Информационные процессы, Том 11, № 3, 2011, стр. 336–347.

Усовершенствованный метод матрицы переноса для подсчета гамильтоновых цепей на прямоугольных решетках и цилиндрах

А.М.Караваев

Петрозаводский государственный университет, Петрозаводск, Россия

Поступила в редколлегию 13.09.2011

Аннотация—В работе предлагается эффективная параллельная вычислительная реализация метода матрицы переноса в задаче подсчета гамильтоновых цепей на прямоугольных решетках и цилиндрах. Получен ряд новых рекуррентных соотношений для числа гамильтоновых цепей на решетках с фиксированной шириной и цилиндрах с фиксированным периметром. Рассчитанные с высокой точностью оценки скорости роста числа гамильтоновых цепей как функций ширины (или периметра) позволили уточнить значения константы связности для указанных семейств графов. Полученные результаты могут быть использованы для расчета термодинамических свойств плотноупакованных макромолекул в решеточных моделях.

1. ВВЕДЕНИЕ

Задача подсчета числа гамильтоновых цепей на прямоугольных сеточных графах имеет ряд важных практических приложений [1,2,3] и является актуальной задачей физики полимеров. Гамильтонова цепь является достаточно точной моделью плотноупакованной макромолекулы. Помимо этого, перечисление цепей является классической задачей как в теории графов, так и в перечислительной комбинаторике, и в силу своей трудности сама по себе представляет значительный интерес.

Задача подсчета количества гамильтоновых цепей на решетках и цилиндрах рассматривалась ранее. Одна из самых первых работ в этой области касалась подсчета всех простых цепей [4]. Если же говорить только о гамильтоновых цепях, то в работе [5] были получены линейные рекуррентные соотношения для решеток с фиксированной шириной $m = 2, \ldots, 6$ и цилиндров с фиксированным периметром $m = 3, 4, 5$. Авторам [6] удалось посчитать точные ответы для решеток шириной от $m = 2$ до $m = 8$ и $n = 2, \ldots, 20$. Точные решения для более широких и длинных решеток ($m \leq 15$ и $n \leq 100$) были получены в работе [7]. Лучший из опубликованных к настоящему времени алгоритмов решения поставленной задачи принадлежит автору [3]. В этой работе вычисления проводились для квадратных решеток размером до 17×17. Отметим, что во всех указанных работах использовался метод матрицы переноса, реализованный в том или ином виде.

Традиционно считается, что проводить расчеты на цилиндре сложнее, чем на решетке, о чем свидетельствует тот факт, что результаты на цилиндре получены для меньших значений m, чем в случае решеток. В то же время, использование техники, предложенной в нашей работе, дает возможность проводить вычисления для цилиндров с меньшими затратами ресурсов, чем для решеток, и получать, в связи с этим, более точные результаты.

Помимо этого, к новым результатам также следует отнести вычисление количества гамильтоновых цепей на решетках размером до 17×100 и цилиндрах размером до 18×100, отыскание рекуррентных соотношений для решеток шириной до $m = 8$ и цилиндров периметром до

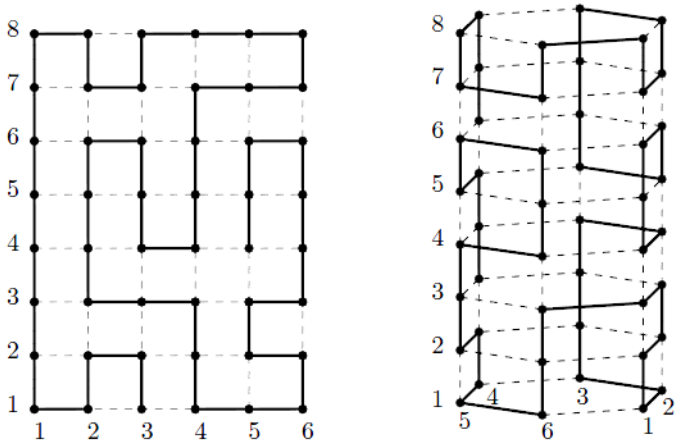

Рис. 1. Гамильтонова цепь на решетке $P_6 \times P_8$ (слева) и на цилиндре $C_6 \times P_8$ (справа)

$m = 10$. Показано, что значение универсальной константы связности для цилиндров с фиксированным периметром m ближе к своему пределу, чем ее значение для решеток шириной m. Таким образом, в нашей работе впервые показано, что наложение циклических граничных условий на решетку существенно упрощает вычисления.

Все вычисления для решеток и цилиндров для $m \geq 17$ проводились на кластере Московского государственного института электронной техники в г. Зеленограде с использованием, в основном, 168 вычислительных ядер (21 узел по 8 ядер и 4 Гб ОП). Способ распараллеливания предлагаемого метода приводится в разделе «Метод матрицы переноса».

2. ОПРЕДЕЛЕНИЯ И ПОСТАНОВКА ЗАДАЧИ

В работе обсуждаются два вида графов: прямоугольные решетки $P_m \times P_n$ и цилиндры $C_m \times P_n$. Прямоугольная решетка представляет собой целочисленные точки координатной плоскости, связанные по принципу ближайшего соседства. Обозначение $P_m \times P_n$ связано с тем, что целочисленный прямоугольник суть прямое произведение двух цепей, первая из которых имеет m звеньев, а вторая — n. Число m — ширина решетки (протяженность слева направо), а n — ее длина (протяженность снизу вверх). Символом $C_m \times P_n$ обозначается цилиндр, представляющий собой прямое произведение цикла C_m из m звеньев (периметр цилиндра) и цепи P_n из n звеньев (высота, или образующая цилиндра). Гамильтоновой цепью в графе называется цепь, проходящая через каждую его вершину в точности один раз. Пример решетки, цилиндра и гамильтоновых цепей в них приводится на рис. 1.

