УДК: 519.688

Название: Разработка программы построения и кластеризации геномных профилей с использованием GPU

Автор(ы):

Никитин С.И. (Новосибирский государственный университет),

Черемушкин Е.С. (Институт систем информатики им. А. П. Ершова СО РАН)

Аннотация: В процессе считывания РНК на определенных участках ДНК — сайтах связывания формируется комплекс белков, называемых транскрипционными факторами. Этот комплекс позволяет закрепиться РНК-полимеразе и начать считывание РНК. Задача поиска сайтов связывания на ДНК является сложной ввиду наличия многих факторов, влияющих на связывание. В их числе — наличие других сайтов связывания на небольшом расстоянии от рассматриваемого сайта. Для исследования этой зависимости авторами были введены в рассмотрение гистограммы распределения плотности сайтов на геноме, названные геномными профилями.

В рамках данной работы реализован алгоритм предсказания сайтов связывания с помощью весовых матриц, написана его параллельная реализация для архитектуры NVidia CUDA, реализован алгоритм построения геномных профилей, алгоритмы иерархической кластеризации и кластеризации К-средних для геномных профилей. Реализован алгоритм, позволяющий строить случайные иерархии транскрипционных факторов на основании существующей биологической классификации для того, чтобы оценить качество полученной классификации геномных профилей. Соответствующая программа написана на языке C++ и предназначена для быстрого построения геномных профилей и их первичного анализа. Проведен анализ сходства классификации геномных профилей с биологической классификацией транскрипционных факторов для исследования влияния взаимного расположения сайтов связывания на ДНК.

Ключевые слова: РНК, ДНК, геном, весовые матрицы, транскрипционные факторы, сайты связывания с факторами транскрипции, кластеризация, программная система, Nvidia CUDA

1. Введение. Появление новых экспериментальных технологий в областях, связанных с обработкой генетической информации, позволяющих с высокой эффективностью исследовать молекулярно-генетические системы и процессы, стимулирует развитие биоинформатики. Задачи определения однонуклеотидных полиморфизмов и даже задачи полной расшифровки генома становятся доступными не только крупным институтам и консорциумам, но и небольшим организациям. В результате в последнее время в

молекулярной биологии и генетике произошел информационный взрыв, сопровождаемый огромным ростом объемов экспериментальных данных[1]. В частности, в ходе работ была получена ценная информация о генном и геномном уровнях организации жизни. Но надежды на то, что с развитием технологии секвенирования мгновенно возникнет четкая картина, описывающая все процессы, происходящие в клетке, пока не оправдываются. Все больше и больше ощущается недостаток алгоритмов и программ, которые бы позволили выделить из этого моря данных биологически значимую, структурированную информацию.

Все больший интерес вызывают такие технологии, как микрочипы (microarray analysis), Chip-on-Chip и Chip-seq[2]. Микрочиповый эксперимент позволяет измерить экспрессию (количество произведенной РНК) практически для всех генов в клетке одновременно. Этот профиль экспрессии различен в клетках, которые выполняют разные функции. Если клетка стала по каким-то причинам функционировать неправильно, например, в результате заболевания или под действием определенных веществ, то это отразится на профиле экспрессии. Таким образом, в теории с помощью таких экспериментов возможно определить массу зависимостей между генами, а также сделать количественные оценки. Chip-on-Chip и Chip-Seq эксперименты позволяют найти набор фрагментов ДНК, к которым присоединяются транскрипционные факторы, что (по плану создателей) позволит определить, как регулируется экспрессия генов. В результате выяснилось, что in vitro (в пробирке), связывание происходит не так, как in vivo (в жизни): транскрипционный фактор может связаться с определенным «сайтом» in vitro, но in vivo этого связывания не будет, т. к. ДНК не будет освобождена от нуклеосомной структуры на этом участке, либо другие транскрипционные факторы будут связаны с этим участком, что помешает связыванию, и т.д.

Таким образом, несмотря на рост количества информации, не теряет актуальности создание и улучшение высокопроизводительных информационно-компьютерных систем и технологий, предназначенных для анализа и интерпретации экспериментальных данных о связывании транскрипционных факторов с ДНК. Эти программы являются необходимыми для решения фундаментальных задач биологии, а тем более — для практического использования в биомедицине и биотехнологии.

Несмотря на то, что биоинформатика является очень молодой наукой, в ней уже существуют свои традиционные направления, такие как компьютерный анализ ДНК, РНК и белковых последовательностей, распознавание функциональных сайтов, реконструкция пространственных структур биополимеров, теоретический и компьютерный анализ

структурно-функциональной организации геномов и белков. Одним из исследуемых в биоинформатике процессов является процесс транскрипции и связанные с ним задачи.

Транскрипцией называется синтез рибонуклеиновой кислоты (РНК) происходящий при непосредственном участии дезоксирибонуклеиновой кислоты (ДНК) [3]. Транскрипция — один из фундаментальных биологических процессов, происходящий в живых клетках, первый этап реализации генетической информации, записанной в ДНК в виде линейной последовательности 4 типов мономерных звеньев — нуклеотидов. Этот процесс осуществляется специальными ферментами — ДНК зависимыми РНК-полимерами. В результате образуется цепь РНК, последовательность звеньев которой повторяет последовательность звеньев одной из двух цепей копируемого участка ДНК. Продуктом транскрипции являются 4 типа РНК, выполняющих различные функции:

- информационные, или матричные, РНК, выполняющие роль матриц при синтезе белка рибосомами (трансляция);
- рибосомальные РНК, являющиеся структурными компонентами рибосом;
- транспортные РНК, являющиеся основными элементами, осуществляющими доставку аминокислот к месту синтеза белка;
- РНК, играющие роль активаторов репликации ДНК.

Транскрипция ДНК происходит отдельными участками, в которые входит один или несколько генов. Фермент РНК-полимераза определяет начало такого участка (промотор), присоединяется к нему, расплетает двойную спираль ДНК и копирует, начиная с этого места, одну из её цепей, перемещаясь вдоль ДНК и последовательно присоединяя мономерные звенья — нуклеотиды — к образующейся РНК в соответствии с принципом комплементарности (т. е. напротив аденина присоединяется урацил, напротив гуанина присоединяется цитозин, и наоборот. Однако в ДНК отсутствует урацил, таким образом, аденин присоединяется напротив тимина). По мере движения РНК-полимеразы растущая цепь РНК отходит от матрицы, и двойная спираль ДНК позади фермента восстанавливается. Когда РНК-полимераза достигает конца копируемого участка (терминатора), РНК отделяется от матрицы. Число копий разных участков ДНК зависит от потребности клеток в соответствующих белках и может меняться в зависимости от условий среды или в ходе развития организма. Это достигается за счет регуляции. Понимание механизма регуляции экспрессии генов — важнейшая задача биологии. При изучении регуляции экспрессии на уровне транскрипции важно не только определить белки-регуляторы (транскрипционные факторы), но и участки их связывания с последовательностью ДНК, а также условия, при которых происходит связывание на данном участке. В настоящее время в открытом доступе

находится большое количество секвенированных геномов и данных по экспрессии генов, что позволяет изучать регуляцию путем анализа последовательностей с помощью вычислительных методов. Задача поиска регуляторных мотивов в наборе последовательностей ДНК — классическая задача биоинформатики. К настоящему моменту создано огромное количество алгоритмов поиска мотивов, однако все они имеют свои ограничения, и не существует универсального алгоритма, который решал бы эту задачу.

