.

4.6 Поддерживаемые операции для мономов и полиномов

Приведем список операций, реализованных для мономов:

- создание монома с заданной мультистепенью
- создание монома на основе строки, например $x1^4 * x2^7$
- вычисление произведения, частного и наибольшего общего делителя мономов
- проверка делимости и равенства
- сравнение в заданном упорядочении
- получение степени конкретной переменной и полной степени
- вывод монома в текстовой форме.

Полиномы поддерживают

- добавление и удаление членов
- загрузку из строки и сохранение в строку
- сложение и умножение на число (коэффициент)
- перебор членов
- получение старшего члена
- сортировку согласно заданного упорядочения.

5 Результаты

5.1 Эффективность СВПдля вычисления значений полиномов

Сравним простейшую схему (МП), обобщенный метод Горнера (МГ) и ММПД по количеству операций сложения и умножения в генерируемых ими схемах. Для тестирования используем набор случайных полиномов (табл. 1).

	# переменных	Степень	# мономов	Размерность базиса	Степень базиса
p_1	2	374	45	4	11
p_2	2	1444	54	12	20
p_3	3	821	267	17	19

Таблица 1: Полиномы над полем рациональных чисел

Результаты приведены в табл. 2. Можно заметить, что метод Горнера дает лучшие результаты.

	МП	МГ	ММПД
p_1	8684	1221	2995
p_2	39246	6865	10345
p_3	121796	34361	41772

Таблица 2: Число операций умножения, необходимое для вычисления полинома, в зависимости от метода построения схемы

5.2 Эффективность РСВП

Рассмотрим сначала редукцию полиномов над полем рациональных чисел. Для тестирования используем те же полиномы, что и в предыдущем примере. В качестве образующих идеала используются базисы Грёбнера наборов случайных полиномов, созданные при помощи Maple. Заметим, что рассматриваемый метод эффективнее алгоритма редукции, реализованного в системе Maple, не только для полиномов высоких степеней, однако данные эксперименты наглядны и исключают возможное влияние различных побочных операций, таких как загрузка полинома.

	МП	МГ	ММПД	Maple
p_1	827	187	684	3270
p_2	1863	320	520	15868
p_3	210557	75846	127671	391264

Таблица 3: Время выполнения редукции полиномов над $\mathbb{Q}$, мс

Также следует обратить внимание на тот факт, что при редукции полиномов над полем рациональных чисел бóльшая часть времени тратится на операции с числами, а не с мономами.

Приведем также результаты сравнения для редукции над конечным полем по модулю $p = 31$. Это число выбрано произвольно и не оказывает какого-либо влияния на эффективность рассматриваемого метода. Полиномы и соответствующие идеалы также выбраны случайным образом.

	# переменных	Степень	# мономов	Размерность базиса	Степень базиса
q_1	3	53	36	30	20
q_2	3	289	72	30	20
q_3	3	126	132	14	30

Таблица 4: Тестовые полиномы над конечным полем

	МП	МГ	ММПД	Maple
q_1	34	62	150	3716
q_2	2908	5552	2366	1674390
q_3	25183	65732	39224	–

Таблица 5: Время выполнения редукции полиномов над конечным полем, мс

Заметим, что в табл. 3 в двух из трех случаев наилуший результат показывает тривиальная схема. Это подтверждает предположение о сложности выбора оптимальной схемы до начала выполнения редукции, и если в случае с полиномами над полем рациональных чисел минимизация числа операций более обоснована, так как сложность операций с коэффициентами «сглаживает» неравномерность истинного критерия качества, то здесь некорректность гипотезы равномерности проявляется в полной мере.

Мы также провели ряд экспериментов с тем, чтобы выявить пути повышения эффективности алгоритма редукции. Как было отмечено в разделе 3.2.1, мы стараемся ограничить число мономов, которые необходимо редуцировать, так как это требует вычислительных ресурсов и дополнительной памяти для их кеширования.

Исходя из этого, мы ищем нормальную форму произведения мономов, умножая один из них на первые степени переменных и нормализуя после каждой такой операции. Если учесть, что в качестве сомножителей берутся лишь нередуцируемые мономы, мы можем гарантировать, что редукция будет выполняться только для мономов достаточного множества (см. 3.2.1).

