Параллельная реализация технологий многомерной робастной регрессии для многоядерных вычислительных архитектур*

О.А. Комалева, А.В. Бухановский

В докладе обсуждаются различные способы распараллеливания вычислительных алгоритмов робастной регрессии на примере методов *L*-регрессии (least absolute value), наименьшей медианы квадратов (least median of squares) и наименьших усеченных квадратов (least trimmed squares). Приводятся результаты сравнительного анализа параллельной производительности для некоторых многоядерных процессоров.

1. Введение

Широкое распространение вычислительных систем с многоядерными процессорами приводит к необходимости разработки и (или) модификации алгоритмов, применяемых для решения различных прикладных задач, связанных с ресурсоемким моделированием или обработкой больших объемов данных [1]. В частности, к ним относятся параллельные алгоритмы многомерного статистического анализа [2–3], в том числе построения и исследования регрессионных зависимостей (например, [4]).

В корреляционном приближении для решения регрессионных уравнений обычно используется метод наименьших квадратов (МНК), распараллеливание которого не представляет особых сложностей [5]. Однако оптимальность этого метода доказана только при выполнении строгих статистических предположений об исходных данных; отклонения от этих ограничений встречаются достаточно часто и приводят к непредсказуемым результатам работы МНК. Это делает оправданным использование других методов регрессионного анализа, эффективность которых слабо зависит от статистических свойств исходных данных. Такие методы называются робастными (robust regression).

В отличие от классических матричных алгоритмов, применяемых для процедуры МНК, технологии робастной регрессии не допускают столь очевидного способа распараллеливания. В общем случае они основаны на решении задачи поиска минимума нелинейной многомерной функции, ресурсоемкость вычисления которой линейно зависит от объема выборочных данных. В работе предлагаются параллельные алгоритмы для нескольких видов робастной регрессии применительно к вычислительным системам с общей памятью.

2. Многомерная робастная регрессия

В многомерном регрессионном анализе [6] предполагается, что можно прямо или косвенно контролировать одну или несколько независимых переменных $X_1, X_2, .., X_n$, и их значения вместе с множеством параметров $a_1, a_2, .., a_n$ определяют математическое ожидание зависимой переменной Y. Задача состоит в вычислении оценок параметров с помощью выборочных данных. Обычно задача записывается в форме предположения об адекватности следующей модели зависимости:

$$EY = a_1 EX_1 + a_2 EX_2 + .. + a_n EX_n . \quad (1)$$

На вход регрессионному методу подается m выборочных значений $\{(x_1^{\,i}, x_2^{\,i}, .., x_n^{\,i}), y_i\}_{i=1}^{m}$. При подстановке исходных данных в модель получаем следующее:

$$y_i = a_1 x_1^{\,i} + a_2 x_2^{\,i} + .. + a_n x_n^{\,i} + r_i , \quad (2)$$

* Работа выполнена при частичной финансовой поддержке гранта Intel SPB/Don/118/2006 «Естественные технологии распараллеливания алгоритмов высокопроизводительной обработки данных для многопроцессорных вычислительных комплексов на основе многоядерных микропроцессоров».

где $r_i = y_i - \sum_{j=1}^{n} a_j x_j^i = -(EY - y_i) + \sum_{j=1}^{n} a_j (EX_j - x_j{}^i)$ суть выборочные значения случайной величины r, характеризующей суммарное отклонение случайных величин от регрессии.

Произвольный регрессионный метод сводится к задаче оптимизации многомерной функции $f(a_1, a_2, .., a_n)$, интегрально характеризующей невязку r. В таблице 1 приведены наиболее популярные методы и соответствующие им целевые функции.

Таблица 1. Регрессионные методы и соответствующие им виды функций невязки [7].

