Модификация алгоритма Копперсмита умножения матриц над GF(2)

М.А. Черепнёв

Аннотация **--- В последнее время все чаще требуется применение алгоритмов эффективно работающих с машинными словами и таких, что основная работа может быть произведена в кэше процессора, то есть время на перезапись меньше. Важно отметить, что объем кэша выпускаемых процессоров быстро растет. В данной работе предлагается новая модификация алгоритма Копперсмита для умножения блочных векторов, ширина которых равна длине машинного слова, друг на друга и на квадратные матрицы. Мы рассматриваем случай, когда коэффициенты лежат в поле из двух элементов. Данные алгоритмы относится к числу блочных алгоритмов, в которых операции производятся с машинными словами, то есть с каждым из 64 битов одновременно. Полученные результаты могут быть применены и к булевым матрицам без значительных изменений. Поскольку данные алгоритмы относится к числу алгоритмов с предвычислениями, то эти оценки представляют практический интерес при использовании процессоров с увеличенным размером кэш памяти первого уровня. Рассмотренные в данной статье алгоритмы дают увеличение скорости пропорционально логарифму объема кэша первого уровня. Для применения в задачах, где данные матричные операции являются базовыми (например, задача целой факторизации) это может оказаться существенным.**

Ключевые слова **--- Умножение матриц, поле из двух элементов, предвычисления.**

Статья получена 12 августа 2019.
Работа поддержана грантом РФФИ 19-01-00294.
М.А.Черепнев - Московский государственный университет имени М.В. Ломоносова, РФ (e-mail: cherepniov@gmail.com)

I. Введение

Умножение матриц, состоящих из нулей и единиц по законам алгебры логики или в поле Галуа, является классической задачей. Кроме того работа с такими матрицами большого размера может потребоваться при реализации прикладных алгоритмов, например алгоритм целой факторизации и др.

Операции в поле F=GF(2) реализуются в машине сложение как логическая операция XOR, умножение - как логическая операция AND. При этом указанные операции со всеми 64 битами машинного слова проделываются параллельно.

Вычисление произведения двух n x n матриц над полем может быть произведено при помощи $O(n^3)$ сложений и умножений в этом поле. Различные способы ускорения этого процесса [6], [7] приводят к сложности $O(n^{2.373})$. Однако данные алгоритмы работают с битами и поэтому для F не эффективны. Сведение к меньшим размерам может быть произведено, например, при помощи алгоритма Штрассена-Винограда [1]. Однако на практике для более эффективного использования арифметики машинных слов и кэш памяти приходится пользоваться другими алгоритмами.

В данной статье мы рассмотрим умножение блочных векторов из $F^{N\times n}$, друг на друга (скалярное произведение), умножение блочных векторов на маленькие матрицы из $F^{n\times n}$ (линейная комбинация).

Согласно статьи [3], в которой эти

операции используются для решения больших разреженных линейных систем, сложность их оценивается величиной $O(nN)$ операций с машинными словами. Однако для более быстрой реализации этих операций можно применить хорошо известный алгоритм "четырёх русских" [5], дающий, как мы покажем ниже, для умножения блока на маленькую матрицу существенно меньшее время.

Копперсмит (см.[4], стр. 342-343) для умножения блока на маленькую матрицу предложил фактически тот же алгоритм, что и [5], а для умножения блоков друг на друга, некий новый алгоритм.

Выбором оптимальных параметров для этих алгоритмов получены алгоритмы с лучшими оценками сложности.

II. Описание алгоритмов

Теорема. *Сложность линейной комбинации не более* $2\frac{nN}{log_2\frac{N}{2}}$, *а сложность скалярного произведения не более*

