Ватутин Эдуард Игоревич, Юго-Западный государственный университет
доцент кафедры вычислительной техники

Кочемазов Степан Евгеньевич, Институт динамики систем и теории управления им В.М. Матросова СО РАН
младший научный сотрудник

Заикин Олег Сергеевич, Институт динамики систем и теории управления им В.М. Матросова СО РАН
старший научный сотрудник

Алгоритмическая оптимизация программной реализации процедуры получения множества трансверсалей для латинских квадратов

Латинским квадратом (ЛК) порядка N называется квадратная таблица $A=\|a_{ij}\|$, $i, j=\overline{1, N}$, элементами a_{ij} которой являются элементы некоторого множества U мощности $N=|U|$ (для определенности далее будем полагать, что $U=\{0, 1, 2, ..., N-1\}$). По определению в каждой строке и в каждом столбце латинского квадрата каждый из элементов множества U встречается в точности один раз. Для диагональных латинских квадратов (ДЛК), являющихся специальным видом ЛК, по определению дополнительно вводятся требования на отсутствие совпадающих элементов на главной и побочной диагонали. ДЛК называется нормализованным, если элементы его первой строки упорядочены по возрастанию. Несложно показать, что путем биективной подстановки (перестановки) элементов множества U любого корректного ДЛК можно добиться его нормализации, а указанное множество квадратов представляет собой класс эквивалентности из $N!$ ДЛК. Для ряда задач (подсчет числа ЛК и ДЛК [1–6], поиск пар и троек попарно (частично) ортогональных ЛК и ДЛК [7–12], сокр. ОЛК и ОДЛК) квадраты в рамках указанного класса эквивалентности не различаются, т.к. обладают одинаковыми свойствами (наличие/отсутствие парного ортогонального квадрата, число трансверсалей и пр.), что существенно экономит машинное время в соответствующих вычислительных экспериментах.

Трансверсалью T в ЛК называется такое множество элементов $T=\{t_1, t_2, ..., t_N\}=\{a_{i_1, j_1}, a_{i_2, j_2}, ..., a_{i_N, j_N}\}$, в котором все индексы различны и все значения элементов также различны. Другими словами, трансверсаль включает в своем составе по одному элементу ЛК из каждой строки и из каждого столбца, причем значения всех элементов различны. Вектора $[i_1, i_2, ..., i_N]$, $[j_1, j_2, ..., j_N]$ и $[a_{i_1, j_1}, a_{i_2, j_2}, ..., a_{i_N, j_N}]$ образуют перестановки элементов множества U. Пример ЛК и множества его трансверсалей приведен на рис. 1.

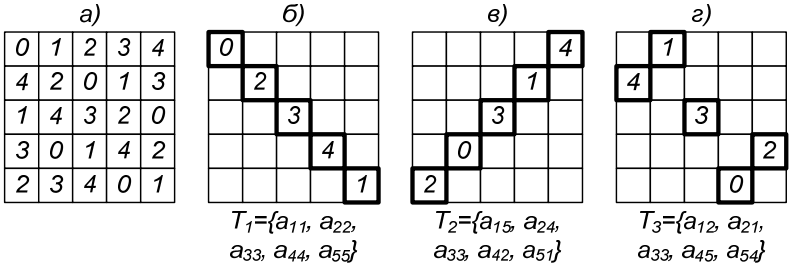

Рисунок 1 – Пример ЛК (а) и множество его трансверсалей (б, в, г). Диагонали ДЛК являются трансверсалями, трансверсаль T_3 является диагональной.

Диагональной трансверсалью $T^{(d)}$ в ЛК называется трансверсаль, которая включает в своем составе ровно по одному элементу с главной и побочной диагонали ЛК (в случае ЛК нечетного порядка данные элементы могут совпадать, как в примере на рис. 1, г). Диагональная трансверсаль является частным случаем трансверсали общего вида. Пример ЛК и множества его диагональных трансверсалей приведен на рис. 2.

