КРАТКОЕ ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

В настоящее время нет возможностей точного сравнения результатов, полученных автором и приведённых в [6], – поскольку в двух случаях использовались различные вычислительные ресурсы. Однако в планах автора настоящей работы – продолжение исследований, улучшение полученных результатов, точное сравнение обоих подходов, создание распределённых версий описанных алгоритмов, а в перспективе – существенное улучшение результатов Andreas'a Distler'a.

Работа автора частично поддержана региональным грантом РФФИ № 13-01-97003, а также частично поддержана Федеральной целевой программой "Научные и научно-педагогические кадры инновационной России" на 2009–2013 годы, соглашение № 14.B37.21.1934.

СПИСОК ЛИТЕРАТУРЫ

1. Шеврин Л.Н. Что такое полугруппа // Соросовский Образовательный Журнал. 1997. № 4. С. 99–104.
2. Полугруппа [Электронный ресурс]: Материал из «Викия-сеть»: Версия 3, сохранённая в 17:17 UTC 2 сентября 2010 / Авторы Викия-сеть// Викия-сеть. – Режим доступа: http://ru.math.wikia.com/wiki/Полугруппа
3. Клиффорд А., Престон Г. Алгебраическая теория полугрупп. – М.: Мир, 1972. – 286 с.
4. Semigroup with two elements [Электронный ресурс]: Материал из Википедии – свободной энциклопедии: Версия 488388842, сохранённая в 20:24 UTC 20 апреля 2012 / Авторы Википедии // Википедия, свободная энциклопедия. – Электрон. дан. – Сан-Франциско: Фонд Викимедиа, 2012. – Режим доступа: http://en.wikipedia.org/w/index.php?title=Semigroup_with_two_elements&oldid=488388842
5. Мельников Б. Мультиэвристический подход к задачам дискретной оптимизации. – Кибернетика и системный анализ (НАН Украины), 2006, № 3, С. 32–42.
6. Classification and enumeration of finite semigroups [Электронный ресурс]: Research@StAndrews Full Text: Версия: 10023/945, сохраненная в июне 2010 / Andreas Distler // Research@StAndrews Full Text – Режим доступа: http://hdl.handle.net/10023/945

ALGORITHMS FOR CALCULATING THE NUMBER OF FINITE SEMIGROUPS: PROBLEM STATEMENT AND SIMPLE HEURISTICS

***E.I. Kuzichkina*,** postgraduate student
Togliatti state university, Togliatti (Russia)

Keywords: finite semigroup; exhaustive search; heuristics.

Annotation: Simple heuristic exhaustive search for calculation of the number of finite semigroups were obtained.

УДК 512.531.2

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ НЕКОТОРЫХ АЛГОРИТМОВ ПОДСЧЁТА ЧИСЛА КОНЕЧНЫХ ПОЛУГРУПП

Е.И. Кузичкина, аспирант
Д.И. Власов, студент
Тольяттинский государственный университет, Тольятти (Россия)

Ключевые слова: конечная полугруппа; алгоритмы перебора; эвристические алгоритмы, распределенные вычисления, параллельные алгоритмы.

Аннотация: Рассмотрен эвристический переборный алгоритм для подсчёта числа конечных полугрупп и вариант его параллелизации.

ВВЕДЕНИЕ

Статья посвящена вопросу разработки параллельного алгоритма для подсчета числа конечных полугрупп. Актуальность разработки параллельного алгоритма обусловлена большими возможностями, предоставляемыми многоядерными и многопроцессорными системами. Создание эффективных программных решений для параллельных систем складывается из трех основных компонентов: параллельных алгоритмов, средств реализации параллельности и систем отладки. Самый важный компонент, без которого все другие средства не смогут сделать программу параллельной, это параллельный алгоритм. Именно этому компоненту и будет посвящена эта статья.

ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ, ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Основные теоретические аспекты, определения и термины приведены в статье ([5]). Приведем только самые важные.

Согласно [2], ассоциативная операция – это бинарная операция $\circ$, обладающая ассоциативностью (т.е. сочетательностью):

$$(x \circ y) \circ z = x \circ (y \circ z) \qquad (1)$$

для любых элементов x, y, z. Полугруппой называют множество с заданной на нем ассоциативной бинарной операцией *(S, *)*. Порядок множества обозначается $n \in N$ (где $N = \{1, 2, ...\}$).