2. Моделирование процесса транскрипции. Инициация транскрипции является сложным процессом, протекание которого зависит от множества факторов. Если говорить об эукариотах, то такими факторами могут являться форма последовательности ДНК вблизи начала транскрибируемой области, а также в более удалённых участках, называемых энхансерами и сайленсерами. Дополнительно на процесс транскрипции влияет наличие или отсутствие различных белковых транскрипционных факторов [4]. Факторы транскрипции — белки, которые регулируют транскрипцию путем связывания со специфичными участками ДНК — сайтами связывания. Транскрипционные факторы выполняют свою функцию самостоятельно либо в комплексе с другими белками. Различают репрессорные и активирующие транскрипционные факторы, которые, соответственно, снижают или повышают константу связывания РНК-полимеразы с регуляторными последовательностями экспрессируемого гена [5]. Определяющая черта факторов транскрипции — наличие в их составе одного или более ДНК-связывающих доменов, которые взаимодействуют с сайтами связывания, расположенными в регуляторных областях генов. Транскрипционные факторы бывают конститутивные (всегда активные в клетке) и активируемые (активируются только при определенных условиях). Активируемые, в свою очередь, разделяют на тканеспецифические (активируются только в определенных тканях организма) и сигнал-зависимые, или рецепторы (требуют внешнего сигнала для активации).

Набор транскрипционных факторов, стимулирующих, в конечном счете, транскрипцию, является различным для разных типов генов. Присоединившись к соответствующему сайту связывания на ДНК, комплекс из транскрипционных факторов позволяет закрепиться РНК-полимеразе на старте транскрипции и произвести считывание РНК. Сайты связывания имеют длину в среднем 10-20 нуклеотидов и для одного и того же транскрипционного фактора имеют схожие последовательности. Это объясняется тем, что транскрипционные факторы имеют специфическую форму, которая позволяет им закрепляться на последовательностях определенного типа. Но, несмотря на кажущуюся простоту, определить, является ли данная последовательность сайтом, сложно. Это обусловлено тем, что на связывание влияют и другие факторы, в частности — другие сайты в окрестности исследуемого.

TRANSFAC (eukaryotic trans-acting Transcriptional regulatory Factors and cis-acting regulatory sites database) — база данных, содержащая информацию о сайтах связывания эукариотических факторов транскрипции в геномах и о связывающихся белках. Содержит экспериментально подтвержденные регуляторные сайты эукариот (от дрожжей до человека), а также белки, действующие как факторы транскрипции или вместе с ними [6].

Весовые матрицы (PWM – position weight matrix) библиотеки Transfac компании Biobase являются самым распространенным средством выявления потенциальных сайтов связывания с транскрипционными факторами на ДНК.

PWM, которые впервые были введены для характеристики сайтов инициации транскрипции и трансляции у E. coli (кишечная палочка) [7,8], хорошо подходят для описания сайтов связывания факторов транскрипции и способны количественно охарактеризовать частые и редкие вариации в последовательности сайтов. PWM представляют собой матрицу L × 4 (L — длина сайта), где номер строки соответствует позиции нуклеотида в сайте, а по столбцам стоят частоты встречаемости данного нуклеотида в данной позиции сайта.

№	A	C	G	T
01	13	4	4	3
02	14	3	5	2
03	2	7	1	14
04	23	0	0	1
05	24	0	0	0
06	1	23	0	0
07	0	1	17	6
08	1	1	21	1
09	11	5	4	4
10	12	5	2	5

Рис.1. Весовая матрица V$VMYB_01. По вертикали — позиция в сайте, по горизонтали нуклеотид. В ячейках частота встречаемости данного нуклеотида в данной позиции в наборе экспериментальных сайтов

Вес, порождаемый матрицей при выравнивании с данным участком последовательности, в нашем случае вычисляется как сумма элементов матрицы, соответствующих нуклеотидам, стоящим в каждой позиции рассматриваемого участка.

Обычно перед вычислением веса матрица конвертируется в скоровую матрицу. Это происходит с помощью домножения каждого элемента матрицы на информационный

коэффициент, соответствующий данной строке, либо подсчетом вероятностной матрицы и логарифмированием значений.

№	A	C	G	T	…
01	13	4	4	**3**	T
02	14	3	5	**2**	T
03	2	7	**1**	14	G
04	23	**0**	0	1	C
05	**24**	0	0	0	A
06	1	23	**0**	0	G
07	0	1	17	**6**	T
08	1	1	**21**	1	G
09	**11**	5	4	4	A
10	12	5	**2**	5	G
					…

Рис. 2. Вычисление веса последовательности TTGCAGTGAG с помощью весовой матрицы V$VMYB_01

Таким образом, PWM предоставляет достаточно полное описание участка ДНК, с которым способен связываться конкретный белок, и может быть применена при сканировании геномной последовательности для поиска сайтов, дающих достаточно хороший вес. Использование PWM позволяет достаточно эффективно предсказывать сайты связывания белков.

Однако следует отметить, что, несмотря на все свои достоинства, PWM все-таки имеет несколько недостатков. Одним из них является то, что PWM не учитывает взаимное влияние соседних позиций сайта, а только учитывает общую форму текущей подпоследовательности. Однако наличие таких зависимостей было показано для некоторых факторов [9,10]. Другим недостатком данного метода поиска потенциальных сайтов связывания является то, что он не учитывает присутствие других сайтов в окрестности текущего. Для решения этой проблемы было предложено исследовать гистограммы распределения плотности сайтов на геноме, названные геномными профилями. Такие профили строятся для каждой весовой матрицы и описывают распределение густоты сайтов связывания, т. е. показывают, сколько фрагментов генома расположены на последовательности ДНК с одной густотой сайтов, сколько с другой и т.д. Перед авторами была поставлена задача реализации алгоритма построения геномных профилей для весовых матриц и обоснования сходства классификации геномных профилей с

биологической классификацией транскрипционных факторов для исследования влияния взаимного расположения сайтов связывания на ДНК.

Алгоритм поиска потенциальных сайтов связывания с факторами транскрипции основан на вычислении веса рассматриваемой подпоследовательности генома и сравнении результата с некоторым наперед заданным порогом. Т.к. геномные последовательности имеют большую длину, процесс поиска занимает длительное время. Эта проблема также рассматривается в данной работе, поскольку алгоритм используется при построении геномных профилей весовых матриц.