$$3\frac{nN}{log_2\frac{N}{2}-log_2log_2\frac{N}{2}}.$$

Доказательство. Без ограничения общности можно доказывать эти оценки в виде, где под знаком log_2 стоит N, но в предположении, что оно является степенью числа 2. Пусть требуется умножить матрицу $D\in \mathsf{F}^{N\times n}$ на блок $M\in \mathsf{F}^{n\times n}$, строки которого представлены машинными словами $a_1a_2\dots a_n$. Разделим матрицу D на вертикальные блоки с горизонтальным размером k. В случае, если соответствующий размер не делится на k, дополняем соответствующие матрицы нулевыми столбцами и строками. На первом этапе алгоритма умножения составим $\frac{n}{k}$ адресных массивов, где каждой строке $b_1b_2\dots b_k, b_i\in\{0,1\}$ из k бит поставим в соответствие сумму машинных слов $\sum_{i:b_i=1}^{k} a_{kt+i}, t=0,1,\dots,\frac{n}{k}-1$. Если упорядочить адресные строки по возрастанию, то очередной элемент массива получается при помощи одного сложения, а для создания всех массивов потребуется не более $\frac{n}{k}2^k$ сложений. На втором этапе умножаем построчно матрицу D на блок M, складывая машинные слова, соответствующие каждому блоку из k бит. Для умножения одной строки на блок M потребуется $\frac{n}{k}$ сложений. Общее число сложений по двум этапам равно $\frac{n}{k}2^k+N\frac{n}{k}$. При $k=log_2N$ эта сумма оценивается величиной $2\frac{Nn}{log_2N}$. Отметим, что если в каком-то алгоритме требуется умножать разные блоки на один и тотже матричный коэффициент из $\mathsf{F}^{n\times n}$, то при втором умножении первый этап можно

опустить, воспользовавшись массивами, полученными при первом умножении. Если блок умножается на два матричных коэффициента, то построение массивов можно осуществлять параллельно, а затем вызывать из них слова с одинаковыми адресами для записи их в соответствующие строки результирующей матрицы. Аналогичные построения для умножения блоков друг на друга дают алгоритм со сложностью $2\frac{Nn}{log_2 n}$.

Пусть теперь требуется умножить два блока $L, M \in \mathsf{F}^{N\times n}$ друг на друга, а именно вычислить $L^T M$. Разделим, как и раньше, левый блок L по столбцам на блоки меньшего размера k. Аналогично тому, как это было сделано раньше, предполагаем что $k \mid n$. Линейно независимых строк в таких блоках, не более чем 2^k. В однопроцессорной версии будем последовательно умножать транспонированные блоки L_i^T с маленьким размером k на блок M, представленный построчно машинными словами. На первом этапе проходим этот блок по столбцам, выбирая из них разные, а именно, проходя слева направо, если очередной столбец ранее не встречался, отмечаем его, присваивая новый порядковый номер (или запоминаем в другом блоке), одновременно аналогично отмечая строку с тем же номером в блоке M. Если столбец встречался ранее, то мы его не отмечаем, а в блоке M прибавляем строку с тем же номером в блоке M к строке с номером, на котором этот столбец встретился впервые. Затем убираем все неотмеченные столбцы в рассматриваемом блоке и все неотмеченные строки в блоке M. Оставшиеся блоки с общим размером не более 2^k перемножаем, умножая каждую строку на столбец машинных слов. Общее число операций равно $\frac{n}{k}(2N + k2^k)$. Выбирая $k = log_2 \frac{N}{log_2 N}$, получим оценку сложности

$$3\frac{Nn}{log_2 N - log_2 log_2 N}.$$

Отметим, что для ускорения построенные матрицы из $\mathsf{F}^{k\times 2^k}$ можно умножать при помощи описанного выше алгоритма "четырёх русских" со сложностью $\frac{k2^k}{log_2 k}$, однако такая замена поможет изменить лишь второе слагаемое в знаменателе полученной оценки.

Лемма доказана.

В обозначениях теоремы для эффективной реализации первого из рассматриваемых алгоритмов требуется быстрая кэш память для хранения $2^к$ машинных слов. Это накладывает естественное ограничение на размер N. При этом к должно делить n=64. При современном объеме кэша первого уровня в 16 Кбайт = 2^{17} бит, получаем 17≥к+6. То есть к=8. Такое значение и было использовано Копперсмитом в [4]. Увеличение до к=16 будет эффективно при объеме кэша первого уровня около половины гигабайта.

Отметим, что во втором алгоритме дополнительная память по сравнению с той, которая содержит условие задачи, нужна только для записи модифицированных правых частей M. В кэш процессора последовательно помещаются блоки L_i^T. Если выполнять всю эту работу параллельно, то требуемая быстрая память должна быть объемом $2N(k+n) \leq 4Nn$. То есть кэша размером 16 Кбайт будет достаточно для обработки матриц с размером $N=2^9$, а для больших размеров при последовательной подаче кусков длины 2^9 в оценке сложности знаменатель заменится на цифру 5=8-3. Получается, что чем больше кэш тем более длинные куски можно обрабатывать на этом процессоре, и общее время убывает пропорционально логарифму этой длины. Тот же вывод дают аналогичные рассуждения для первого алгоритма.

