Ватутин Э.И.
Юго-Западный государственный университет
305040, г. Курск, ул. 50 лет Октября, 94

Стратегия распределенной диагонализации латинских квадратов

В комбинаторике известным является понятие т.н. «комбинаторного взрыва» – ситуации, при которой с ростом размерности задачи N ее вычислительная сложность резко возрастает и применение ряда методов, работавших до этого для малых размерностей, становится невозможным из-за необходимости огромных вычислительных затрат. На примере решаемых в настоящее время задач, связанных с построением спектров быстровычислимых числовых характеристик диагональных латинских квадратов (ДЛК) [1], данная ситуация для диагонализации (построения всех возможных ДЛК, изоморфных заданному латинскому квадрату (ЛК)) и поквадратного обхода окрестностей [2–4] начинает проявляться начиная с размерности N=13. Поквадратный обход окрестностей при этом становится невозможен (приблизительная оценка необходимых вычислительных затрат – около 80 лет в проекте добровольных вычислений [3]), диагонализацию возможно реализовать по частям путем разработки специализированной распределенной программной реализации указанной процедуры.

В решаемых в настоящее время задачах, связанных с построением спектров, диагонализация применяется совместно с анализом окрестностей и позволяет как увеличить мощность результирующего спектра, так и сдвинуть его верхнюю и нижнюю границы (супремум и инфимум) в ситуациях, когда спектр близок в пределу и просто обходом окрестностей границы уже не сдвигаются. Для размерности N=12 диагонализация требует от нескольких часов до нескольких десятков часов на квадрат (для ДЛК с максимально известным числом трансверсалей (198 144) – более недели в 1 поток на Core i7 4770). Для размерности N=13 самые «легкие» по числу трансверсалей квадраты будут диагонализироваться несколько часов, «тяжелые» – до нескольких месяцев, что неприемлемо.

Схематично процедура диагонализации выглядит следующим образом: из множества трансверсалей ЛК находятся подходящие пары трансверсалей, по которым производится ряд целенаправленных перестановок строк и столбцов исходного ЛК с целью получения результирующего ДЛК, что на много порядков быстрее построения перестановок всех возможных пар строк и столбцов. Далеко не все пары трансверсалей подходят для выполнения преобразования, однако проверять необходимо все, при этом пары $\left(T_i, T_j\right)$ и $\left(T_j, T_i\right)$, $i \neq j$, проверять дважды смысла нет, что при изображении соответствия трансверсалей в виде бинарной матрицы (0 – не подходят для диагонализации, 1 – подходят) необходимо обойти не всю матрицу, а ее половину – верхнюю или нижнюю треугольную подматрицу (для определенности обходится верхняя). В программировании подобный обход применяется очень часто в ряде алгоритмов, а соответствующий псевдокод выглядит следующим образом:

```
for (int i = 0; i < NT; i++)
  for (int j = i+1; j < NT; j++)
    ...
```

Простейшим способом распараллеливания является разбиение по внешнему циклу (по i), однако при этом возникает проблема, связанная с тем, что в таком случае в каждом расчетном задании (англ. Work Unit, сокр. WU) потребуется хранить полное множество трансверсалей (для порядка N=13 их уже бывает более миллиона, для б**о**льших порядков их количество может быть сильно б**о**льшим[1]), что потребует минимум сотен МБ – единиц–десятков ГБ оперативной памяти на WU. Более подходящей представляется следующая стратегия распараллеливания: матрица разбивается на квадраты заданного размера $K \times K$, в рамках одного WU производится обработка одного квадрата (см. рис., на нем изображен один из самых «легких» ДЛК с $N_T \approx 43$ тыс. трансверсалей и разбиение на WU по 20 тыс. × 20 тыс. трансверсалей в квадрате, всего 6 WU).

[1] https://oeis.org/A287644

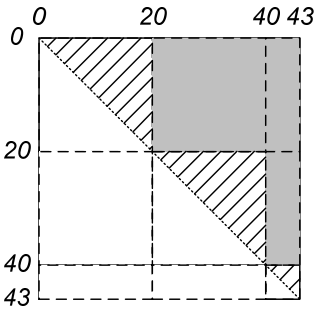

Рис. Схематичное изображение разбиения исходной задачи на подзадачи

При программной реализации в соответствии с данной стратегией возникает ряд особенностей.