2. Постановка задачи. Целью данной работы являлось создание программы, позволяющей для каждой весовой матрицы строить геномные профили, графически отображать полученные результаты в виде гистограмм распределения, обрабатывать полученные профили с помощью некоторых алгоритмов кластеризации; написать параллельную реализацию алгоритма поиска потенциальных сайтов связывания для GPU; написать алгоритм создания случайных деревьев по существующему дереву классификации транскрипционных факторов для исследования применимости геномных профилей. Сделать вывод о сходстве классификации геномных профилей с биологической классификацией транскрипционных факторов. В качестве языка программирования выбран C++ с использованием инструментария Qt, так как он позволяет упростить процесс создания приложений, обладает обширной документацией, а также является кроссплатформенным, что расширяет возможности использования программы на отличных от Windows платформах. Для параллельной реализации алгоритма поиска потенциальных сайтов связывания была выбрана технология NVidia CUDA, т.к. программно она основана на языке C, использует принципы SIMD архитектуры (одним набором инструкций независимо обрабатывается большой объем данных); к тому же существуют специализированные видеоадаптеры Tesla, отличающиеся от обычных видеоадаптеров качеством реализации, точностью расчетов и большей вычислительной мощностью.

3. Геномные профили. В настоящее время физические свойства многих транскрипционных факторов подробно изучены, благодаря большому количеству экспериментальной информации о сайтах связывания с транскрипционными факторами. Это позволило получить весовые матрицы, предсказывающие потенциальные сайты связывания на ДНК с большой точностью. Однако на связывание влияет не только форма факторов транскрипции, но и другие условия, такие как доступность сайта, наличие кофакторов, внутрицепочечные взаимодействия, влияние других сайтов связывания в окрестности данного и другие. Таким образом, затруднительно на основании последовательности генома

достоверно предсказать реальное место посадки транскрипционного фактора на ДНК in vivo. В связи с этим изучение условий, влияющих на связывание транскрипционных факторов и ДНК, является актуальной задачей. Для анализа влияния сайтов связывания, находящихся в окрестностях данного, нами было предложено ввести новую конструкцию «геномные профили». Геномные профили представляют собой гистограммы распределения плотности сайтов на геноме и характеризуют кучность расположения потенциальных сайтов связывания с факторами транскрипции. Пока рассматривается влияние сгруппированности сайтов одного типа, независимо от других сайтов.

Алгоритм построения геномных профилей для весовой матрицы состоит из 3 этапов:

1. Предсказание потенциальных сайтов связывания.

Пусть M — весовая матрица размера $4xN$. Зафиксируем некоторую позицию i в последовательности генома G и будем рассматривать его подпоследовательность длины N: $G(i, N)$. С помощью весовой матрицы вычислим вес выбранного отрезка генома w_i как сумму элементов матрицы, соответствующих нуклеотидам, стоящим в каждой позиции рассматриваемого участка, а затем нормируем полученное значение на отрезок $[0, \dots, 1]$ следующим образом:

$$w = \frac{(w_i - w_{min})}{(w_{max} - w_{min})}$$

где w_{min} и w_{max} — минимальный и максимальный вес последовательности

Полученный вес w сравнивается с некоторым наперед заданным порогом c. Если $w \geq c$, то сайт связывания в данном месте на последовательности считается распознанным, в противном случае нераспознанным.

2. Построение профиля распознанных сайтов.

Зафиксируем весовую матрицу M и порог c, с которым мы предполагаем сайт связывания распознанным. Разобьем последовательность генома на участки длины L (в нашем случае L=100000) нуклеотидов, в каждом из них последовательно проведем поиск потенциальных сайтов связывания. Таким образом, мы получим профиль распознанных сайтов, то есть количество $V_{c,M}(i)$ найденных сайтов для каждого участка.

В последовательной реализации на фиксированном участке поиск сайтов происходит последовательно от начала до конца отрезка ДНК. Параллельная реализация алгоритма заключается в следующем: зафиксируем участок длины L. Произведем поиск сайтов связывания для каждой позиции i с помощью отдельного потока на GPU. Благодаря возможности запускать сотни потоков одновременно происходит одновременная обработка

большого объема данных, что приводит к значительному увеличению скорости обработки участка.

3. Построение профиля матрицы.

Упорядочим полученный профиль распознанных сайтов $V_{c,M}(i)$ по возрастанию. Исключим нулевые элементы, т.к. они образуются на участках ДНК, заполненных полиNсигналом. Далее, отбросим 5% значений сверху и снизу, тем самым удаляя выбросы распределения $V_{c,M}(i)$. Полученное распределение $V'_{c,M}$ преобразуем db гистограмму следующим образом: найдем V'_{max} и V'_{min} — максимум и минимум $V'_{c,M}$. Разобьем отрезок $[V'_{min}, V'_{max}]$ на $T = 20$ равных фрагментов $d_1 \dots d_T$. Далее посчитаем количество $V'_{c,M}(i)$, попавших в каждый из d_t. Повторим процедуру для каждой хромосомы. Получим искомый геномный профиль $P_{c,M}(t)$.

3.1. Кластеризация. Кластерный анализ (англ. Data clustering) — задача разбиения заданной выборки объектов на подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались. Рассматриваем в качестве элементов выборки геномные профили весовых матриц, которые по сути являются векторами одной длины. Соответственно, стоит задача разбиения набора векторов на подмножества близких по метрике элементов. Для реализации алгоритмов кластеризации выбраны две основные метрики пространства R^n:

- евклидова: $S(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$;
- $S(x,y) = \sum_{i=0}^{n}|x_i - y_i|$.

В качестве методов кластеризации выбраны два известных в литературе и широко распространенных алгоритма: метод К-средних (K-means) и иерархический метод.

3.1.1. Метод К-средних.

1. Задать число K кластеров. Выбрать случайным образом K элементов, которые будут являться «центрами масс» кластером на начальном этапе.
2. Отнести каждый элемент к кластеру с ближайшим «центром масс».
3. Пересчитать «центры масс» согласно текущему составу элементов в кластерах. Координаты «центров масс» вычисляются как средние арифметические соответствующих координат элементов кластера.
4. Если состав кластеров не изменился, то стоп, иначе на шаг 2.

Итерация алгоритма имеет сложность $O(N * K)$, где N — число элементов выборки, K — число кластеров.

3.1.2. Иерархическая кластеризация. Метод иерархической кластеризации заключается в следующем:

1. Задать число K кластеров. Каждый элемент выборки является отдельным кластером. Таким образом, перед началом работы имеем N кластеров.
2. Выбираем два кластера, расстояние между которыми минимально, и объединяем их в один. Таким образом уменьшаем общее число кластеров. В качестве расстояния между кластерами можно выбрать один из следующих вариантов:
   - расстояние между двумя наиболее отдаленными элементами;
   - расстояние между двумя наиболее близкими элементами;
   - среднее расстояние между всеми парами объектов в них.
3. Если достигнуто необходимое число K кластеров, то стоп, иначе шаг 2.

Алгоритм имеет сложность $O(N^2)$.

4. Программная система для расчета геномных профилей. Для получения и обработки геномных профилей весовых матриц, была разработана программная система GenomeSignal. Программа написана на языке C++ с использованием кросс-платформенного инструментария разработки ПО Qt, а также с использованием технологии параллельных вычислений NVidia CUDA. Кроме этого, в программе использована технология параллельных вычисления для систем с общей памятью OpenMP. Тестирование проводилось на операционных системах Windows 7 x86/x64, Windows 8 x64.

4.1. Внутренне строение программной системы. В связи с тем, что C++ является объектно-ориентированным языком программирования, внутреннее устройство программы реализовано в виде классов. Взаимодействие между классами показано на рис. 3.

