ВЫЧИСЛИТЕЛЬНЫЕ И ИНФОРМАЦИОННО-ИЗМЕРИТЕЛЬНЫЕ СИСТЕМЫ

УДК 573.22

ПОИСК ПЕРЕПРЕДСТАВЛЕННЫХ ХАРАКТЕРИСТИК ГЕНОВ: ОПЫТ РЕАЛИЗАЦИИ ПЕРЕСТАНОВОЧНОГО ТЕСТА С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ*

А. А. Якименко[1], К. В. Гунбин[2], М. С. Хайретдинов[3]

[1]*Новосибирский государственный технический университет, 630073, г. Новосибирск, просп. К. Маркса, 20*
[2]*Институт цитологии и генетики СО РАН, 630090, г. Новосибирск, просп. Академика Лаврентьева, 10*
[3]*Институт вычислительной математики и математической геофизики СО РАН, 630090, г. Новосибирск, просп. Академика Лаврентьева, 6*
E-mail: al--le@yandex.ru

Предлагаются алгоритмический подход и программа для решения задачи поиска статистически значимых перепредставленных биологических характеристик генов из заданного множества. Задача связана с реализацией известного в биологии перестановочного (рандомизационного) теста. На основе учёта потенциальной возможности распараллеливания перестановочного теста разработана параллельная программа с реализацией на графических процессорах. Приводятся оценки эффективности применения разработанной программы на эмпирическом материале.

Ключевые слова: графические процессоры, матрично-векторные операции, перестановочный тест.

Введение. Представленная работа относится к решению важной для биологии задачи анализа генетической детерминации признаков. Обычно такое исследование проводят на основе аналитических критериев (таких как t-тест, ANOVA, корреляция Пирсона и т. д.), построенных на замкнутой логике. Чтобы получить представление о распределении данных в генеральной совокупности используются выборки, которые полагаются распределёнными согласно субъективно заданному закону распределения данных. Выход из этого замкнутого круга предлагают методы ресамплинга, так как они не требуют никакой дополнительной информации о законе распределения данных в генеральной совокупности, а исследуют выборочные данные в различных комбинациях, как бы рассматривая их с разных сторон [1, 2]. Кроме того, ещё одним важным преимуществом методов ресамплинга перед аналитическими методами является отсутствие необходимости постоянно корректировать уровень статистической значимости для одновременного тестирования многих статистических гипотез, отражающих в случае анализа биологических данных одновременный вклад многих факторов в формирование одного признака [3]. Таким образом, в большинстве биологических исследований методы ресамплинга более корректны в сравнении с аналитическими методами, но для достаточно точных оценок различных статистик анализируемых выборок они подчас требуют огромных вычислительных ресурсов.

Значительное ускорение вычислений при ресамплинге может быть достигнуто за счёт

*Работа выполнена при поддержке Министерства образования и науки РФ (ГК № П-857 «Разработка программного обеспечения для высокопроизводительных вычислений в биоинформатике») и Российского фонда фундаментальных исследований (грант № 11-04-01771).

распараллеливания, причём наиболее экономным является распараллеливание с использованием графических процессоров (GPU — Graphic Processor Unit) [4–6]. В связи с этим были созданы специальные программные пакеты, позволяющие исследовать биологические объекты (в основном группы генов) методами ресамплинга, среди которых можно отметить свободно доступные программные продукты RandTestGPU [7] и permGPU [8]. Однако существенный недостаток вышеуказанных программных продуктов — возможность тестирования лишь очень простых случаев, например сравнение двух выборок измеряемых величин для генов (уровень экспрессии, скорость эволюции, количество известных полиморфных состояний и т. д.). Такой упрощённый подход к представлению биологических данных не позволяет учесть более детальную уже известную экспериментальную информацию о принадлежности генов к различным функциональным и/или структурным группам (аннотациям) и, следовательно, определить тонкую составляющую детерминации признака при разных внешних или внутренних условиях.

Целью данной работы является программная реализация и исследование эффективности распараллеливания перестановочного теста, нацеленного на поиск статистически значимых перепредставленных свойств генов при разных внешних или внутренних условиях с применением графических процессоров. Полученное при этом ускорение даёт возможность существенно повысить производительность решения обозначенной задачи.

