ОБ АЛГОРИТМЕ СТРОГОГО КОНСЕНСУСНОГО РАНЖИРОВАНИЯ

В. А. Антонюк[1]

Рассматривается комбинаторный подход к задаче отыскания оптимального строгого консенсусного ранжирования для заданной совокупности нестрогих упорядочений альтернатив. Вводится понятие «облегчённой» матрицы потерь (антисимметричной), позволяющей сделать процесс (и результат) оптимизации более простым и наглядным. Сформулированы процедуры поиска оптимальных строгих ранжирований (в т. ч. — всех множественных) в различных ситуациях.

Ключевые слова: ранжирования, медиана Кемени, антисимметричные матрицы, язык Julia, OpenCL.

1. Введение. Процесс ранжирования [1, стр. 16] предполагает сопоставление независимыми акторами (они называются обычно экспертами) набору некоторых объектов, событий, явлений и пр. в количестве N — обобщённо именуемых альтернативами — какого-то своего порядка (порядка предпочтительности).

Представлением результата такого ранжирования (он здесь тоже будет именоваться ранжированием, поскольку это слово может в языке обозначать как сам процесс, так и его результат) считают либо вектор рангов, где перечислены ранги, сопоставленные альтернативам, — от 1 до N, либо вектор порядка, где перечислены сами альтернативы — в порядке от наилучшей до самой худшей.

Если эксперт сопоставляет разным альтернативам различные значения рангов, то принято говорить о полном ранжировании (complete ranking) и линейном (или полном) упорядочении альтернатив (linear/full ordering). Если же какие-нибудь альтернативы для эксперта представляются неразличимыми, т. е., не получается сопоставить им различные ранги или порядок следования, то говорят о связанном ранжировании (tied ranking) или слабом упорядочении (weak ordering). Набор из K ранжирований, предоставленных K экспертами, часто называют профилем индивидуальных предпочтений или просто профилем.

Цель анализа исходных ранжирований альтернатив (профиля) — получение некоторого консенсусного ранжирования, минимизирущего суммарное (по всем экспертам) «расхождение» между предпочтениями экспертов (исходными ранжированиями) и получаемым консенсусом (результирующим ранжированием). В качестве расстояния между ранжированиями часто используется т. н. расстояние Кемени, при этом найденное консенсусное ранжирование именуется медианой Кемени [1, стр. 73]; оно имеет вид перестановки (или перестановок, если их несколько) альтернатив в нужном порядке (или же их условных номеров).

В работе показано, что для отыскания оптимальных консенсусных (строгих[2]) ранжирований удобнее работать не со стандартной матрицей потерь, а с т. н. «облегчённой». Её элементарный анализ позволяет установить, с какой ситуацией приходится иметь дело, каково примерно ожидаемое количество оптимальных перестановок. Предложен простой эвристический алгоритм поиска оптимальных консенсусных ранжирований. Для практической работы с ранжированиями и реализации предложенного алгоритма был использован язык Julia [2] — по причинам, которые станут понятны в дальнейшем.

2. «Облегчённая» матрица потерь (антисимметричная). Матрица потерь (также profile matrix [3, стр. 2928–2929]) является удобным способом представления информации о совокупности ранжирований альтернатив экспертами и, обладая полезными свойствами, часто используется при отыскании медианы Кемени. Цитируя [1, стр. 78]: *"Задача отыскания медианы Кемени для ранжирований может быть сформулирована как задача отыскания такого упорядочения альтернатив, а следовательно, строк и столбцов матрицы потерь, чтобы сумма её элементов, расположенных над диагональю, была минимальна."*

Если обратиться к определению матрицы потерь [1, стр. 77], [3, стр. 2929], то выяснится, что каждое из значений в ней (за исключением нулей на главной диагонали) включает в себя и количество исходных ранжирований (экспертов), поскольку там суммируются модули элементов каждой из матриц отношений[3] за вычетом значения 1: $r_{ij} = \sum_{k=1}^{K} |a_{ij}^{(k)} - 1|$, $i \neq j$, что реально просто отображает область значений $\{+1, 0, -1\}$ внедиагональных элементов всех K матриц $\|a_{ij}^{(k)}\|$ в диапазон $\{0, 1, 2\}$ перед суммированием.

---

[1] Московский государственный университет им. М. В. Ломоносова, Физический факультет. ГСП-1, Ленинские горы, д. 1, стр. 2, 119991, Москва; доцент, e-mail: antonyuk@physics.msu.ru

[2] В настоящей статье исходные ранжирования считаются полными или связанными, но результирующие ранжирования — всегда полными; для заголовка же статьи вместо слова «полное» выбрано слово «строгое» — как менее дезориентирующее.

[3] Для элементов a_{ij} матрицы отношений (для ранжирования частичного или линейного порядка) величина $+1$ означает предпочтительность альтернативы i перед альтернативой j, 0 — равноценность альтернатив i, j (или равенство $i = j$), -1 — наоборот, предпочтительность альтернативы j перед альтернативой i.

Однако для минимизации суммы наддиагональных элементов (производимой путём «перемещения» внедиагональных значений в пределах внедиагональных же областей) не имеет никакого значения, что ко всем внедиагональным добавлена одна и та же величина. Можно вычесть количество ранжирований из этих элементов, тогда каждый элемент этой преобразованной матрицы потерь станет просто суммой соответствующих элементов матриц отношений — но с противоположным знаком. А так как все матрицы отношений для ранжирований (частичного или линейного порядка) антисимметричны, их сумма тоже будет антисимметричной[4]; далее будем называть её (условно) «облегчённой» матрицей потерь[5].

Расположение в подобной матрице отрицательных значений только над главной диагональю означает, во-первых, что все альтернативы расположены в таком порядке, что каждая из них предпочтительнее (или как минимум равноценна) каждой из последующих, а, во-вторых, что найдена такая перестановка альтернатив, при которой сумма наддиагональных элементов матрицы минимальна.

Таким образом, ясно, что традиционная неотрицательность элементов стандартной матрицы потерь — это следствие метрического подхода к задаче поиска консенсусных ранжирований, потому что вполне естественно работать с неотрицательными величинами, трактуя их как некоторые «расстояния». Если же взглянуть на это немного под другим углом, с более «комбинаторной» точки зрения, то окажется, что и отрицательность значений имеет свои преимущества.

Поскольку речь идёт об антисимметричной матрице, минимально возможное (хотя бы теоретически) значение суммы наддиагональных элементов известно заранее: оно равно сумме всех её отрицательных элементов. Такое значение может быть достигнуто, если все эти отрицательные элементы удастся сосредоточить в верхнем треугольнике матрицы, и меньше эту сумму сделать уже никак нельзя. Но удастся ли расположить все отрицательные значения таким образом — заранее совершенно неочевидно. Кроме того, нас будет интересовать не какое-то одно решение этой задачи, а все возможные решения, поскольку часто в данной задаче они многочисленны. При этом вопрос, как следует поступать с этими множественными решениями для получения одного итогового консенсусного ранжирования, здесь не рассматривается.