Наша задача состоит в подсчете точного числа гамильтоновых цепей на решетке и цилиндре заданного размера. Известно, что она является труднорешаемой [8], то есть для ее решения не существует эффективного алгоритма. По этой причине любые усовершенствования, позволяющие решить поставленную задачу для как можно бóльших значений m и n, оказываются актуальной темой исследования.

Любую цепь в графе $P_m \times P_n$ можно строить последовательно по *слоям* $P_m \times P_k$ ($k = 0, 1, 2, \ldots, n$). Для наглядности на рис. 2 слева изображена линия, разрезающая граф на две части, снизу от нее граф $P_6 \times P_5$, на котором еще недостроенная гамильтонова цепь представляет собой множество непересекающихся простых цепей. Сверху от линии изображен один из вариантов завершения недостроенной цепи. Ту часть цепи, которая находится ниже линии разреза будем называть *прошлым*, а ту, что находится выше — *будущим*.

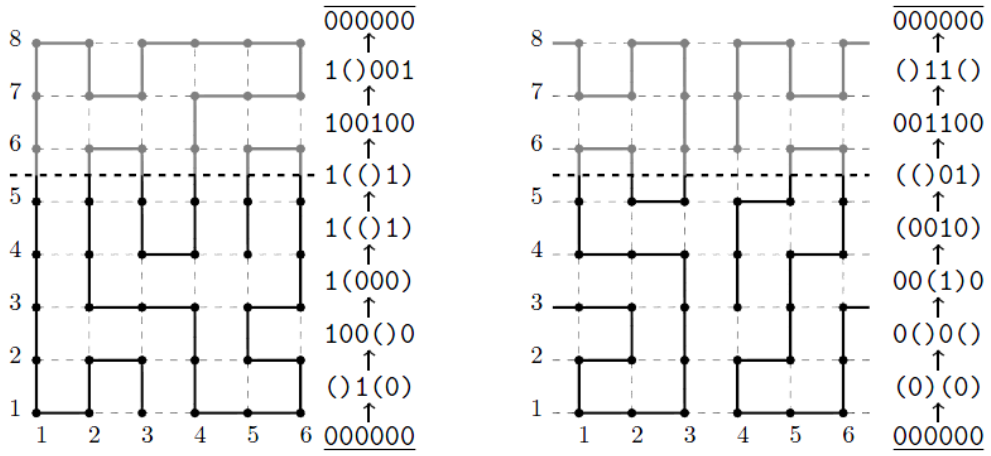

Рис. 2. Последовательность разрезов, образующихся при построении гамильтоновой цепи из рис. 1 на решетке $P_6 \times P_8$ (слева) и на развернутом цилиндре $P_6 \times P_8$ (справа). Пунктирной линией показан разрез слоя $k = 5$

Горизонтальный разрез делит гамильтонову цепь на прошлое и будущее. При этом прошлое может состоять из простых цепей двух типов. Цепь первого типа обоими концами пересекает линию разреза, а сама каким-либо образом проходит ниже линии разреза. Цепь второго типа начинается где-либо в прошлом и пересекает линию разреза только один раз. Например, на рис. 2 слева прошлое состоит из двух цепей первого типа и двух цепей второго типа, а справа — из двух цепей первого типа и одной цепи второго типа.

Само пересечение цепи линией разреза можно представить в виде правильной скобочной последовательности, разреженной нулями и единицами, и имеющей не более двух единиц. В такой последовательности пара соответствующих друг другу скобок соотносятся с двумя концам одной и той же цепи первого типа, нули означают, что в данных точках нет пересечения линии разреза с цепью, а единица показывает, что в этом месте линия разреза пересекает цепь второго типа. Например, на рис. 2 слева (разрез по слою $k = 5$) такая последовательность имеет вид `1(()1)`. Ограничение на количество единиц обусловлено тем, что у гамильтоновой цепи должно быть в точности два конца, но находиться они могут на любых слоях рассматриваемых графов.

В случае с цилиндром удобно представлять разрез тогда, когда цилиндр развернут на плоскости, а левая и правая границы изображаются так, как указано на рис. 2 справа. Поскольку цепи неориентированные, нет разницы, какой из концов цепи первого типа соотносить с левой скобкой, а какой — с правой, поэтому будем всегда считать, что левый конец цепи соотносится с левой скобкой, даже если в процессе построения гамильтоновой цепи концы простой цепи первого типа меняются местами за счет циклических граничных условий. По этой причине справедливо говорить, что любой разрез гамильтоновой цепи на цилиндре также будет соответствовать некоторой правильной скобочной последовательности, разреженной нулями и не более чем двумя единицами.

Двигая линию разреза снизу вверх, мы переходим от одного разреза к другому, как это показано на рис. 2 справа от решетки и аналогично для цилиндра. Для того, чтобы определить число гамильтоновых цепей на решетке необходимо посчитать число способов, начав построение с разреза `000000` (если $m = 6$), закончить его так же. Поскольку начальный и конечный разрезы шифруются одной и той же последовательностью из нулей, мы различаем их, записывая начальный разрез с чертой снизу, а конечный — с чертой сверху.