Методика решения задачи. Общая идеология организации перестановочного теста, нацеленного на поиск статистически значимых перепредставленных свойств генов, заключается в следующем:

1) из эмпирического материала рассчитывается значение статистики $G_0(x_1, x_2, x_3, \ldots, x_n)$, где x — измеренная количественная характеристика гена (уровень экспрессии, скорость эволюции и т. д.);

2) осуществляется случайная перестановка количественных характеристик генов между выборками измеряемых величин;

3) для тех же выборок со случайно переставленными величинами $x_1, x_2, x_3, \ldots, x_n$ рассчитывается значение той же статистики $G_u(x'_1, x'_2, x'_3, \ldots, x'_n)$ — отражающей выборки после перестановки, индекс u отвечает номеру перестановки;

4) процедуры перестановки 2 и расчёта статистики 3 повторяются U раз;

5) определяется вероятность ошибки при отклонении нулевой гипотезы (p-значение) как доля значений G_u, превышающих G_0.

Для реализации перестановочного теста на графической карте необходимо учитывать особенности её архитектуры [5]. Важно выделить участки кода и алгоритмы, поддающиеся распараллеливанию. Проблемными местами остаются неделимые операции над массивами данных и чтение/запись в файл. Циклы обработки независимых данных целесообразно распределить по мультипроцессорам видеокарты и минимизировать количество условных выражений внутри таких блоков.

Общая структура программы представлена на рис. 1. Программа имеет на входе два текстовых файла: один с параметрами перестановочного теста (количество итераций и перестановок в итерацию) и второй непосредственно с входными данными.

Алгоритм перестановочного теста для последовательной версии на языке С. Весь алгоритм работы перестановочного теста можно разбить на три основных этапа: 1) чтение входного файла и формирование массивов данных в удобном для дальнейших расчётов виде; 2) расчёт сумм (или любых других величин, например средних, дисперсий и т. д.) измеренных значений гена для его различных свойств (в частности, функциональных аннотаций (ФА)), цикл с перемешиванием элементов массива и сбором необходимых для статистики величин; 3) расчёт p-значений и формирование файла с результатами. Наиболее трудоёмким и хорошо поддающимся распараллеливанию является второй этап. Подробнее он представлен на рис. 2, *b* и рис. 3.

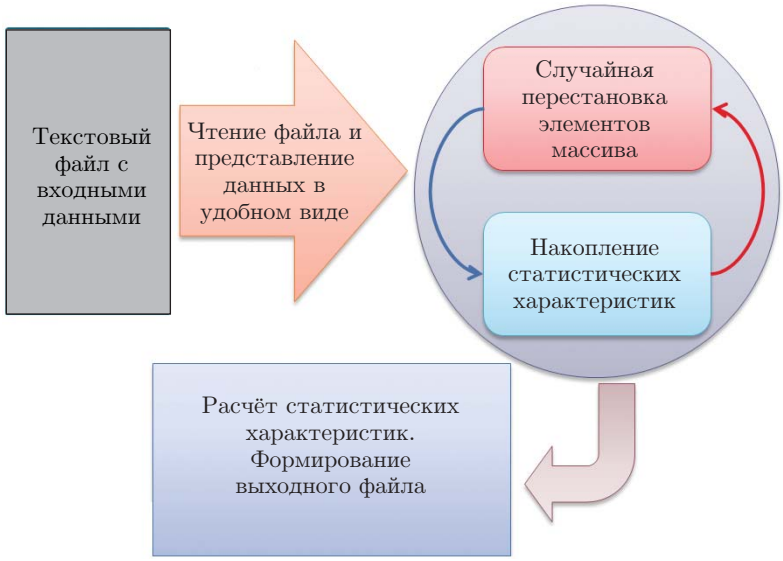


Рис. 1. Обобщённый алгоритм работы перестановочного теста

Рассмотрим более детально пример, в котором в качестве свойств генов будут функциональные аннотации GeneOntology [9]. Перед итерациями с перестановками необходимо рассчитать массив сумм реальных значений измеренных характеристик генов для каждой из имеющихся ФА. Для этого в последовательной версии алгоритма созданной программы организован цикл над структурой данных std::map, в которой хранятся ключи — идентификаторы генов и значения — числовые характеристики генов. После выполнения этой процедуры запускается цикл, повторяющий заданное количество раз случайную перестановку характеристик генов и дальнейшее накопление статистических величин.

По завершении описанных процедур выводится p-статистика и формируется выходной файл с результатами, из которого делается вывод о значимости определённых ФА генов.

