УДК 004.421.2:004.032.24:512.714+

Параллельная реализация алгоритма вычислений базисов Гребнера и Жане на уровне редукции полиномов

Д. А. Янович

Лаборатория информационных технологий
Объединённый институт ядерных исследований
ул. Жолио-Кюри д.6, Дубна, Московская область, 141980, Россия

В предыдущих работах был представлен алгоритм для параллельного вычисления базисов Гребнера и Жане, работающий в терминах параллельного вычисления нормальных форм. Реализация выглядела многообещающе, но столкнулась с проблемой «голодания»(т.е. в некоторые моменты времени загружено мало вычислительных ядер машины). В этой работе представлен один подход повышения масштабируемости и избежания «голодания». Представлены данные по ускорению вычислений на восьмиядерной SMP машине.

Ключевые слова: базис Гребнера, базис Жане, параллелизация, масштабируемость.

1. Введение

1.1. Основные обозначения и определения

Будем использовать следующие обозначения:
$\mathbb{X} = \{x_1, \ldots, x_n\}$ — множество полиномиальных переменных;
$\mathbb{R} = \mathbb{Q}[\mathbb{X}]$ — кольцо многочленов с рациональными коэффициентами;
$Id(F)$ — идеал кольца $\mathbb{R}$, порождённый многочленами $F \subset \mathbb{R}$;
$\mathbb{M}$ — множество мономов, т.е. произведение степеней переменных из $\mathbb{X}$ с целыми неотрицательными показателями;
$\deg_i(u)$ — степень переменной x_i в мономе $u \in \mathbb{M}$;
$\deg(u) = \sum_{i=1}^{n} \deg_i(u)$ — полная степень монома u;
$\succ$ — допустимый порядок на мономах, такой что $x_1 \succ x_2 \succ \cdots \succ x_n$;
$u \mid v$ — обычное отношение делимости монома v мономом u. Если $u \mid v$ и $\deg(u) < \deg(v)$, т.е. если u является *собственным делителем* v, будем записывать это условие как $v \sqsupset u$;
$\mathrm{lm}\,(f)$ и $lt(f)$ — старший моном и старший одночлен многочлена $f \in \mathbb{R} \setminus \{0\}$, соответственно;
$\mathrm{lm}\,(F)$ — набор старших мономов полиномиального множества $F \subset \mathbb{R} \setminus \{0\}$.
Для каждого $1 \leqslant i \leqslant n$ разобьём конечное множество многочленов $F \in \mathbb{R} \setminus \emptyset$ на группы, индексированные неотрицательными целыми числами $d_1, \ldots, d_i$

$$[d_1, \ldots, d_i] = \{f \in F \mid d_j = \deg_j(\mathrm{lm}\,(f)),\ 1 \leqslant j \leqslant i\}.$$

Переменная x_1 является *J-мультипликативной (мультипликативной по Жане)* [1] для многочлена $f \in F$, если $\deg_1(\mathrm{lm}\,(f)) = \max\{\deg_1(\mathrm{lm}\,(g)) \mid g \in F\}$. Для $i > 1$ x_i является *J-мультипликативной* для $f \in [d_1, \ldots, d_{i-1}]$, если

$$\deg_i(\mathrm{lm}\,(f)) = \max\{\deg_i(\mathrm{lm}\,(g)) \mid g \in [d_1, \ldots, d_{i-1}]\}.$$

В дальнейшем будем обозначать через $M_J(f, G) \subset \mathbb{X}$ и $NM_J(f, G) = \mathbb{X} \setminus M_J(f, G)$ множество J-мультипликативных и J-немультипликативных переменных для $f \in G$ соответственно.

Работа частично поддержана грантом РФФИ 07-01-00660 и грантом 1027.2008.2 Министерства образования и науки РФ.

Данное разделение переменных на немультипликативные и мультипликативные переменные порождает инволютивной деление мономов [2,3]. Это деление — деление Жане — определяется по заданному конечному набору многочленов F и порядку на мономах $\succ$ следующим образом: если мономы $u \in lm(F)$, v и w связаны соотношением $w = u \cdot v$ и при этом моном v содержит только мультипликативные (по Жане) переменные для u, то u является *делителем Жане или J-делителем* монома w. В этом случае будем записывать отношение инволютивной делимости по Жане как $u \mid_J w$.