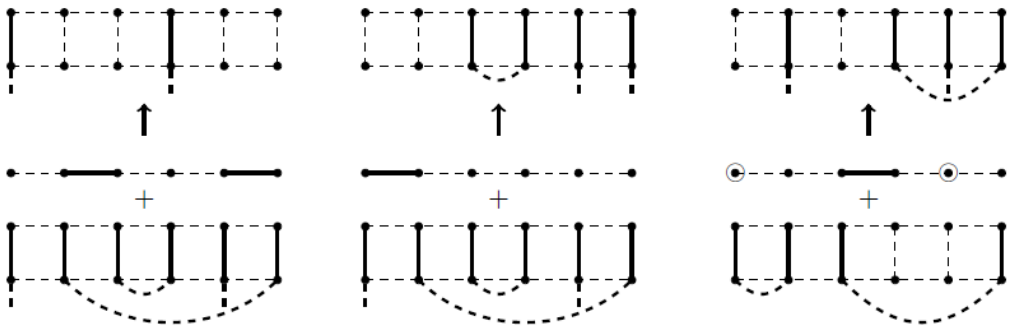

Рис. 3. Три примера перехода: от разреза `1(()1)` к `100100` и `00()11`, и от разреза `()(00)` к `010(1)` с образованием цепей второго типа

Множество всех таких последовательностей, имеющих длину m, является, очевидно, конечным для фиксированного m. Верхняя граница на мощность этого множества предлагается в работе [6], где (с точностью до «+1») приводится формула (без асимптотики)

$$M_m = \sum_{k=0}^{\lfloor m/2 \rfloor} \frac{1}{k+1} \binom{2k}{k} \binom{m}{2k} \left(1 + \binom{m-2k}{1} + \binom{m-2k}{2}\right) + 1 = O\left(3^m \sqrt{m}\right). \qquad (1)$$

Таблица значений чисел M_m приводится в разделе «Количество разрезов». Из (1) следует, что при увеличении m на 1, число возможных разрезов увеличивается приблизительно в три раза. На практике реальное количество разрезов оказывается существенно меньшим, чем предсказывает формула (1), о чем будет подробно говориться далее.

3. ПЕРЕХОД ОТ ОДНОГО РАЗРЕЗА К ДРУГИМ

Описанная здесь техника перехода от одного разреза к другим кратко обсуждалась в [6]. В этом разделе мы приведем более полное описание и расширим технику для случая цилиндров. Начнем описание с решеток.

Предположим, что ширина решетки m фиксирована, и зафиксирован некоторый разрез гамильтоновой цепи s. Для примера пусть $s =$ `1(()1)`, это разрез слоя $k = 5$ на рис. 2. Продолжить построение цепи можно путем добавления горизонтальных ребер, с последующим переходом на один шаг в будущее. В нашем примере (если следовать рис. 2) можно добавить два горизонтальных ребра (рис. 3 слева). В этом случае мы перейдем к слою $k = 6$, разрез которого будет иметь вид `100100`. Другой способ перехода состоит в том, чтобы добавить лишь одно ребро (рис. 3 по центру). Тогда осуществится переход к разрезу `00()11`.

Для того, чтобы получить все возможные разрезы, необходимо, как минимум, перебрать все $O(2^m)$ способов установить горизонтальные ребра прежде, чем сделать очередной шаг в будущее. Однако этим не исчерпываются все возможности перехода от одного разреза к другим. В каждом слое может начаться или закончиться одна из простых цепей. Эта ситуация показана на рис. 3 справа, где первый символ «$\odot$» обозначает запрет на переход соответствующего конца цепи в будущее, а второй такой символ показывает, что в этом месте начинается цепь второго типа.

У будущей гамильтоновой цепи должно быть два конца. Поэтому нельзя формировать более 2-х цепей второго типа. Таким образом, после того, как выбрано некоторое множество горизонтальных ребер, нужно перебрать не более $O(m^2)$ способов установить символы «$\odot$». То есть общее число комбинаций, которое необходимо проверить для перехода из фиксирован-

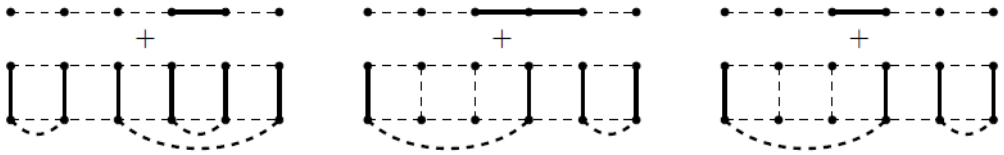

Рис. 4. Первый, второй и третий случаи недопустимых переходов

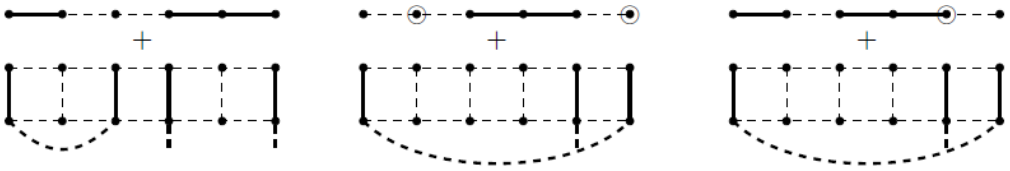

Рис. 5. Четвертый, пятый и шестой случаи недопустимых переходов

ного разреза во все возможные не превосходит $O(m^2 2^m)$. Однако существуют 6 случаев, когда выбранная комбинация недопустима.