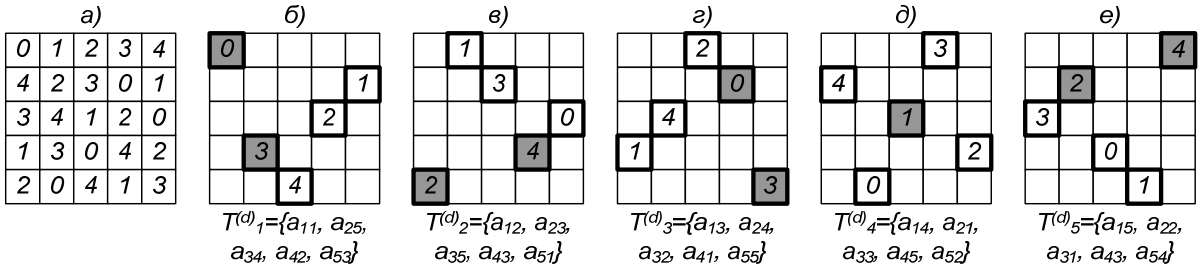

Рисунок 2 – Пример ЛК (а) и множество его диагональных трансверсалей (б, в, г, д, е). Элементы трансверсалей, лежащие на диагоналях, выделены серым

Построение множества трансверсалей играет важную роль в ряде задач (например, при получении пар ОЛК/ОДЛК и более сложных комбинаторных структур с использованием алгоритма Эйлера-Паркера или при отыскании ЛК и ДЛК с экстремальным количеством трансверсалей [13]), а время его выполнения является бутылочным горлышком и по большей части определяет время работы соответствующих алгоритмов и их программных реализаций. В связи с этим вызывает интерес выполнение алгоритмической оптимизации с целью его сокращения.

За основу для доработки была взята известная реализация LAT03C, разработанная А. Беляевым на языке Pascal и доступная с открытыми исходными кодами. В ней используется рекуррентный алгоритм полного перебора с возвратами, реализованный в виде хитроумной итеративной программы с группой меток (англ. labels) и по логике работы напоминающий конечный автомат. С незначительными изменениями она была портирована в более современную среду разработки Delphi, имеющую более эффективный в плане оптимизации машинного кода компилятор. Кроме того, была разработана аналогичная реализация на языке C++, откомпилированная с использованием компилятора, поставляемого в составе среды Microsoft Visual Studio 2015. Также была разработана рекуррентная программная реализация, соответствующая тому же алгоритму, но описанная в конструкциях структурного программирования без использования меток. Результаты тестовых замеров времени работы данных реализаций приведены в табл. 1 (процессор Intel Core i7 4770, ядро Haswell, запуск в один поток, фоновая нагрузка отсутствует).

Таблица 1 – Результаты сравнения времени, необходимого для построения множеств трансверсалей для 1000 ДЛК порядка 10

Реализация	Время, с	Темп, ДЛК/с
LAT03C (А. Беляев, итеративная реализация, Delphi)	9,2	109
Рекуррентная реализация, Delphi	5,7 (4,8) *	175 (208)
Итеративная реализация, C++	5,9	106

Примечание *. Время 5,7 с получено для построения множества трансверсалей общего вида. При построении диагональных трансверсалей оно снижается до 4,8 с за счет возможности раннего отсечения неперспективных решений, содержащих более одного элемента с каждой диагонали, в рамках стратегии ветвей и границ [14].

В ряде задач было показано, что вариация порядка рассмотрения элементов решения оказывает сильное влияние либо на качество результирующего решения (для оптими-

зационных задач [2, 15, 16]), либо на время его получения (для задач на пересчет [17]). С целью апробации данного подхода в задаче построения множества трансверсалей было произведено варьирование порядка заполнения элементов трансверсали в соответствии с принципом минимума возможностей [18], при этом выбор очередного элемента трансверсали производился из еще не рассмотренной ранее строки ДЛК, в которой присутствует минимально возможное число элементов-кандидатов (в базовом варианте (табл. 1) заполнение элементов трансверсалей производится в порядке возрастания). Схематично данный принцип изображен на рис. 3.

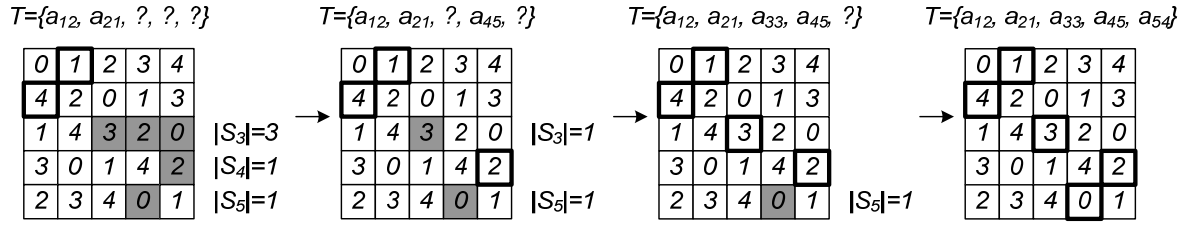

Рисунок 3 – Выбор порядка заполнения элементов трансверсали по принципу минимума возможностей: после выбора первого и второго элементов формируемой трансверсали (выделены жирным) четвертый и пятый элементы трансверсали имеют минимальное число возможностей (по одной, возможности выделены серым) и их внеочередное заполнение является предпочтительным

