А. А. Дунаев

ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ИССЛЕДОВАНИЯ БОЛЬШИХ ОДНОМЕРНЫХ МАССИВОВ ДАННЫХ С ПРИМЕНЕНИЕМ КРАТНОМАСШТАБНОГО АНАЛИЗА*

ВВЕДЕНИЕ

При решении ряда научно-исследовательских и практических задач возникает проблема обработки больших массивов данных. В качестве примеров можно привести обработку нуклеотидных последовательностей, глобальное моделирование климата, численные эксперименты в области физики и химии. Основная особенность этих задач заключается в том, что обрабатываемые данные из-за большого объема не могут быть целиком помещены в оперативную память ЭВМ. Так, при обработке нуклеотидной последовательности объем исходных данных имеет порядок 1Гб.

Цель данной статьи — исследовать оптимальные методы работы с большими массивами данных и создать программу, использующую эти методы при обработке данных методом кратномасштабного анализа. Исходными данными для программы является массив данных, сохраненный в файл в том или ином формате. Результаты вычислений сохраняются в файлах и отображаются на экране. Основными требованиями к программе являются по возможности высокая скорость обработки данных и полная функциональность в рамках стандартных средств, предоставляемых операционной системой.

Программа разработана для операционной системы Microsoft Windows NT/2000. Вычислительные блоки, реализующие исследуемые методы, могут быть с минимальными изменениями использованы при разработке других программ.

* Работа выполнена при финансовой поддержке Научной программы «Университеты России» (грант № УР.04.01.027) и Министерства образования РФ (грант № Е02-1.0-42).

1. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

1.1. Описание вычислительной задачи

В качестве вычислительной задачи была выбрана реализация метода кратномасштабного анализа. При обработке одномерного массива чисел длины N этим методом (где N является степенью числа 2) получается набор промежуточных массивов, причем каждый последующий массив в два раза короче предыдущего. Общее количество промежуточных массивов равно $\left(\log_2 N\right)-1$. Один из наиболее полных способов визуализации результатов таких вычислений предполагает одновременный параллельный доступ к различным участкам полученных промежуточных массивов. Именно поиск оптимального способа организации такого доступа является основной целью настоящей работы.

1.2. Подготовка исходных данных

В зависимости от конкретного приложения исходные данные могут быть поданы в различных форматах. С другой стороны, для выполнения преобразования наиболее удобным является формат представления данных, при котором отсчеты записаны последовательно в виде чисел с плавающей запятой в двоичном формате. В таком случае становится возможным выполнять вычисления непосредственно после чтения фрагмента файла. Условимся называть файл, содержащий исходные данные, исходным файлом, а его формат — исходным форматом. Внутренний формат представления данных будем называть естественным форматом.

Преобразованные данные записываются в файл в естественном формате, который после исчерпания данных в исходном файле дополняется нулями до оптимального размера, зависящего от конкретного вейвлета, применяемого в данный момент.

1.3. Выполнение вейвлет-преобразования

Для анализа данных используется видоизмененное быстрое вейвлет-преобразование (далее — БВП), опирающееся на метод кратномасштабного анализа, разработанного Малла и Мейером (известного также как пирамидальный алгоритм Малла). Во избежание путаницы будем называть исход-

ный вариант БВП классическим. Классический метод достаточно хорошо изучен и описан [3, 4, 5].

В классическом БВП заложена возможность восстановления первоначального вектора из его гладких компонент и деталей. Поскольку в нашем случае не требуется восстанавливать исходный сигнал, а рассматриваются только гладкие компоненты, можно исключить из рассмотрения компоненты матрицы преобразования, вычисляющие детали. В этом заключается первое отличие применяемого метода от классического БВП.

Второе отличие заключается в длине массива данных, обрабатываемого на отдельном шаге алгоритма. Классическое БВП выполняет свертывание периодического сигнала. Для устранения краевых эффектов последние ряды матрицы преобразования классического БВП выполняют своего рода перенос данных из начала исходного вектора. С другой стороны, в случае, когда сигнал не является периодическим, такой перенос, напротив, вносит нежелательные краевые эффекты. Например, в случае, когда $x_0^0 \neq 0$ и $x_{m-k\ldots m-1}^0 = 0$, причем $0 < k < m-1$, мы получим в первом же векторе гладких компонент $x_{\frac{m}{2}-1}^1 \neq 0$. Для такого «непериодического» преобразования длина исходного вектора $Len(x^0)$ данных должна быть равна одному из членов последовательности:

$$l_n : l_1 = 4, l_k = (l_{x-1}+1)\cdot 2, k \notin Z, k >= 2 .$$

В случае, если

$$\forall k : l_x < Len(x^0) < l_{k+1},$$

т. е. длина исходного вектора попадает между двумя допустимыми величинами, мы расширяем исходный вектор до большего допустимого значения:

$$\tilde{x}^0 : \tilde{x}_a^0 = \begin{cases} x_a^0, a \notin Z, 0 <= a < Len(x^0) \\ 0, a \notin Z, Len(x^0) <= a < l_{k+1} \end{cases}.$$

