УДК 519.71 Вестник СПбГУ. Сер. 1. 2010. Вып. 4

ПАРАМЕТРИЧЕСКИ РАЗДЕЛИМЫЕ АЛГОРИТМЫ*

С. М. Ермаков

С.-Петербургский государственный университет,
д-р физ.-мат. наук, профессор, ermakov@math.spbu.ru

1. Введение

Работа обобщает и развивает полученные ранее результаты [10]. Напомним кратко, но с более общих позиций, существо рассматриваемой проблемы. При этом мы будем использовать (без доказательств) ряд результатов работ [1–3].

Когда ставится вопрос о привлечении многих процессоров к решению некоторой задачи, обычно рассматривается класс последовательных алгоритмов, и выбирается алгоритм, наиболее подходящий для распараллеливания. Это, как правило, «хороший» алгоритм (высокая степень аппроксимации, высокая точность интегрирования и т. п.). Можно ожидать, однако, что при очень большом числе процессоров ($\approx 10^4 - 10^5$) более простые (более грубые) алгоритмы могут оказаться более выгодными с точки зрения загрузки оборудования, чем «хорошие» в упомянутом выше смысле.

С другой стороны, непомерно большие объемы данных, возникающие при решении ряда современных задач, нуждаются в сжатии, что может быть достигнуто методами статистической обработки как предварительной, так и применяемой в процессе решения задачи. Это приводит к развитию стохастических вычислительных методов — различных модификаций метода Монте-Карло в том числе.

Известно, что алгоритмы метода Монте-Карло, как правило, просты и обладают свойствами естественного параллелизма. Если решается задача вычисления интеграла $\int f(x)P(dx),\ x \in X$, по вероятностной мере $P(dx)$, то алгоритм состоит в

1) получении реализаций случайных величин y_i, распределенных по закону P, $i = 1, \ldots, N$;

2) вычислении среднего (оценки интеграла)

$$\frac{1}{N}\sum_{i=1}^{N} f(y_i). \tag{1}$$

Отмечавшийся ранее параллелизм очевиден. Относительно синхронности следует заметить следующее. Обычно N очень велико ($> 10^6$). Если время вычисления f при различных y_i сильно различается, то при числе N_1 процессоров при $N_1 \ll N$ полагаем $N \approx N_1 N_2$. Поручая каждому процессору вычисление суммы N_2 слагаемых, мы можем в подавляющем большинстве случаев существенно улучшить свойства синхронности алгоритма, так как среднее время вычисления f по группам, как правило, подвержено меньшему разбросу.

Далее, алгоритм вычисления f может быть последовательным. В этом случае не нужно специальных средств параллельного программирования. Разумеется, могут воз-

*Работа выполнена при финансовой поддержке РФФИ (грант №08-01-00194).

никать ситуации, когда возможность распараллеливания этого алгоритма непременно должна быть учтена.

2. Параметрически разделимые алгоритмы

Рассматривается алгоритм решения некоторой задачи. Предполагается, что это решение принадлежит нормированному пространству, и погрешность, о которой идет речь далее, понимается в смысле нормы этого пространства.

Относительно алгоритмов делаются следующие предположения.

- Для вычислений доступны N процессоров, каждый из которых в общем случае имеет отдельную память. Число N достаточно велико (практически неограничено).
- Алгоритм $A(z)$ решения задачи зависит от параметра z из некоторого параметрического множества $Z \in \mathbb{N}$. При каждом значении z алгоритм получает решение задачи на i-ом процессоре с погрешностью $\varepsilon_i(z)$.
- В результате обмена данными между процессорами, который осуществляется достаточно редко, может быть вычислено решение с погрешностью

$$\varepsilon_{(N)}(z_1, \ldots, z_N) = \varepsilon_{(N)}(\varepsilon_1(z_1), \ldots, \varepsilon_N(z_N)), \quad \varepsilon_{(N)}(z_1, \ldots, z_N) \longrightarrow 0,\ N \longrightarrow \infty. \quad (2)$$

Рассмотренный алгоритм (1) является частным случаем семейства алгоритмов, удовлетворяющих этим предположениям. Эти алгоритмы мы будем называть параметрически разделимыми (п.р. алгоритмами).

Очевидно, что для алгоритма (1) y_i играет роль параметра, а после вычисления каждого значения f на отдельном процессоре обмен данными для получения окончательного результата производится один раз.