Первый случай (рис. 4 слева) состоит в том, что нельзя замыкать какую-либо из цепей в цикл. Второй случай запрещает соединение в одной вершине более чем двух ребер (рис. 4 по центру). Третий случай возникает в связи с тем, что цепь должна получиться гамильтоновой, поэтому незатронутых вершин быть не должно (рис. 4 справа).

Четвертый случай запрещает связывать концы будущей гамильтоновой цепи, если только данный шаг не является последним (рис. 5 слева). На последнем шаге, при $k = n$, может оказаться допустимым соединить концы цепей второго типа, что приведет к образованию гамильтоновой цепи и переходу к финальному разрезу (рис. 6 слева снизу).

Следующие два случая связаны с комбинациями, в которые входит символ начала / конца цепи «⊙». В процессе построения гамильтоновой цепи, ни в каком разрезе не может быть более двух простых цепей второго типа. Недопустимыми оказываются любые попытки породить 3 и более таких цепей (рис. 5 по центру). Помимо этого, недопустимым может оказаться случай, когда символ «⊙» оказывается посередине строящейся цепи (рис. 5 справа).

На последнем шаге (когда $k = n$) требуется выполнить проверку на то, что полученный разрез может перейти в финальный разрез, путем соединения всех входящих в него простых цепей первого и второго типа с последующим запретом (если потребуется) перехода к следующему разрезу. Два примера таких переходов указаны на рис. 6 слева.

Для цилиндра ситуации абсолютно аналогичны за исключением тех случаев, когда оказывается задействованным *заднее ребро*. В таких случаях может потребоваться разворачивать скобки в обратную сторону. Например, пусть $s =$ `(11)()` (рис. 6 по центру) и устанавливается заднее ребро. В этом случае мы должны будем получить разрез `011)(0`, однако, принимая во внимание тот факт, что получившаяся цепь первого типа неориентирована и нет разницы, какой из ее концов считать левым, а какой — правым, мы можем развернуть скобки, которые потеряли свои пары, в обратную сторону: `011()0` — и они по-прежнему соответствуют концам той же самой цепи. Похожий пример изображен на рис. 6 справа. Из разреза `0()0()` получается разрез `)()1(0`; развернув скобки, получим `(()1)0`.

4. МЕТОД МАТРИЦЫ ПЕРЕНОСА

Один из наиболее эффективных способов решения поставленной задачи называется *методом матрицы переноса*, который достаточно детально обсуждался в [5,6]. Общий смысл метода сводится к следующему. Зафиксируем m. Пронумеруем все возможные разрезы для решетки

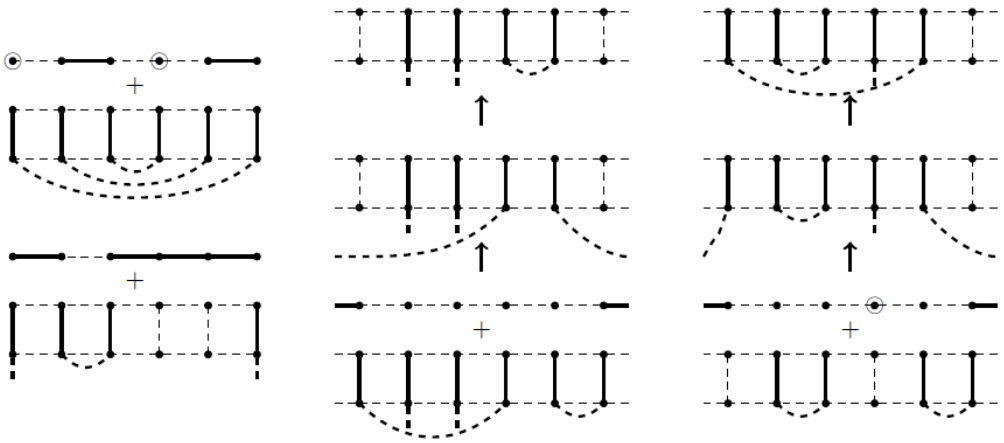

Рис. 6. Слева два примера переходов к финальному разрезу. По центру и справа два примера переходов от одного разреза к другому в случае цилиндра

(цилиндра) числами от 1 до N любым способом ($N \leq M_m$), но так, чтобы начальный разрез имел номер 1, а финальный — номер N (для удобства). Далее построим матрицу T, в которой числа T_{ij} будут равны количеству способов получить из разреза с номером i разрез с номером j. Решением задачи для решетки $P_m \times P_n$ (и цилиндра $C_m \times P_n$) будет число $(T^n)_{1,N}$. Сама матрица T называется *матрицей переноса*.

Однако на практике такой способ решения задачи применим лишь для относительно небольших значений m, когда матрица переноса имеет не слишком большой размер (см. таблицу 1 в разделе «Количество разрезов») и умещается в оперативной памяти ЭВМ. По указанной причине следует отказаться от явного хранения матрицы переноса, а хранить только количество способов получить каждый из возможных разрезов на каждом шаге последовательного построения цепей. Это приводит к методу динамического программирования, который по своей сути является одним из вариантов метода матрицы переноса, но сама матрица в нем явно не фигурирует.

Подобная схема (в словесной форме), где матрица переноса явно не используется, приводится и в работах [7, 3], с той лишь разницей, что авторы указанных работ предлагают иной способ построения и шифрования разрезов графа, более затратный по использованию памяти (о чем подробнее написано в разделе «Количество разрезов»).