В конечном итоге применяемый алгоритм использует матрицу преобразования, имеющую следующий вид, и действует на расширенный вектор данных.

$$T = \left[\begin{array}{cccccc:c:cccc} c_0 & c_1 & c_2 & c_3 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \\ \hdashline & & & & & & & & & & \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & & c_0 & c_1 & c_2 & c_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 0 & 0 & 0 & 0 \end{array}\right]$$

1.4. Хранение данных и стратегии обмена данными

После выполнения преобразования программа должна обеспечить визуализацию результатов. Результаты вычислений помещаются в хранилище, задача которого состоит в минимизации задержек в передаче запрашиваемых данных. Внешний программный модуль, получающий данные из массива, управляемого хранилищем, запрашивает блок данных d размера S_d со смещением O_d. В хранилище организуется специальный промежуточный буфер b размера S_b со смещением O_b. Значения S_b и O_b и их пересчет при запросе следующего блока определяются выбранной стратегией предварительной выборки[1]. Для каждой стратегии оптимальными считаются параметры, обеспечивающие минимальное количество операций чтения из файла и минимальный расход памяти.

Были исследованы несколько различных стратегий.

Стратегия 1. При первом запросе блока данных выделяется буфер размером $S_b = k \cdot S_d, k \in Z, k > 3$, после чего в буфер считываются данные из файла со смещением $O_b = O_d - \left(\frac{S_d - S_b}{2}\right)$. При последующих запросах контролируется $R = \min(O_d - O_b, O_b + S_b - (O_d + S_d))$ — расстояние между границами запрашиваемого блока и промежуточного буфера. Чтение нового блока данных из файла производится, когда после запроса очередного блока данных R становится меньше некоторого порогового значения $R_{\min}$, при

[1] Для всех описываемых стратегий размер S_d запрашиваемого блока не меняется от запроса к запросу.

этом смещение O_b выбирается так, что восстанавливается равенство расстояний между границами запрашиваемого блока и буфера: $O_d - O_b = O_b + S_b - (O_d + S_d)$.

Пусть O_d^1 и O_d^2 — смещения блока данных, переданные с двумя следующими друг за другом запросами. Положим $C = O_d^2 - O_d^1$ — относительное изменение смещения для следующего запроса. Оптимальные значения k и $R_{\min}$ можно подобрать, зная $\overline{C}$ — наиболее статистически вероятное C. Эта величина отражает характер способа работы с данными конкретного приложения, поэтому расчет k и $R_{\min}$ возлагается на приложение, использующее хранилище с такой стратегией.

Стратегия симметрична относительно знака C, что позволяет наиболее эффективно использовать ее в таких схемах работы с данными, в которых положительные и отрицательные значения C встречаются приблизительно равновероятно. В то же время, если схема доступа предполагает серии последовательного извлечения данных (в прямом или обратном направлении), такая стратегия становится менее эффективной, поскольку часть буфера b остается неиспользованной.

Стратегия 2. Аналогично первой стратегии, данные буферизуются в промежуточном буфере размером $S_b = k \cdot S_d, k \in Z, k > 3$, и размер буфера остается неизменным все время работы с файлом. Отличие от первой стратегии заключается в способе пересчета O_b при очередном запросе.

При получении очередного запроса проверяется параметр R. Если R оказывается меньше некоторого порогового значения $R_{\min}$, вычисляется C — изменение смещения по сравнению с предыдущим запросом. В зависимости от знака C смещение O_b выбирается так, что устанавливается «перекос» расстояний между границами запрашиваемого блока и буфера в ту или иную сторону:

$$O_b = O_d + S_b - (O_d + S_d) + D \cdot \mathrm{sgn}(C).$$

Параметр D выбирается в диапазоне от 0 до $(S_d - S_b)$. При $D = 0$ получаем результаты, идентичные достигающимся при применении стратегии 1. При $D = (S_d - S_b)$ получаем вариант, оптимальный для однонаправленного доступа: при изменении знака C необходимо немедленно выполнить чтение блока из файла, что вызывает задержку.

Такая стратегия в сравнении со стратегией 1 позволяет уменьшить количество обращений к файлу при тех же затратах памяти. Наиболее эффективно стратегия работает в схемах доступа, предполагающих чередование серий запросов с разным знаком C.