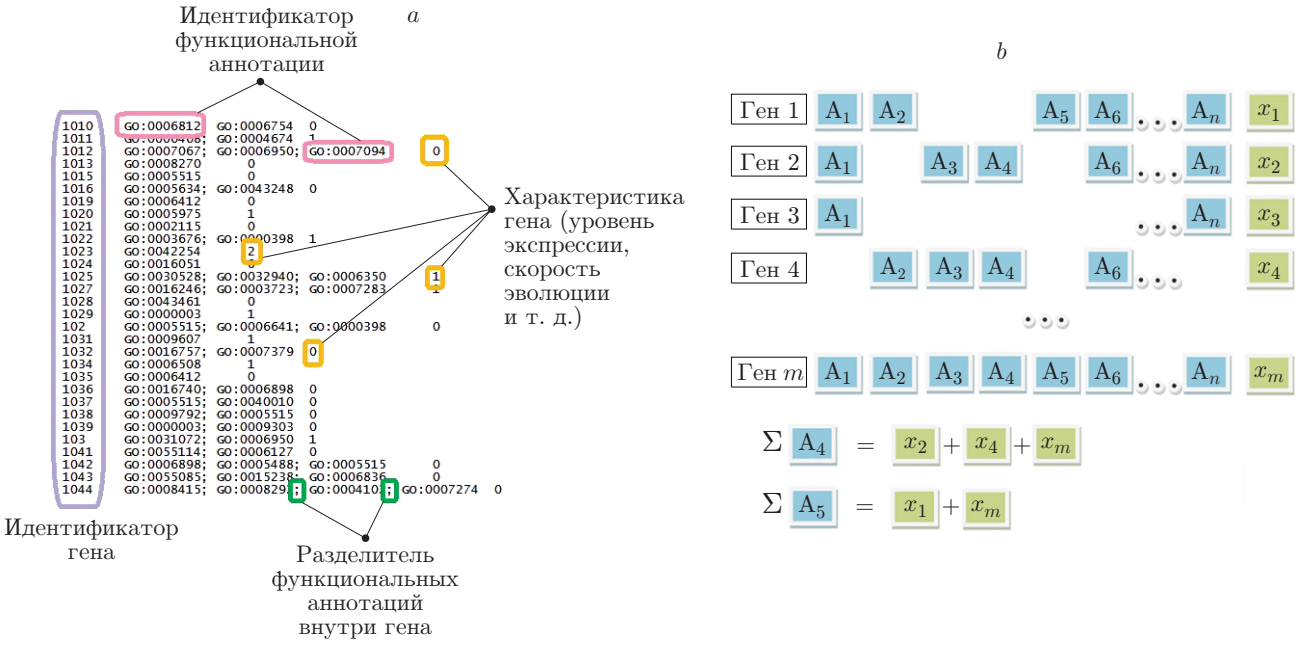


Рис. 2. Структуры хранения данных: a — представление гена во входном файле; b — понятие суммы значений гена для ФА

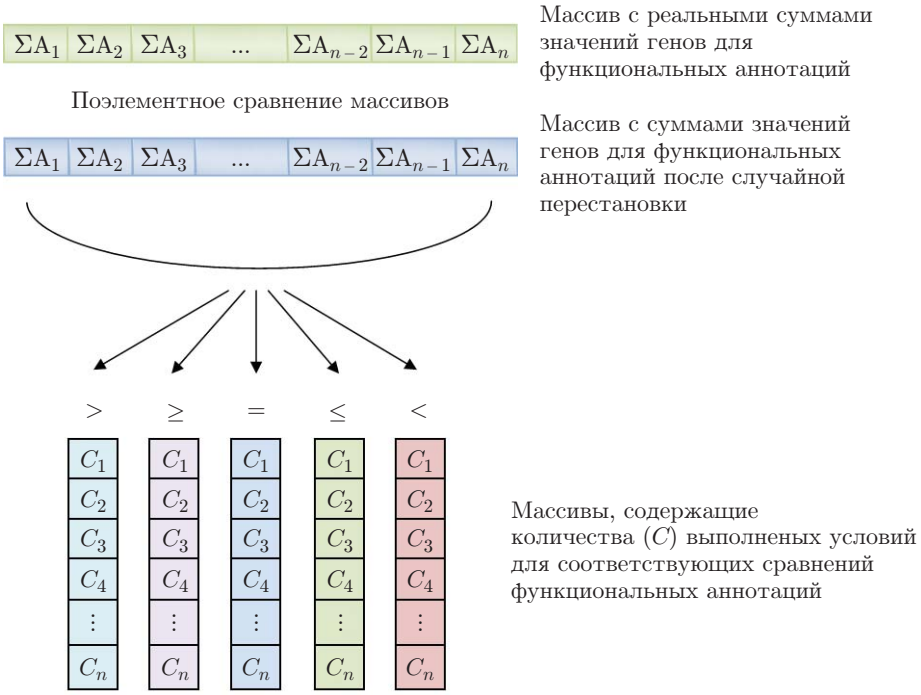


Рис. 3. Алгоритм накопления статистических величин

Кроме p-статистики приводятся идентификаторы генов, которые имеют указанные ФА.

