О применении системы компьютерной алгебры в параллельных средах для решения одной задачи теории групп*

А.И. Макосий, А.В. Тимофеенко
Хакасский государственный университет
Красноярский педагогический университет

На примере задачи поиска порождающего множества инволюций с определенными условиями рассматриваются вопросы применения параллельных вычислений в теории групп.

Немногим более пятидесяти лет назад в исследовании по теории групп были использованы компьютеры. С этого момента, когда впервые была продемонстрирована мощь компьютеров в алгебраических исследованиях, компьютерная индустрия существенно поменяла свой облик и возможности. Наряду с метаморфозой в технологии, параллельно в алгебре, произошла разительная перемена в способах и методах исследования. Сегодня компьютерные вычисления не только облегчают математическое открытие, они могут являться неотъемлемой частью доказательства теорем.

И если вначале опыт применения вычислений в каждом случае был уникальным, то сейчас этот процесс «индустриализирован» компьютерными системами *GAP* и *MAGMA*. Их широкое использование как инструментов исследования алгебраических и комбинаторных структур, в свою очередь, поставило множество алгоритмических и иных вопросов, повлекших дальнейшее развитие самой теории групп. Очень важным и интересным аспектом компьютерных доказательств является то, что после получения результата часто его можно проверить без использования компьютеров или переформулировать результат, не используя при его доказательстве вычисления или минимизируя их использование.

Рассмотренная ниже задача может быть отнесена к классу комбинаторных задач. За редким исключением перебор (порождение) элементов группы и их опробование (тестирование на обладание нужным свойством) является единственным способом решения такого рода задач. Чаще всего алгоритмы решения имеют экспоненциальную сложность и требуют, как правило, больших вычислительных ресур-

* Работа поддержана грантами РФФИ № 09-01-00395, 10-01-00509, грантом Красноярского госпедуниверситета № 16-11-2/МП.

сов. Поэтому даже в параллельном варианте они имеют предел для своего применения. Тем не менее «параллелизация» вычислений приводит к существенному уменьшению времени расчетов.

Модели групп и все вычисления с ними были реализованы с использованием системы компьютерной алгебры *GAP* [1]. Вычисления проводились на суперкомпьютере Межведомственного суперкомпьютерного центра, а также вычислительном кластере Красноярского государственного педагогического университета и кластере Института вычислительного моделирования СО РАН (г. Красноярск).

Предварительные сведения

Напомним определения некоторых основных используемых понятий из теории групп. Недостающие сведения могут быть найдены в учебнике [3].

Пусть G – произвольная группа. Элементы $a,b \in G$ называются сопряженными в G, если $a=g^{-1}bg$ для некоторого элемента $g \in G$. В этом случае также используется обозначение $a=b^g$. Множество $h^G = \{h^g \mid g \in G\}$ называется классом сопряженных с h в группе G элементов. Если H – подгруппа группы G, то множество $g^{-1}Hg$ называется *сопряжением* подгруппы H с помощью элемента g и обозначается как H^g.

Подгруппа H удовлетворяющая условию $H^g \subseteq H$ для любого элемента $g \in G$ называется *нормальной подгруппой* (нормальным делителем) группы G, и записывается как $H \lhd G$. Группа, не содержащая нормальных подгрупп, называется *простой*. Исключительная роль конечных простых групп объясняется тем, что из них может быть построена любая конечная группа.

Пересечение любого количества подгрупп вновь является подгруппой. Это позволяет определить подгруппу, порожденную множеством M, как наименьшую подгруппу, содержащую подмножество M, т.е. пересечение всех подгрупп группы G, содержащих множество M. Подгруппа, порожденная множеством M, будет обозначаться как $<M>$. Легко проверить, что $<M>$ является множеством всевозможных произведений элементов из M и обратных к ним. *Инволюцией* группы G называется ее элемент порядка два, т.е. $i \in G, i \neq 1$, такой что $i^2=1$.