Формальная схема метода на псевдокоде приводится на рис. 7. В этой схеме используются множества Q и H. В Q хранятся разрезы и число способов получить их на шаге $k-1$. В H хранятся разрезы и число способов получить их на шаге k. Элементы этих множеств будут иметь вид (s, d), где s — разрез, а d — число способов построить его. Названия Q и H связаны с тем, что первое множество удобно реализовывать в программе в виде очереди, а второе — в виде хэш-таблицы (для быстрой проверки существования в H нужного элемента).

На нулевом слое, когда построение цепи еще не началось, строится начальный разрез одним способом (строка 1). Цикл в строках 2–14 перебирает один слой графа за другим с помощью переменной k. В начале тела этого цикла множество H полагается пустым (строка 3). Для каждого элемента (s, d) множества Q (цикл в строке 4) в строке 5 с помощью функции To(s) для разреза s строятся все разрезы, в которые можно перейти из s (как в разделе «Переход от одного разреза к другим»). Структура данных T в строках 5–6 является списком, а не множеством, то есть в нем могут быть одинаковые элементы. Для каждого полученного из s разреза t (цикл в строке 6) возможны два случая: либо разрез t встречался ранее для данного слоя k, тогда к числу способов его построить e, добавляется число способов d построить разрез

```
IMPROVED–TRANSFER–MATRIX(m, n)
 1  Q ← {(0^m, 1)}
 2  for k = 1 to n
 3        do H ← ∅
 4           for ∀ (s, d) ∈ Q
 5                 do T ← To(s)
 6                    for ∀ t ∈ T
 7                          do if ∃ e : (t, e) ∈ H
 8                                then H ← (H \ {(t, e)}) ∪ {(t, e + d)}
 9                                else H ← H ∪ {(t, d)}
10           Q ← H
11           if ∃ x : (0̄^m, x) ∈ Q
12              then Answer[k] ← x
13                   Q ← Q \ {(0̄^m, x)}
14              else Answer[k] ← 0
```

Рис. 7. Схема метода динамического программирования на псевдокоде

s (строка 8), либо разрез t встречается впервые, тогда он и число способов его построить d добавляются в H. После того, как все элементы множества Q обработаны, оно освобождается и в него записываются элементы полученного множества H (строка 10), чтобы затем начать обработку следующего слоя графа. Но, прежде чем снова перейти к строке 3, необходимо проверить, был ли на текущем шаге k достигнут финальный разрез (строка 11). Если был, то число способов x достигнуть его является числом гамильтоновых цепей на решетке $P_m \times P_k$ (или цилиндре $C_m \times P_k$). В этом случае финальное состояние нужно удалить из множества Q (строка 13), так как иначе будут суммироваться все способы построить гамильтоновы цепи на решетках (или цилиндрах) всех размеров от $m \times 1$ до $m \times n$. Если финальный разрез не был достигнут на шаге k, значит число гамильтоновых цепей в этом случае равно 0 (строка 14). По завершении метода число Answer$[i]$ будет равно числу цепей на решетке (цилиндре) размером $m \times i$ $(i = 1, \ldots, n)$.

Таким образом, достоинством этого подхода является экономия оперативной памяти (вместо матрицы переноса необходимо хранить два множества чисел для соседних слоев), однако недостаток метода заключается в том, что для каждого разреза s необходимо каждый раз снова определять, какие разрезы t из него можно построить.

Указанный недостаток компенсируется с помощью параллельных вычислений. Рассмотрим схему распараллеливания предложенного метода. Пусть все параллельные процессы (их количество будем обозначать P) делятся на две группы. Процессы первой группы будут называться вычислительными, а второй — вспомогательными. Пусть всего A вычислительных процессов. Каждый вычислительный процесс использует B вспомогательных, то есть общее количество процессов $P = A(B + 1)$.

Для начала рассмотрим схему с одним вычислительным процессом $(A = 1)$. Цикл в строке 4 должен пройти по всем элементам (s, d) множества Q, для каждого разреза s он должен построить список To(s). Пусть такой список построен для самого первого разреза из множества Q. В тот момент, пока вычислительный процесс выполняет строки 6–9, B вспомогательных процессов строят списки T для B следующих элементов из множества Q. Таким образом, к моменту, когда вычислительный процесс снова подойдет к строке 5, вспомогательный процесс

передает ему уже построенный список T, а сам вычислительный процесс уже не задерживается в этой строке. На практике получается, что построение списка T — более сложная задача, чем выполнение суммирования в строках 6–9, поэтому данная оптимизация имеет смысл.

Теперь пусть имеется несколько (A) вычислительных процессов, каждый из которых связан с B вспомогательными. Перед выполнением операций строки 4 множество Q разбивается на приблизительно одинаковые части $Q_1, Q_2, \ldots, Q_A$. Каждый вычислительный процесс (с номером i) получает свою часть этого множества Q_i и выполняет дальнейшие вычисления только с ним, а результат записывает в свое локальное множество H_i. В строке 10 все локальные копии множества H_i объединяются в единое множество Q на одном из вычислительных процессов (например, на самом первом), а перед выполнением строки 4 оно снова разбивается на части.

Таким образом, глобальный обмен данными между процессами происходит дважды для каждого слоя графа (для каждого значения $k = 1, \ldots, n$): до начала вычислений и в конце, когда вся информация собирается в единое целое.