При практической реализации данного действия необходима оценка числа возможностей и хранение множества уже использованных строк. Указанные действия являются дополнительными по сравнению с предыдущей рекуррентной реализацией с последовательным заполнением элементов трансверсалей, а, следовательно, на них необходимо дополнительное машинное время. Соотношение данных затрат и выигрыша из-за уменьшения арности узлов дерева комбинаторного перебора [17] определяет целесообразность вариации порядка. Измерение времени выполнения данной реализации в тех же условиях показало, что времена построения множеств трансверсалей и диагональных трансверсалей составляют соответственно 11,2 с и 8,5 с, что хуже приведенных в табл. 1. По-видимому, данный эффект объясняется тем, что затраты вычислительного времени на выбор оптимального порядка оказываются б**о**льшими, чем выигрыш от уменьшения числа узлов в дереве комбинаторного перебора.

На каждом ярусе рекурсии в разработанной реализации в цикле производится сканирование возможных элементов a_{ij} текущей *i*-й строки ДЛК, а затем из них исключаются те, которые нарушают определение корректной (диагональной) трансверсали. При подобной проверке в коде программы присутствуют нерегулярно срабатывающие условные переходы, которые приводят к частым сбросам конвейера CPU и неэффективному исполнению кода. Чтобы этого избежать, можно воспользоваться битовыми операциями. При этом для текущей *i*-й строки производится построение множества доступных элементов $S_i = U \setminus C \setminus A_i \setminus \{\alpha a_{ii}\} \setminus \{\beta a_{i,\,N-i+1}\}$, где C – множество уже использованных на данный момент столбцов; A_i – множество номеров столбцов, элементы в составе которых уже присутствуют в формируемой трансверсали; $\alpha, \beta \in \{0,1\}$ – двоичные признаки присутствия в трансверсали элемента с главной и с побочной диагонали соответственно (в случае их единичного значения из формируемого множества удаляются элементы, соответствующие *i*-му и $(N-i+1)$-му столбцу соответственно). Хранение указанных множеств производится в целочисленных переменных, попадающих в РОН процессора в результате оптимизации компилятора, в виде битовых строк, а операции над ними реализуются посредством простейших логических команд CPU (AND, OR, NOT, SHL).

Для быстрого определения состава элементов множества A_i необходимо иметь ин-

формацию о том, какое из значений элементов располагается в каком столбце. Данную информацию можно получить путем однократного построения подстановок, обратных к подстановкам, образованным строками ДЛК. Например, второй строке ДЛК на рис. 2а соответствует подстановка $\sigma = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 4 & 2 & 3 & 0 & 1 \end{pmatrix}$, а обратной к ней является подстановка $\sigma^{-1} = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 3 & 4 & 1 & 2 & 0 \end{pmatrix}$. С использованием подстановки σ^{-1}, хранимой в программе в виде одномерного массива, можно быстро определить, что, например, значение элемента 3 располагается в данной строке в столбце 2 (при условии нумерации столбцов с нуля). Интересной особенностью является то, что квадрат, составленный из подобных обратных подстановок, также является корректным ДЛК. Указанное преобразование при необходимости может быть проведено и для столбцов, при этом получаемые квадраты имеют отличное от исходного квадрата число трансверсалей, следовательно, они не являются разновидностью *M*-преобразований по Чебракову. Пример подобного квадрата из обратных подстановок строк приведен на рис. 4.

а)

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

→

б)

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

Рисунок 4 – Исходный ДЛК (а) и ДЛК, полученный замещением строк обратными подстановками (б)

С применением указанных выше битовых операций циклы организуются только по элементам, присутствующим в множестве S_i, с использованием известных формул булевой алгебры для выделения граничных значений (например, выделение младшего нуля в битовой строке может быть произведено как $\overline{S_i} \wedge (S_i + 1)$), при этом в теле цикла отсутствуют условные переходы и, соответственно, не происходит сброса конвейера CPU.