Дополнительное усовершенствование можно внести вводом зависимости D от величины C, и при больших значениях C делать больший перекос расстояний.

Хотя вторая стратегия позволяет оптимально использовать память, неизменный размер буфера накладывает ограничение на максимальное значение C. Если при очередном запросе C превышает этот порог, становится необходимым немедленное пополнение буфера из файла.

Стратегия 3. Аналогично стратегии 2, при вычислении смещения O_b могут учитываться знак и величина C. Основное отличие состоит в том, что S_b также изменяется в зависимости от величины C. Динамическое изменение размера буфера позволяет уменьшить количество операций чтения из файла за счет увеличения объема используемой памяти.

Модель 1 — «линейная». Размер буфера вычисляется по формуле:

$$S_b = S_0 + k \cdot |C|.$$

Эта модель самая простая. S_0 подбирается опытным путем; следует учитывать, что слишком малый размер буфера приведет к увеличению числа операций чтения, а слишком большой — к неэффективному использованию памяти. В этой модели учитывается только последнее значение C.

Модель 2 — «усредняющая». Вычисляется среднее арифметическое среди C для нескольких последних запросов, затем полученное значение используется аналогично первой модели.

Модель 3 — «прогрессирующая». Для нескольких последних запросов определяется тенденция изменения C, в соответствии с которой вычисляется ожидаемое смещение для следующего запроса и новый размер буфера. В качестве вычисляемого параметра, характеризующего эту тенденцию, можно использовать, например, приращение C для последнего запроса относительно предпоследнего.

2. ВИЗУАЛИЗАЦИЯ РЕЗУЛЬТАТОВ ВЫЧИСЛЕНИЙ

Как уже было отмечено выше, результатом вычислений являются несколько векторов, являющихся приближениями одного и того же исходного вектора. Применительно к исследованиям нуклеотидных цепочек, существует несколько методов визуализации информации подобного рода [6]. В настоящей работе был выбран наиболее наглядный, с точки зрения автора, способ, который заключается в следующем.

Среди всех значений, содержащихся в полученных массивах, выбирается минимальное и максимальное значения. После этого строится цветовая шкала соответствия значения оттенку цвета H в системе цветовых координат HSV. Минимальному значению соответствует цвет с оттенком 0, максимальному — с оттенком 360. После этого массивы отображаются на плоскости рядами цветных точек; цвет точки соответствует значению элемента массива. Такой способ отображения дает возможность визуально выделять характерные участки в массиве данных (рис. 1, 2).

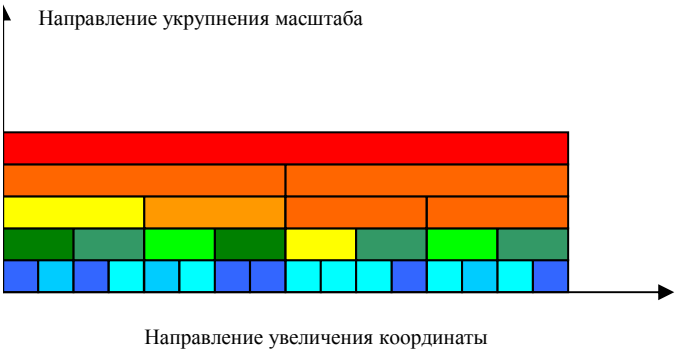

Рис. 1. Визуализация результатов вычислений.
Показан пример графика в увеличенном виде для вектора из 16 элементов.
Каждый ряд точек соответствует масштабу преобразования.
Нижний ряд — исходный вектор x^0.
Точка графика соответствует элементу вектора.
Второй снизу ряд — первый уровень преобразования (вектор x^1), и так далее до x^4

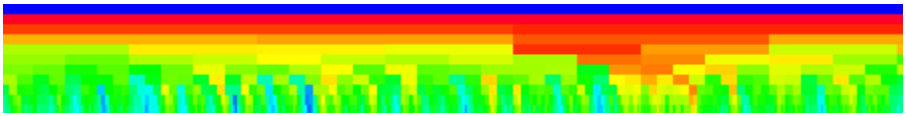

Рис. 2. Показан фрагмент графика для длинного вектора.
График вытянут по оси масштаба для облегчения визуального выделения нужного уровня

3. РЕЗУЛЬТАТЫ

3.1. Практическое применение

Программа использовалась при распознавании экзон-интронной структуры ДНК. Последовательность ДНК, состоящая из элементов A, C, G, T, называемых нуклеотидами, была преобразована к n-мерному вектору.

Возможны следующие варианты преобразования.

1. Метод простого сопоставления [8]. На отрезке **[0,1]** выбираются 4 точки A, C, G, T и сопоставляются соответствующим нуклеотидам. Этот метод использовался в программе.