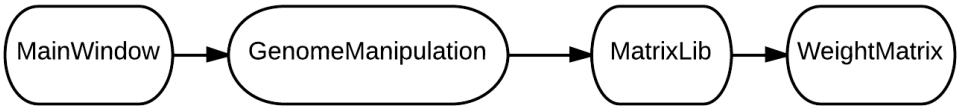

Рис.3. Взаимодействие классов в программе GenomeSignal

Класс **WeightMatrix** является хранилищем значений количеств предсказанных сайтов транскрипционных факторов для некоторой фиксированной весовой матрицы, содержит набор геномных профилей, соответствующих данной матрице, а также предоставляет процедуры для построения данных структур.

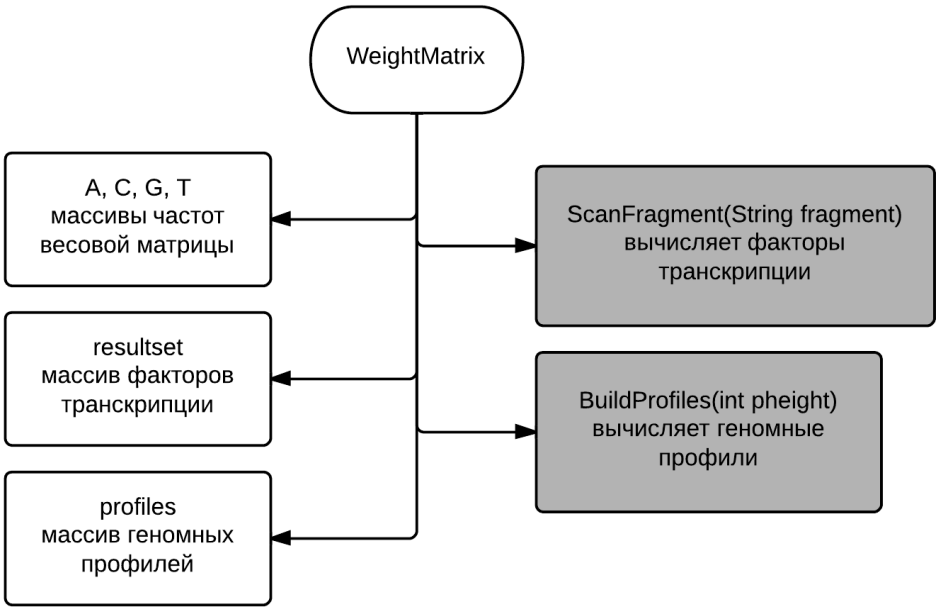

Рис.4. Основные поля и методы класса WeightMatrix

Класс **MatrixLib** является хранилищем результатов работы для всех весовых матриц, т.к. содержит массив элементов типа WeightMatrix для всех загруженных весовых матриц и общие методы получения профилей и предсказания факторов транскрипции.

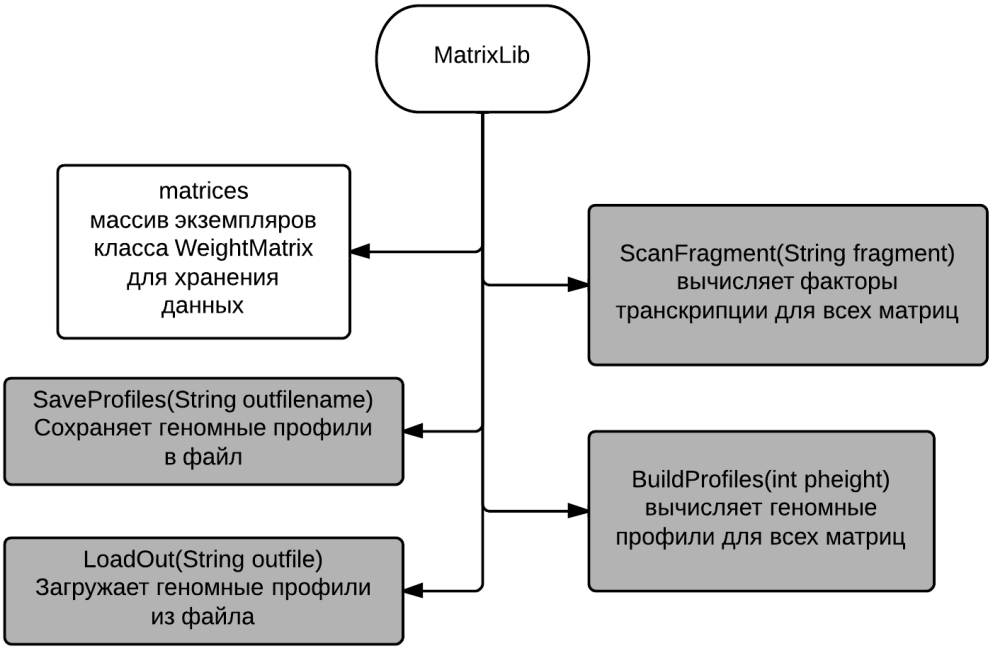

Рис.5. Основные поля и методы класса MatrixLib

Класс **GenomeManipulation** является основным классом программы GenomeSignal. Он включает в себя реализации всех алгоритмов для получения и обработки геномных профилей.