Поэтому фактически можно считать, что в настоящей работе анализируется такая комбинаторная

Задача. *Для заданной антисимметричной матрицы найти такие (одновременные) перестановки её строк и столбцов, чтобы после любой из них сумма всех наддиагональных элементов преобразованной матрицы была минимально возможной.*

Компактная запись одновременной перестановки строк и столбцов квадратной матрицы R выглядит так: $\Pi^\top R\,\Pi$, где Π — некоторая матрица перестановки[6]. К такому же виду приводится и последовательность подобных преобразований, поскольку

$$\Pi_n^\top(\dots(\Pi_2^\top(\Pi_1^\top R\,\Pi_1)\,\Pi_2)\dots)\,\Pi_n = (\Pi_1\Pi_2\dots\Pi_n)^\top R\,(\Pi_1\Pi_2\dots\Pi_n),$$

а произведение нескольких матриц перестановки есть тоже некоторая матрица перестановки, т. к. класс этих матриц замкнут относительно операции умножения матриц [4, стр. 36]. Легко также убедиться, что после подобного преобразования антисимметричная матрица $A : A^\top = -A$ таковой и остаётся:

$$(\Pi^\top A\,\Pi)^\top = \Pi^\top A^\top(\Pi^\top)^\top = \Pi^\top(-A)\,\Pi = -(\Pi^\top A\,\Pi).$$

3. Преимущества использования антисимметричных матриц. Переход от уже традиционных матриц потерь к «облегчённым» (антисимметричным) не влияет на «перестановочную» оптимизацию суммарного значения в верхнем треугольнике матрицы, однако позволяет чётко разделить три возможные при этом ситуации.

I. *В антисимметричной матрице нет внедиагональных нулевых элементов и все отрицательные элементы могут быть сгруппированы над главной диагональю матрицы.* Отметим, что в этом случае число отрицательных элементов в столбцах линейно нарастает (от 0 до $N-1$), а в строках их число линейно убывает (от $N-1$ до 0).

II. *В антисимметричной матрице есть нулевые элементы вне главной диагонали и все отрицательные элементы могут быть сгруппированы над этой диагональю.* В таком случае число отрицательных элементов в столбцах (строках) не превышает указанных выше для ситуации I значений линейного нарастания

[4] Если $A_i^\top = -A_i,\ i = 1,\dots,n$, то $(A_1+A_2+\dots+A_n)^\top = A_1^\top + A_2^\top + \dots + A_n^\top = -A_1 - A_2 - \dots - A_n = -(A_1+A_2+\dots+A_n)$.

[5] В «облегчённой» матрице потерь любую её строку (соответствующую некоторой альтернативе) надо трактовать так: положительными значениями в каких-то столбцах указаны более предпочтительные альтернативы (соответствующие этим столбцам), отрицательными — менее предпочтительные, нулевыми — равноценные (или сама альтернатива).

[6] При домножении R на Π справа — переставляются столбцы R, а при домножении R слева на $\Pi^\top$ (транспонированную матрицу перестановки Π) — точно так же переставляются строки матрицы R.

(убывания), потому что в противном случае это означало бы принципиальную невозможность размещения всех отрицательных элементов над диагональю матрицы. Однако здесь — в отличие от предыдущей ситуации — появляются возможности для такого обмена местами каких-то строк/столбцов, который не нарушает при этом «группировку» отрицательных элементов над диагональю, что означает появление некоторого количества перестановок, дающих требуемый результат.

III. *Отрицательные элементы не удаётся полностью сосредоточить над главной диагональю.* Это значит, что «идеальные» значения линейного нарастания (убывания) количеств отрицательных элементов в столбцах (строках) матрицы при этом где-то обязательно превышены.

Каждая из упомянутых здесь трёх ситуаций иллюстрируется на рис. 1 диаграммами[7] со строчными (красные ромбики) и столбцовыми (синие квадратики) количествами отрицательных значений трёх конкретных «облегчённых» матриц потерь (подробнее о самих матрицах — ниже). Розовые и голубые фоновые треугольники показывают допустимые области расположения отсортированных должным образом количеств отрицательных элементов, при которых в принципе возможно сосредоточить их над главной диагональю. Это, конечно, не значит, что подобное размещение отрицательных величин будет обязательно возможно, так как данные ограничения на них являются необходимыми, но не достаточными.

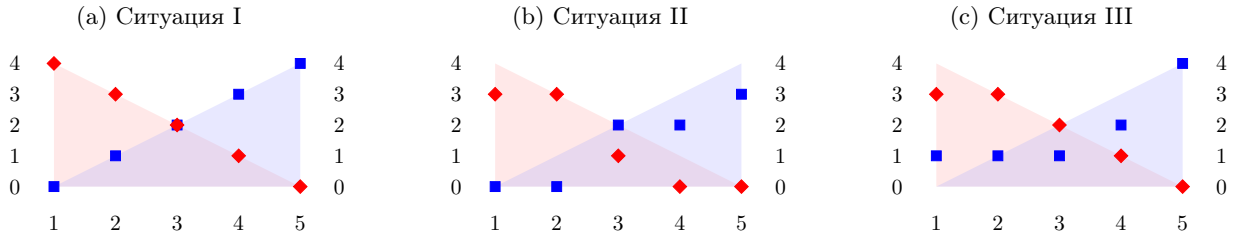

Рис. 1. Строчные/столбцовые отрицательные значения в антисимметричных матрицах (см. рис. 2, 4, 3)

Для ситуаций I и II никакого превышения количеств отрицательных значений не наблюдается (красные ромбики не выходят за пределы розовых треугольников, а синие квадратики — за пределы голубых), в то время как в ситуации III все столбцовые количества отрицательных значений больше нуля, значит в матрице нет столбца без отрицательных величин: в любом столбце всегда будет отрицательный элемент.

Кстати, стоило бы отметить, что до тех пор, пока все отрицательные элементы антисимметричной матрицы можно расположить над главной диагональю, не имеет никакого значения, чему они в точности равны. Но в случае принципиальной невозможности сосредоточить их все над диагональю, значения остающихся под диагональю уже весьма существенны: если среди них есть самые большие (по модулю) отрицательные, то стоит озаботиться именно их перемещением в верхний треугольник, тем самым уменьшая наддиагональную сумму.

Кроме того, для антисимметричных «облегчённых» матриц потерь становятся более заметными существенные различия нечётного и чётного количества экспертов: в первом случае в «облегчённой» матрице потерь никогда не будет нулевых значений за пределами главной диагонали, а во втором они возможны — в случае некоторого несовпадения мнений экспертов по поводу отдельных альтернатив.

4. Антисимметричные матрицы и минимизация суммы их наддиагональных элементов. Начнём с некоторых несложных, но довольно полезных утверждений.

Теорема 1. *Для антисимметричной матрицы, у которой все элементы над главной диагональю отрицательны, не существует одновременной перестановки её строк и столбцов (кроме тождественной), после которой все наддиагональные элементы снова будут отрицательными.*

Доказательство. Будем доказывать «от противного». Отметим прежде всего, что отрицательность всех наддиагональных элементов матрицы означает, что все элементы под диагональю — положительные, так как после подобных перестановок матрица всё равно остаётся антисимметричной.

Предположим противное: существует ещё одна перестановка строк и столбцов матрицы, при которой все наддиагональные элементы матрицы опять отрицательны. Покажем, что это вступает в противоречие с тем фактом, что ни один из столбцов рассматриваемой матрицы не может быть переставлен на другое

[7] На всех трёх диаграммах по горизонтали указаны условные номера строк/столбцов «облегчённых» матриц потерь, а по вертикали — количества отрицательных элементов в них (после сортировки строчных количеств — по убыванию и столбцовых — по возрастанию). Цель диаграмм — оценка возможности «правильного» расположения отрицательных элементов.

место[8], иначе под диагональю обязательно окажется хотя бы один отрицательный элемент, что будет противоречить либо тому, что матрица остаётся антисимметричной, либо тому, что после перестановки все наддиагональные элементы опять отрицательны.