Сокращение	**Название метода**	**Функция для оптимизации**
МНК	Метод наименьших квадратов (least squares)	$\sum_{i=1}^{m} r_i^2$
МНАЗ	Метод наименьших абсолютных значений (L-регрессия, least absolute value)	$\sum_{i=1}^{m} \lvert r_i \rvert$
МНМК	Метод наименьшей медианы квадратов (least median of squares)	$\underset{i}{med}(r_i^2)$
МНУК	Метод наименьших усеченных квадратов (least trimmed squares)	$\sum_{k=1}^{h} r_{i_k}{}^2$, где $m/2 < h < m$, $\{i_k\}_{k=1}^{m}$ - перестановка $\{i\}_{i=1}^{m}$, причем $r_{i_w}{}^2 \le r_{i_v}{}^2$, если $w < v$

На практике обычно используется МНК, в основном из-за своей эффективности. Однако результаты работы МНК будут оптимальными только при выполнении стандартных статистических предположений, одним из которых является предположение о нормальности распределения величины r. Это предположение часто не выполняется, например, в случае наличия ошибок в исходных данных. Поэтому оправдано применение робастных регрессионных методов (МНАЗ, МНМК, МНУК), результаты работы которых значительно более устойчивы как к наличию ошибок в исходных данных, так и к виду функции распределения величины r.

3. Вычислительные методы регрессионного анализа

Вычислительную сложность МНК можно оценить как $O(n^2 m + n^3)$, так как задача оптимизации целевой функции $\sum_{i=1}^{m} r_i^2$ может быть сведена к системе линейных уравнений $n \times n$ за линейное относительно m время. Отметим, что на практике $n \le 100$, а m может быть очень большим.

В отличие от МНК, время работы точных алгоритмов для робастных методов пропорционально m^n, для приближенных алгоритмов – m^{n-1} [8]. Они сводятся в общем виде к решению задачи глобальной оптимизации функции невязки из таблицы 1. Для задачи глобальной оптимизации многомерной функции также не существует точных эффективных алгоритмов. Потому для ее решения рекомендуется использовать методы, основанные на случайном поиске [9]. После изучения существующих алгоритмов случайного поиска был выбран алгоритм рекурсивного случайного поиска (РСП, recursive random search) [10].

В том случае, если вектор параметров $a \in D$, D – ограниченная область из R^n, и в области D значения функции изменяются на отрезке $[f_{\min}, f_{\max}]$, то алгоритм рекурсивного случайного поиска состоит из двух этапов: изучение (процедура `exploration(`D`)`) и эксплуатация (`exploitation(`a`,` *radius*`)`). Целью первого этапа является макроскопическое исследование поведение целевой функции и выявление областей, в которых функция может принимать оптимальное значение. Сначала генерируется N точек, по которым определяется граница y_r 10%-множества значений функции. После этого генерирование точек продолжается. Если в очеред-

ной точке a значение функции $f(a) \le y_r$, то в окрестности этой точки запускается процедура `exploitation` (второй этап), которая выполняет локальный случайный поиск. Эта процедура генерирует несколько точек из данной области. Если существует точка y такая, что $f(y) < f(a)$, тогда процесс эксплуатации запускается рекурсивно для окрестности с центром в y, иначе окрестность сжимается и т.д. Процедура выполняется до тех пор, пока радиус окрестности не будет достаточно малым.

После проверки очередных N точек в процедуре `exploration` обновляется значение y_r. Процесс продолжается, пока не будет выполнено заданное количество итераций.

4. Параллельные алгоритмы

Наиболее трудоемким этапом алгоритма РСП применительно к задачам робастного регрессионного анализа является вычисление значения целевой функции, именно его и имеет смысл распараллеливать. Это можно сделать двумя различными способами:

- распараллеливать фрагмент работы процедуры случайного поиска, когда нужно вычислить значения целевой функции в нескольких точках;
- непосредственно распараллеливать вычисления значения целевой функции.

4.1. Параллельная реализация случайного поиска

В исходном варианте возможно распараллелить только цикл вычислений между уточнениями порогового значения y_r. В среднем процедура `exploitation` будет вызвана один раз за этот цикл. Экспериментально установлено, что трудоемкость ее вычисления превышает трудоемкость вычислений всего цикла. Сама процедура `exploitation` распараллеливанию не поддается, так как на каждом ее шаге используются значения, полученные на предыдущих этапах.