В результате экспериментов мы выяснили, что порядок выполнения умножений на переменные также имеет значение. Последовательность переменных (или, в более общем случае, мономов), на которые мы домножаем рассматриваемый моном, определяет его

траекторию в пространстве мономов. Наихудшим способом является, судя по всему, домножение на переменные по порядку – например, домножение на моном x^3y^5z заменяется последовательным домножением на $\{x, x, x, y, y, y, y, y, z\}$. В качестве альтернативы, мы можем чередовать переменные в этой последовательности, например, $\{x, y, z, x, y, x, y, y, y\}$ – при таком подходе траектория монома меньше отклоняется от заданного направления, что, интуитивно, снижает среднюю сложность редукций промежуточных мономов. Более того, мы обнаружили, что объединение различных подряд идущих переменных в группы ($\{xyz, xy, xy, y, y\}$) хотя и нарушает границы достаточного множества, но в большинстве случаев дает прирост производительности по сравнению с двумя другими рассмотренными вариантами. Например, для базиса из четырех полиномов десятой степени в четырехмерном пространстве, редукция некоторого монома сороковой степени выполняется последним способом на 60% быстрее. Именно эта последняя стратегия используется в наших экспериментах в начале раздела.

Заключение

Итак, мы рассмотрели понятие схем вычисления полиномов, некоторые методы их построения, а также применение таких схем для повышения эффективности алгоритма редукции полиномов.

Описанные методы и алгоритм редукции реализованы, исследована их эффективность, а также выполнено сравнение производительности алгоритма редукции, использующего схемы вычисления полиномов, с алгоритмом, реализованным в системе компьютерной алгебры Maple 13.

На основании выполненных экспериментов и некоторых теоретических соображений, мы можем заключить, что рассматриваемый метод редукции демонстрирует высокую производительность и гибкость, а также предложить потенциально более эффективный его вариант – с построением схемы в процессе выполнения редукции.

A Примеры базисов и полиномов, использованных при тестировании

A.1 Полином p_1 над полем рациональных чисел

$G = xy^4+x^6+x^5y^3, y^9-x^5y+y^4, x^{10}y-y^7x^2+x^8+x^6y^2+x^6y+x^3y^4, x^{11}-x^8y^2-x^3y^6+x^7y+x^7$

$p = xy - x^{69}y^{53} - x^{74}y^{49} - x^{73}y^{52} + x^{85}y^{43} + x^{90}y^{39} + x^{89}y^{42} + x^{84}y^{51} - x^{151}y^{16} + x^{146}y^{19} + y^{56}x^{84} - y^{65}x^{74} + x^{58}y^{147} - x^{63}y^{139} + x^{58}y^{142} + x^{146}y^{24} + y^{68}x^{69} + x^{76}y^{177} - x^{81}y^{169} + x^{76}y^{172} + x^{69}y^{73} + x^{194}y^{177} - x^{189}y^{180} + x^{185}y^{103} - x^{180}y^{106} - x^{89}y^{48} - x^{189}y^{185} - x^{175}y^{115} - 2x^{180}y^{111} - x^{179}y^{114} + x^{124}y^{61} + x^{129}y^{57} + x^{128}y^{60} + x^{97}y^{117} + x^{102}y^{113} + x^{101}y^{116} + x^{110}y^{130} + x^{115}y^{126} + x^{114}y^{129} + y^{76}x^{52} + y^{73}x^{53} + y^{77}x^{48} + y^{20}x^{103} + y^{17}x^{104} + y^{21}x^{99}$