Действительно, последний столбец (где имеется $N-1$ отрицательное значение) не может оказаться на месте предпоследнего (или более левого) столбца, поскольку тогда над диагональю в столбце не более $N-2$ мест, а отрицательных значений в последнем столбце — $N-1$. Точно так же предпоследний столбец не может попасть на место последнего, поскольку тогда некуда будет поставить последний столбец, а перемещение предпоследнего столбца левее тоже невозможно — по соображениям, только что изложенным для последнего столбца: мест над диагональю будет меньше, чем отрицательных значений в столбце. Применяя аналогичные рассуждения к каждому из более левых столбцов матрицы, мы получим, что ни один из столбцов не может быть перемещён на другое место, т. е., другой перестановки реально получить не удаётся, что противоречит предположению. Значит, это предположение неверно и столбцы матрицы нельзя переставить *по-другому*, чтобы **все** её наддиагональные элементы снова были отрицательными. □

Теорема 2. *Если все столбцовые*[9] *количества отрицательных величин антисимметричной матрицы* $N \times N$ $(N > 1)$ *различны, то одновременными перестановками её строк и столбцов матрица может быть приведена к состоянию, когда все отрицательные значения находятся в её верхнем треугольнике.*

Доказательство. Если все количества отрицательных величин в разных столбцах разные, значит они принимают все целые значения от 0 (минимально возможное количество отрицательных величин столбца) до $N-1$ включительно (максимально возможное, т. к. один элемент, расположенный на главной диагонали матрицы, всегда равен нулю). Всего отрицательных величин в матрице в этом случае будет $0+1+\ldots+(N-2)+(N-1) = N(N-1)/2$, т. е., столько же, сколько и наддиагональных элементов матрицы.

Приведём матрицу с помощью перестановок её строк и столбцов к виду, когда столбцовые количества отрицательных величин расположены по возрастанию (слева направо)[10]. Рассмотрим последний столбец матрицы. В нём должно быть $N-1$ отрицательное значение и одно нулевое, расположенное в последней строке. Значит, все отрицательные значения последнего столбца расположены выше главной диагонали, т. е., в верхнем треугольнике. При этом вся последняя строка, за исключением последнего элемента в ней, состоит из положительных значений — в силу антисимметричности матрицы: элементы этой строки противоположны по знаку элементам последнего столбца. Аналогичным образом все отрицательные значения предпоследнего столбца расположены в начальных $N-2$ строках, потому что оставшиеся два элемента столбца неотрицательны: предпоследний (на главной диагонали) равен нулю, а последний — положителен (как уже выяснилось). При этом в предпоследней строке все значения, за исключением последних двух, положительны — опять из-за антисимметричности матрицы. Продолжая рассматривать один за другим остальные столбцы и используя аналогичные соображения, мы приходим к выводу, что все различные столбцовые количества отрицательных значений расположены над главной диагональю матрицы (в первом столбце отрицательных значений нет), т. е., в её верхнем треугольнике. Строчные же количества отрицательных величин окажутся расположенными в убывающем порядке, поскольку все значения над главной диагональю отрицательны. □

Хотя эта теорема сформулирована для столбцовых количеств отрицательных элементов антисимметричной матрицы (либо может быть сформулирована только для строчных количеств), из её доказательства видно, что в любом случае оба набора количеств будут удовлетворять условиям теоремы — из-за антисимметричности матрицы. Если же вспомнить, что анализируемая здесь задача для таких матриц — просто переформулированная задача поиска медианы Кемени по традиционной матрице потерь, то мы получаем простое

Следствие. *Если в «облегчённой» матрице потерь некоторого набора ранжирований все столбцовые (строчные) количества отрицательных величин различны, то медиана Кемени данного профиля в этом случае будет единственна и её можно определить перестановкой номеров столбцов по возрастанию в них (или перестановкой номеров строк по убыванию в них) количеств отрицательных величин.*

[8] Достаточно говорить только о столбцах, поскольку используемая операция сохраняет все элементы столбца неизменными, лишь изменяя их расположение в нём, хотя с таким же успехом можно было рассматривать и строки, ибо они обладают теми же свойствами.

[9] Совершенно аналогично можно сформулировать этот результат и для строковых количеств отрицательных величин.

[10] Это всегда можно сделать не более чем $N-1$ проходом по столбцам с перестановкой (в стиле сортировки «пузырьком») соседних столбцов и соответствующих строк, если столбцы оказались расположены не надлежащим образом.

Подобная ситуация встречается не очень часто, но её примеры имеются в литературе ([1, стр. 75], [5, стр. 23]). Рассмотрим первый из них: пятью экспертами указаны такие ранжирования пяти альтернатив a_1, a_2, a_3, a_4, a_5: 1) a_1, a_2, a_3, a_4, a_5; 2) a_2, a_5, a_1, a_4, a_3; 3) a_3, a_2, a_1, a_4, a_5; 4) a_1, a_5, a_3, a_2, a_4; 5) a_4, a_3, a_1, a_5, a_2.

На рис. 2 приведены: исходная «облегчённая» матрица потерь для этих ранжирований (вместе с обозначениями альтернатив); она же, но с количествами отрицательных элементов; результирующее состояние «облегчённой» матрицы потерь после «оптимальной» перестановки альтернатив.

(a) Т. н. «облегчённая» матрица потерь и обозначения альтернатив

	a_1	a_2	a_3	a_4	a_5
a_1	0	−1	−1	−3	−3
a_2	1	0	1	−3	−1
a_3	1	−1	0	−1	−1
a_4	3	3	1	0	−1
a_5	3	1	1	1	0

(b) Количества отрицательных элементов в её столбцах и строках

	0	2	1	3	4
4	0	−1	−1	−3	−3
2	1	0	1	−3	−1
3	1	−1	0	−1	−1
1	3	3	1	0	−1
0	3	1	1	1	0

(c) Перестановка `[1,3,2,4,5]`, применённая к матрице потерь

	1	3	2	4	5
1	0	−1	−1	−3	−3
3	1	0	−1	−1	−1
2	1	1	0	−3	−1
4	3	1	3	0	−1
5	3	1	1	1	0

Рис. 2. Пример из книги Б. Литвака [1, стр. 75] (ситуация I)

В соответствии с теоремой 2, как только обнаруживается различие всех столбцовых или же строчных количеств отрицательных элементов «облегчённой» матрицы потерь, задачу поиска медианы Кемени можно считать решённой: остаётся лишь узнать порядок следования номеров альтернатив после сортировки какого-либо из количеств отрицательных элементов. Поэтому совершенно аналогичная ситуация[11] из [5] может быть проанализирована и разрешена существенно проще и быстрее, чем это сделано в статье.

Действия, необходимые для выяснения, имеет ли место для заданного профиля предпочтений столь простой случай, здесь записаны на языке Julia, но следует знать, что они во многих синтаксических аспектах не сложнее записи кода в MATLAB (синтаксис Julia был сознательно сделан авторами языка близким к его синтаксису).

```
col_negs(R::Matrix) = [sum(R[:,j].<0) for j=1:size(R)[2]]
row_negs(R::Matrix) = [sum(R[i,:].<0) for i=1:size(R)[1]]

straight_cols(R) = all(sort(col_negs(R)).==Vector(0:size(R)[2]-1))
straight_rows(R) = all(sort(row_negs(R)).==Vector(0:size(R)[1]-1))
is_straight(R) = straight_cols(R) && straight_rows(R)
```

Все эти строки — однострочные определения функций в Julia, выполняющих нужные действия над передаваемой им матрицей `R`: подсчёт столбцовых/строчных количеств отрицательных элементов, проверку линейного изменения этих отсортированных количеств (по отдельности и вместе).