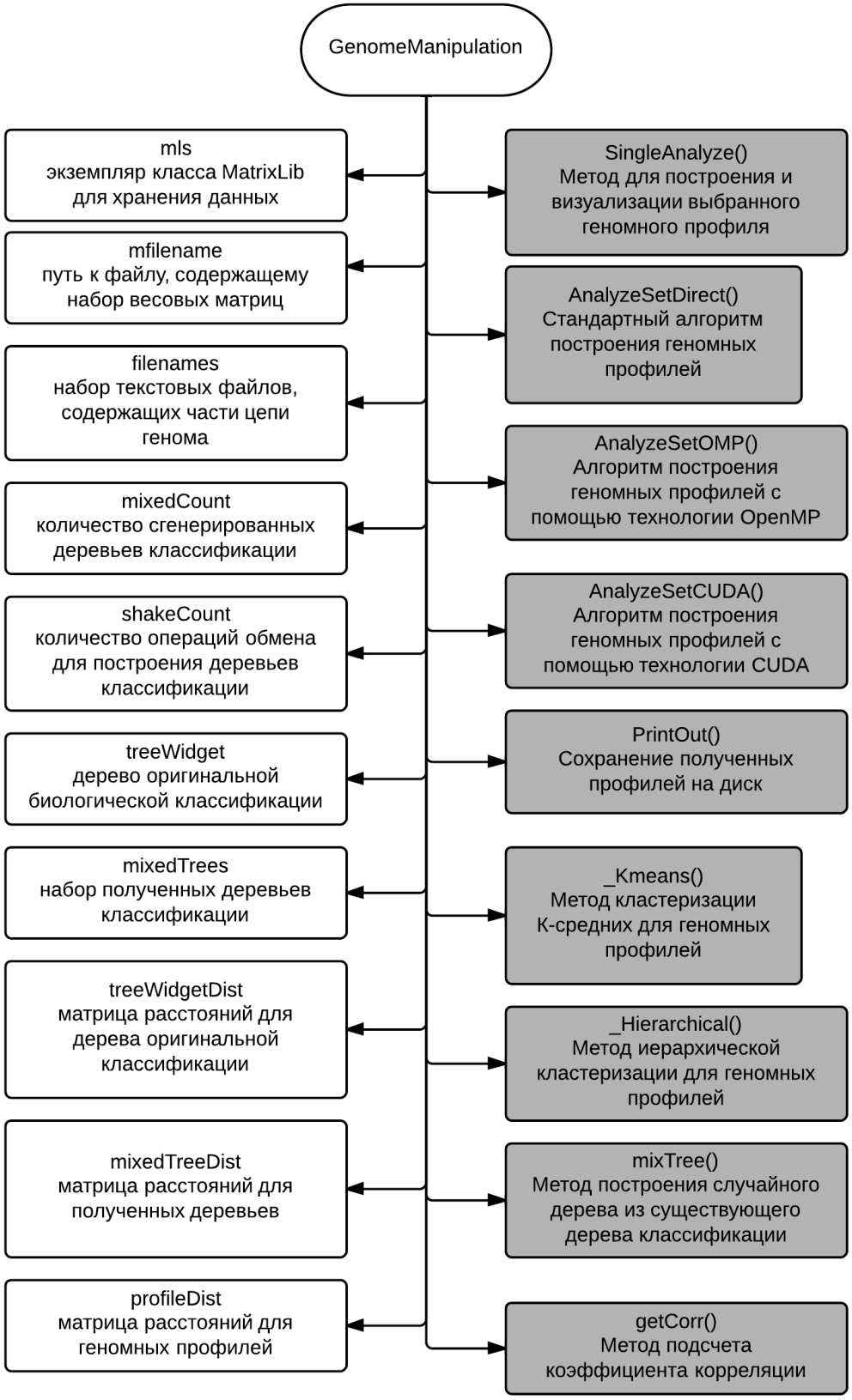

Рис.6. Основные поля и методы класса GenomeManipulation (параметры методов опущены для сокращения записи)

Класс **MainWindow** является классом взаимодействия программы с пользователем. В нем собраны методы работы интерфейса, реализованы возможности хранения последних путей к файлам, логирования последних произведенных вычислений, а также проверки системы на возможность использования тех или иных технологий. Для независимой работы в каждом режиме и во избежание потери выполненных вычислений, в программе используется 4 экземпляра класса GenomeManipulation.

4.2. Пользовательский интерфейс программной системы. Внешний вид программы представляет собой диалоговое окно, в котором вкладки «пакетная обработка данных», «визуализация геномных профилей», «кластеризация», «сравнение» отвечают соответствующим режимам работы.

4.2.1. Режим пакетной обработки данных. На рис. 7. программа GenomeSignal настроена в режим «пакетная обработка данных».

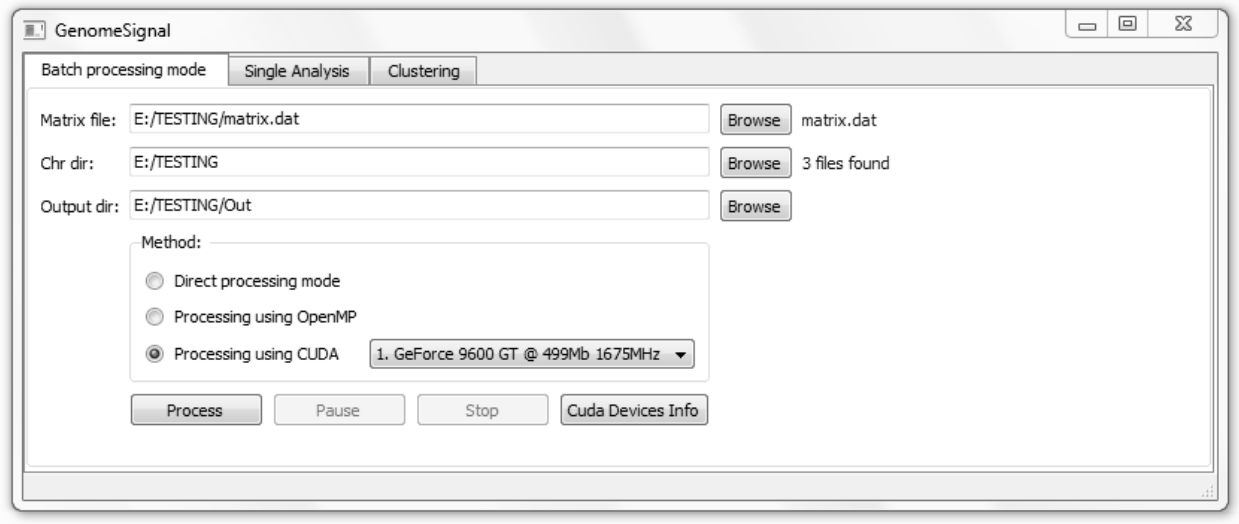

Рис.7. Режим пакетной обработки данных

Входными данными этого режима являются файл с библиотекой матриц, а также набор файлов с последовательностями генома. Пользователь может выбрать директорию, в которую будут записаны результаты работы. Программа обработает входные данные и сохранит полученные геномные профили с соответствующими именами в указанную директорию. Это длительная процедура, поэтому в программе реализована возможность останавливать процесс или ставить обработку на паузу. Также пользователь может выбрать один из вариантов работы программы: последовательная обработка, обработка с применением технологии OpenMP, обработка с применением технологии CUDA (если в

системе нет установленных совместимых с CUDA видеоадаптеров, данный метод будет недоступен).

4.2.2. Режим визуализации геномных профилей. На рис. 8. программа GenomeSignal настроена в режим «визуализация геномных профилей».

Рис.8. Режим визуализации геномных профилей

Входными данными этого режима является файл с библиотекой матриц, а также файл с последовательностью ДНК. Пользователь выбирает весовую матрицу и порог, для которых строится профиль. После обработки программа отображает результат в виде двух графиков. График справа (синий цвет) отображает распределение сайтов на данном участке ДНК. График слева (красный цвет) отображает соответствующий геномный профиль. Также пользователь может загрузить уже сохраненный в файле геномный профиль. В этом случае будет отображен только график геномного профиля.

Пользователь имеет возможность экспортировать полученный профиль в Microsoft Excel.

4.2.3. Режим кластеризации. На рис. 9 программа GenomeSignal настроена в режим «кластеризация».