МЕТОДИКА ПРОВЕДЕНИЯ ВЫЧИСЛИТЕЛЬНОГО ЭКСПЕРИМЕНТА

Как было сказано в [5], общая постановка задачи следующая: необходимо описать эффективные алгоритмы подсчёта количества неизоморфных полугрупп на *n*-элементном множестве.

Ниже представлена таблица в первой строке, которой представлена информация о размерности полугруппы, во второй строке количество всех возможных операций на множестве, в третьей строке количество полугрупп на множестве и в четвертой частота появления полугрупп.

Таблица 1. Анализ полугрупп

N	1	2	3	4	5	6	7
Количество претендентов на полугруппу	1	16	19 683	$4{,}29 \cdot 10^{9}$	$2{,}98 \cdot 10^{17}$	$1{,}03 \cdot 10^{28}$	$2{,}57 \cdot 10^{41}$
Количество полугрупп	1	8	113	3 492	183 732	17 061 118	7 743 056 064
Отношение	100%	50%	0,6%	$8{,}1 \cdot 10^{-5}$%	$6{,}1 \cdot 10^{-11}$%	$1{,}6 \cdot 10^{-19}$%	$3{,}0 \cdot 10^{-30}$%

Отсюда понятно, что задача поиска количества полугрупп на больших размерностях становится нетривиальной.

Метод 1 – полный перебор.

С точки зрения асимптотической оценки сложности алгоритма имеем:

$$f(n) \in O(n^{n^2} \cdot n^3) \qquad (2)$$

Здесь n^{n^2} – число, получающееся при переборе всех возможных операций на n-элементном множестве, а n^3 – при проверке ассоциативности каждой операции.

Для подсчёта полугрупп в более высоких порядках множества необходимо применять специальные эвристики.

Метод 2 – авторский, мультиэвристический.

Был рассмотрен в предыдущей статье [5]. Для предыдущего метода применение распределенных вычислений затруднительно, поэтому был разработан метод 3.

Метод 3 – авторский, мультиэвристический, возможный для параллелизации.

В данной задаче были использованы ряд эвристик, которые позволили сократить время выполнения программы в разы. Многие из этих эвристик согласованы с подходом, предложенным в [6].

Для того чтобы каждая тройка для проверки ассоциативности операции выполнялась в матрице элементы должны стоять на определенных местах. Для каждой тройки для проверки ассоциативности можно построить «правильное» расположение элементов. Например: для 2-ух элементного множества, чтобы выполнилась тройка *(2*1)*2 = 2*(1*2)*, нужно:

1 вариант:

Если *a = 2*1 = 1; b = 1*2 = 1*, то элементы должны быть расположены в соответствии с рис. 1.

1	(2*1)*2 = 1 * 2 = 1
1	2*(1*2) = 2 * 1 = 1

Рис. 1. Матрица операции для тройки 212 при a = 1; b = 1.

2 вариант:

Если *a = 2*1 = 1; b = 1*2 = 2*, то элементы должны быть расположены в соответствии с рис. 2.

2	(2*1)*2 = 1 * 2 = 2
1 2	2*(1*2) = 2 * 2 = 2

Рис. 2. Матрица операции для тройки 212 при a = 1; b = 2.

3 вариант:

Если *a = 2*1 = 2; b = 1*2 = 1*, то элементы должны быть расположены в соответствии с рис. 3.

1	2*(1*2) = 2 * 1 = 2
2	(2*1)*2 = 2 * 2 = 2

Рис. 3. Матрица операции для тройки 212 при a = 2; b = 1.

4 вариант:

Если *a = 2*1 = 2; b = 1*2 = 2*, то элементы должны быть расположены в соответствии с рис. 4.

Z – может быть любым числом.

2	(2*1)*2 = 2 * 2 = z
2 Z	2*(1*2) = 2 * 2 = z

Рис. 4. Матрица операции для тройки 212 при a = 2; b = 2.

Если повторить предыдущее рассуждение для каждой тройки, то получается набор «кусочков» матриц, изображенный на рис. 5.