(a) Т. н. «облегчённая» матрица потерь и обозначения альтернатив

	a_1	a_2	a_3	a_4	a_5
a_1	0	−2	1	1	0
a_2	2	0	3	1	1
a_3	−1	−3	0	1	−2
a_4	−1	−1	−1	0	1
a_5	0	−1	2	−1	0

(b) Количества отрицательных элементов в её столбцах и строках

	2	4	1	1	1
1	0	−2	1	1	0
0	2	0	3	1	1
3	−1	−3	0	1	−2
3	−1	−1	−1	0	1
2	0	−1	2	−1	0

(c) Перестановка `[3,5,4,1,2]`, применённая к матрице потерь

	3	5	4	1	2
3	0	−2	1	−1	−3
5	2	0	−1	0	−1
4	−1	1	0	−1	−1
1	1	0	1	0	−2
2	3	1	1	2	0

Рис. 3. Пример из книги Б. Литвака [1, стр. 86–88] (ситуация III)

[11] Исходные ранжирования разбираемого там примера: $c \succ e \succ d \succ b \succ a$, $c \succ e \succ d \succ b \succ a$, $c \succ e \succ b \succ d \succ a$, $c \succ e \succ d \succ b \succ a$, $e \succ c \succ d \succ a \succ b$, $e \succ d \succ c \succ a \succ b$, $e \succ c \succ d \succ a \succ b$, $b \succ e \succ d \succ c \succ a$, $d \succ b \succ e \succ c \succ a$.

Иллюстрация наиболее часто встречающейся ситуации III (невозможности размещения над главной диагональю всех отрицательных элементов) приведена на рис. 3 (пример[12] взят из [1, стр. 86–88]). Обнаружить подобную ситуацию помогут такие фрагменты Julia-кода:

```
crossing_cols(R) = any(sort(col_negs(R)).>Vector(0:size(R)[2]-1))
crossing_rows(R) = any(sort(row_negs(R)).>Vector(0:size(R)[1]-1))
is_crossing(R) = crossing_cols(R) || crossing_rows(R)
```

Здесь опять присутствуют однострочные определения минимально необходимых вспомогательных функций. Из-за того, что часть отрицательных элементов неизбежно останется под главной диагональю (на это указывают ненулевые столбцовые суммы), теоретически возможный минимум суммы наддиагональных элементов (его можно найти с помощью функции `sum_negs(R) = sum(R[R.<0])`) не может быть достигнут.

Ситуация II иллюстрируется рис. 4: наличие дополнительных нулевых элементов антисимметричной матрицы уменьшает число отрицательных (превышение становится менее вероятным), однако появляется возможность дополнительных перестановок получаемого результата, приводящая к множественности медиан Кемени (в примере для профиля 5 альтернатив — $ABCDE, BDAEC, BAECD, ADBCE$ — 6 консенсусных ранжирований: `[1,2,3,4,5]`, `[1,2,4,3,5]`, `[1,2,4,5,3]`, `[2,1,3,4,5]`, `[2,1,4,3,5]`, `[2,1,4,5,3]`).

(a) Т. н. «облегчённая» матрица потерь и обозначения альтернатив

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 0 & -4 & -2 & -4 \\ 0 & 0 & -4 & -2 & -4 \\ 4 & 4 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & -2 \\ 4 & 4 & 0 & 2 & 0 \end{pmatrix} \end{array}$$

(b) Количества отрицательных элементов в её столбцах и строках

$$\begin{array}{c} \\ 3 \\ 3 \\ 0 \\ 1 \\ 0 \end{array} \begin{array}{c} \begin{array}{ccccc} 0 & 0 & 2 & 2 & 3 \end{array} \\ \begin{pmatrix} 0 & 0 & -4 & -2 & -4 \\ 0 & 0 & -4 & -2 & -4 \\ 4 & 4 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & -2 \\ 4 & 4 & 0 & 2 & 0 \end{pmatrix} \end{array}$$

(c) Перестановка `[2,1,4,5,3]`, применённая к матрице потерь

$$\begin{array}{c} \\ 2 \\ 1 \\ 4 \\ 5 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccccc} 2 & 1 & 4 & 5 & 3 \end{array} \\ \begin{pmatrix} 0 & 0 & -2 & -4 & -4 \\ 0 & 0 & -2 & -4 & -4 \\ 2 & 2 & 0 & -2 & 0 \\ 4 & 4 & 2 & 0 & 0 \\ 4 & 4 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Рис. 4. Пример из пакета `RankAggreg` [6] для языка R

Таким образом, количество отрицательных элементов в столбцах и строках матрицы, подвергаемой рассматриваемым здесь преобразованиям (одновременным перестановкам строк и столбцов), — это важные инварианты этих строк и столбцов при таких преобразованиях[13]. Полезно также рассмотреть чуть подробнее, что может происходить при этом с отдельными элементами вообще. Т. к. порядок следования столбцов и строк в любой матрице потерь определяется общим порядком расстановки альтернатив, будем иногда говорить не о столбцах и строках матриц отдельно, а о позиционировании этих строк и столбцов, определяемом некоторой перестановкой их изначального расположения.

5. Перестановка двух произвольных позиций. Действие этой «базовой» для рассматриваемых здесь матриц операции проиллюстрировано на рис. 5. Так как при обмене i-го и j-го столбцов и i-й и j-й строк часть элементов матрицы переставляется один раз, а те, что находятся на пересечении строк и столбцов, — дважды, получается, что элементы строк/столбцов вне области «перекрытия» останутся в «своей» треугольной половине, в пределах самой области «перекрытия» (за исключением «угловых» элементов) — перейдут в другую половину, а «угловые» — диагонально поменяются местами.

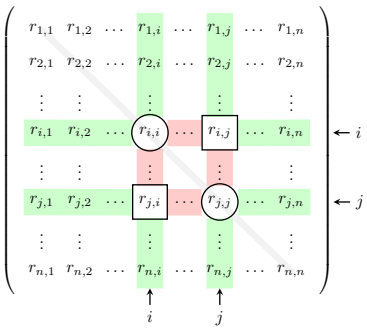

Рис. 5. Операция перестановки двух строк и столбцов (i и j)

На рисунке 5 элементы области «перекрытия» закрашены розовым цветом, а элементы вне её — светло-зелёным. Переставляемые «угловые» элементы там выделены, причём (поскольку на главной диагонали матрицы потерь находятся нули) перестановка одной пары вообще никак не сказывается на результате, а во второй паре элементы (не находящиеся на главной диагонали) попадают в противоположную половину. При

[12] В примере пятью экспертами указаны такие ранжирования пяти альтернатив: 1) $a_2 \sim a_5, a_4, a_1 \sim a_3$; 2) $a_1 \sim a_3, a_2, a_5, a_4$; 3) $a_1, a_4, a_3 \sim a_5, a_2$; 4) $a_4, a_3, a_1 \sim a_2 \sim a_5$; 5) a_3, a_5, a_4, a_1, a_2. Консенсусных ранжирований в этом примере три: `[3,5,4,1,2]`, `[4,3,1,5,2]`, `[4,3,5,1,2]`.

[13] Поскольку при любых перестановках строк (столбцов) элементы столбцов (строк), оставаясь неизменными, могут лишь менять своё местоположение в них.

этом, если обмениваются местами соседние строки/столбцы, то область «перекрытия» сводится лишь к «угловым» её элементам, а потому в таком случае в матрице потерь просто меняются местами наддиагональный и поддиагональный элементы, а остальные изменения происходят лишь в рамках «своих» половин и не изменяют ситуацию в смысле формирования суммы наддиагональных элементов.