Важным свойством п.р. алгоритмов является то, что эффективная загрузка оборудования при сколь угодно большом N не требует параллельного программирования. При каждом значении параметра Z программа может носить последовательный характер.

Естественными вопросами, которые возникают далее, являются следующие.

1. Не является ли класс п.р. алгоритмов слишком узким? Может быть он ограничивается алгоритмом (1)?
2. Не являются ли п.р. алгоритмы, как правило, слишком трудоемкими по сравнению с другими?

Далее мы постараемся ответить на эти вопросы, опираясь частично на полученные ранее результаты, которые будут приведены без доказательств.

3. Примеры п.р. алгоритмов

Отметим сразу же, что многие прикладные задачи допускают представление решения в виде интеграла по траекториям. Последние аппроксимируются кратными интегралами. Алгоритм вычисления кратных интегралов методами Монте-Карло и Квази Монте-Карло является п.р. алгоритмом. Параметрическое множество состоит из параметров датчика псевдослучайных чисел или параметров квазислучайных последовательностей. Для метода Монте-Карло $\varepsilon_N = O\left(1/\sqrt{N}\right)$, для методов Квази Монте-Карло $\varepsilon_N = O\left((\ln^d N)/N\right)$, где d — кратность вычисляемого интеграла.

Очевидно, что многие задачи вычислительной математики не укладываются в эту схему. Тем не менее, как было показано в работах [1–3], возможно построение п.р. алгоритмов в достаточно широком классе случаев. Действительно, случай обобщенного интегрирования, возникающий, когда не существует интеграл $\int |f(x,y)|\mu(dx)\nu(dy)$, но существует повторный интеграл

$$\int \left(\int |f(x,y)|\nu(dy) \right) \mu(dx), \tag{3}$$

значительно расширяет класс интересующих нас алгоритмов. Особенность, возникающая здесь, состоит в том, что внутренний интеграл должен быть вычислен достаточно точно для всех x, участвующих в дальнейшем расчете. Возникает дополнительная проблема синхронизации. При этом алгоритмы вычисления как внутреннего интеграла при фиксированном x, так и внешнего, могут быть п.р. алгоритмами.

Класс задач, где решение допускает представление в виде (3), уже очень широк, что следует из приведенных ниже фактов.

1. Решение системы линейных алгебраических уравнений

$$X = AX + F, \quad A = \{a_{ij}\}_{i,j=1}^{n}, \quad F = (f_1, \ldots, f_n)^T, \quad X = (x_1, \ldots, x_n)^T \tag{4}$$

допускает представление в виде интеграла по траекториям однородной цепи Маркова с n состояниями при условии

$$\rho(|A|) < 1, \tag{5}$$

где ρ — наибольшее по модулю собственное число матрицы $|A| = \{|a_{ij}|\}_{ij=1}^{n}$.

2. При условии

$$\rho(A) < 1 \tag{6}$$

решение (4) может быть представлено в виде повторного интеграла по аналогичным траекториям.

3. Вычисление повторного интеграла методом Монте-Карло осуществляется рекуррентно. Последовательно строятся оценки векторов

$$Y_k = AY_{k-1} + F, \quad Y_0 = F, \quad k = 1, 2, \ldots \tag{7}$$

Фиксируется N_1, при каждом $k > 0$ вычисляется N_1 независимых оценок Ξ_k и берется их среднее. Таким образом, Ξ_k вычисляется по формуле

$$\Xi_k = \widetilde{A}\Xi_{k-1} + F. \tag{8}$$

Здесь $\widetilde{A}$ — случайная матрица, не зависящая от Ξ_k, и такая, что

$$\mathbb{E}\widetilde{A} = A. \tag{9}$$

Подробное описание одного из алгоритмов такого рода можно найти в [2].

Относительно статистической погрешности вектора Ξ_k заметим следующее. Как следует из утверждений 5.1–5.4 руководства [5], для n^2 вектора Σ_k, полученного в результате векторизации матрицы ковариаций вектора ошибок $\varepsilon_k = \Xi_k - Y_k$, справедливо соотношение

$$\Sigma_k = \left(L + \frac{1}{N_1} M^{'} \right) \Sigma_{k-1} + \mathcal{F}_k; \tag{10}$$

здесь $L = A \otimes A^T$ — кронекеровское произведение матриц A и A^T, а $M^{'}$ — матрица ковариаций оценок $\widetilde{A}$ элементов матрицы A, $\mathcal{F}_k$ — некоторый вектор, не зависящий от Σ_{k-1}.