Соотношение между A и B следует подбирать эмпирически, так как это зависит от реализации функции построения списка T в строке 5. Чем медленнее реализована эта функция, тем бóльшим должно быть B, но с другой стороны, оно не должно быть настолько большим, чтобы вспомогательные процессы обгоняли вычислительный процесс, с которым связаны. В нашей реализации для решеток шириной $m = 17$ оптимальным оказалось значение $B = 3$. При этом мы запускали программу с $A = 42$, то есть было задействовано 168 ядер.

С ростом m количество разрезов резко увеличивается, поэтому в следующем разделе и в разделе «Оптимизация» будут даны рекомендации для тех случаев, когда необходимо экономить оперативную память и вычислительные ресурсы.

5. КОЛИЧЕСТВО РАЗРЕЗОВ

На практике реальное число разрезов меньше, чем предсказывает формула (1). Это связано с тем, что не все последовательности скобок, нулей и единиц могут оказаться допустимыми разрезами какой-либо гамильтоновой цепи. Например, на решетке шириной $m = 4$ не может получиться разрез `(0)0`. То есть ни одна гамильтонова цепь не может дать такой разрез ни по какому слою решетки. Но даже учет невозможных разрезов не сильно уменьшает их количество, необходимое для хранения. Так, например, в [6] автор показывает, что для решетки шириной $m = 14$ получается 2 110 824 разрезов, а с вычетом невозможных разрезов их становится 2 046 816, что лишь немногим меньше. В той же работе предлагается эвристическое правило, основанное на четности / нечетности позиций концов простых цепей, проходящих через прошлое. Но это правило не отсеивает все невозможные разрезы. Наша программа гарантировано хранит только такие разрезы, которые могу быть получены (что следует непосредственно из самой схемы алгоритма) и показывает, что на самом деле допустимых разрезов 1 810 665. Далее будет представлена техника, позволяющая для решетки с такой шириной получить 906 642 разрезов, а для цилиндров всего 129 522.

К сожалению, в работах других исследователей не предлагается очевидный метод, позволяющий существенно сэкономить оперативную память — учет симметрии разрезов (в случае решетки) и циклических сдвигов разрезов (в случае цилиндра).

Смысл этой оптимизации для решетки состоит в том, чтобы симметричные друг другу разрезы, например, `()1` и `1()`, хранить как один разрез, поскольку число способов построить их одинаково. Любой несимметричный разрез s можно симметрично отразить (переписать слева направо, обернув скобки в обратную сторону) и получить разрез s'. Среди этих двух разрезов *каноническим* будем называть тот, который лексикографически меньше. Например, если считать, что «`(`» < «`)`» < «`0`» < «`1`» из $s =$ `((1)1)` и $s' =$ `(1(1))` каноническим будет

первый. То есть в описании метода динамического программирования на рис. 7 в строке 5, все разрезы из списка T заменяются на соответствующие канонические разрезы.

В случае с цилиндрами ситуация оказывается еще более значительной, так как любые разрезы, получающиеся друг из друга циклическим сдвигом, можно считать одинаковыми. То есть разрезы `1()1`, `()11`, `(11)`, `11()` можно считать одним и тем же разрезом. Среди них можно выбрать канонический (то есть лексикографически наименьший) `()11`, и хранить только его. Аналогично, в строке 5 программы в случае цилиндра все разрезы из списка T заменяются на соответствующие канонические.

С учетом вышесказанного, мы представляем таблицу 1, в которой в первой колонке указана ширина решетки (периметр цилиндра), во второй — число M_m, то есть верхняя граница числа всех разрезов. В третьем и четвертом столбцах указано максимальное число разрезов, которые необходимы для хранения в процессе вычислений на решетках и цилиндрах соответственно (с учетом указанной оптимизации). Следует обратить внимание, насколько значительным оказался учет циклических сдвигов разрезов на цилиндре. Для $m = 18$ их оказалось почти в 9 раз меньше, чем разрезов на решетке.

Важное замечание: на решетках с нечетной шириной число разрезов, необходимых для хранения при четном k и нечетном k различное (например, при $m = 9$ при четном k требуется хранить 3248 разрезов, а при нечетном — 3654). Из этих двух чисел мы выбираем максимальное. В остальных случаях, как и в случае цилиндра, разница между множествами разрезов на четных и нечетных шагах метода отсутствует.

Таблица 1. Количество всех возможных разрезов, возникающих при построении гамильтоновых цепей на решетках и цилиндрах в сравнении с ожидаемым количеством. Значение для цилиндра с периметром $m = 2$ пропущено в силу неоднозначности его определения

m	M_m	**Решетка**	**Цилиндр**
2	6	4	
3	14	8	5
4	38	22	11
5	107	53	22
6	313	158	54
7	926	422	133
8	2768	1129	323
9	8315	3654	926
10	25074	11317	2265

m	M_m	**Решетка**	**Цилиндр**
11	75792	32478	6891
12	229496	100854	16791
13	695723	292026	53518
14	2110826	906642	129522
15	6407758	2635978	427193
16	19458700	8177381	1021993
17	59103511	23832333	3476678
18	179538856	73843106	8204802

Для решеток программа запускалась для всех $m \leq 18$, что позволило определить точное число разрезов. Несомненным преимуществом описанного нами метода является значительная экономия памяти. Для сравнения, в [3] используется менее эффективный способ кодирования разрезов, что приводит к значительным затратам памяти. Как указывает сам автор, имевшиеся вычислительные ресурсы позволяли работать с числом разрезов, не превышающим 10^8. Его метод для решетки шириной 18 порождает большее их количество. В нашем методе критичным является количество вычислительных ядер, тогда как памяти для решеток шириной 18 требуется немногим больше 2 Гб (с учетом дополнительных оптимизаций из раздела «Оптимизация»).