6. Циклический сдвиг некоторых соседних позиций. Для простоты будем рассматривать пока только вариант циклического сдвига на одну позицию в ту или другую сторону. В принципе, перестановка двух соседних позиций вроде бы является частным случаем разбираемого здесь, но существенное отличие заключается в том, что для столь малого числа позиций вообще нет внедиагональных элементов «общего» фрагмента матрицы размером 2×2, остающихся в «своей» треугольной половине после подобного сдвига, если не считать — формально — случай нулевых элементов под и над диагональю.

1	2	3	4	5	
6	7	8	9	10	6
11	12	13	14	15	11
16	17	18	19	20	16
21	22	23	24	25	21
	2	3	4	5	1

Рис. 6

Легко проверить, что циклический сдвиг одновременно нескольких соседних строк и столбцов на одну позицию в сторону, скажем, уменьшения их номеров эквивалентен диагональному перемещению влево-вверх «общего» участка из этих строк и столбцов, «свёрнутого» в тор, т. е., когда первая из группы строк переходит в последнюю (с перемещением диагонального элемента), а первый столбец — в последний столбец (см. рис. 6).

Теперь, пользуясь этими двумя приведёнными преобразованиями, можно уже сформулировать некоторые полезные критерии относительно сумм наддиагональных элементов антисимметричных матриц.

Критерий «неизменности». *Если в антисимметричной матрице $\|a_{ij}\|$ с размерами $N\times N$ для строки i при некотором целом $k>0$ выполняется какое-либо из двух условий*

$$\sum_{j=i}^{i+k} a_{ij}=0 \quad (0<k\leqslant N-i), \qquad \sum_{j=i-k}^{i} a_{ij}=0 \quad (0<k<i),$$

то перестановка строки и столбца i на место $i+k$ (или $i-k$, соответственно) не изменяет сумму наддиагональных элементов матрицы[14].

Доказательство. Применяемая здесь перестановка является циклическим сдвигом строк и столбцов (см. ранее), это значит, что строка i попала на место $i+k$ (или $i-k$) и наоборот, а остальные части строк и столбцов (между ними, но по разные стороны от главной диагонали) переместились в пределах «своих» треугольных половин матрицы. То же самое справедливо и для столбцов i и $i+k$ (или $i-k$): они перешли каждый в противоположную половину, но сумма элементов в этих перемещённых отрезках равна нулю: для строк — по условию, для столбцов — в силу антисимметричности матрицы. Значит, это изменение никак не могло повлиять на сумму её наддиагональных элементов. □

	5	4	6	1	7	2	3
5	0	0	−1	−1	1	0	−1
4	0	0	−2	1	−2	2	1
6	1	2	0	−1	0	−1	−2
1	1	−1	1	0	−1	−1	−1
7	−1	2	0	1	0	−1	−1
2	0	−2	1	1	1	0	−4
3	1	−1	2	1	1	4	0

Рис. 7. Наличие дополнительных «равноправных» ранжирований

В качестве пояснения использования критерия можно привести пример из [7], где для 6 ранжирований[15] семи альтернатив получается 13 оптимальных перестановок; на рисунке 7 показано состояние «облегчённой» матрицы потерь этих ранжирований после одной из них.

Для каждого диагонального элемента антисимметричной матрицы ищутся (вправо и влево) отрезки из последовательных элементов его строки с нулевой накопленной суммой (рис. 7). Они будут указывать, куда можно переставлять номер диагонального элемента в текущем порядке столбцов и строк, чтобы сумма элементов верхнего треугольника матрицы оставалась неизменной. Такие отрезки здесь выделены фоновым цветом, а суммы элементов равны: $a_{11}+a_{12}=0$, $a_{22}+a_{21}=0$, $a_{22}+a_{23}+a_{24}+a_{25}+a_{26}+a_{27}=0$, $a_{44}+a_{43}+a_{42}=0$. Соответственно, в перестановке `[5,4,6,1,7,2,3]`, отвечающей за демонстрируемое состояние матрицы, первый номер можно поставить на второе место, второй — на первое и на последнее, четвёртый — на второе, что даёт (без учёта повторений) такие новые перестановки: `[4,5,6,1,7,2,3]`, `[5,6,1,7,2,3,4]`, `[5,1,4,6,7,2,3]`.

[14] Интересно, что антисимметричная матрица со всеми отрицательными значениями в верхнем треугольнике не будет удовлетворять критерию неизменности, поскольку в каждой строке справа и слева от диагональных элементов находятся величины одного знака (если они есть), — в полном соответствии с теоремой 1.

[15] Это ранжирования $x_1\succ x_2\succ x_3\succ x_4\succ(x_5,x_6,x_7)$, $x_5\succ x_1\succ x_2\succ x_4\succ x_6\succ(x_3,x_7)$, $x_7\succ x_6\succ x_2\succ x_3\succ(x_1,x_4,x_5)$, $x_5\succ x_3\succ x_4\succ(x_1,x_2,x_6,x_7)$, $x_4\succ x_7\succ x_5\succ x_6\succ x_2\succ(x_1,x_3)$, $x_6\succ x_1\succ x_7\succ x_2\succ(x_3,x_4,x_5)$, записанные здесь чуть более традиционно, чем в оригинале (в статье [7] вместо знака $\succ$ используется $\prec$).

Критерий «улучшаемости». *Если в антисимметричной матрице $\|a_{ij}\|$ с размерами $N\times N$ для строки i при некотором целом $k>0$ выполняется какое-либо из двух условий*

$$\sum_{j=i}^{i+k} a_{ij} > 0 \qquad (0<k\leqslant N-i), \qquad\qquad \sum_{j=i-k}^{i} a_{ij} < 0 \qquad (0<k<i),$$

то перестановка строки и столбца i на место $i+k$ (или $i-k$, соответственно) уменьшает сумму её наддиагональных элементов на удвоенную абсолютную величину получаемой суммы, т. е., на $2|\sum a_{ij}|$.

Доказательство. Аналогично соображениям из доказательства предыдущего критерия можно сказать, что при указанных циклических сдвигах строка с положительной (отрицательной) суммой попадает из наддиагональной (поддиагональной) части матрицы в противоположную часть, уменьшая сумму наддиагональных элементов на (абсолютную) величину суммы в строке; то же самое происходит и с соответствующими столбцами (с такой же абсолютной величиной суммы): они переходят в противоположную часть, тоже уменьшая сумму наддиагональных элементов; остальные изменяющие своё положение части строк и столбцов никак на эту сумму не влияют, поскольку перемещаются в пределах «своих» частей. □

Располагая такими критериями, можно уже «сконструировать» простейший («наивный») алгоритм минимизации суммы наддиагональных элементов антисимметричной матрицы (путём одновременного изменения положений её строк и столбцов): сначала — в соответствии с критерием «улучшаемости» — уменьшать её наддиагональную сумму, пока это возможно, а затем — с помощью критерия «неизменности» — выявить и многочисленные «равноправные» перестановки полученного «решения» (если они есть).

Конечно, здесь имеются некоторые трудности, связанные с тем, что путей уменьшения минимизируемой величины обычно не так мало (но можно рассматривать «жадную» версию, когда выбирается путь, дающий наибольший эффект), достигнутый минимум вполне может оказаться локальным, а не глобальным (но можно попытаться стартовать с какого-нибудь «хорошего приближения» к минимизирующей перестановке), да и вообще возможно, что существуют какие-то другие (пока не обнаруженные) критерии, позволяющие установить, что значение в локальном минимуме можно уменьшить, и указывающие, как это сделать.