Как известно [4], наибольшее собственное число матрицы L есть $(\rho(A))^2$ и, следовательно, для наибольшего собственного числа матрицы $L + 1/N_1 M^{'}$ справедливо равенство

$$\Lambda_{N_1} = (\rho(A))^2 + \frac{c}{N_1}, \tag{11}$$

где константа c зависит лишь от величины ковариаций оценок $\widetilde{A}$.

Если две независимые оценки (при одинаковых N_1) получены на разных компьютерах, то в силу их несмещенности их среднее также будет несмещенной оценкой с дисперсией, равной половине дисперсии каждой из оценок.

Но при $\Lambda_{N_1} > 1$ с ростом k дисперсии растут экспоненциально, и вычисления практически невозможны. Таким образом, условие $\Lambda_{N_1} < 1$ является необходимым для устойчивости (стохастической) алгоритма и, следовательно, возможности его распараллеливания. Если N_1 таково, что $\Lambda_{N_1} < 1$, мы можем конструировать п.р. алгоритм, состоящий в независимом проведении вычислений нужное число раз при разных значениях параметров (крупнозернистый параллелизм [3]).

4. Метод Монте-Карло с предобусловливанием

Далее мы приступим к изложению основного результата работы.

Пусть система (4) такова, что

$$\rho(|A|) < 1, \quad \text{но} \quad |\rho(A)| = 1 - \delta, \quad \text{где} \quad \delta > 0 \quad \text{мало}. \tag{12}$$

В этом случае решение (4) представимо в виде интеграла по траекториям согласованной с $|A|$ цепи Маркова [5].

Будем рассматривать один из простейших алгоритмов вычисления этого интеграла.

Если однородная цепь Маркова задана вектором начального распределения $p^0 = (p_1^0, \ldots, p_n^0)$ и переходной стохастической матрицей $\mathbb{P} = \|p_{ij}\|_{ij=1}^n$, то ее траектория $i_0 \longrightarrow i_1 \longrightarrow \ldots \longrightarrow i_k$ длины $k+1$ имеет вероятность $p_{i_0}^0 p_{i_0 i_1} \ldots p_{i_{k-1} i_k}$.

Случайный вектор

$$\left\| \sum_{k=0}^{M} \frac{f_{i_0}^0 a_{i_0 i_1} \ldots a_{i_{k-1} i_k}}{p_{i_0}^0 p_{i_0 i_1} \ldots p_{i_{k-1} i_k}} \mathcal{X}_{i_k}(i) \right\|_{i=1}^{n} \tag{13}$$

является несмещенной оценкой Y_M; здесь $a_{i_{-1} i_0} = p_{i_{-1} i_0} = 1$, $\mathcal{X}_{i_k}(i) = \begin{cases} 1, & i = i_k \\ 0, & i \neq i_k \end{cases}$.

Трудоемкость T_1 вычисления Y_M с помощью формулы (13) есть

$$T_1 = NMt. \tag{14}$$

Здесь N — необходимое число независимых испытаний (число повторений вычисления при различных независимых реализациях траекторий цепи Маркова); N имеет порядок $N \sim \sigma^2/\varepsilon^2$, где σ^2 — максимальная дисперсия компонент вектора (13). Мы считаем σ^2 не зависящей от ε и $N = O(\varepsilon^{-2})$.

Согласно общей теории итерационных методов (см., например, [4]), имеем $M = O\left(\log\varepsilon/\log\rho(A)\right)$. Величина t складывается из трудоемкости моделирования очередного i_k при известном i_{k-1}, трудоемкости операции умножения на $a_{i_{k-1}i_k}/p_{i_{k-1}i_k}$ и трудоемкости пересылки. При полном заполнении строк и точных вычислениях имеем $t \le \log_2 n + c$, где c — натуральное число (умножение, пересылка). Для разреженных матриц n следует заменить на r, где r — максимальное число элементов в строке. Наконец, если строки матрицы P подвергнуть соответствующей предварительной обработке, то t не будет превосходить нескольких единиц и не будет зависеть от n (или r соответственно)(метод Уокера [6]).