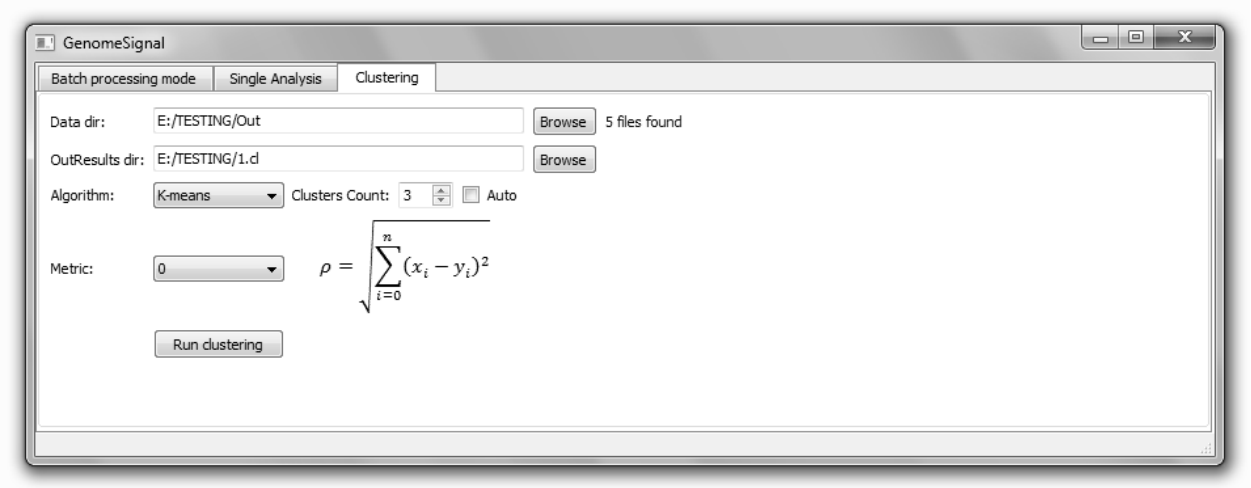

Рис.9. Режим кластеризации

Входными данными этого режима является набор файлов, содержащих построенные геномные профили. Пользователь выбирает метод кластеризации, количество кластеров, а также метрику, с помощью которой будут производиться расчеты. Программа обрабатывает данные и сохраняет результат разбиения в указанный пользователем файл.

4.2.4. Режим сравнения. На рис. 10 программа GenomeSignal настроена в режим «сравнение».

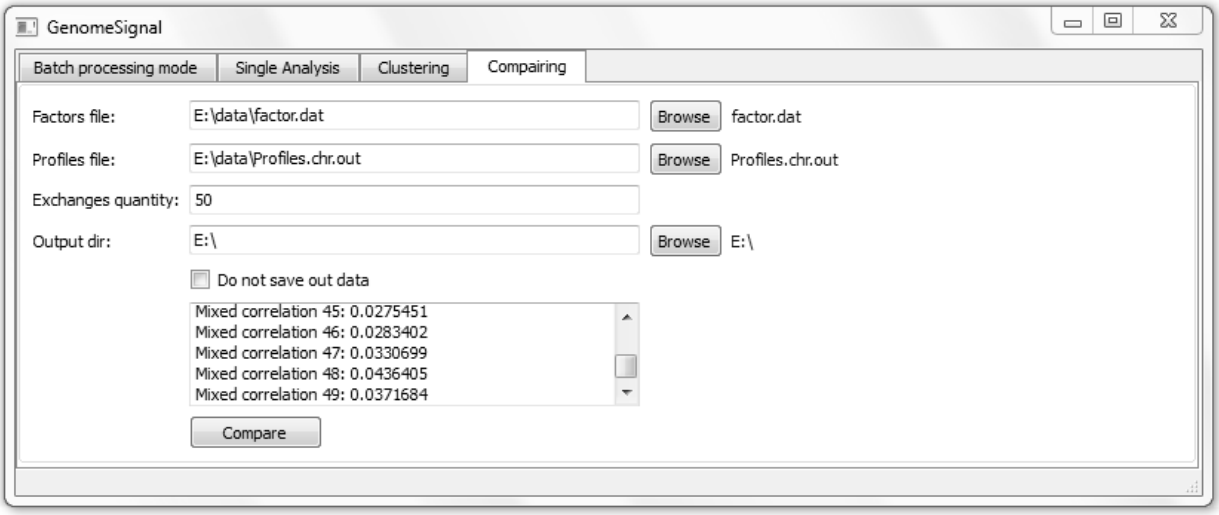

Рис.10. Режим сравнения

Входными данными этого режима являются файл с иерархией транскрипционных факторов, а также файл с геномными профилями. Пользователь может выбрать директорию, в которую будут записаны сгенерированные деревья и полученные матрицы расстояний, либо отказаться сохранять данные, выбрав соответствующее поле программы. Расстояние между весовыми матрицами соответствует расстоянию между их геномными профилями и

вычисляется как расстояние между векторами длины 20 с евклидовой метрикой. Иерархия транскрипционных факторов представляет собой дерево, в вершинах которого записаны соответствующие транскрипционным факторам весовые матрицы. Соответственно, в данном случае расстоянием между двумя весовыми матрицами является кратчайшее по количеству рёбер расстояние между ними в дереве иерархии. Результатом работы программы является набор коэффициентов корреляции между матрицей геномных профилей и матрицами иерархии факторов транскрипции (загруженной биологической иерархии и сгенерированных случайных иерархий).

5. Реализация на GPU. Развитие современной вычислительной техники предопределило два совершенно разных подхода к обработке данных, используемых в компьютерах в настоящее время. Еще 10 лет назад центральные процессоры испытывали стремительный рост быстродействия и параллельной обработки данных. Были введены специализированные векторные возможности (SSE2 и SSE3). Однако рост частот центральных процессоров резко замедлился из-за физических ограничений и высокого энергопотребления. К настоящему времени развитие ЦПУ сводится к размещению нескольких ядер на одном процессоре. Сейчас на рынке представлены центральные процессоры с количеством ядер до 16.

Противоположностью центральным процессорам выступают процессоры для видеокарт, которые изначально были спроектированы для параллельных векторных вычислений, используемых в 3D-графике. Современные видеочипы содержат сотни математических исполнительных блоков, выполняющих операции одновременно, что может значительно ускорить множество вычислительно интенсивных приложений. Для того чтобы задействовать возможности видеочипов для вычислений общего назначения, были разработаны технологии неграфических расчётов GPGPU (General-Purpose computation on GPUs). Наибольшее распространение на данный момент получила программно-аппаратная вычислительная архитектура CUDA от компании NVidia. По данным компании, около 20% мощности всего списка быстрейших суперкомпьютеров Top500 обеспечивают NVidia GPU [11].

Архитектура CUDA основана на расширении языка C и даёт возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений. Архитектура видеочипов, изначально разработанных из идеи SIMD (одни инструкции для большого потока данных) вычислений, делает перспективным их использование для решения многих научных задач. Еще одним немаловажным преимуществом применения видеокарт для неграфических вычислений является применение более быстрой памяти, поэтому видеочипам доступна в разы большая

пропускная способность памяти, что играет важную роль при обработке большого потока данных. На данный момент существует множество вычислительных кластеров на GPU, построенных на основе специализированных видеоадаптеров Tesla от компании NVidia с поддержкой CUDA, а также возможна установка нескольких видеокарт в персональные компьютеры с помощью технологии SLI, что позволяет получать еще большую производительность путем использования нескольких видеочипов одновременно.