Предлагаемая модификация алгоритма заключается в том, что все вызовы процедуры `exploitation` производятся после завершения процесса изучения. В рассматриваемом случае это возможно, так как критерием завершения этапа изучения выбрано заданное ограничение на количество итераций. На рис. 1 приведена схема параллельного алгоритма.

4.2. Параллельное вычисление значения целевой функции

Трудоемкость вычисления целевой функции зависит от вида регрессионного метода. Например, для МНУК и МНМК наиболее ресурсоемкой процедурой является сортировка массива невязок, которую можно выполнять параллельно. Предлагается использовать алгоритм параллельной быстрой сортировки [11]. На рис. 2 приведена схема параллельного алгоритма вычисления значений целевой функции для МНУК на рабочей станции с P процессорами.

5. Сравнительный анализ производительности параллельных алгоритмов

На рис. 3 представлены усредненные результаты измерения времени выполнения итерации программы, реализующей алгоритм МНУК, для разного объема исходных данных m. Вычисления производились на рабочей станции с двухъядерным процессором *Intel Pentium D 930*. Видно, что распараллеливание процедуры случайного поиска привносит значительный выигрыш в производительность: для $m > 1000$ ускорение на двух ядрах составляет 1.8–1.9. В то же время распараллеливание процедуры расчета целевой функции (быстрая сортировка) при $m < 3000$ проигрывает в производительности последовательному алгоритму, а при $m \ge 3000$ ускорение на двух ядрах составляет в среднем всего 1.3. Столь низкий эффект от распараллеливания обуславливается тем, что в ходе выполнения программы постоянно осуществляется переход из последовательного режима выполнения в параллельный. При использовании инструментария *OpenMP* это требует дополнительных затрат по времени на порождение и синхрони-

зацию потоков. Кроме того дополнительный дисбаланс нагрузки появляется из-за особенностей алгоритма параллельной быстрой сортировки.