Но, как выясняется, даже столь тривиальный подход работает на удивление неплохо для эвристического алгоритма, особенно учитывая его крайнюю простоту и применимость к ситуациям, которые нельзя назвать малоразмерными. А самым интересным для множественных оптимальных перестановок является то, что преобразования из критерия «неизменности» позволяют получить не просто какие-то другие оптимальные перестановки, а, похоже, дают возможность найти их все. Или, выражаясь более осторожно: пока не обнаружены примеры, в которых бы это было не так.

7. Эксперименты с ранжированиями. Настоящая работа начиналась с попыток отыскания консенусных ранжирований (медиан Кемени) для различных наборов ранжирований. Довольно быстро выяснилось, что эвристический алгоритм, предложенный в [1, стр. 82–83], работает не всегда, да и не рассчитан на поиски множественных оптимальных ранжирований. Попытки трансформировать его в рекурсивную версию, способную обнаруживать множественные результирующие ранжирования, привели к обнаружению случаев, в которых процедура из [1, стр. 82], находит не медиану Кемени, а другое (довольно близкое к оптимальному) ранжирование [8]. После этого по доступной литературе были изучены приводимые там примеры, особенно те, где имеются множественные оптимальные ранжирования [3, 7], [9, стр. 29][16] и др.

Разумеется, для повторения и анализа уже опубликованных примеров и результатов потребовались эффективные средства как ввода находимых наборов ранжирований, так и преобразования их в матрицы потерь. Подобный инструментарий — Julia-пакет `Rankings.jl` [10] — был создан, опробован на различных найденных в литературе примерах и постепенно усовершенствовался. Это позволило легко и просто выверять находимые примеры. При помощи макросов этого пакета ранжирования могут быть записаны во вполне привычном для исследователя виде, причём чаще всего — почти так же, как и в большинстве публикаций[17]: например, $a_2\sim a_5, a_4, a_1\sim a_3$ — для примера из [1, стр. 86], $BDAEC$ — для примера из [6], $x_1\succ x_2\succ x_3\succ x_4\succ (x_5, x_6, x_7)$ — для примера из [7] и т. д., — только с «обрамлением» `ranking"..."`.

[16] Два ранжирования, приводимые в книге Кемени и Снелла (по немного другому поводу), можно записать с помощью созданного пакета `Rankings.jl` так: `ranking"abcidefjgh"` ⍉ `ranking"gaiebchjfd"`. Как оказывается, они имеют целых 1224 ранжирования, максимально близких к ним (в смысле расстояния Кемени), являясь тем самым превосходным примером для проверки любых алгоритмов получения всех оптимальных ранжирований.

[17] Некоторым исключением являются французские (и, по всей видимости, канадские) статьи, поскольку в них часто знаком приоритетности альтернативы выбран $\prec$ (знак предшествования), а не $\succ$ (по аналогии с $>$), как в остальном мире.

Дополнительно (для удобства) была введена специальная операция, обозначаемая выбранным для этой цели символом ⊘ (в Julia такое тоже возможно), — для комбинирования отдельных ранжирований сразу в («облегчённую») матрицу потерь. Этот символ можно использовать и как специфическое имя функции:

```
⊘(ranking"ABCDE", ranking"BDAEC", ranking"BAECD", ranking"ADBCE")
```

и как инфиксный символ бинарной операции (наподобие сложения или вычитания):

```
ranking"ABCDE" ⊘ ranking"BDAEC" ⊘ ranking"BAECD" ⊘ ranking"ADBCE"
```

В любом случае будет сформирована квадратная целочисленная антисимметричная матрица, каждый размер которой равен количеству альтернатив в отдельном ранжировании (совпадение их обозначений и согласованность количеств обязательно проверяется). Тем самым ввод практически любого найденного набора экспертных оценок сводился к «правильной» записи соответствующего выражения на языке Julia, создающего в качестве результата матрицу потерь для набора, что позволяло переходить от ранжирований непосредственно к процедурам определения консенсусных вариантов для них.

Реализованный в рамках пакета `Rankings.jl` упомянутый выше «наивный» алгоритм минимизации суммы наддиагональных элементов антисимметричной матрицы оформлен в виде функции `consensus()`, принимающей в качестве единственного параметра «облегчённую» матрицу потерь заданного профиля (набора ранжирований). Ситуация I в функции обрабатывается отдельно, а в ситуациях II и III (которые иногда[18] трудно отличить друг от друга) в качестве первоначальных приближений взяты перестановки строк и столбцов матрицы, при которых должным образом отсортированы не сами количества отрицательных элементов в строках и столбцах, а суммы этих элементов в них[19]. При невозможности продолжить дальнейшее «улучшение» предусмотренным способом (используется «жадный» вариант, т. е., максимальный шаг) функция возвращает набор перестановок, получаемый на основе многократного применения критерия «неизменности».

Проверка работы функции `consensus()` на опубликованных в литературе примерах показала, что в большинстве случаев обнаруживается глобальный минимум наддиагональной суммы, а также все сопутствующие ему перестановки [8]. В некоторых примерах определяется локальный минимум (как правило, это происходит на матрицах, получаемых из т. н. турнирных[27]); в одном из примеров, приводимом в статье [3], возникли трудности с получением всех оптимальных перестановок — поскольку их количество весьма велико (почти полмиллиона по оценкам автора статьи); сейчас в коде количество выводимых перестановок специально ограничено и процесс их поиска прерывается, если количество стало большим, хотя основная проблема заключается не в самом их формировании, а во времени, затрачиваемом на это.

Надёжная верификация результатов работы алгоритма при поиске оптимальных ранжирований (и тем более — всех) возможна с помощью последовательного перебора различных перестановок альтернатив, но практически осуществима лишь в случае их небольшого количества: уже десять альтернатив будут перебираться Julia-программой в течение нескольких секунд; распараллеливание проверок даёт возможность ускорить процесс, но не слишком значительно (можно перебрать до пятнадцати альтернатив), поскольку требует быстро нарастающего количества независимых «исполнителей» с каждой новой альтернативой. Тем не менее, параллельная реализация перебора также была опробована — из-за того, что в литературе имеются результаты поиска оптимальных ранжирований для 13 и 14 альтернатив [3, 11].

8. Распределённое (параллельное) формирование перестановок. Если для ускорения верификации результатов искать подходящие перестановки параллельно, то возникает необходимость формировать их раздельно и независимо друг от друга, а, значит, понадобится сопоставлять условному номеру каждого вычислительного элемента свою перестановку. Использованный в настоящей работе (ввиду отсутствия готового кода для параллельного формирования перестановок) способ такого сопоставления оказался — как выяснилось впоследствии — вариантом т. н. факториальной системы [12], но с обратным порядком следования «разрядов» представления; поясним его вкратце.

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
. . . . . .

Рис. 8. О перестановках

«Треугольник» на рис. 8 наглядно иллюстрирует рекурсивное «строение» перестановок. Возьмём, скажем, его вторую строчку. Числа 1, 2 показывают две возможные позиции вто-

[18] Так бывает при формальном отсутствии превышения прямых линейного нарастания/убывания отсортированными количествами отрицательных элементов, но при невозможности расположить все отрицательные элементы над диагональю.

[19] Такой подход позволяет «улучшить» приближения в случае большого диапазона изменения значений элементов; если же все значения отличаются незначительно, то приближения практически совпадут с полученными просто по количеству.

рого элемента (впереди/позади) при размещении его относительно позиции первого (первая строчка), одновременно нумеруя их. Точно так же третья строка показывает все три возможные позиции размещения третьего элемента относительно (любого размещения!) первых двух (эти позиции тоже пронумерованы). Совершенно аналогичные выводы можно сделать и по поводу любой из последующих строк по сравнению с ей предшествующей: там будут все позиции размещения очередного элемента относительно уже расположенных ранее. Первая строка показывает (и нумерует) единственное возможное расположение самого первого элемента: перед ним никаких элементов нет, он может располагаться произвольно, но фактически — безвариантно. Тем самым (в силу принципа индукции) мы имеем способ перебора всех возможных перестановок.