A.2 Полином q_1 над полем конечным полем характеристики 30

$G = x^3y^3z^6+z^2x^9, xy^3z^6+x^6y^9z, 30zx^9+y^9x^8, xy^6z^{11}+30z^3x^{13}, x^3y^{11}z^4+xy^3z^{13}+x^{11}z^5, x^8y^6z^4+x^{10}y^3z^4+xy^8z^4, x^{14}z^5+30x^{10}z^9+30xy^5z^9, xy^{14}z^4+x^9y^3z^5+x^{11}z^5, x^{12}y^3z^4+30xy^{11}z^4+x^3y^8z^4+30x^9z^5, x^9y^8z^2+30xy^3z^{15}+30x^{11}z^7, x^{14}y^3z^2+z^2x^{16}+30xy^8z^6, xy^4z^{15}+x^{11}yz^7+30z^3x^{10}, xy^{11}z^8+x^7y^8z^4, x^{16}z^4+30x^5y^8z^4+30xy^8z^8+30xy^3z^{13}, z^3x^{17}+30z^7x^{13}+x^{10}y^2z^3, x^{12}y^6z^2+30xy^3z^{11}, z^2x^{18}+xy^{11}z^6+z^2y^5x^9+x^9z^7, xy^3z^{17}+x^{11}y^5z^4+z^9x^{11}+x^2y^{10}z^4, x^{10}z^{11}+xy^{11}z^7+y^5z^3x^9+x^9z^8+xy^5z^{11}+30z^3y^2x^{11}, x^{12}z^9+30x^5y^8z^5+30xy^8z^9+30xy^3z^{14}+30x^9z^5y^2, x^{13}z^8+30x^6y^8z^4+30x^2y^8z^8+30x^2y^3z^{13}+30x^{10}y^2z^4, x^{13}yz^7+30x^9yz^{11}+30z^3x^{12}, x^9z^{13}+30x^2y^{13}z^4+30x^6y^8z^5+30x^2y^8z^9+30x^2y^3z^{14}+29x^{10}y^2z^5, x^9yz^{12}+30x^2y^9z^8+30x^2y^4z^{13}+30x^{10}y^3z^4+x^{12}z^4+xy^3z^9$

$p = -x^{66}y^{28}z^{70}+x^{86}y^{81}z^{84}-x^8y^{39}z^{61}+x^{55}y^{23}z^{22}+x^{16}y^{42}z^{76}-x^{83}y^{77}z^{77}+x^{45}y^{40}z^{68}-x^{27}y^8z^{91}+x^{92}y^{89}z^{19}-x^{66}y^{15}z^9+x^{64}y^{87}z^{27}+x^{89}y^{95}z^{20}-x^{37}y^{86}z^{98}-x^{27}y^{22}z^{61}-x^7y^{28}z^{41}-x^{50}y^{89}z^{32}+x^{53}y^{58}-x^{95}y^{25}z^{47}+x^{78}y^{92}z^{23}-x^{91}y^{59}z^{34}-xy^{31}z^{77}+x^{65}y^{83}z^{47}+x^{65}y^{58}z^{20}-x^{19}y^{56}z^{18}+x^{27}y^{58}z^{80}+x^3y^{53}z^{76}+x^{81}y^{38}z^{35}+x^{84}y^{20}z^{30}+x^{85}y^{73}z^{35}+x^{77}y^{86}z^{43}-x^{10}y^{51}z^8-x^{92}y^{35}z^{48}-x^{99}y^{99}z^{26}+x^{93}y^{62}z^{44}-x^{72}y^{24}z^{96}+x^{38}y^6z^4+x^{16}y^{21}z^{37}-x^{25}y^{75}z^{76}+x^{66}y^{92}z^9+x^{38}y^{76}z^{26}+x^{73}y^{47}z^{10}-x^{98}y^{76}z^{76}-x^{25}y^{87}z^7+x^{89}y^2z^{75}-x^{15}y^{87}z^{24}+x^{61}y^{69}z^{88}+x^{97}y^{32}z^{92}-x^{49}y^{86}z^{89}+x^{45}y^{59}z^{86}-x^{98}y^{93}z^{98}+x^8y^{66}z^{90}-x^{72}y^{89}z^{24}-x^{26}y^{86}z^{66}-x^{81}y^{74}z^{52}-x^{13}y^{64}z^{20}-x^{67}y^{56}z^{39}+x^{87}y^{57}z^{32}-x^4y^{38}z^{11}+x^{44}y^8z^{83}+x^{57}y^{17}z^{52}-x^{80}y^{80}z^{31}-x^{37}y^{94}z^{13}+x^{91}y^{26}z^{49}-x^{39}y^{15}z^{45}+x^{85}y^{87}z^{15}-x^{59}y^5z^{48}-x^{29}y^{53}z^{86}-x^{92}y^{90}z^{68}-x^{77}y^{90}z^{76}+x^{90}y^{12}z^{84}-x^{66}y^{27}z^{28}+x^{99}y^{12}z^7$

Список литературы

[1] Bachmann O., Schoenemann H. Monomial Representations for Groebner Basis Computation. *Proc. International Symposium on Symbolic and Algebraic Computation*, ACM 1998

[2] Bernstein D. Pippenger's Exponentiation Algorithm. `http://cr.yp.to/papers.html#pippenger`

[3] Bini D., Pan V. Parallel polynomial computations by recursive processes. *Proc. ACM SIGSAM Intern. Symp. on Symbolic and Algebraic Comp.*, ACM 1990