Теория конечных групп – старейшая и в то же время наиболее активно развивающаяся ветвь теории групп. Одним из крупнейших результатов явилось завершение классификации конечных простых неабелевых групп, включающей серии групп лиева типа, знакопеременные группы A_n при $n \geq 5$ и 26 спорадических групп. Среди спорадических одна из наибольших групп Baby Monster имеет порядок

$2^{41}\cdot3^{13}\cdot5^{6}\cdot7^{2}\cdot11\cdot13\cdot17\cdot19\cdot23\cdot31\cdot47$, а ее элементы могут быть представлены как матрицы размерности 4370 и уже обычное умножение элементов этой группы представляет собой нетривиальную вычислительную задачу.

При исследовании групп, прежде всего, интересен вопрос о строении (описании) множества групп с некоторыми заданными свойствами. Структурная теория групп в настоящее время представляет одну из самых развитых областей алгебры и математики в целом. Наиболее важные свойства конечных групп стали оформлять в виде сборника – атласа конечных групп [2]. В наш век повсеместного использования Интернета появился его электронный аналог – электронный атлас конечных групп [4].

О порождении групп тремя инволюциями, две из которых перестановочны

Ряд задач теории групп и ее приложений сводятся к проблеме нахождения множества порождающих элементов, удовлетворяющих определенным свойствам. Для конечных простых групп и близких к ним наибольший интерес вызывают порождающие множества минимальной мощности, в которых особую роль играют инволюции.

Всякая конечная простая неабелева группа содержит инволюции и порождается любым классом сопряженных инволюций. Естественно возникает вопрос: каково минимальное число инволюций (необязательно сопряженных), порождающих конечную простую неабелеву группу? К 90-м годам прошлого века стало известно, что тремя инволюциями порождена каждая конечная простая неабелева группа, исключая группу $U_3(3)$ [4].

Несколько лет назад в работах Я.Н. Нужина и В.Д. Мазурова было выяснено, какие конечные простые неабелевы группы порождаются тремя инволюциями, две из которых перестановочны. Такие тройки инволюций, если они существуют в группе, называются (2×2,2) тройками инволюций этой группы. Препринт [5] положил начало циклу работ, посвященных следующему вопросу:

Указать алгоритмы поиска (2×2,2) троек инволюций в конечных простых неабелевых группах и создать электронный атлас таких троек.

Если (i, j, k) – (2×2,2) тройка инволюций конечной простой группы G и $|ij|=2$, $|ik|=p$, $|jk|=q$, то определим множество $C_2(G)$ как множество всех таких упорядоченных пар чисел (p,q). Таким образом, две (2×2,2)

тройки инволюций не различаются, если соответствующие числа p и q равны.

В работе [6] указан алгоритм поиска (2×2,2) троек инволюций в любой конечной (простой) группе, вычислительные возможности которого ограничены только возможностями программного и аппаратного обеспечения. Схема последовательного алгоритма в общем виде может быть описана как процесс формирования массива троек инволюций и цикла перебора-проверки: являются ли они (2×2,2) тройками и формирования выходных данных. Эти процессы условно реализуются процедурой *SubmitTaskInput()* – инициализация данных и организация перебора троек инволюций, *TestTriples()* – проверка на то, что некоторая тройка инволюций является (2×2,2) тройкой и *UpdateData()* – формирование множества $C_2(G)$.

Методы и программное обеспечение распараллеливания расчетов

Для распараллеливания расчетов применялись два подхода. Первый подход базируется на использовании *ParGAP (Parallel GAP)* [6] – сторонней разработки, предназначенной для работы с *GAP* и позволяющей создавать параллельные программы на языке *GAP*. *ParGAP* реализован на подмножестве стандарта *MPI*.

Декомпозиция последовательного алгоритма была осуществлена сочетанием функциональной декомпозиции и декомпозиции данных. Уменьшение объема обмена данными между процессами было получено за счет того, что подчиненному процессу фактически выделялся некоторый пул номеров, по которому он формировал множество троек инволюций группы, и проверялось, являются ли они (2×2,2) тройками, а уже главный процесс решал – пополнять такой тройкой (если ее там еще нет) множество $C_2(G)$ или нет. При этом также была получена полная загрузка узлов во время счета. Для возобновления расчетов была организована обработка прерываний.