Конечное множество F ненулевых многочленов является *J-авторедуцированным* (или авторедуцированным по Жане), если каждый моном, входящий в $f \in F$, не имеет J-делителей среди $\mathrm{lm}(F) \setminus \{\mathrm{lm}(f)\}$. *$J$-нормальная форма* $NF_J(p, F)$ многочлена $p \notin F$ по отношению к J-авторедуцированному множеству F определяется как

$$NF_J(p, F) = \tilde{p} = p - \sum_{ij} \alpha_{ij} m_{ij} g_j,$$

где $\alpha_{ij} \in \mathbb{K}$, $g_j \in F$, $m_{ij} \in L(\mathrm{lm}(g_j), \mathrm{lm}(F))$, $\mathrm{lm}(m_{ij} g_j) \preceq \mathrm{lm}(p)$ и $\tilde{p}$ не содержит мономов, имеющих Жане делителя среди $\mathrm{lm}(F)$.

Вычитание из многочлена p указанных в выражении для $NF_J(p, F)$ слагаемых под знаком суммы называется *мультипликативным приведением (редукцией) p* многочленами из F. Когда $\mathrm{lm}(f)$ ($f \notin F$) не имеет J-делителей среди элементов $\mathrm{lm}(F)$, многочлен f находится в *головной J-нормальной форме* по отношению к F. Этот факт будем записывать как $f = HNF_J(f, F)$.

Для заданного идеала $I \subset \mathbb{R}$ и порядка на мономах $\succ$ конечное J-авторедуцированное множество ненулевых многочленов $G \subset \mathbb{R}$, порождающее I, является его *базисом Жане*, если выполнено следующее [2]

$$(\forall f \in G)(\forall x_i \in NM_J(f, G))[NF_J(x_i \cdot f, G) = 0].$$

Произведение $x_i \cdot f$ многочлена $f \in F$ и $x_i \in NM_J(f, F)$ называется *немультипликативным продолжением f*. Тем самым для базиса Жане любое его немультипликативное продолжение приводится (редуцируется) мультипликативно к нулю.

Базис Жане является базисом Гребнера, хотя и не обязательно авторедуцированным в смысле гребнеровских редукций [2]. Подобно редуцированному базису Гребнера, минимальный базис Жане с нормированными на единицу старшими коэффициентами, определяется единственным образом по идеалу и порядку на мономах [4].

1.2. Предыдущие результаты

Приведём сокращённый вариант параллельного алгоритма для вычислений базисов Гребнера и Жане (подробное описание, теоретические доказательства его корректности и практические результаты вычислений можно найти в [5–8]). Также в работах [5, 8] можно найти обзор результатов попыток параллелизации в мировой науке.

Ассоциируем каждый полином f в алгоритме с тройкой $p = \{f, u, vars\}$, где $\mathrm{pol}(p) = f$ — сам полином, $\mathrm{anc}(p) = u$ — лидирующий моном предка f из T, $\mathrm{nmp}(p) = vars$ — набор переменных (возможно пустой).

1: $T := \emptyset \quad Q := F \quad G := \emptyset$
2: **while** $Q \neq 0$ **do**
3: $\quad S := \emptyset \;\; P := \{\, q_i \in Q \mid i \leqslant K_{thr}, q_i - \min \in Q \,\}$
4: $\quad Q := Q \setminus P$
5: $\quad$ **create** $\min\{\mathrm{card}(P), K_{thr}\}$ **WorkerThread**$(P, T, S, NFL, mutex)$
6: $\quad$ **wait** for return of all threads
7: $\quad Q := Q \cup S$
8: $\quad T := T \cup \{p \mid lm(pol(p)) = \min(lm(Q))\} \;\; Q := Q \setminus \{p\}$
9: $\quad Q := Q \cup \{p \cdot x_i \mid x_i \in \mathrm{nmp}(p)\}$
10: $\quad$ **if** $\mathrm{lm}(\mathrm{pol}(p)) = \mathrm{anc}(p)$

11: **for all** $\{ r \in T \mid lm(\mathrm{pol}(r)) \sqsupset \mathrm{lm}(\mathrm{pol}(p)) \}$
12: $Q := Q \cup \{r\}; \quad T := T \setminus \{r\}$
13: **od**
14: **create** $\min\{\mathrm{card}(T), K_{thr}\}$ **WorkerThread**$(T, T, S, PNF, mutex)$
15: **wait** for return of all threads
16: $T := S$
17: **fi**
18: **od**

Схематично алгоритм состоит из трёх шагов:

3-7 Параллельное вычисление нормальных форм полиномов с точностью до лидирующего монома.
8-13 Обновление набора полиномов, которые необходимо редуцировать в ходе алгоритма.
14-16 Параллельное вычисление полной нормальной формы новых элементов базиса.