Важно понимать, что особенности программного алгоритма не позволяют распараллелить бóльшую часть кода. Более того, особенности вычислений на графических картах ограничивают использование структур хранения данных (СХД) до массивов (библиотека thrust не рассматривается) [5]. Для решения этой проблемы необходимо пересмотреть подходы к реализации наиболее выгодных в распараллеливании алгоритмов и перейти от более сложных СХД к массивам. Такой метод был применён в алгоритме расчёта сумм числовых характеристик генов для различных ФА.

На рис. 2, *a* показана структура файла с входными данными. Её необходимо считать в оперативную память и представить в удобных для работы СХД. Важно отметить, что на рисунке приведён тестовый образец входных данных. В реальных задачах идентификатор ФА может иметь более сложное имя и разделитель между ФА может быть любым символом или комбинацией символов. Информативными данными здесь являются идентификаторы ФА и характеристики гена. Именно их представление играет большую роль в повышении быстродействия всей программы.

На рис. 2, *b* приведена структура гена в схематическом виде и показан смысл понятия сумма числовых значений генов для ФА. К примеру, для функциональной аннотации A_4 эта сумма будет складываться из характеристик генов x_2, x_4 и x_m.

Кроме того, из рис. 2, *b* можно сделать вывод о целесообразности применения двух массивов вместо структуры std::map для хранения информации по генам: двумерный массив, отражающий вхождения ФА в ген, и одномерный, содержащий измеренные характеристики генов. Такой подход позволит сократить количество передаваемых данных в память графического процессора и использовать стандартную библиотеку cuBLAS для выполнения матрично-векторного перемножения.

Параллельная реализация программы перестановочного теста на графическом процессоре. Общая структура программы представлена на рис. 4. Здесь блоки GPU исполнены на графическом процессоре. Важно отметить, что для вычисления реальных и