1. Необходимость построения разбиения на квадраты, некоторые из которых частично попадают под главную диагональ матрицы – в составе соответствующих WU необходима проверка и отсечение пар трансверсалей из нижней треугольной подматрицы (обрабатываемая часть выделена штриховкой, полностью обрабатываемые квадраты/прямоугольники выделены серым).
2. Наличие маленьких прямоугольников и квадратов по краям (им будут соответствовать более короткие WU с рядом дополнительных проверок, чтобы не выйти за пределы анализируемой области вправо и вниз).
3. Необходимость хранения множества трансверсалей по частям (точнее, в двух частях в диапазонах $[x_{\min}, x_{\max}]$ и $[y_{\min}, y_{\max}]$, где $x_{\min}, y_{\min}$ – координаты верхнего левого угла анализируемого квадрата/прямоугольника, $x_{\max} = \min(x_{\min} + K, N_T)$, $y_{\max} = \min(y_{\min} + K, N_T)$, диапазоны иногда могут пересекаться (в данной задаче – совпадать полностью, в перспективе в ряде других практических применений в задачах по эвристической работе со спектрами – частично пересекаться).

Неоспоримым плюсом подхода является его универсальность: на квадраты можно разбить как «легкие», так и «тяжелые» ДЛК, в последнем случае возрастет лишь число подзадач (для ДЛК порядка 13 с максимально известным числом трансверсалей – около 2500 WU при текущем способе разбиения), при этом возможно управлять средним временем счета WU и затратами памяти путем изменения размера квадрата K (критичным в данной задаче является именно время счета, которое при организации расчета в проекте грид-вычислений не желательно делать как сильно маленьким (минуты и меньше), так и сильно большим (десятки часов)).

Кроме описанного выше распараллеливание также допустимо по парастрофическим преобразованиям (по 3 на ДЛК из 6 возможных за исключением транспонирования, т.к. оно дает ДЛК из того же главного класса и не представляет интереса) и т.н. slice'ам для каждого из них (также 3, итого $3 \times 3 = 9$ комбинаций).

В соответствии с вышеизложенным была разработана программная реализация, в которой для 200 «легких» ДЛК было сформировано около 10 тыс. WU, которые были обсчитаны в проекте добровольных распределенных вычислений RakeSearch[2]. При $K = 20 \cdot 10^3$ среднее время счета составило 7–8 минут на WU, максимальное – 20 минут. В перспективе дальнейших исследований планируется увеличение значения K с целью увеличения времени счета WU и выполнение диагонализации всех ДЛК, входящих в состав спектров (на данный момент спектр числа диагональных трансверсалей включает в своем составе 12751 элемент, числа трансверсалей общего вида – 75265 элементов, числа интеркалятов – 152 элемента).

При организации подобного распределенного расчета в грид есть еще одна особенность, связанная с построением необходимого множества трансверсалей: его можно однократно получить в процессе генерации WU, а затем передать на клиент, а можно получить первым этапом при обсчете WU. В данном эксперименте выбран второй вариант (на генерацию нужных трансверсалей требуется 1–2 секунды в каждом WU, при этом происходит экономия нескольких сотен КБ исходных данных WU. В перспективе с ростом размерности задачи N вполне вероятна

[2] https://rake.boincfast.ru/rakesearch/

ситуация, в которой получение нужных подмножеств трансверсалей подобным образом в каждом WU станет неприемлемо долгим и потребуется переход к первому варианту.

Литература

1. Keedwell A.D., Dénes J. Latin Squares and their Applications. Elsevier, 2015. 438 p. DOI: 10.1016/C2014-0-03412-0.

2. Ватутин Э.И., Титов В.С. и др. Оценка мощностей спектров быстровычислимых числовых характеристик диагональных латинских квадратов порядков N>9 // Наука и образование в развитии промышленной, социальной и экономической сфер регионов России. Муром, 2022. С. 314–315.

3. Ватутин Э.И., Титов В.С. и др. Эвристический метод построения аппроксимаций спектров числовых характеристик диагональных латинских квадратов // Интеллектуальные информационные системы: тенденции, проблемы, перспективы (ИИС – 2022). Курск: изд-во ЮЗГУ, 2022. С. 35–41.

4. Ватутин Э.И., Никитина Н.Н. и др. Методы построения спектров быстровычислимых числовых характеристик диагональных латинских квадратов // Национальный суперкомпьютерный форум (НСКФ – 2022). Принята к опубликованию.