Отметим, что случай очень большого числа переменных ($n \sim 10^r$, где r может достигнуть сотни), также укладывается в нашу схему, но требует использования MCQMC-техники (метод Монте-Карло на марковских цепях).

В случае, который мы рассматриваем (δ мало), исходную систему обычно преобразуют так, чтобы матрица $\widehat{A}$ преобразованной системы имела меньшую норму $|\rho(\widehat{A})|$, где $|\rho(\widehat{A})|$ — наибольшее по модулю собственное число (путем введения предобусловливателя). Однако при этом чаще всего оказывается, что $\rho(|A|)$ больше единицы, так что следует использовать алгоритм (7). Возможна альтернатива.

1. Для исходной системы $\rho(|A|) > 1$. Предобусловливатель улучшает ситуацию так, что $\rho(A) > \rho(\widehat{A})$. Дальнейшее зависит от сравнения вычислительной работы при оценке итераций для A и для $\widehat{A}$.
2. $\rho(|A|) < 1$, но $\rho(\widehat{A}) < \rho(A)$. Уменьшение может быть весьма значительным или относительно небольшим.

   Существенно, что в одном случае используется алгоритм (13), а в другом — (7).

Вычислительная работа T_2 в случае алгоритма (7) определяется по формуле

$$T_2 = \widehat{M}\widehat{N}\widehat{t}N_1, \tag{15}$$

при этом

$$\widehat{M} = O\left(\frac{\log\varepsilon}{\log\rho(\widehat{A})}\right), \quad \widehat{N} = O\left(\frac{1}{\varepsilon^2}\right), \tag{16}$$

$\widehat{t}$ зависит, как и t, от способа моделирования распределения соответствующей переходной плотности и от заполненности $\widehat{A}$.

Число повторений N_1 при оценке очередной итерации Y_k оценивается из следующих соображений: для стохастической устойчивости алгоритма необходимо и достаточно, чтобы Λ_{N_1} было меньше единицы. Из формулы (11) следует, что

$$\left(1 - (1 - \rho(\widehat{A}))\right)^2 + \frac{c}{N_1} < 1 \Longleftrightarrow (1 - \hat{\delta})^2 + \frac{c}{N_1} < 1,$$

где $\widehat{\delta} = 1 - \rho(\widehat{A})$. Отсюда следует, что N_1 должно иметь порядок $1/\widehat{\delta}$ в предположении $\widehat{\delta} \ll 1$, то есть

$$N_1 = O(\widehat{\delta}^{-1}). \tag{17}$$

Изложенные соображения создают основу для сравнения трудоемкостей различных вариантов итерационных процессов.

5. Пример

В качестве примера мы сравним асимптотическую трудоемкость методов (7) и (13) при $\rho(|A|) < 1$ для малых $\varepsilon, \delta, \widehat{\delta}$.

При этом предполагается:

1) дисперсии оценок в (7) и (13) примерно одинаковы; в случае (7) имеется в виду дисперсия единичного испытания при переходе от k к $k+1$;

2) вычислительная работа при оценке произведения вектора на матрицу A и $\widehat{A}$ также примерно одинакова; из сказанного выше следует, что речь может идти самое большее о множителе порядка $\log_2 n$;

3) величина $\widehat{\delta} = \delta + \gamma,\ \gamma > 0$, достаточно мала, так что величиной $\widehat{\delta}^2$ можно пренебречь.

При сделанных предположениях

$$T_2 = \frac{\log \varepsilon}{\log \rho(\widehat{A})} \cdot N \cdot \frac{1}{\delta + \gamma} \cdot \widehat{t}.$$

Полагая $t/\widehat{t} = O(1)$ и имея в виду, что $1/\log(1-(\gamma+\delta)) \sim 1/(\gamma+\delta)$, получаем

$$\frac{T_2}{T_1} = \frac{\delta}{(\delta+\gamma)^2}. \tag{18}$$

Мы видим, что хороший результат достижим, если γ имеет порядок $\sqrt{\delta}$. Формальное решение неравенства $0 < \delta < (\delta+\gamma)^2$ есть $\sqrt{\delta} - \delta < \gamma < 1 - \delta$.

Остается выяснить, существуют ли реальные ситуации, для которых выполняются сделанные нами предположения и равенство (18).