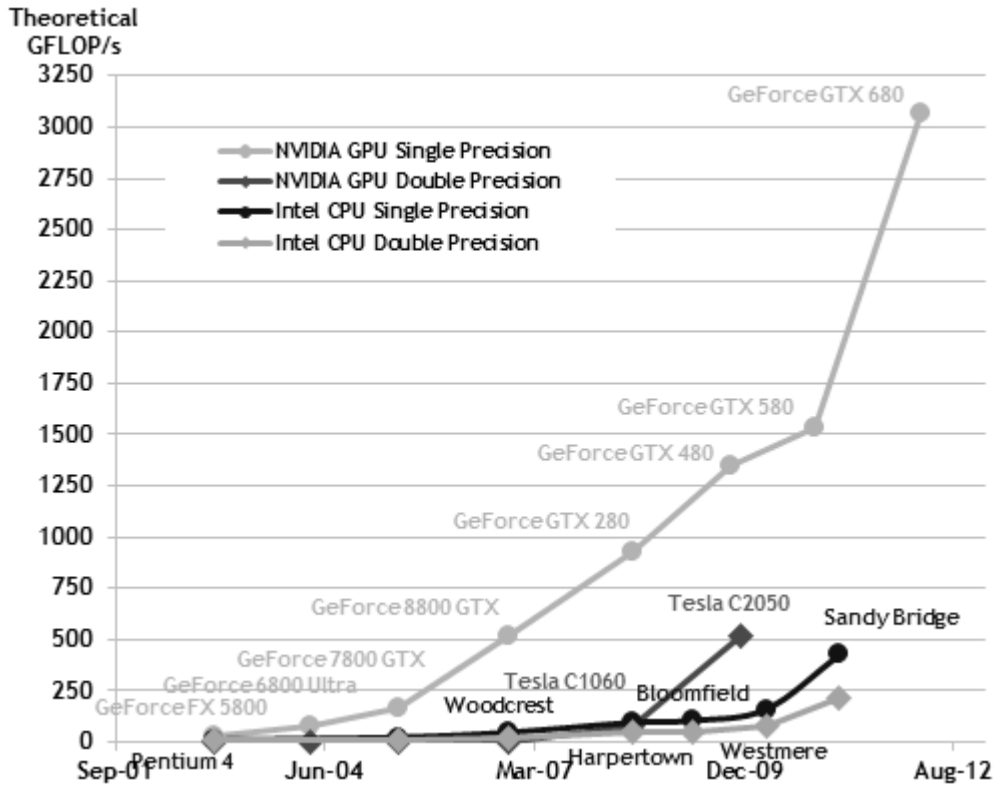

Рис. 11. Пиковая мощность для операций с плавающей точкой на CPU и на GPU [12]

В рамках нашей научной работы одной из подзадач является поиск потенциальных сайтов связывания с транскрипционными фактора на геноме. Данная подзадача является перспективной для параллельной реализации вычислений на видеочипе, так как требует обработки большого объема данных (геном) с помощью весовых матриц. Это позволяет ожидать многократного прироста производительности алгоритма по сравнению с его стандартной реализацией.

Блок-схема стандартного алгоритма поиска потенциальных сайтов связывания показана на рис. 12. Т.к. весовые матрицы имеют длину в среднем 10-20, мы имеем не меньше 5000 одинаковых операций вычисления веса участка генома для каждой весовой матрицы, которые выполняются последовательно. При параллельной реализации алгоритма мы

создаем сразу необходимое количество потоков, при этом создание и переключение между потоками на видеокарте происходит быстрее, чем на центральном процессоре. Каждый поток вычисляет вес участка со своим смещением, чем и достигается ускорение алгоритма.

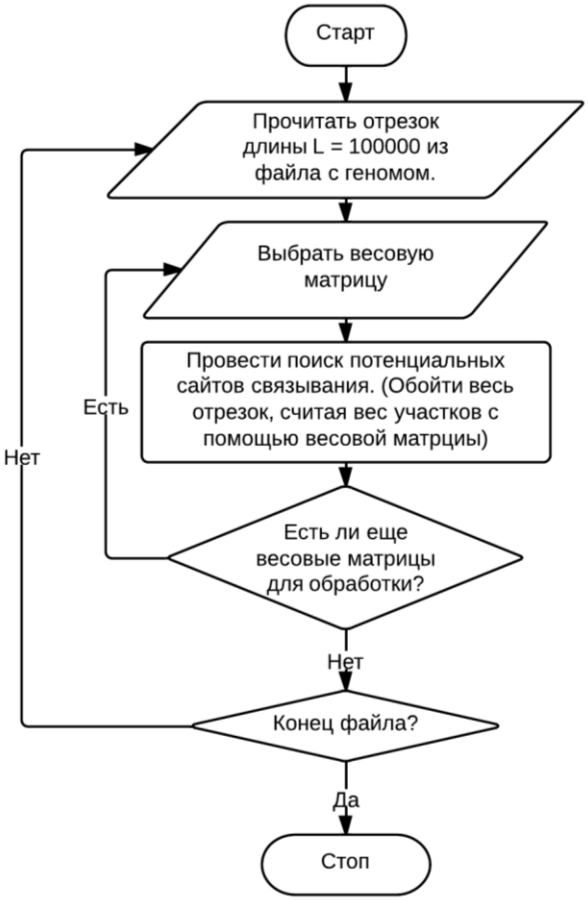

Рис.12. Алгоритм поиска потенциальных сайтов связывания

При достаточно большом количестве ядер CUDA все потоки произведут вычисления одновременно, в противном случае параллельно будет запущено максимально возможное количество потоков, остальные будут ожидать своей очереди.

Код на CUDA пишется в отдельном файле с расширением .cu, так как для его компиляции с основной программой требуется предварительная сборка с помощью CUDA компилятора nvcc. Заголовки функций, которые будут вызываться из основной программы, помещаются с заголовочный файл. Этот файл можно подключить в любом месте программы, как любой заголовочный файл. Синтаксис кода является расширением синтаксиса языка C. Для того чтобы работать с памятью на видеокарте, существуют функции **cudaMalloc** и **cudaFree.** Копирование из оперативной памяти в память видеокарты и обратно происходит с помощью функции **cudaMemcpy**, где последний параметр определяет направление копирования (**host** – оперативная память, **device** – видеопамять).

```
void ParallelStart(int *iA, int *iC, int *iG, int *iT, int *iDim, int *iW, char *iGene, int imcount, int *oRes)
{
    int Nthreads = 500;
    int Nblocks = 20;

    .......

    int *GlobRes = new int[imcount * 15];

    cudaMalloc((void**)&dlocRes, sizeof(int)*(Nblocks*Nthreads*15));
    for ( int i = 0; i < imcount; i++ )
    {
        for (int cl = 0; cl < Nblocks*Nthreads*15; cl++) hlocRes[cl] = 0;
        cudaMemcpy(dlocRes, hlocRes, sizeof(int)*(Nblocks*Nthreads*15), cudaMemcpyHostToDevice);
        SearchSites<<<Nblocks, Nthreads>>>(iA, iC, iG, iT, iDim, iW, iGene, i, dlocRes);
        cudaMemcpy(hlocRes, dlocRes, sizeof(int)*(Nblocks*Nthreads*15), cudaMemcpyDeviceToHost);

        .......

    }
    cudaFree( dlocRes );

    .......
    delete [] hlocRes;
    delete [] GlobRes;
}
```