Таким образом, основной результат нашей работы заключается в новой технике вычислений на цилиндре, где удалось добиться значительной экономии вычислительных ресурсов благодаря удачно выбранному способу шифрования разрезов (в виде последовательности скобок, нулей и единиц), а также учету циклических сдвигов.

6. ОПТИМИЗАЦИЯ

Одной лишь оптимизации, связанной с учетом симметрии и циклических сдвигов (из предыдущего раздела) недостаточно, чтобы говорить о значительном повышении эффективности алгоритма, хотя эта оптимизация и является самой важной.

Из анализа работ других авторов следует, что главной проблемой при решении поставленной задачи является нехватка оперативной памяти. Прямолинейное применение метода матрицы переноса является не самым эффективным. В частности, автор работы [6], по-видимому, хранил матрицу переноса явно, поэтому смог представить результаты лишь для решеток шириной до $m = 8$. Авторы [7, 3] не хранили матрицу явно, а пользовались методом динамического программирования, но недостаток их метода состоял в неэффективном способе шифрования разрезов. Поэтому в первой из указанных работ заявлены результаты вычислений на решетках до $P_{15} \times P_{100}$, а во второй — на квадратных решетках до $P_{17} \times P_{17}$.

Особенно много памяти занимает хранение *длинных чисел*. Так, например, количество гамильтоновых цепей на решетке $P_{17} \times P_{100}$ является 262-значным числом, для хранения которого требуется 872 бита. Всего требуется хранить (см. таблицу 1) порядка $24 \cdot 10^6$ таких чисел, причем для двух смежных слоев, что в итоге дает около 5 Гб (только на хранение чисел, без учета затрат на хранение разрезов). Поэтому вместо хранения длинных чисел удобно использовать модулярную арифметику. Программа вычисляет количество гамильтоновых цепей по модулю нескольких простых чисел: 2147483647, 2147483629, 2147483587, ... Затем ответ восстанавливается с помощью китайской теоремы об остатках. Для этого можно использовать любую систему компьютерной алгебры, поддерживающую длинные числа, например, Maple. Для расчетов на $P_{17} \times P_{100}$ требуется чуть меньше 30 последовательных убывающих простых чисел, начинающихся с 2147483647 (самое большое число, вмещающееся в 32 бита со знаком). Далее можно запускать программу сразу для нескольких простых чисел, насколько позволяют вычислительные ресурсы. Например, в случае решетки шириной 18 нам удавалось запускать программу сразу для двух простых чисел, вычисления для $A = 42$ и $B = 3$ (т. е. $P = 168$ ядер) занимают около 5 суток для $n = 100$, а, поскольку для восстановления ответа потребуется немногим меньше 40 остатков, вычисления займут слишком много времени. Для расчетов требуется, чтобы каждому ядру было доступно около 3.5 Гб оперативной памяти (достаточно около 2 Гб, если считать по только по одному модулю). Модулярная арифметика также использовалась в [3].

7. РЕЗУЛЬТАТЫ

Помимо получения точного решения задачи для решеток размером до 17×100 и цилиндров размером до 18×100 (данные находятся в свободном доступе на сайте [9] в разделе о гамильтоновых цепях), мы получили ряд новых рекуррентных соотношений и уточнили универсальную константу связности для гамильтоновых цепей на решетках и цилиндрах.

7.1. Рекуррентные соотношения

Когда параметр m фиксирован, количество гамильтоновых цепей удовлетворяет линейному однородному рекуррентному соотношению с постоянными коэффициентами [5]. Например, для решетки $P_3 \times P_n$, число гамильтоновых цепей (a_n) удовлетворяет соотношению

$$a_1 = 1,\ a_2 = 8,\ a_3 = 20,\ a_4 = 62,\ a_5 = 132,\ a_6 = 336,\ a_7 = 688,\ a_8 = 1578,$$
$$a_n = 3a_{n-1} + 2a_{n-2} - 12a_{n-3} + 4a_{n-4} + 12a_{n-5} - 8a_{n-6}, \quad n \geq 9.$$

Нам удалось получить эти соотношения для решеток шириной до $m = 8$ и цилиндров шириной до $m = 10$. Порядки этих соотношений указаны в таблице 2.

Таблица 2. Порядки рекуррентных соотношений для решеток и цилиндров с фиксированным параметром m. Символом «*» отмечены новые результаты.

m	2	3	4	5	6	7	8	9	10
$P_m \times P_n$	3	6	13	27	70	*183	*431		
$C_m \times P_n$		3	7	12	*24	*50	*93	*252	*452

Автор работы [5] предлагает другие рекуррентные соотношения для решеток шириной $m = 5$ и $m = 6$. Они имеют порядки 36 и 114 соответственно, то есть эти соотношения не являются минимальным. Наши решения гарантировано являются минимальными по порядку в силу способа их нахождения. Все рекуррентные соотношения доступны для скачивания в виде рабочих файлов Maple на нашем сайте [9].

7.2. Универсальные константы связности

В физике для расчета такого термодинамического потенциала, как энтропия, необходимо знать количество возможных конформаций макромолекулы [1]. Пусть Z_{mn} — число возможных конфигураций плотноупакованной макромолекулы (число гамильтоновых цепей). Тогда энтропия определяется как $\mathbb{S} = \ln Z_{mn}$. Таким образом, при невозможности точно вычислять значения Z_{mn} для различных m и n, необходимо знать порядок роста этой величины.