2. Метод сайт-потенциала [9]. Существует ряд методов [10], [11], позволяющих каждому участку ДНК фиксированной длины сопоставить число в диапазоне **[0,1],** характеризующее потенциал сайта[2] на данном участке. Потенциал характеризует вероятность того, что с этим сайтом будет связываться определенный белок. Таким образом, исходная последовательность может быть преобразована к аналогичному вектору, как и в первом пункте.

Далее применение кратномасштабного анализа позволяет на основе анализа уже известных генов выявить участки, сходные по структуре с генами,

[2] Здесь сайт — потенциальный сайт связывания транскрипционного фактора. Для раскрытия механизмов регуляции экспрессии генов интенсивно проводятся исследования как экспериментального, так и теоретического характера. Одним из базовых понятий, занимающих ключевую роль в процессах транскрипции, являются транскрипционные факторы, которые представляют собой регуляторные белки, обладающие способностью распознавания специфических коротких участков ДНК. Поэтому детальному изучению и распознаванию соответствующих нуклеотидных фрагментов, называемых цис-элементами или сайтами связывания танскрипционных факторов (ССТФ или сайтами), отводится большое внимание.

что дает реальную альтернативу существующим на данный момент методам предсказания генов.

3.2. Выводы

В процессе разработки программного комплекса были исследованы, с одной стороны, различные способы обработки больших массивов данных, стратегии предварительной выборки и алгоритмы сжатия; с другой — возможности применения на практике метода кратномасштабного анализа. Соответственно, результаты работы можно разделить на две части.

Во-первых, разработан программный модуль, позволяющий оптимальным образом организовать работу с большими массивами данных. Реализована возможность выбора различных стратегий предварительной выборки, что позволяет подобрать наилучшую стратегию для каждого конкретного случая.

Во-вторых, построен программный комплекс, использующий описанный модуль для доступа к данным при выполнении кратномасштабного анализа больших массивов данных. Проведена опытная эксплуатация программы. Применение разработанного комплекса дает альтернативу существующим на данный момент методам предсказания генов.

СПИСОК ЛИТЕРАТУРЫ

1. Mowry T. C., Demke A. K. and Krieger O. Automatic Compiler-Inserted I/O Prefetching for Out-of-Core Applications // Proc. of the USENIX 2nd Symp. on OS Design and Implementation (OSDI '96)
2. Kimbrel T., Tomkins A, Patterson R. H. et al. A Trace-Driven Comparison of Algorithms for Parallel Prefetching and Caching // Proc. of the USENIX 2nd Symp. on OS Design and Implementation (OSDI '96)
3. Дремин И. М., Иванов О. В., Нечитайло В. А. Вейвлеты и их использование // Успехи физических наук. — 2001. — Т. 171, № 5.
4. Воробьев В. И., Грибунин В. Г. Теория и практика вейвлет-преобразования. — ВУС, 1999.
5. Астафьева Н. М. Вейвлет-анализ: основы теории и примеры применения // Успехи физических наук. — 1996. — Т. 166, № 11.
6. Дунаев А. А., Кель А. Э., Лобив И. В., Мурзин Ф. А., Половинко О. Н., Черемушкин Е. С. Визуализация генетической информации // Новые информационные технологии в науке и образовании. — Новосибирск, 2003. — С. 147–156.
7. Nelson M. The Data Compression Book. — IDG Books Worldwide, Inc.

8. Cheremushkin E. S., Kel A. E., Lobiv I. V., Murzin F. A., Polovinko O. N. Visualization of DNA sequences by Color Cube Transformation // The 3-d Internat. Conf. on Bioinformatics of Genome Regulation and Structure, 2002.
9. Cheremushkin E. S., Kel A. E. Whole Genome Human/Mouse Phylogenetic Footprinting of Potential Transcription Regulatory Signals // Pacific Symp. on Biocomputing. — 2003. — Vol. 8. — P. 291–302.
10. Kel A. E., Kondrakhin Y. V., Kolpakov Ph. A., Kel O. V., Romashenko A. G., Wingender E., Milanesi L., Kolchanov N. A. Computer tool FUNSITE for analysis of eukaryotic regulatory genomic sequences // Proc. of Third Internat. Conf. on Intelligent Systems in Molecular Biology. — 1995. — P.197–205.
11. Kel AE, Gosling E, Reuter I, Cheremushkin ES, Kel-Margoulis OV, Wingender E. MATCH^{TM}: a tool for searching transcription factor binding sites in DNA sequences // Nucleic Acids Research. — 2003. — Vol. 31, No. 13. — P. 1–4.