VIII. Заключение.

Мы показали, как размер кэша процессора влияет на скорость матричных операций с матрицами, состоящими из нулей и единиц. Для рассмотренных алгоритмов скорость растет пропорционально логарифму объема кэша.

БИБЛИОГРАФИЯ

[1] Albrecht, M. R., G. V. Bard, and W. Hart. "Efficient multiplication of dense matrices over GF (2). CoRR abs/0811.1714 (2008)."

[2] Криптографические протоколы/ Черепнёв М.А.// Москва: МАКС Пресс, 2018.

[3] Montgomery P.L. A Block Lanczos Algorithm for Finding Dependencies over GF(2)// Advances in Cryptology - EuroCrypt'95 / Louis C.Guillou and Jean-Jacques Quisquater, editors. Berlin: Springer-Verlag, Lect. Notes in Comp. Sci.- 1995.-v.921- p.106-120.

[4] Coppersmith D. Solving homogeneous linear systems over GF(2) via block Widemann algorithm.// Mathematics of Computation.- 1994-v.62-no.205-p.333-350.

[5] Об экономном построении транзитивного замыкания ориентированного графа./ Арлазаров В.Л., Диниц Е.А., Кронрод М., Фараджев И.А.//ДАН СССР.-1970.- т.194-№3-с.487-488.

[6] Powers of tensors and fast matrix multiplication. / Le Gall F.// International Symposium on Symbolic and Algebraic Computation, pp. 296–303 (2014)

[7] Multiplying matrices faster than coppersmith-winograd./ Virginia Vassilevska Williams// Proceedings of the 44th Symposium on Theory of Computing Conference, pp. 887–898 (2012)

Modification of the Coppersmith algorithm for matrix multiplication over GF(2)

M.A. Cherepniov

***Abstract*--- In recent years, increasingly requires the use of algorithms that work effectively with machine words and such that the main work can be done in the processor cache, that is, the time to data overwrite less. It is important to note that the size of the released processor cache in modern computers is growing rapidly. In this paper, we propose a new modification of the Coppersmith algorithm for multiplying block vectors whose width is equal to the length of the machine word, on each other and on square matrices. We consider the case when the coefficients lie in a field with two elements. These algorithms are among the block algorithms in which operations are performed with machine words, that is, with each of the 64 bits in parallel. Our results can be applied to Boolean matrices without significant changes. Since these algorithms are of the algorithms with pre-calculations, these estimates are of practical interest when using processors with an increased size of the memory cache of the first level. The algorithms considered in this article give an increase in speed in proportion to the logarithm of the first level cache volume. For use in tasks where these matrix operations are basic (for example, the integer factorization), this can be significant.**

***Key words*---Matrix multiplication, field with two elements, pre-calculation.**

REFERENCES

[1] Albrecht, M. R., G. V. Bard, and W. Hart. "Efficient multiplication of dense matrices over GF (2). CoRR abs/0811.1714 (2008)."

[2] Kriptograficheskie protokoly/ Cherepnjov M.A.// Moskva: MAKS Press, 2018.

[3] Montgomery P.L. A Block Lanczos Algorithm for Finding Dependencies over GF(2)// Advances in Cryptology - EuroCrypt'95 / Louis C.Guillou and Jean-Jacques Quisquater, editors. Berlin: Springer-Verlag, Lect. Notes in Comp. Sci.-1995.-v.921- p.106-120.

[4] Coppersmith D. Solving homogeneous linear systems over GF(2) via block Widemann algorithm.// Mathematics of Computation.-1994-v.62-no.205-p.333-350.

[5] Ob jekonomnom postroenii tranzitivnogo zamykanija orientirovannogo grafa./ Arlazarov V.L., Dinic E.A., Kronrod M., Faradzhev I.A.//DAN SSSR.-1970.- t.194-#3-s.487-488.

[6] Powers of tensors and fast matrix multiplication. / Le Gall F.// International Symposium on Symbolic and Algebraic Computation, pp. 296–303 (2014)

[7] Multiplying matrices faster than coppersmith-winograd./ Virginia Vassilevska Williams// Proceedings of the 44th Symposium on Theory of Computing Conference, pp. 887–898 (2012)