2. Профилирование и оптимизация

После того, как в нашем распоряжении появился восьмиядерный SMP-сервер, были проведены эксперименты по определению масштабируемости реализации описанного алгоритма. К сожалению, при высоком теоретическом значении, мы получили низкое практическое. Фактически, реализация на любом количестве задействованных ядер ускорялась не более, чем в два раза на большинстве тестовых примеров из баз [9, 10].

Для исследования причин подобного поведения необходим был профилировщик потоков. Испытания общеизвестных инструментов показали их недостаточную эффективность, информативность и сильное влияние на время вычислений. В результате был самостоятельно написан подобный инструмент с использованием подсистемы `/proc/pid/stat` ядра Linux со следующими характеристиками:

- размер: около 30 строк C-кода в основной программе, 1, 5 Kb кода на языке Python для разбора результатов и построения графиков загрузки;
- влияние на производительность: один системный вызов `fork()`, +1% нагрузки на CPU (при профилировании восьми потоков).

В результате исследований поведения различных примеров был сделан вывод, что при параллельной редукции некоторых наборов полиномов возникает «голодание» — быстро вычисляются нормальные формы всего набора, за исключением нескольких полиномов. При этом совокупная нагрузка на систему падает с полной загрузки всех ядер до загрузки одного–двух ядер, и общая эффективность сильно снижается.

Рассмотрим подробнее вычисление нормальной формы:

1: **while** $\exists p \in T \ : \ p \mid_L f$ **do**
2: $f := reduce(f, p)$
3: **od**

Таким образом, само вычисление нормальной формы можно разбить на более мелкие независимые части: на первом уровне получаем разбиение на отдельные элементарные редукции одного полинома по другому, на втором — можно разбить саму редукцию на отдельные операции с термами, входящими в состав полиномов.

3. Результаты

Эффективность двух подходов к параллелизации, измеренная на 2xXeon-5410@ 2.33Ghz, 8 ядер, 16Gb RAM. В колонке `1thr` — время вычисления последовательной версии алгоритма, в колонке `NF` вычисления проведены до уровня редукций полиномов, в колонке `RD` — до уровня редукций термов. Колонка `up` — ускорение вычислений в разах.

Как видно из табл. 1, более глубокая параллелизация не всегда полезна — нарастают накладные расходы (ожидание на мьютексах, обращения к общей области памяти и т.п.).

Таблица 1

Эффект различных оптимизаций параллельных вычислений

Пример	1 thr	NF	up	RD	up
cyclic8	4050.68	1231.01	3.29	1280.64	3.16
discret3	6491.38	1079.31	6.01	1079.24	6.01
eco11	714.53	146.86	4.87	166.68	4.29
eco12	7808.16	1287.37	6.07	1860.08	4.20
extcyc6	194.17	37.88	5.13	40.51	4.79
hcyclic8	2136.40	1230.01	1.74	1233.00	1.73
hf855	544.09	181.16	3.00	188.94	2.88
ilias13	2712.76	564.93	4.80	569.67	4.76
ilias_k_3	157.37	36.10	4.36	37.15	4.24
jcf26	75.65	11.01	6.87	10.93	6.92
katsura10	2119.75	649.08	3.27	681.11	3.11
noon8	336.88	168.94	1.99	206.58	1.63
noon9	23615.50	14590.20	1.62	16594.30	1.42
redcyc7	544.27	115.41	4.72	111.57	4.88
redeco11	110.59	31.08	3.56	33.71	3.28
redeco12	1019.84	261.92	3.89	282.64	3.61
reimer7	879.98	524.12	1.68	518.68	1.70
Всего	54426.13	22460.50	2.42	25216.12	2.16

Если провести профилирование потоков после применения оптимизаций, то в «хорошем» случае получим картину загрузки сервера, представленную на рис. 1.

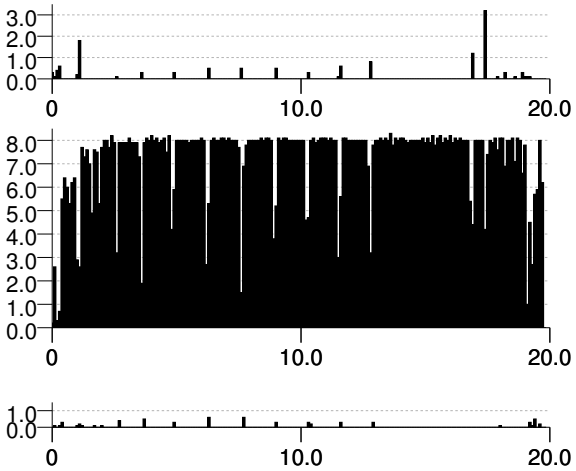

Рис. 1. **График нагрузки на CPU при вычислении примера jcf**

На рис. 1 графики (сверху вниз) визуализируют вычисления полных нормальных форм, вычисления нормальных форм с точностью до лидера, вспомогательные вычисления. Видно, что машина полностью загружена и эффективность высока.