Спускаясь из «вершины» этого треугольника вниз и выбирая в каждой строке (вплоть до последней) номер одной из позиций, мы получаем и подтверждение тому, что перебираются все перестановки (количество «путей» будет равно $1 \cdot 2 \cdot 3 \cdot \ldots \cdot N = N!$ для N строк), и уникальный числовой код каждой перестановки (образованный уже выбранными номерами — это и будет представление её условного номера в некоторой числовой системе; правда, в отличие от числа в «настоящей» факториальной системе разряды здесь идут в обратном порядке[20]). Последовательность перестановок четырёх элементов, формируемую при этом, воспроизводит рисунок 9; предполагается, что выбор каждого номера новой позиции в какой-либо из строк (при фиксированных позициях в предыдущих строках) происходит слева направо.

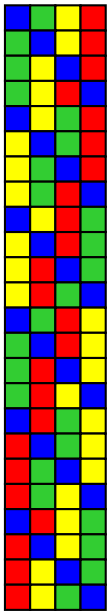

Рис. 9

Несмотря на то, что данное представление есть почти факториальная форма[21] (некоторого) числа, только записанная «цифрами» в обратном порядке, преобразование перестановки снова в уникальное первоначальное значение должно производиться не обычным переводом факториального представления в десятичное (хотя получится тоже уникальное значение из того же диапазона, но иное), а немного другим способом. Скажем, для четырёхзначных представлений веса разрядов будут не $(1), 1, 2, 6$, а $(24), 12, 4, 1$ ($2 = 1\times2,\ 6 = 1\times2\times3$; но $24 = 4\times3\times2,\ 12 = 4\times3$)[22].

Одним из переносимых способов организации параллельных вычислений является стандарт OpenCL [13], предполагающий использование набора вычислительных единиц для выполнения одинакового кода (т. н. ядра OpenCL); именно наличие в настоящее время многих различных реализаций этого стандарта на распространённых платформах и позволило протестировать параллельный вариант перебора перестановок. Поясним, как организовано необходимое ядро.

Глобальный идентификатор ядра (т. е., уникальная целая величина, соответствующая ему; т. н. номер «исполнителя») сначала преобразуется в изложенное выше (квази)факториальное представление, из которого, в свою очередь, формируется соответствующая перестановка[23]. Она далее используется для преобразования порядка следования строк и столбцов заданной матрицы перед подсчётом суммы её наддиагональных элементов. В массиве с результатами проверки фиксируются только те перестановки, при которых сумма наддиагональных элементов преобразованной матрицы не превышает заданной пороговой величины. Если перестановок будет больше, чем предусмотрено памяти для хранения их условных номеров, то последующие перестановки сохранены не будут, однако их общее количество, тем не менее, подсчитывается.

Полный код использованного OpenCL-ядра для формирования некоторой конкретной перестановки и вычисления для заданной матрицы — после воздействия на её строки и столбцы этой сформированной перестановки — суммы её наддиагональных элементов приводится ниже (листинг 1). Он содержит собственно функцию ядра `Permutation()` (строки 13–42) и вспомогательную функцию `UTSum()` (строки 3–11). Последняя подсчитывает сумму наддиагональных элементов передаваемой ей матрицы `A` после переупорядочивания её строк и столбцов с помощью передаваемой перестановки `P` размера `N`.

Параметрами функции ядра являются: количество всех перестановок `Total`, указатель на исследуемую матрицу `AMatrix`, задаваемая пороговая величина `Level`, указатель на инкрементируемую (атомарно[24]) величину `NumOf`, которая может быть изменена лишь одним из многочисленных «исполнителей», давая ему одновременно уникальное значение для расположения его результата в массиве, и ограничивающее её значение `NoMore`, определяемое реальным размером результирующего массива, а также сам этот массив `Result` для номеров подходящих перестановок — в виде указателя.

[20] Что не имеет принципиального значения, поскольку нас интересует просто весь набор возможных перестановок.

[21] В факториальном представлении какого-либо числа его «цифры» можно считать координатами соответствующей целочисленной точки многомерного параллелепипеда со всеми разными (целыми) размерами по различным измерениям.

[22] Это соответствует процедуре «выстраивания в линию» целочисленных точек упомянутого многомерного параллелепипеда со всеми разными (целыми) размерами по разным измерениям — только начиная с других («старших») измерений.

[23] Следует отметить, что она будет содержать величины, начиная с нуля, — поскольку ими индексируется массив языка C.

[24] Для того, чтобы функция атомарного инкремента `atom_inc()` была доступна коду ядра, необходима также специальная директива разрешения использования нужного расширения в коде (она располагается в строке 1 листинга 1).

Суммарное количество «исполнителей» должно быть точно равно факториалу числа альтернатив — чтобы при обработке каждая перестановка была использована ровно один раз, для этого в функции ядра производится проверка (строка 24): должно ли оно (в зависимости от значения своего глобального идентификатора) исполняться или нет. Из-за быстрого роста (при увеличении количества альтернатив) числа необходимых «исполнителей» конфигурационное пространство OpenCL надо использовать максимально полно и понадобятся все три его возможных измерения. При этом приведение адресации конкретного «исполнителя» в трёх измерениях к некоторому уникальному идентификатору осуществляется с помощью параметров конфигурационного пространства стандартным образом (строки 19–21).

```
1  #pragma OPENCL EXTENSION cl_khr_int64_base_atomics : enable
2
3  int UTSum(__global int A[], uint P[], uint N)
4  {
5      int S = 0;
6
7      for (uint i = 0; i < N; i++)
8          for (uint j = i+1; j < N; j++)
9              S += A[P[i]*N + P[j]];
10     return S;
11 }
12
13 __kernel void Permutation(ulong Total, __global int *AMatrix, int Level,
14                           volatile __global ulong *NumOf, ulong NoMore,
15                           __global ulong *Result)
16 {
17     uint factcode[SIZE] = {0};
18     uint permutation[SIZE] = {0};
19     ulong gid = get_global_id(0)+
20                 get_global_id(1)*get_global_size(0)+
21                 get_global_id(2)*get_global_size(0)*get_global_size(1);
22     ulong gid0 = gid;
23
24     if (gid < Total) {
25         uint n = SIZE;
26         while (0 < n) {
27             factcode[n-1] = gid % n + 1;
28             gid /= n--;
29         }
30         n = SIZE;
31         while (1 < n--) {
32             uint k = 0;
33             for (uint i = 0; i < SIZE; i++)
34                 if (permutation[i] == 0 && ++k == factcode[n])
35                     permutation[i] = n;
36         }
37         if (UTSum(AMatrix, permutation, SIZE) <= Level) {
38             ulong MyNo = atom_inc(NumOf);
39             if (MyNo < NoMore) Result[MyNo] = gid0;
40         }
41     }
42 }
```

Листинг 1. Код ядра OpenCL

Воспользоваться этим ядром можно как в рамках OpenCL-программы на языке C/C++, так и из языков Python (с помощью пакета PyOpenCL), Julia (с помощью пакета OpenCL.jl) и т. п. В экспериментах настоящей статьи применялись различные версии пакета PyOpenCL на платформах x86_64 и ARM.

Поскольку ядро OpenCL компилируется уже в процессе работы программы (непосредственно перед запуском необходимого количества экземпляров ядра), при его компиляции можно (и нужно!) в командной строке указать значение ещё не определённого параметра: в данном случае — размера перестановки `SIZE`.