	111	**112**	**121**	**122**	**211**	**212**	**221**	**222**
a=1 b=1	1	1 1	z 1 1	1 1	1 1	1 1	1 1 1	Z z 1
a=1 b=2		1 2	1 1 2	1 2	1 2	2 1 2	1 2 1	
a=2 b=1		2 1 2	1 2 1	1 2 1	2 1 2	1 2 2	1 2	
a=2 b=2	2 z Z	2 2 2	2 2	2 2	2 2 2	2 2 Z	2 2	2

Рис. 5. Таблица матриц операций для всех возможных троек на двухэлементном множестве.

Теперь остается найти все пути прохода в этом лабиринте. Из каждого столбца нужно выбрать одну матрицу, и наложить ее на имеющуюся. Если выход из лабиринта найти удалось, то полугруппа готова. Более подробно ниже.

Авторами предложено два варианта решения.

Первый вариант:

Нам нужно взять по одной матрице из каждого столбца и наложить их друг на друга, так чтобы не было противоречий. Например, возьмем путь, изображенный на рис. 6.

	111	112	121	122	211	212	221	222
a=1 b=1	1							z z 1
a=1 b=2		1 2			1 2		1 2 1	
a=2 b=1				1 2 1				
a=2 b=2			2 2			2 2 z		

Рис. 6. Таблица с непротиворечивым набором матриц.

При наложении всех матриц друг на друга получается итоговый вариант – рисунок 7.

1	2
2	1

Рис. 7. Итоговая матрица операции

Это полугруппа, т.к. на ней выполнятся все ассоциативные операции (которые мы бы перебирали при простом переборе). В данном случае мы их не перебираем, мы заранее знаем, как могут быть расположены элементы. И для каждого столбца большой матрицы (или для каждой операции) нужно выбрать одну подходящую матрицу, если хоть один столбец будет пропущен, то полугруппы не будет. Обязательно, в каждом столбце нужно найти одну матрицу, которая подойдет.

Как видно, наполненная матрица-претендент на полугруппу в данном случае получается на 4 шаге (рис. 8).

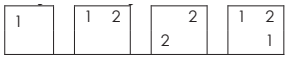


Рис. 8. Матрицы операций на 4-ом шаге прохода.

Получается, что весь процесс разбивается на 2 части: заполнение матрицы – претендента на полугруппу, затем поиск пути до последнего столбца большой матрицы.

После заполнения матрицы-претендента на полугруппу можно найти не один путь, а несколько, а нам не интересны они все. Интересно есть он или нет. Т.е. нам надо найти всего лишь один из них.

С точки зрения алгоритма, возврат по дереву решений происходит в точку, где было последнее заполнение, представим в виде рисунка 9. Кружочки это матрицы с правильными расположениями элементов.

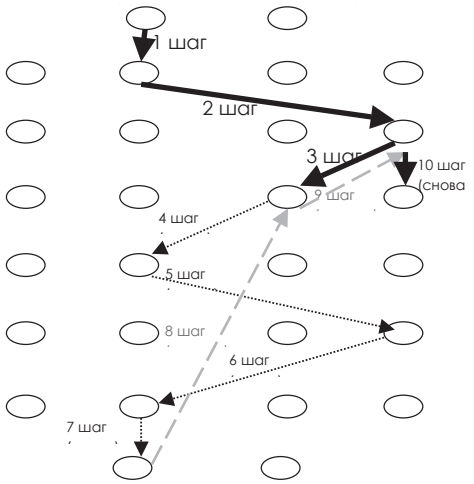


Рис. 9. Схема прохода по таблице матриц при поиске в глубину.

Получается несколько срезанный алгоритм поиска в глубину.

Второй вариант.

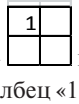
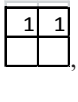
У нас есть большая матрица, изображенная на рис. 8. Создадим вторую такую, в которой элементами будут бинарные матрицы.

Например, в строке «a=1 b=1» в столбце «111» будет находится элемент, значение которого представлено на рис. 10.

1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
	0	1	1	0	1	1	
	0	1	1	0	1	1	

Рис. 10. Бинарная матрица наложения для тройки 111; a = 1; b =1.

Как видно структура повторяет структуру большой матрицы. Цифра 1 ставится в том случае, если элемент, находящийся в позиции – определенная строка и определенный столбец в большой матрице хорошо накладывается на какой-либо другой элемент, 0 – не накладывается.