Но найденная оптимизация не покрывает все примеры: один из примеров серии `noon` (рис. 2) выглядит как «плохой» случай — очень много участков, где нагрузка на процессор низка.

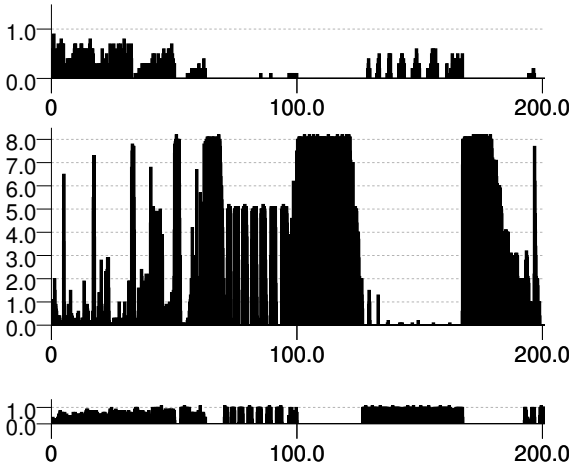

Рис. 2. **График нагрузки на CPU при вычислении примера noon8**

4. Планы на будущее

Дальнейшие работы будут посвящены повышению эффективности вычислений. Первоочередной задачей будет поиск дополнительных оптимизаций для примеров с поведением, характерным для `noon8`.

Литература

1. *Janet M.* Leçons sur les Systèmes d'Equations aux Dérivées Partielles. Cahiers Scientifiques. IV. — Gauthier-Villars, Paris, 1929.
2. *Gerdt V. P., Blinkov Y. A.* Involutive Bases of Polynomial Ideals // Math. Comp. Sim. — 1998. — Vol. 45. — Pp. 519–542.
3. *Гердт В. П., Блинков Ю. А.* Инволютивные деления мономов // Программирование. — 1998. — № 6. — С. 22–24.
4. *Gerdt V. P., Blinkov Y. A.* Minimal Involutive Bases // Math. Comp. Sim. — 1998. — Vol. 45. — Pp. 543–560.
5. *Gerdt V. P., Yanovich D. A.* Parallelism in Computing Janet Bases // Proceedings of CAAP'01, JINR E5,11-2001-279. — Dubna: 2002. — Pp. 93–103.
6. *Янович Д. А.* О распараллеливании алгоритма вычисления инволютивных базисов Жане // Программирование. — 2002. — № 2. — С. 16–21.
7. *Gerdt V. P., Yanovich D. A.* Parallel Computation of Involutive and Gröbner Bases. // «Computer Algebra in Scientific Computing/CASC 2004» / Ed. by E. V. V. V. G. Ganzha, E. W. Mayr; Institute of Informatics, Technical University of Munich. — Garching, 2004. — Pp. 185–194.
8. *Янович Д. А.* Оценка эффективности распределенных вычислений базисов Гребнера и инволютивных базисов // Программирование. — 2008. — № 4. — С. 32–40.
9. *Bini D., Mourrain B.* Polynomial Test Suite. — 1996. — `http://www-sop.inria.fr/saga/POL`.
10. *Verschelde J.* The Database with Test Examples. — `http://www.math.uic.edu/~jan/demo.html`.

UDC 004.421.2:004.032.24:512.714+

Reduction-Level Parallel Computations of Gröbner and Janet Bases

D. A. Yanovich

Laboratory of Information Technologies
Joint Institute for Nuclear Research
Joliot-Curie 6, 141980 Dubna, Moscow region, Russia

In previous papers we presented algorithm for parallel calculation of Gröbner and Janet bases that works in terms of parallel normal forms computations. The realization was quite promising but faced problem of "starvation" (e.g. in some moments of time only few processors was fully loaded). In this talk one approach to raise scalability and avoid "starvation" will be presented. Experimental results of parallel computations on eight core SMP machine will be shown. Work was partially supported by the RFBR grant 07-01-00660 and by the grant 1027.2008.2 of the Ministry of Education and Science of the Russian Federation.

Key words and phrases: Gröbner bases, Janet bases, parallel calculation, scalability.