Следует отметить, что реализованный выше алгоритм может исполняться не только на многих ядрах, но и параллельно на многочисленных независимых вычислительных устройствах, поскольку позволяет легко распределять работу между ними путём разделения всего диапазона необходимых идентификаторов «исполнителей» на отдельные отрезки, тем самым дополнительно ускоряя процесс перебора.

9. Предшествующие работы по рассматриваемым вопросам. То, что здесь названо критерием «улучшаемости», в несколько упрощённом виде уже использовалось ранее и в эвристическом алгоритме Б.Литвака: ситуация, формулируемая критерием для ближайших к главной диагонали элементов, там называлась нарушением необходимого условия оптимальности [1, стр. 82]; здесь оба сформулированных критерия составляют взаимодополняющую пару способов «перемещения» по возможным «уровням» суммы наддиагональных элементов «облегчённой» матрицы потерь.

Идею возможной циклической перестановки альтернатив можно обнаружить в статье [7], но там нет никаких численных соотношений, характеризующих подобные перестановки. Мысль об использовании количеств определённых элементов в строках и столбцах матрицы (по)парных сравнений[25] каких-либо объектов для определения такого их упорядочения, при котором количество противоречивых ответов минимально, появлялась в давней статье [14, p. 308], но не получила дальнейшего логического развития[26], а приводимый там же пример наглядно продемонстрировал, что работает такой подход далеко не всегда. В настоящей работе подобный подход применяется к (произвольным) антисимметричным матрицам (как вариантам матриц потерь некоторых профилей ранжирований) — но для их предварительного анализа и получения начального приближения, с которого начинается процесс дальнейшего уточнения и поиска оптимальных ранжирований.

Вообще надо сказать, что задача отыскания медиан Кемени родственна немалому количеству задач, по-разному названных и рассматриваемых в различных предметных областях: и только что упомянутой задаче упорядочения альтернатив по матрице парных сравнений (которая антисимметрична с элементами ± 1), и задаче ранжирования команд по результатам игр в (однокруговом) турнире[27] (от матрицы парных сравнений турнирная матрица отличается лишь отсутствием отрицательных значений — вместо них присутствуют нули, но её легко привести к антисимметричному виду, если сформировать матрицу[28] $A = T^\top - T$; в записи на языке Julia можно использовать выражение `A = T'-T`, где `T'` — операция, эквивалентная для вещественных матриц транспонированию) и многим другим, не говоря уже о том, что турнир можно рассматривать и как полный орграф, где каждой паре вершин (двум игрокам) соответствует ребро из одной вершину в другую (результат их игры), а тогда задача уже переходит в область теории графов. Поэтому значительная часть исследований может не попадать в поле зрения специалистов, так как находится за пределами областей их традиционного интереса, а, значит, требует внимательного перекрёстного анализа разнообразных математических и прикладных разделов.

Иллюстрации вывода некоторых известных алгоритмов генерации перестановок для случая четырёх элементов приводятся в [15] и [16]; использованный здесь алгоритм порождает другую последовательность (см. рис. 9). Она отличается от уже известных тем, что в ней перестановки упорядочены в соответствии с факториальным представлением их индекса, но начиная не со «старших», а с «младших» разрядов этого представления. Впрочем, для алгоритма параллельного формирования всех возможных перестановок, использованного в работе, их порядок не является существенным, главное, чтобы перестановки были сформированы все и без повторений.

10. Заключение. В настоящей работе рассмотрен комбинаторный подход к задаче отыскания медиан Кемени с использованием антисимметричных аналогов матриц потерь, проанализированы возникающие при этом типы ситуаций и сформулирован эвристический алгоритм поиска оптимальных консенсусных ранжирований, включая множественные. Для эффективной работы с ранжированиями и возможности быстрой верификации сторонних результатов был создан специализированный пакет на языке Julia и опробован на имеющихся в литературе примерах.

---

[25] В статье подобная матрица содержит символ + (плюс) в столбце j строки i и символ − (минус) в строке j столбца i — если объект i предпочтительнее объекта j. По сути она антисимметрична (с элементами ± 1), а при «правильном» порядке расположения объектов все последующие менее предпочтительны, чем предшествующие, и, значит, правее главной диагонали в каждой строке в идеале должны быть лишь символы плюс. Ситуация, когда там имеются символы минус, трактуется как противоречивость отдельных сравнений; «правильный» порядок должен иметь минимальное число противоречий.

[26] Хотя там уже содержится наблюдение о возможности перемещения строки матрицы ниже, если число символов минус справа от диагонального элемента превышает число символов плюс, — для уменьшения количества символов минус в верхней треугольной половине, в чём можно усмотреть «предка» сформулированного в настоящей работе критерия «улучшаемости».

[27] Турнир (tournament) — это круговой турнир, в котором каждый игрок из N играет ровно одну игру с каждым из остальных игроков, причём ничьих не бывает. Результаты турнира обычно записываются в квадратную матрицу $\|t_{ij}\|$ порядка N, где $t_{ij} = 1$, если игрок i побеждает j, и $t_{ij} = 0$, если игрок i проигрывает j; диагональные элементы считаются равными нулю: $t_{ii} = 0$. Такая (0,1)-матрица T называется турнирной матрицей (tournament matrix).

[28] Антисимметричность её следует непосредственно из определения: $A^\top = (T^\top - T)^\top = T - T^\top = -(T^\top - T) = -A$.

Работа выполнена при поддержке гранта 18-07-00424 и по предложению его научного руководителя профессора Ю.П. Пытьева, за что автор хотел бы выразить ему искреннюю благодарность.

Список литературы

1. *Литвак Б.Г.* Экспертная информация: Методы получения и анализа. М.: Радио и связь, 1982.

2. Язык Julia. https://julialang.org.

3. *Muravyov, S.V.* Ordinal measurement, preference aggregation and interlaboratory comparisons. // Measurement 46 (2013) 2927–2935.
doi 10.1016/j.measurement.2013.04.044.

4. *Тыртышников Е.Е.* Основы алгебры. М.: Физматлит, 2017.

5. *Болтенков В.А., Куваева В.И., Червоненко П.П.* Метод экспертного выбора цифровых компонентов систем промышленной автоматики на основе марковской модели. // Технология и конструирование, 2018, № 2, с. 21–28.
doi 10.15222/TKEA2018.2.21.

6. Пакет RankAggreg. https://cran.r-project.org/web/packages/RankAggreg/RankAggreg.pdf.

7. *Guénoche, A.* Analyse dés Préférences et Tournois Pondérés. // J. of Interd. Method. and Issues in Science, vol. 2, 2017.

8. Эксперименты с ранжированиями. https://github.com/vaa-msu/ranking-tests.

9. *Кемени, Дж., Снелл, Дж.* Кибернетическое моделирование. Некоторые приложения. М.: Советское радио, 1972.

10. Пакет Rankings.jl. https://github.com/vaa-msu/Rankings.jl.

11. *Двоенко, С.Д., Пшеничный, Д.О.* О метрических свойствах медианы Кемени. // Машинное обучение и анализ данных, т. 1, № 11, 1619–1631, 2015.

12. Factorial number system. https://en.wikipedia.org/wiki/Factorial_number_system.

13. Open Standard for Parallel Programming. https://www.khronos.org/opencl/.

14. *Slater, P.* Inconsistencies in a Schedule of Paired Comparisons. // Biometrika, 1961, vol. 48, no. 3/4, 303–312.

15. Перестановка. https://en.wikipedia.org/wiki/Permutation.

16. Генерация перестановок. http://combos.org/perm.