При использовании указанных битовых операций время выполнения рекуррентной программной реализации с константным порядком просмотра строк сокращается до 4,3 с (темп – 232 ДЛК/с), а время рекуррентной реализации с вариацией порядка рассмотрения строк – до 2,8 с, темп – 357 ДЛК/с (вторая по-прежнему оказывается быстрее первой).

Анализируя дерево комбинаторного перебора, можно заметить, что вариация порядка и сопутствующие проверки эффективны не на всех глубинах рекурсии (аналогичный эффект был зафиксирован и при генерации ДЛК методом полного перебора). Так, например, на нижних ярусах дерева комбинаторного перебора (при большой глубине рекурсии) заполнение элементов трансверсали либо комбинаторный возврат во многих случаях происходят тривиально и проверка мощности множества S_i не нужна. Для анализа того, в каком диапазоне глубин $d_b \le d \le d_t$ вариация порядка работает эффективно, был организован соответствующий вычислительный эксперимент, результаты которого приведены в табл. 2.

Таблица 2 – Зависимость вычислительного времени от диапазона глубин, секунд

		d_b									
		1	2	3	4	5	6	7	8	9	10
d_t	1	3,22	3,17	3,18	3,20	3,15	3,32	3,73	4,13	4,27	4,33
	2	–	3,15	3,19	3,19	3,13	3,33	3,73	4,13	4,28	4,32

	3	–	–	3,17	3,17	3,16	3,34	3,76	4,15	4,28	4,34
	4	–	–	–	3,16	3,22	3,43	3,83	4,20	4,32	4,37
	5	–	–	–	–	3,21	3,58	4,01	4,35	4,44	4,48
	6	–	–	–	–	–	3,19	3,76	4,05	4,15	4,16
	7	–	–	–	–	–	–	3,17	3,46	3,54	3,54
	8	–	–	–	–	–	–	–	3,18	3,27	3,30
	9	–	–	–	–	–	–	–	–	3,17	3,21
	10	–	–	–	–	–	–	–	–	–	3,22

Полученные результаты позволяют сделать вывод о том, что минимальное время составляет 3,13 с и достигается при значениях глубин $2 \le d \le 5$, что наглядно демонстрирует неэффективность выполнения вариации порядка на всех глубинах рекурсии (наихудшим является сочетание $5 \le d \le 10$, с временем обработки 4,48 с). Однако даже при этом постоянный порядок заполнения имеет незначительное преимущество. По сравнению с исходной программной реализацией LAT03C выигрыш во времени построения множества трансверсалей составляет 3,3 раза.

В настоящее время разработанная программная реализация используется при поиске пар ОДЛК порядка 10, выполняемом в рамках проекта добровольных распределенных вычислений Gerasim@home на платформе BOINC. Целью данного подпроекта является классификация комбинаторных структур на множестве бинарного отношения ортогональности ДЛК [19], в том числе попытка отыскания тройки попарно ОДЛК, существование которой является открытой математической проблемой. В настоящее время (21.11.2017) в проекте найдено 150 тыс. уникальных канонических форм ОДЛК, поиск активно продолжается.

Библиографический список

1. Vatutin E.I., Zaikin O.S., Zhuravlev A.D., Manzuk M.O., Kochemazov S.E., Titov V.S. Using grid systems for enumerating combinatorial objects on example of diagonal Latin squares // Distributed computing and grid-technologies in science and education (GRID'16). Dubna: JINR, 2016. pp. 114–115.
2. Ватутин Э.И., Заикин О.С., Журавлев А.Д., Манзюк М.О., Кочемазов С.Е., Титов В.С. О влиянии порядка заполнения ячеек на темп генерации диагональных латинских квадратов // Информационно-измерительные диагностирующие и управляющие системы (Диагностика – 2016). Курск: изд-во ЮЗГУ, 2016. С. 33–39.
3. Ватутин Э.И., Титов В.С., Заикин О.С., Кочемазов С.Е., Валяев С.Ю., Журавлев А.Д., Манзюк М.О. Использование грид-систем для подсчета комбинаторных объектов на примере диагональных латинских квадратов порядка 9 // Информационные технологии и математическое моделирование систем 2016. М.: ЦИТП РАН, 2016. С. 154–157.
4. Vatutin E.I., Zaikin O.S., Zhuravlev A.D., Manzyuk M.O., Kochemazov S.E., Titov V.S. Using grid systems for enumerating combinatorial objects on example of diagonal Latin squares // CEUR Workshop proceedings. Selected Papers of the 7th International Conference GRID'17. 2017. Vol. 1787. pp. 486–490.
5. Kochemazov S.E., Vatutin E.I., Zaikin O.S. Fast Algorithm for Enumerating Diagonal Latin Squares of Small Order // arXiv:1709.02599 [math.CO], 2017. 31 p.
6. Vatutin E.I., Kochemazov S.E., Zaikin O.S. Applying volunteer and parallel computing for enumerating diagonal Latin squares of order 9 // Proc. of The Eleventh International Conference on Parallel Computational Technologies, Vol. 753 of Communications in Computer and Information Science, Springer, 2017, pp. 114–129. DOI: 10.1007/978-3-319-67035-5_9.
7. Заикин О.С., Ватутин Э.И., Журавлев А.Д., Манзюк М.О. Применение высокопроизводительных вычислений для поиска троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 // Параллельные вычислительные техноло-