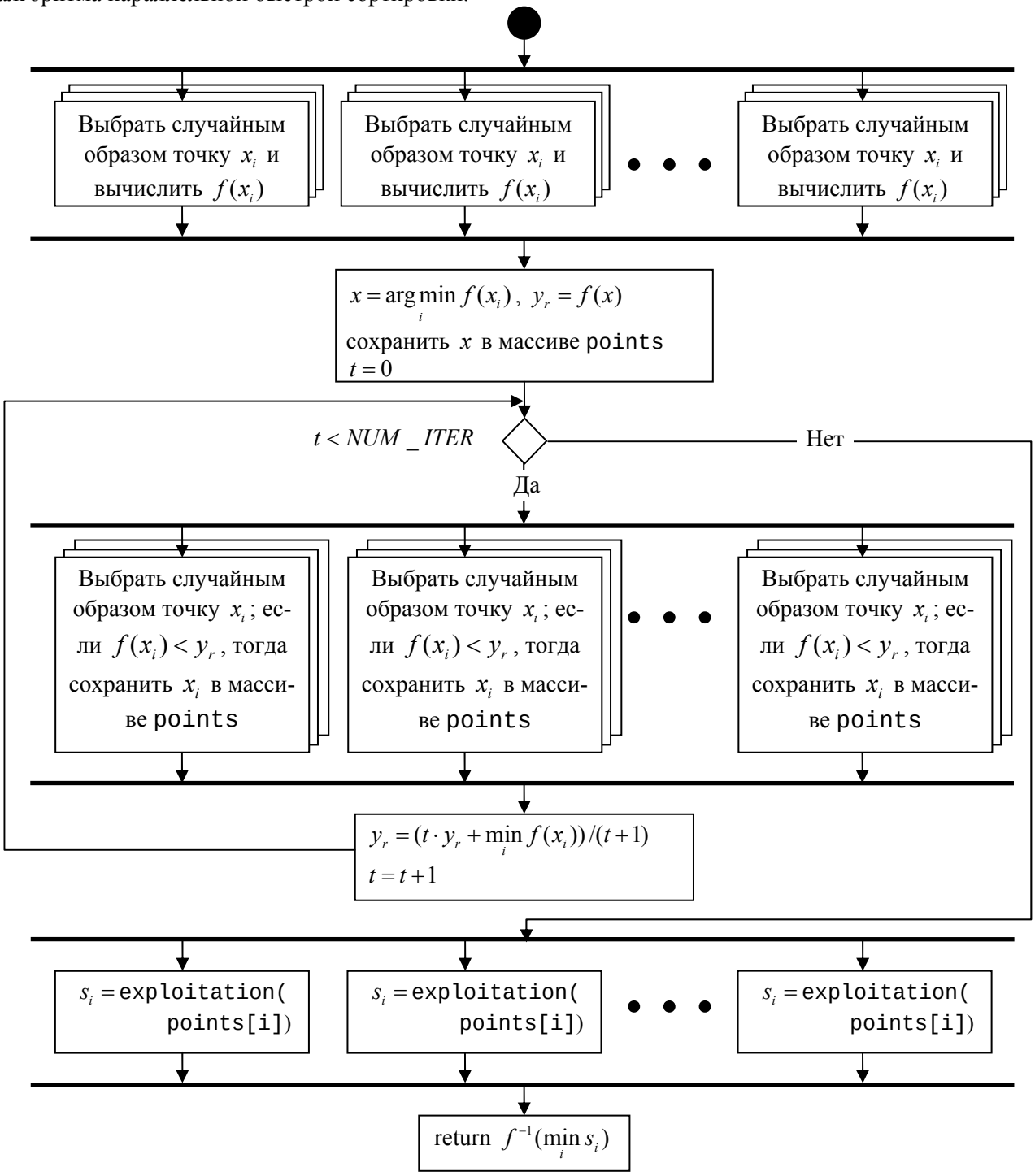

Рис. 1. Схема алгоритма распараллеливания рекурсивного случайного поиска

6. Заключение

Таким образом, в работе предложены два параллельных алгоритма многомерной робастной регрессии для систем с общей памятью. По испытаниям на двухъядерной вычислительной системе алгоритм, основанный на декомпозиции процедуры рекурсивного случайного поиска, показывает лучшие результаты, чем метод на основе декомпозиции процедуры сортировки при вычислении целевой функции.