Например, строка «a=1 b=1» столбец «111»  накладывается на элемент в строке «a=1 b=1» столбец «112»  , поэтому в этом месте единица (рис.11).

1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
	0	1	1	0	1	1	
	0	1	1	0	1	1	

Рис. 11. Операции троек 111; a = 1; b = 1 и 112; a = 1; b = 1 накладываются.

Например, строка «a=1 b=1» столбец «111»  не накладывается на элемент в строке «a=2 b=1» столбец «112»  , поэтому в этом месте 0 (рис. 12).

1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1
	0	1	1	0	1	1	
	0	1	1	0	1	1	

Рис. 12. Операции троек 111; a = 1; b = 1 и 112; a = 2; b = 1 накладываются.

Получается массив бинарных массивов (рис. 13).

Теперь, точно также ищем путь, как и в варианте № 1. Меняется то, что по второй таблице можно определить побитово, какие матрицы можно накладывать.

Для того же примера, что и в варианте 1 (рис. 9), на каждом шаге для принятия решения, (какую матрицу дальше можно взять) необходимо иметь значение следующего побитово сложенного столбца.

Например, для принятия решения на 5 шаге в примере, нам необходимо взять 5 столбец из всех битовых матриц, которые были выбраны на предыдущих столбцах (рис. 14).

Теперь необходимо побитово перемножить столбцы (рис. 15).

Соответственно можно выбрать только 2 матрицу в 5 столбце.

	111	112	121	122	211	212	221	222
a=1 b=1	1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1	0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1	0 0 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0	….	….	….	….	….
a=1 b=2	0 0 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 1 1 1 1 1 1	0 0 0 0 1 0 1 0 1 0 0 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1	0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1	….	….	….	….	….
a=2 b=1		0 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 1	0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0	….	….	….	….	
a=2 b=2		0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1	0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 1	….	….	….	….	

Рис. 13. Часть таблицы бинарных матриц наложения.

На 1 шаге	На 2 шаге	На 3 шаге	На 4 шаге	На 5 шаге	…	…	…
1 1 1 1	1 1 0 0	0 1 0 1	0 1 0 0	…	…	…	…

Рис. 14. Таблица пятых столбцов из выбранных матриц наложения.

1	1	0	0	**0**
1	1	1	1	**1**
1	0	0	0	**0**
1	0	1	0	**0**

Рис. 15. Результат побитового умножения столбцов.

Параллелизм алгоритма:

Ветки поиска путей по лабиринту друг от друга не зависят, поэтому параллелить можно начинать из любой точки. На старте получается (для двух элементного множества) 2 ветки, далее можно еще распараллелить каждую ветку на 4 ветки. Каждая нить будет искать количество путей (в соответствии с рисунком 9) из ее точки создания.

КРАТКОЕ ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

В настоящее время ведется разработка параллельной программы для данного алгоритма. В планах автора настоящей работы – продолжение исследований, улучшение полученных результатов, точное сравнение трех подходов, а в перспективе – существенное улучшение результатов Andreas'a Distler'a.

Работа первого автора частично поддержана региональным грантом РФФИ № 13-01-97003, а также поддержана программой Министерства образования и науки в рамках Госзадания Тольяттинского государственного университета на 2012 год (шифр 6.3072.2011).

Работа второго автора частично поддержана программой Министерства образования и науки в рамках Госзадания Тольяттинского государственного университета на 2012 год (шифр 6.3072.2011), а также поддержан Федеральной целевой программой «Научные и научно-педагогические кадры инновационной России» на 2009–2013 годы (соглашение № 14.B37.21. 0233).