[4] Bini D., Gemignani L., Pan V. Improved parallel computations with matrices and polynomials. *Proc. 18-th Intern. Symposium on Automata, Languages and Programming*, Springer 1991, 520-531

[5] Bos J., Coster M. Addition chain heuristics. *Proc. CRYPTO'89*, Springer 1995

[6] Brauer, O.M. Addition-Subtraction Chains. *Diplomarbeit*, University Paderborn 2001

[7] Cruz-Cortes N., Rodriges-Henriques F., Jurez Morales R., Coello C. Finding optimal addition chains using a genetic algorithm approach. `http://delta.cs.cinvestav.mx/~francisco/cis2005_nareli.pdf`, 2005

[8] Dimitrov V.S., Jullien G.A., Miller W.C. Algorithms for multiexponentiation based on complex arithmetic. *Proc. 13th Symposium on Computer Arithmetic*, IEEE 1997

[9] Downey P., Leong B., Sethi R. Computing sequences with addition chains. *SIAM Journal on Computing 10 (3)*, SIAM 1981, 638–646

[10] Gordon D. A survey of fast exponentiation methods. *Journal of Algorithms 27*, Elsevier 1998, 129-146

[11] Knuth D. The Art of Computer Programming, vol. 2. Addison-Wesley 1981

[12] Knuth D. Evaluation of Polynomials by Computer. *Communications of the ACM, vol. 5, issue 12*, ACM 1962, 595-599

[13] Moeller B. Algorithms for Multiexponentiation. *Technical report*, TU Darmstadt 2001

[14] Olivos J. On vectorial addition chains. *Journal of Algorithms*, Elsevier 1981, 13-21

[15] Vasiliev N.N. Complexity of monomial evaluations and duality. *Computer Algebra in Scientifig Computing*, Springer 1999, 479-485

[16] Vasiliev N.N. Fast polynomial reduction. *Proc. Workshop on Computer Algebra and Differential Equations*, 2009

[17] Vasiliev N.N. Joint Evaluations of Sparse Polynomials and the Synthesis of Evaluating Programms. *Proc. International Conference Systems and Methods of Analytical Computations with Computers in Theor. Phys.*, 1985, 154-159

[18] Vasiliev N.N. Creation of efficient symbolic-numeric interface. *Lecture Notes in Computer Science V378*, Springer 1989, 118-119

[19] Vasiliev N.N. The System for the Synthesis of evaluating programms for polynomial-trigonometric expressions. *Proc. International Conference Analytical Computations with Computers in Theor. Phys.*, 1983, 120-124

[20] Александров Д.Е., Галкин В.В., Зобнин А.И., Левин М.В. Распараллеливание матричных алгоритмов вычисления базисов Грёбнера. *Фундаментальная и прикладная математика, 2008, том 14, № 4*, МГУ 2008, 35—64

[21] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов, Москва, Мир 1979

[22] Белага Э.Г. О вычислении значений многочленов от одного переменного с предварительной обработкой коэффициентов. *Проблемы кибернетики, вып. 5. М.*, Физматлит 1961, 7-16

[23] Клискунов А.Н. Эффективные вычисления нормальных форм в алгоритме Бухбергера. *Магистерская диссертация*, СПбГПУ ФизМех, 2006

[24] Кокс Д., Литтл Дж., О'Ши Д. Идеалы, многообразия и алгоритмы. Москва, Мир 2000

[25] Кочергин В.В. Об асимптотике сложности аддитивных вычислений систем целочисленных линейных форм, *Дискретный анализ и исследование операций, Апрель-Июнь 2006. Серия 1. Том 13, № 2*, 38–58

[26] Кочергин В.В. О вычислении наборов степеней, *Дискретная математика, том 6, выпуск 2*, 1994, 129-137

[27] Пан В.Я. Некоторые схемы для вычисления многочленов с действительными коэффициентами. *Проблемы кибернетики, вып. 5. М., Физматлит*, 1961, 17-30

[28] Axiom: the scientific computation system. `http://www.axiom-developer.org/`

[29] The GNU multiple precision arithmetic library. `http://gmplib.org/`

[30] Indigo API. `http://opensource.scitouch.net/indigo/`

[31] Maple, computer algebra system. `http://www.maplesoft.com/`

[32] Singular, computer algebra system. `http://www.singular.uni-kl.de/`

[33] Sage, open-source mathematics software systems. http://www.sagemath.org/