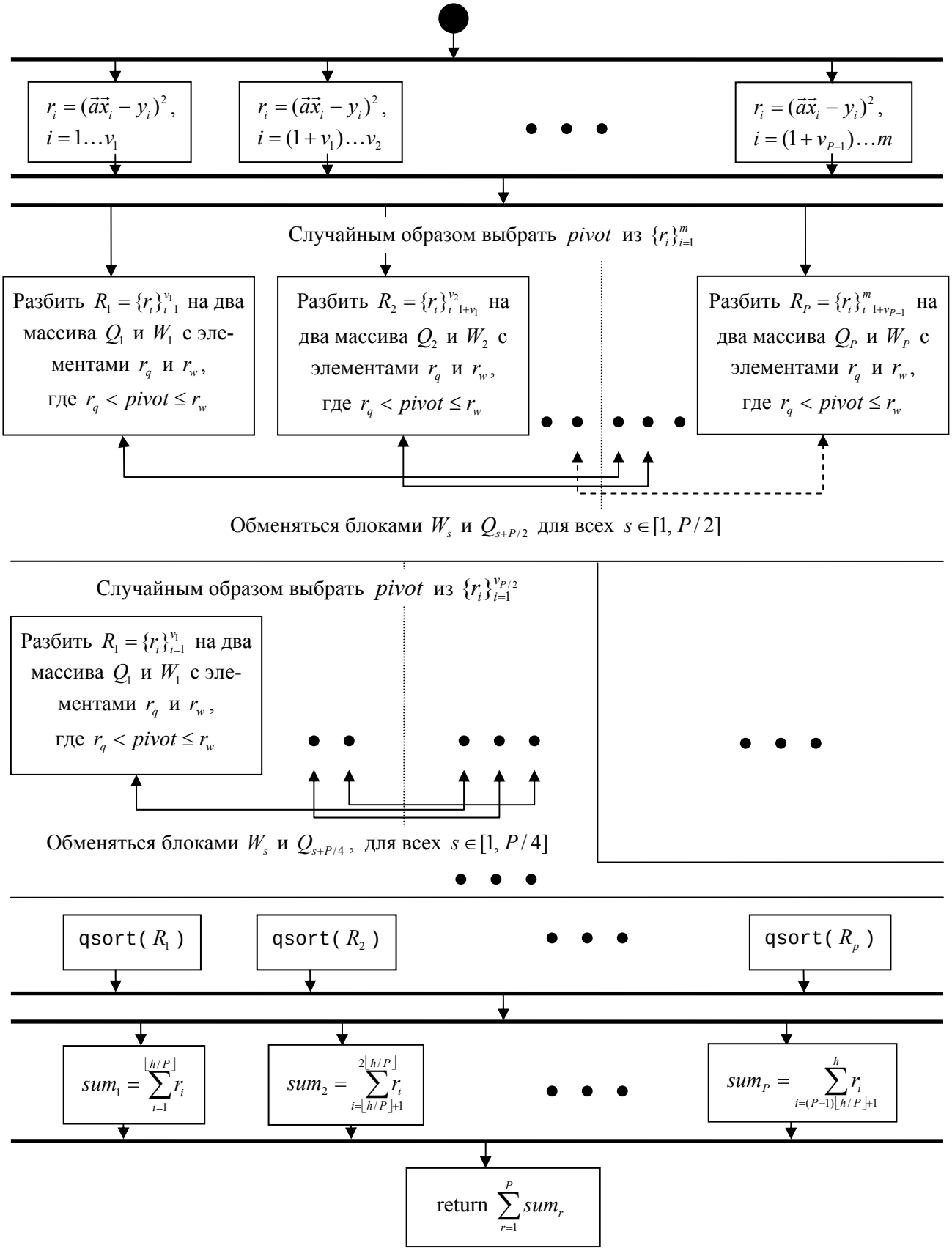

Рис. 2. Схема алгоритма распараллеливания вычисления значений целевой функции

Следует отметить, что оба алгоритма являются стохастическими: при каждом запуске программа завершается через разные интервалы времени. Потому анализ производительности требует проведения дополнительных исследований в соответствии с подходом работы [12].