Проверка данных была разбита на две части – локальную (предварительную) проверку *TestTriples()* на узлах и окончательную *CheckTaskResult()* на главном процессе. Вначале главный процесс инициализирует глобальные и разделяемые данные. Глобальные данные доступны всем узлам. Они используются в процессе расчетов в режиме «только для чтения». Разделяемые данные доступны процедуре *UpdateData()* в режиме «чтение–запись», в то время как для *SubmitTaskInput(), TestTriples()* и *CheckTaskResult()* в режиме «только для чтения». Разделяемые данные должны быть вначале проиниция-

лизированы на всех процессах, а затем модифицироваться только процедурой *UpdateData()* на главном процессе. Обработка прерываний реализована при помощи стандартного механизма организации контрольных точек. «Зависания» подчиненных процессов обрабатываются внутренними механизмами *ParGAP*. Для старта задачи потребовалась небольшая адаптация скрипта инициализации заданий *mpirun*. «Жизненный цикл» работы алгоритма представлен на рис. 1:

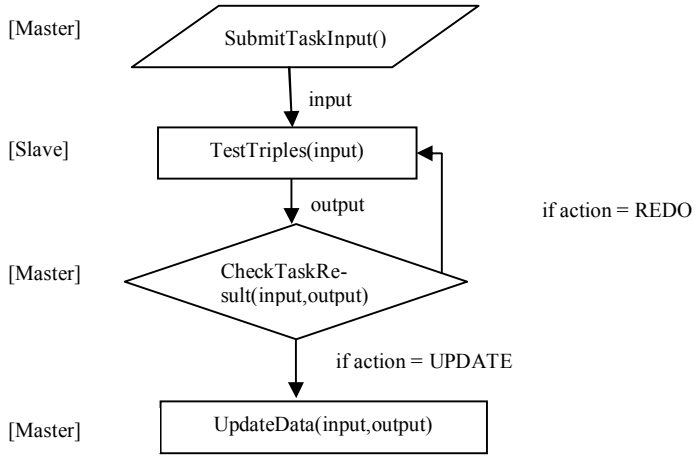

Рис. 1

Второй способ параллелизации основан на возможности прямой декомпозиции исходных данных и выполненения последовательного кода на каждом из узлов. Множества инволюций разбиваются на непересекающиеся множества и для каждого из них выполняется поиск (2×2,2) троек инволюций и построение множества $C_2(G)$. Полученная при этом избыточность расчетов (на разных узлах могут быть получены одинаковые пары чисел (p,q) (порядков неперестановочных инволюций) устраняется проверкой выходных данных после прерывания расчетов и формированием нового множества глобальных данных.

Для запуска *GAP* в параллельной среде была написана на языке *C* программа-стартер. Стартер запускал нужное количество *GAP*-процессов. Каждый процесс использовал свое множество данных и создавал выходные файлы в отдельном каталоге. После прерывания (выполнения) расчетов эти данные модифицировались и формировались единые для всей задачи выходные файлы.

По указанному алгоритму были выполнены расчеты для знакопеременных групп A_{12}–A_{16} и ряда спорадических групп.

Литература

1. **The GAP Group**. The GAP Group, GAP – Groups, Algorithms, and Programming, v. 4.4.12, 2008. http://www.gap-system.org.
2. **Conway J. H., Curtis R. T., Norton S. P. et al.** Atlas of finite groups. – Oxford: Clarendon Press, 1985. – 252 p.
3. **Каргополов М.И., Мерзляков Ю.И.** Основы теории групп. – М.: Наука, 1996.
4. **Wilson R., Parker R.A., Bray J.N.** ATLAS of Group Representations. http://web.mat.bham.ac.uk/atlas/v2.0.
5. **Нужин Я.Н., Тимофенко А.В.** Порождающие тройки инволюций некоторых спорадических групп. Препринт №13–99. – Красноярск: ИВМ СО РАН. 1999. – 20 с.
6. **Макосий А.И.** О порождающих множествах инволюций конечных групп и смежные вопросы. Вычислительный подход. – LAP LAMBERT Academic Publishing GmbH, 2011. – 78 с.
7. **Cooperman G.** Parallel GAP/MPI (ParGAP/MPI), College of Computer Science, Northeastern University, http://www.ccs.neu.edu/home/gene/pargap.html