Число Z_{mn} гамильтоновых цепей на решетке с фиксированной шириной и цилиндре с фиксированным периметром имеет порядок роста

$$Z_{mn} = O\big(n^{\gamma} \cdot \mu_m^{mn}\big),$$

при этом $\mu_m > 1$, а γ — целое неотрицательное. Величина μ_m называется универсальной константой связности для гамильтоновых цепей на решетке (цилиндре) шириной (периметром) m. Качественно, константа связности показывает, скольким числом способов гамильтонова цепь проходит через одну вершину. В пределе при увеличении ширины решетки или периметра цилиндра константа связности для решеток и цилиндров стремится к одному и тому же числу μ, то есть существует [10] предел $\lim_{m\to\infty} \mu_m = \mu$. Вычисленное с высокой точностью значение μ, может лечь в основу более точных оценок термодинамических свойств макромолекул. Примечательным является и тот факт, что значение μ не зависит от граничных условий.

В тех случаях, когда известно точное рекуррентное соотношение, значение константы связности можно определить с любой точностью. В остальных случаях требуется пользоваться методами аппроксимации последовательностей [11]. Рассчитав точное количество гамильтоновых цепей для решеток (до 17×100) и цилиндров (до 18×100), удалось отыскать достаточно точные значения универсальных констант связности для $m \le 18$. Также удалось обнаружить, что $\gamma = 0$ во всех случаях, кроме случая решеток с четной стороной, где $\gamma = 1$. Это обстоятельство затрудняет применение аппроксимаций в случае решеток, но упрощает в случае цилиндров (для которых, как мы показали, $\gamma = 0$ при $m \le 18$).

Таблица 3. Значения универсальных констант связности для решеток и цилиндров с различной шириной и периметром. Символы $P_m \times P_\infty$ и $C_m \times P_\infty$ означают, что данные приводятся в пределе при $n \to \infty$

m	$P_m \times P_\infty$	$C_m \times P_\infty$
3	1.259921	1.442250
4	1.262261	1.452371
5	1.330377	1.463479
6	1.334651	1.463285
7	1.366751	1.468410
8	1.369621	1.467337

m	$P_m \times P_\infty$	$C_m \times P_\infty$
9	1.388400	1.470244
10	1.390388	1.469265
11	1.402728	1.471125
12	1.404172	1.470330
13	1.412904	1.471616
14	1.413996	1.470978

m	$P_m \times P_\infty$	$C_m \times P_\infty$
15	1.420502	1.471918
16	1.421354	1.471401
17	1.433530	1.472117
18		1.471692

Для определения величины μ выполнялась экстраполяция данных, приведенных в таблице 3. Как для решеток, так и для цилиндров исходные последовательности делились на 2 подпоследовательности по признаку четности (нечетности) параметра m. К каждой из полученных подпоследовательностей применялось 2 способа ускорения сходимости — таблицы Невилля и ρ-алгоритм Вайна [12]. Достоверными считались те разряды в десятичной записи величины μ, которые присутствовали в итоговых столбцах обоих методов. В результате была получена оценка константы связности

$$\mu = 1.47281 \pm 0.00002.$$

Следует обратить внимание на то, что константы связности μ_m для цилиндров ближе к ожидаемому пределу, чем аналогичные значения для решеток. При этом мы показали, что расчеты для цилиндров выполнять проще, чем для решеток. Оба этих факта показывают, что для более точных оценок константы связности для гамильтоновых цепей следует проводить расчеты на цилиндрах, а не на решетках.

СПИСОК ЛИТЕРАТУРЫ

1. Cloizeaux J., Jannink G. *Polymers in Solution: Their Modelling and Structure*. Oxford: Clarendon Press, 1990.
2. Madras N., Slade G. *The Self-Avoiding Walk*. Boston, MA: Birkhauser, 1993.
3. Jacobsen J.L. Exact enumeration of Hamiltonian circuits, walks, and chains in two and three dimensions. *Journal of Physics A: Mathematical and Theoretical*, 2008.
4. Derrida B. Phenomenological renormalisation of the self avoiding walk in two dimensions. *Journal of Physics A: Mathematical and General*, 1981, vol. 14, no. 1, pp. L5–L9.
5. Faase, F.J. On the number of specific spanning subgraphs of the graphs $G \times P_n$. *Ars Combinatoria*, 1998, vol. 49, pp. 129–154.
6. Kloczkowski A., Jernigan R.L. Transfer matrix method for enumeration and generation of compact self-avoiding walks. I. square lattices. *Journal of Chemical Physics*, 1998, vol. 109, pp. 5134–5146.
7. Jensen. I. Enumeration of compact self-avoiding walks. *Computer Physics Communications*, 2001, vol. 142, pp. 109–113.
8. Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи*. М.: Мир, 1982.
9. FlowProblem [электронный ресурс]. Copyright c 2008–2010. Artem M. Karavaev. URL: `http://www.flowproblem.ru`.
10. Hammersley J.M. Percolation processess. II. The connective constant. *Proceedings of the Cambridge Philosophical Society*, 1957, vol. 53, pp. 642–645.
11. Guttmann A.J. In: *Phase Transitions and critical Phenomena. Vol. 13*. Ed. Domb C., Lebowitz J.L. Eds. London: Academic Press, 1989.
12. Wynn P. On a procrustean technique for the numerical transformation of slowly convergent sequences and series. *Mathematical Proceedings of the Cambridge Philosophical Society*, 1956, vol. 52, pp. 663–671.