Одной из простейших содержательных задач является разностный аналог первой краевой задачи для уравнения Лапласа (Пуассона).

Если область есть s-мерный куб, покрытый сеткой с равным шагом $h = 1/n$ по каждой переменной, то матрица простых итераций A имеет для всех s одинаковые $\rho(A)$, и достаточно рассмотреть одномерный случай.

Имеем

$$y_k = \frac{1}{2}(y_{k+1} + y_{k-1}), \quad k = 1, \ldots, n-1, \quad \rho(A) = \cos \pi h \approx 1 - \frac{(\pi h)^2}{2}. \tag{19}$$

При малых h процесс сходится удручающе медленно. Известным приемом улучшения сходимости является использование метода Зейделя с введением релаксационного параметра (метод последовательной верхней релаксации, описанный в [7]). При этом трудоемкость вычисления оценок решения изменяется, хотя и в большую сторону, но незначительно (например, [8]). Как известно [7], при оптимальном выборе параметра релаксации

$$\rho(\widehat{A}) = \frac{\cos \pi h}{1 + \sin \pi h} \approx 1 - \pi h$$

и γ имеет порядок $\sqrt{\delta}$.

6. Выводы и обобщения

Приведенный пример свидетельствует лишь о том, что вариант метода Монте-Карло с использованием метода верхней релаксации сравним по трудоемкости с обычным методом блуждания по сетке. Заметим, однако [1], что использование метода Квази Монте-Карло в первом случае позволяет получить порядок сходимости $O(1/N)$. В случае же блуждания по сетке мы имеем $O(N^{-1/2})$.

Таким образом, из наших исследований можно сделать вывод, что п.р. алгоритмы могут быть построены и эффективны для широко класса задач. Можно напомнить также [1, 9], что с ростом размерности систем даже в однопроцессорном варианте п.р. алгоритмы решения систем линейных алгебраических уравнений могут быть более эффективными, чем классические итерационные методы.

Очевидно, что многие соображения относительно п.р. алгоритмов для решения линейных систем справедливы в общем случае для линейных операторных уравнений (см. [5]). Многие алгоритмы стохастической аппроксимации также относятся к числу п.р. алгоритмов, но это отдельная обширная тема.

Автор благодарит Ю. К. Демьяновича, прочитавшего рукопись статьи и сделавшего ряд полезных замечаний.

Литература

1. *Ermakov S. M., Rukavishnikova A. I.* Quasi Monte-Carlo Algorithms for Solving Linear Algebraic Equation // MC Methods and Applications. Vol. 12, N 5, 2006. P. 363–384.

2. *Ермаков С. М., Тимофеев К. А., Рукавишникова А. И.* О некоторых стохастических и квазистохастических методах решения уравнений // Вестн. С.-Петерб. ун-та. 2008. Сер. 1. Вып. 4. С. 75–85.

3. *Wagner W., Ermakov S.* Monte-Carlo difference schemes for the wave equations // Monte-Carlo Methods and Appl. 2002. Vol. 18. P. 1–19.

4. *Гантмахер Ф. Р.* Теория матриц. М.: Наука, 1988.

5. *Ермаков С. М.* Метод Монте-Карло в вычислительной математике. Вводный курс. СПб.: Невский Диалект; М.: Бином, 2009. 192 с.

6. *Walker A. J.* New Fast method for generating discrebe random numbers with arbitrary fiquency distributions. Elektronic Letters., 1974 Vol. 10. P. 127–128.

7. *Марчук Г. И.* Методы вычислительной математики. М.: Наука, 1977. 456 с.

8. *Ермаков С. М., Беляева А. А.* О методе Монте-Карло с запоминанием промежуточных результатов // Вестн. С.-Петерб. ун-та. 1996. Сер. 1. Вып. 29, № 3. С. 5–8.

9. *Ермаков С. М.* Дополнение к одной работе по методу Монте-Карло // Ж. Выч. Математики и Мат. Физики. 2001. Т. 41, № 6. С. 991–992.

10. *Ермаков С. М.* О конструировании параллельных алгоритмов в задачах вычислительной математики // Вестн. С.-Петерб. ун-та. 2009. Сер. 1. Вып. 2. С. 15–22.

Статья поступила в редакцию 20 апреля 2010 г.