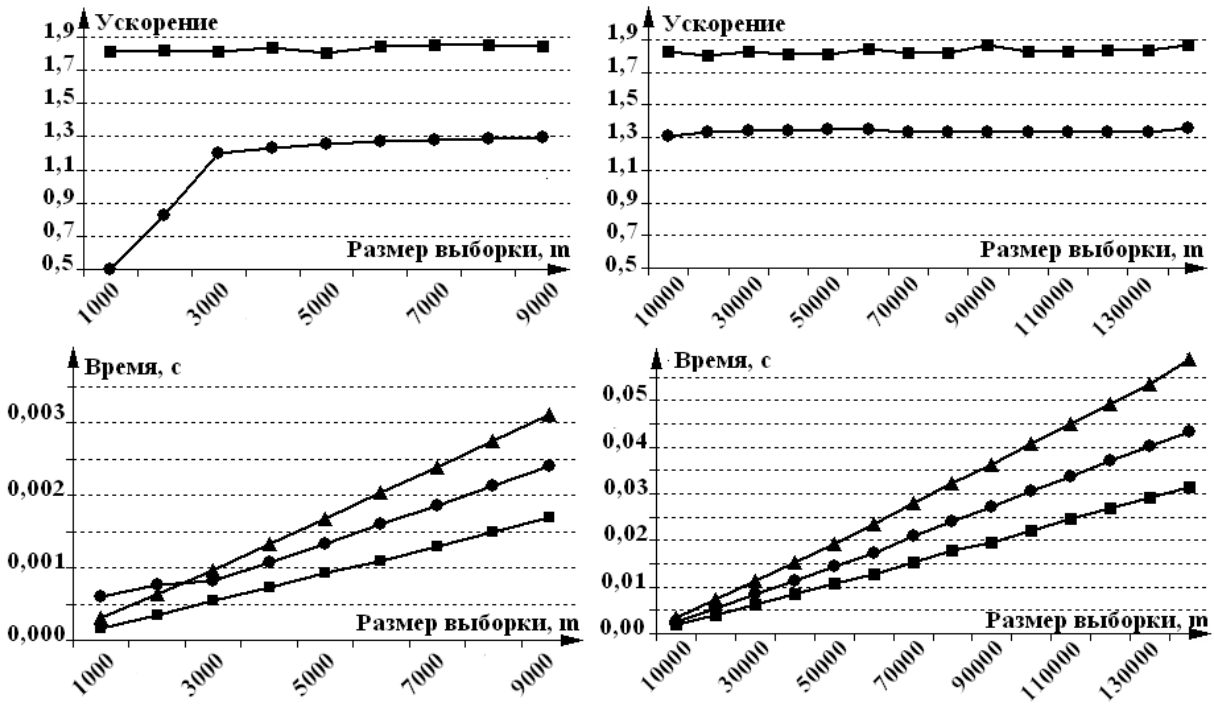

Рис. 3. Время выполнения и ускорение параллельных и последовательного алгоритмов

Литература

1. Akhter S., Roberts J. Multi-core programming: increasing performance through software multi-threading. –Intel Press. 2006. –334 p.

2. Boukhanovsky A.V., Ivanov S.V. Stochastic simulation of inhomogeneous metocean fields. Part III: High-performance parallel algorithms // ICCS'03, Proceedings. LNCS. Springer-Verlag. 2003. –Vol. 2658. –P. 234–243.

3. Boukhanovsky A.V., Bogdanov A.V. High performance parallel algorithms for data processing. // ICCS'04, Proceedings. LNCS. Springer-Verlag. 2004. –Vol. 3036. –P. 239–246.

4. Gatu C., Kontoghiorghes E.J. Parallel algorithms for computing all possible subset regression models using the QR decomposition // Parallel Computing. 2003. –Vol. 29. –P. 505–521.

5. Голуб Дж., Лоун Ван Ч. Матричные вычисления. –М.: Мир, 1999. –548 с.

6. Джонсон Н., Лион Ф. Статистика и планирование эксперимента в технике и науке: методы обработки данных. –М.: Мир, 1980. –520 с.

7. Rousseuw P.J., Leroy A.M. Robust regresion and outlier detection. –John Wiley & Sons, 1987. –341 p.

8. Erickson J., Har-Peled S., Mount D.M. On the Least Median Square Problem // Discrete & Computational Geometry, Springer-Verlag. – December 2006. –Vol. 36, N. 4. –P. 593–607.

9. Растригин Л.А. Адаптация сложных систем. –Рига: Зинатне, 1981. –375 с.

10. Tao Ye, Shivkumar Kalyanaraman. A recursive random search algorithm for large-scale network parameter configuration // Joint International Conference on Measurement and Modeling of Computer Systems, ACM SIGMETRICS'03. Proceedings. ACM Press. 2003. –P. 196–205.

11. Sanders P., Hansch T. On the efficient implementation of massively parallel quicksort // Proceedings. LNCS. Springer-Berlin. 1997. –Vol. 1253. –P. 13–24.

12. Ларченко А.В., Дунаев А.В., Бухановский А.В. Анализ и моделирование производительности параллельных стохастических алгоритмов, адаптированных к особенностям многоядерных вычислительных архитектур // Научный сервис в сети Интернет: многоядерный компьютерный мир. 15 лет РФФИ, 24–29 сентября, 2007. Новороссийск. Труды Всероссийской научной конференции. –М.: Издательство МГУ, 2007. –С. 156.