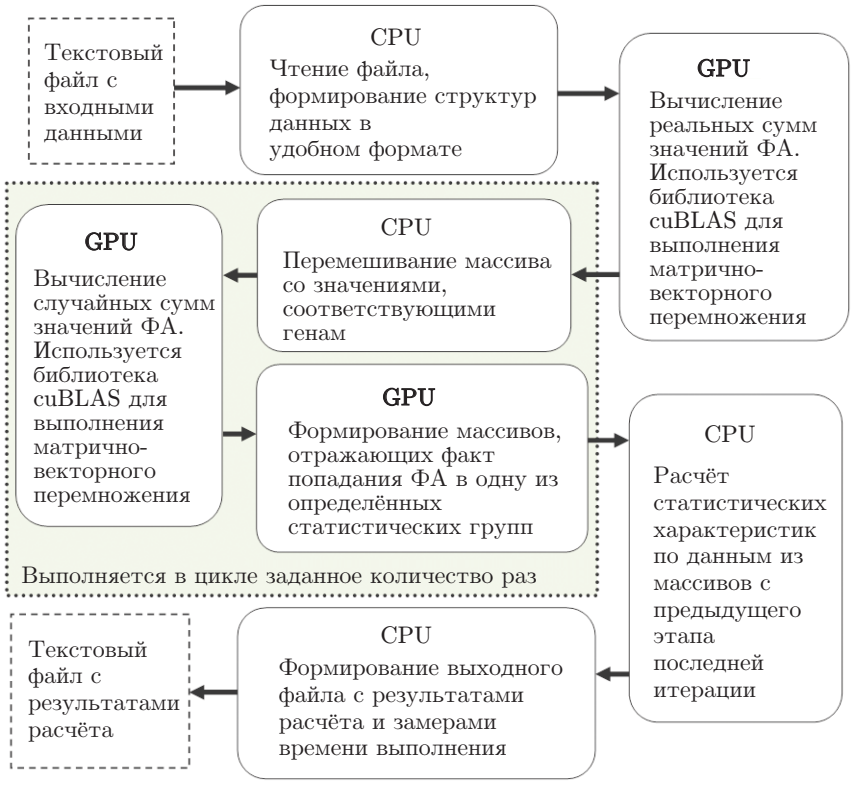


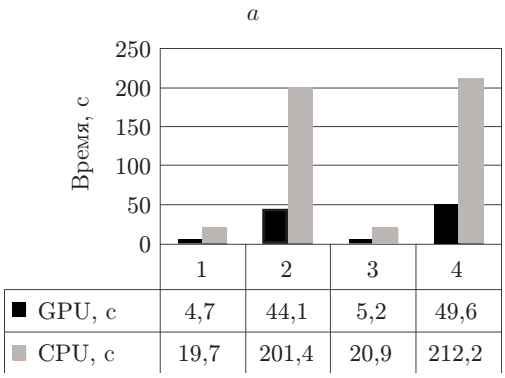
Рис. 4. Структура выполнения программы

случайных сумм каждый раз берётся первоначальная матрица вхождений функциональных аннотаций, изменяется только массив значений характеристик генов. Таким образом, в каждую итерацию необходима пересылка только одного одномерного массива, что снижает накладные расходы расчёта. Использование центрального процессора для случайной перестановки массива значений связано с невозможностью организации параллельного неупорядоченного доступа к элементам в массиве на графическом процессоре. Это, в свою очередь, требует передачи управления от GPU к CPU (Central Processor Unit) на каждой итерации цикла.

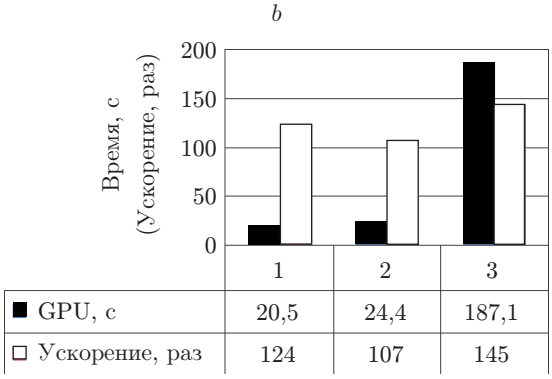
Алгоритм подсчёта реальных и случайных сумм заключается в представлении набора функциональных аннотаций в виде двумерного массива с нулями и единицами (рис. 5). В строках будут располагаться имеющиеся гены, а в столбцах — входящие в них ФА.

$$\begin{bmatrix} \text{ФА5} & \text{ФА3} & \text{ФА1} & & \\ \text{ФА2} & \text{ФА4} & & & \\ \text{ФА1} & \text{ФА2} & \text{ФА3} & & \\ \text{ФА3} & \text{ФА5} & \text{ФА4} & \text{ФА1} & \text{ФА2} \\ \text{ФА1} & \text{ФА5} & & & \\ \text{ФА2} & & & & \\ \text{ФА4} & & & & \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Рис. 5. Представление ФА в памяти графического процессора



	1	2	3	4
GPU, с	4,7	44,1	5,2	49,6
CPU, с	19,7	201,4	20,9	212,2



	1	2	3
GPU, с	20,5	24,4	187,1
Ускорение, раз	124	107	145

Рис. 6. Время выполнения программы: для первой (a) и второй (b) задач

Так, на место ФА, входящей в ген, будет ставиться единица, в противном случае запишется нуль. Тогда матрично-векторное перемножение такого транспонированного массива размерностью $M \times N$ (M — количество строк, N — количество столбцов) с одномерным массивом размерностью N, содержащим значения характеристик генов, даст в результате массив с суммами ФА по всем генам.

В алгоритме для сбора статистических величин широко применяются условные операции, что приводит к снижению эффективности выполнения на графическом процессоре. Обусловливается это тем, что все потоки GPU будут выполнять код полностью, отсекая ложное условие по завершении блока IF [5]. Тем не менее достигнутое ускорение S будет пропорционально $(\mathrm{Sg} \times \mathrm{Rm})/\mathrm{Cp}$, где под Rm понимается количество сравниваемых реальных сумм числовых значений для ФА со случайными суммами этих значений; Cp — степень параллелизации алгоритма; Sg в нашем случае равно пяти и определяется количеством случайных сумм значений ФА для категорий больших, меньших, равных, больших или равных, меньших или равных реальных сумм значений ФА.