СПИСОК ЛИТЕРАТУРЫ

1. Шеврин Л.Н. Что такое полугруппа // Соросовский Образовательный Журнал. 1997. № 4. С. 99–104.
2. Полугруппа [Электронный ресурс]: Материал из «Викия-сеть»: Версия 3, сохранённая в 17:17 UTC 2 сентября 2010 / Авторы Викия-сеть// Викия-сеть. – Режим доступа: http://ru.math.wikia.com/wiki/Полугруппа
3. Клиффорд А., Престон Г. Алгебраическая теория полугрупп. – М.: Мир, 1972. – 286 с.
4. Semigroup with two elements [Электронный ресурс]: Материал из Википедии – свободной энциклопедии: Версия 488388842, сохранённая в 20:24 UTC 20 апреля 2012 / Авторы Википедии // Википедия, свободная энциклопедия. – Электрон. дан. – Сан-Франциско: Фонд Викимедиа, 2012. – Режим доступа: http://en.wikipedia.org/w/index.php?title=Semigroup_with_two_elements&oldid=488388842
5. Кузичкина Е.И. Переборные алгоритмы подсчёта числа конечных полугрупп – постановка задачи и простейшие эвристики//Вектор науки ТГУ №4, 2012
6. Мельников Б. Мультиэвристический подход к задачам дискретной оптимизации. – Кибернетика и системный анализ (НАН Украины), 2006, № 3, С. 32–42.
7. Classification and enumeration of finite semigroups [Электронный ресурс]: Research@StAndrews Full Text: Версия: 10023/945, сохраненная в июне 2010 / Andreas Distler // Research@StAndrews Full Text – Режим доступа: http://hdl.handle.net/10023/945

PARALLEL IMPLEMENTATION OF SOME ALGORITHMS OF COUNTING THE NUMBER OF NONISOMORPHIC FINITE SEMIGROUPS

E.I. Kuzichkina, postgraduate student
D.I. Vlasov, student
Togliatti state university, Togliatti (Russia)

Keywords: finite semigroup; exhaustive search; heuristics; distributed computing; parallel algorithms.

Annotation: Considered heuristic search algorithms for calculation of the number of finite semigroups and options for parallelization.

УДК 519.178

НЕКОТОРЫЕ ПОДЗАДАЧИ ЗАДАЧИ ВЕРШИННОЙ МИНИМИЗАЦИИ НЕДЕТЕРМИНИРОВАННЫХ КОНЕЧНЫХ АВТОМАТОВ

М.В. Кукеев, аспирант
Тольяттинский государственный университет, Тольятти (Россия)

Ключевые слова: базисный автомат; покрывающий автомат; параллельная модель вычисления CUDA.

Аннотация: В статье рассматриваются подзадачи нахождения циклов базисного автомата и проверка их наличия в покрывающем автомате, а также алгоритмизация этих подзадач для параллельной модели вычисления.

ВВЕДЕНИЕ

Решение задачи минимизации детерминированного конечного автомата Рабина-Скотта было придумано уже достаточно давно. Тогда казалось, что это является конечной точкой в теории регулярных языков. Однако, со временем, стала очевидна необходимость использования и других формализмов в теории регулярных языков. Недетерминированный конечный автомат оказался очень удобным средством во многих прикладных задачах теории регулярных языков. Например, недетерминированные конечные автоматы активно используются для построения компиляторов, также удобно использовать недетерминированность в построении доказательств теорем в теории регулярных языков. Системы поиска слов, подстрок, предложений в разных приложениях используют недетерминированные конечные автоматы, как удобное средство построения таких алгоритмов поиска.

В современных условиях возникают задачи построения минимального по какому-либо критерию недетерминированного конечного автомата. Вершинная минимизация позволяет минимизировать состояния конечного автомата. Это полезно, например, для компактного представления этого автомата в памяти компьютера.

Так как для задачи вершинной минимизации недетерминированных конечных автоматов пока не существует эффективных алгоритмов, имеет смысл пытаться улучшить существующие алгоритмы или написать вспомогательные, которые позволят в какой-нибудь степени расширить возможности практического применения этих алгоритмов. Для расширения практического применения можно попытаться уменьшить время исполнения алгоритмов за счет применения различных технологий. Например, можно распараллелить алгоритм для его исполнения в параллельных моделях вычислений.

В данной работе используется модель параллельного вычисления, основанная на архитектуре CUDA (Compute Unified Device Architecture – унифицированная архитектура вычислений) [1], которая является перспективной во многих научных направлениях. И, хотя появилась на рынке эта архитектура сравнительно недавно (примерно в 2006 году), уже сейчас она активно внедряется в научные «круги».

Использование архитектуры CUDA в задачах с объёмными вычислениями часто даёт приличный прирост производительности, а если уж задача изначально располагает к параллелизму, то равных архитектуре CUDA по параметру цена/производительность нет.

ОБЩИЕ ПОНЯТИЯ

Под минимизацией конечного автомата понимают нахождение эквивалентного конечного автомата,