гии (ПаВТ'2016). Челябинск: ЮУрГУ, 2016. С. 155–166.
8. Заикин О.С., Ватутин Э.И., Журавлев А.Д., Манзюк М.О. Применение высокопроизводительных вычислений для поиска троек взаимно частично ортогональных диагональных латинских квадратов порядка 10 // Вестник ЮУрГУ. Серия: вычислительная математика и информатика. Т. 5. № 3. 2016. С. 54–68. DOI: 10.14529/cmse160304.
9. Zaikin O.S., Vatutin E.I., Zhuravlev A.D., Manzyuk M.O. Applying high-performance computing to searching for triples of partially orthogonal Latin squares of order 10 // CEUR Workshop Proceedings. Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies "Parallel Computing Technologies" (PCT 2016). Vol. 1576. 2016. pp. 155–166.
10. Zaikin O., Zhuravlev A., Kochemazov S., Vatutin E. On the Construction of Triples of Diagonal Latin Squares of Order 10 // Electronic Notes in Discrete Mathematics. Vol. 54C. 2016. pp. 307–312. DOI: 10.1016/j.endm.2016.09.053.
11. Ватутин Э.И., Кочемазов С.Е., Заикин О.С. Оценка комбинаторных характеристик диагональных латинских квадратов // Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации (Распознавание – 2017). Курск: ЮЗГУ, 2017. С. 98–100.
12. Ватутин Э.И., Кочемазов С.Е., Заикин О.С., Манзюк М.О., Титов В.С. Оценка комбинаторных характеристик для пар ортогональных диагональных латинских квадратов // Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов (МППОС'2017). Барнаул: Изд-во Алт. ун-та, 2017. С. 104–111.
13. Vatutin E.I., Kochemazov S.E., Zaikin O.S., Valyaev S.Yu. Enumerating the Transversals for Diagonal Latin Squares of Small Order // CEUR Workshop Proceedings. Proceedings of the Third International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2017). Vol. 1973. Technical University of Aachen, Germany, 2017. pp. 6–14.
14. Land A.H., Doig A.G. An automatic method of solving discrete programming problems // Econometrica. Vol. 28 (3). 1960. pp. 497–520. DOI:10.2307/1910129.
15. Ватутин Э.И., Романченко А.С., Титов В.С. Исследование влияния порядка рассмотрения пар на качество расписаний при использовании жадного подхода // Известия Юго-Западного государственного университета. 2013. № 1 (46). С. 58–64.
16. Ватутин Э.И., Бобынцев Д.О., Романченко А.С. Исследование влияния частичного упорядочивания пар и локального улучшения окрестности пары на качество расписаний при использовании жадного подхода // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2014. № 1. С. 8–16.
17. Ватутин Э.И., Титов В.С., Емельянов С.Г. Основы дискретной комбинаторной оптимизации. М.: Аргамак-Медиа, 2016. 270 с.
18. Golomb S.W., Baumert L.D. Backtrack Programming // Journal of ACM. Vol. 12. Iss. 4. 1965. pp. 516–524. DOI: 10.1145/321296.321300.
19. Ватутин Э.И., Титов В.С., Заикин О.С., Кочемазов С.Е., Манзюк М.О. Анализ комбинаторных структур на множестве отношения ортогональности диагональных латинских квадратов порядка 10 // Информационные технологии и математическое моделирование систем 2017. М.: ЦИТП РАН, 2017. С. 167–170.