Производительность параллельной и последовательной версий программ. Чтобы оценить производительность работы программы, рассматривались параллельная и последовательная реализации с различными параметрами для запуска на двух реальных задачах разного размера. На рис. 6, a показано время решения первой задачи с 2256 исследуемыми генами, содержащими 782 функциональные аннотации. На графике представлены времена выполнения программы для параллельной версии на GPU и последовательной версии на CPU. Проводились четыре запуска для каждой версии программы: 1 — 492 перестановки и 10000 итераций; 2 — 492 перестановки и 100000 итераций; 3 — 2256 перестановок и 10000 итераций; 4 — 2256 перестановок и 100000 итераций. Для данной задачи достигается ускорение около 5 раз, что определено небольшими размерами задачи. Количество итераций линейно увеличивает время выполнения программы. Тогда как рост количества перестановок лишь незначительно замедляет процесс счёта.

На рис. 6, b представлены времена выполнения второй задачи, в которой существенно больше исследуемых генов (19147) при немного большем количестве функциональных аннотаций (898). В отличие от первой задачи вместо времени выполнения программы на CPU приведено достигнутое ускорение параллельной версии относительно последовательной. Количество запусков сократилось до трёх: 1 — 5000 перестановок и 10000 итераций; 2 — 19147 перестановок и 100000 итераций; 3 — 5000 перестановок и 100000 итераций. На этом примере достигается ускорение до 145 раз. Ожидается, что в задачах большей размерности ускорение будет расти.

Заключение. Разработаны последовательный и параллельный варианты программ для проведения перестановочного теста, нацеленного на поиск статистически значимых пе-

репредставленных свойств генов при разных внешних или внутренних условиях на вычислительных устройствах: ПК с GPU NVIDIA и гибридный суперкомпьютер НКС-30Т+GPU Сибирского суперкомпьютерного центра СО РАН. В ходе выполнения работы решена задача распараллеливания наиболее трудоёмких алгоритмов программы перестановочного теста для реализации на GPU. Для этого использовалась библиотека матрично-векторного умножения cuBLAS, позволившая перенести данный алгоритм на архитектуру графических процессоров.

По результатам оценивания быстродействия показано ускорение программы на рассматриваемых двух задачах (см. рис. 6) с применением графического процессора относительно последовательной версии до 150 раз. Отмечено, что на время выполнения программы влияют размеры входных данных (количество генов и функциональных аннотаций) и количество итераций с перестановками.

Продемонстрировано несущественное влияние количества перестановок, выполняемых перед каждой итерацией расчёта случайных сумм значений для функциональных аннотаций, на время работы программы.

В настоящее время программа находится в опытной эксплуатации в Институте цитологии и генетики Сибирского отделения РАН.

СПИСОК ЛИТЕРАТУРЫ

1. **Эфрон Б.** Нетрадиционные методы многомерного статистического анализа. М.: Финансы и статистика, 1988. 263 с.
2. **Good P.** Permutation, Parametric and Bootstrap Tests of Hypotheses. N. Y.: Springer Verlag, 2005. 315 p.
3. **So H. C., Sham P. C.** Multiple testing and power calculations in genetic association studies // Genetics of Complex Human Diseases: A Laboratory Manual /Eds. A. Al-Chalabi, L. Almasy. N. Y.: Cold Spring Harbor Laboratory Press, 2010. P. 49–59.
4. **Dematté L., Prandi D.** GPU computing for systems biology // Brief. Bioinform. 2010. **11**, N 3. P. 323–333.
5. **Боресков А. В., Харламов А. А., Марковский Н. Д.** Параллельные вычисления на GPU. Архитектура и программная модель CUDA. М.: Изд-во Московского ун-та, 2012. 336 с.
6. **Пустовалов Е. Ф., Войтенко О. В., Грудин Б. Н., Плотников В. С.** Графические процессоры в задачах электронной томографии // Автометрия. 2012. **48**, № 1. С. 72–79.
7. **Van Hemert J. L., Dickerson J. A.** Monte Carlo randomization tests for large-scale abundance datasets on the GPU // Comput. Methods Programs Biomed. 2011. **101**, N 1. P. 80–86. URL: https://subversion.vrac.iastate.edu/Subversion/RandTestGPU/svn/RandTestGPU/ (дата обращения: 22.04.1013).
8. **Shterev I. D., Jung S. H., George S. L., Owzar K.** permGPU: Using graphics processing units in RNA microarray association studies // BMC Bioinformatics. 2010. **11**. P. 329. URL: https://code.google.com/p/permgpu/ (дата обращения: 22.04.2013).
9. **Ashburner M., Ball C. A., Blake J. A. et al.** Gene ontology: tool for the unification of biology // The Gene Ontology Consortium. Nature Genet. 2000. **25**, N 1. P. 25–29.

Поступила в редакцию 22 апреля 2013 г.