Рис.13. Работа с памятью в CUDA программе

Запуск функции на видеочипе происходит с указанием двух параметров: числа блоков и числа потоков в каждом блоке (на рис. 13 функция **SearchSites** запускается на видеочипе). Соответственно, будет создано число потоков, равное произведению этих параметров. Во время выполнения можно определить, какой поток выполняет функцию, и использовать номер потока как параметр (на рис. 14 в **m_starter** помещается номер потока, в котором мы находимся).

```
__global__ void SearchSites(int *iA, int *iC, int *iG, int *iT, int *iDim, int *iW, char *iGene, int iMSel, int *oRes)
{
    int m_starter = blockIdx.x * 5000 + threadIdx.x * 10; // Вычисляем, в каком потоке мы находимся

    .......

if (denominator != 0)
{

    for ( int i = 0; i < 10; i++ )
    {
        record = 0;
        for ( int j = 0; j < iDim[m_matrIndex]; j++ )
        {
          if(( iGene[m_starter + i + j] == 'A' )||( iGene[m_starter + i + j] == 'a' )) record = record + iA[m_matrArrIndex + j];
          if(( iGene[m_starter + i + j] == 'C' )||( iGene[m_starter + i + j] == 'c' )) record = record + iC[m_matrArrIndex + j];
          if(( iGene[m_starter + i + j] == 'G' )||( iGene[m_starter + i + j] == 'g' )) record = record + iG[m_matrArrIndex + j];
          if(( iGene[m_starter + i + j] == 'T' )||( iGene[m_starter + i + j] == 't' )) record = record + iT[m_matrArrIndex + j];
        }
        record = record - iW[m_matrIndex * 2 + 0];
        for ( int coff = 15; coff > 0; coff--)
        {
            rt = 1.0 - (0.025*(float)coff);

            if ( (float)(record) >= rt*(float)(denominator) ) oRes[blockIdx.x*500*15 + threadIdx.x*15 + coff - 1]++;
            else break;
        }
    }
}

}
```

Рис.14. Функция, которую можно запустить на видеокарте

Как видно, процесс написания программы на CUDA слабо отличается от обычной программы на языке C. Это позволяет быстро адаптировать алгоритм для работы на видеокарте. Для нашей задачи поиска потенциальных сайтов связывания параллельная реализация алгоритма показала значительный прирост производительности. При этом тестирование было проведено также для видеокарт разных поколений —– GeForce 9600GT выпускается с 2008 года, построена на базе процессора G94, имеет 64 ядра CUDA; GeForce GTX650 выпускается с 2012 года, построена на базе процессора GK107, имеет 384 ядра CUDA. Результаты можно видеть на рис. 15.

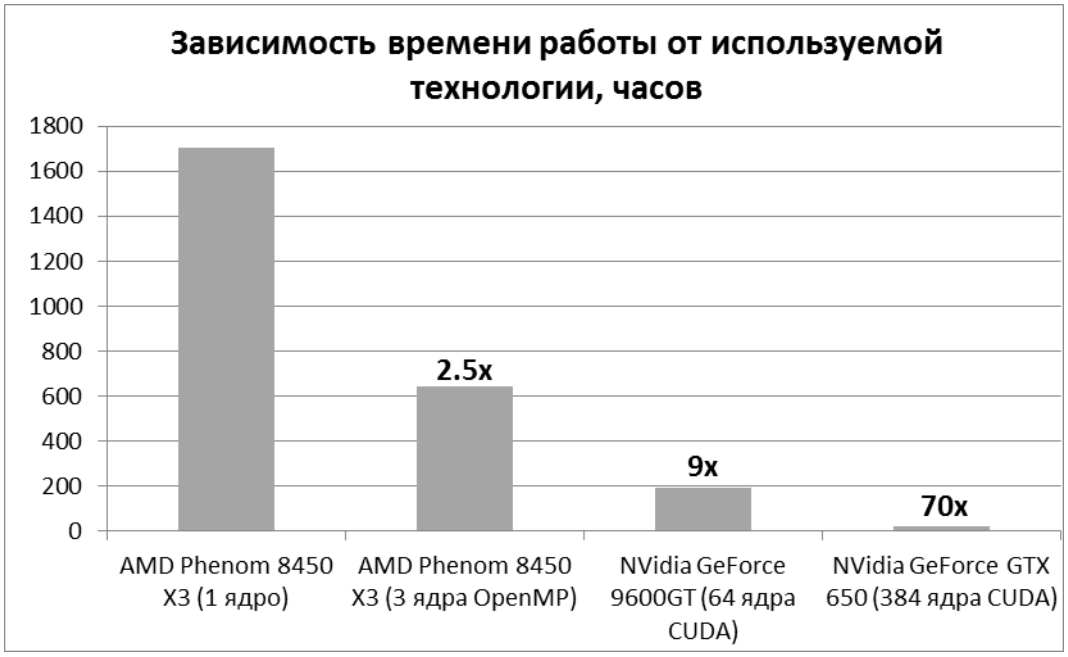

Рис.15. Работа алгоритма в разных режимах

Видно, что реализация алгоритма для работы на видеокарте позволила получить скачек в производительности до 70 раз. Можно предположить, что на более мощных видеокартах рост производительности будет продолжаться.

Таким образом, использование технологии CUDA оправданно и позволяет добиться сокращения времени работы программы, что является важным фактором для задач биоинформатики, где необходимо обрабатывать большие массивы данных.

6. Семплирование иерархий транскрипционных факторов. Для того чтобы проанализировать взаимосвязь геномных профилей и факторов транскрипции, определенных экспериментальным путем, воспользуемся существующей биологической классификацией.

Вообще говоря, транскрипционные факторы можно классифицировать несколькими способами:

- по механизму действия
- по белковой структуре;
- по происхождению.

По механизму действия транскрипционные факторы разбиваются на три класса:

1. **Главные факторы транскрипции**, вовлеченные в образование инициационного комплекса. Они присутствуют во всех клетках и взаимодействуют с кор-промотором генов, транскрибируемых РНК-полимеразой.

2. Факторы, взаимодействующие с upstream-участками ДНК – областями, расположенными до промотора, лежащими относительно него с другой стороны от кодирующей области гена.

3. Индуцируемые факторы — сходны с факторами, взаимодействующими с upstream-участками ДНК, но требуют активации либо ингибирования.

- По регуляторной функции.

1. Конститутивные — присутствуют всегда во всех клетках – главные факторы транскрипции.

2. Активируемые — активны в определенных условиях.

2.1. Участвующие в развитии организма (клеточно-специфичные) – факторы, экспрессия которых строго контролируется, но, начав экспрессироваться, не требуют дополнительной активации.

2.2. Сигнал-зависимые — требующие внешнего сигнала для активации**.**

2.2.1. Внеклеточные сигнал-зависимые – ядерные рецепторы.

2.2.2. Внутриклеточные сигнал-зависимые — активируются низкомолекулярными внутриклеточными соединениями.

2.2.3. Мембраносвязанные рецептор-зависимые — фосфорилируются киназами сигнального каскада.

2.2.3.1. Резидентные ядерные факторы – находятся в ядре независимо от активации.

2.2.3.2. Латентные цитоплазматические факторы – в неактивном состоянии локализованы в цитоплазме, после активации транспортируются в ядро.

- Структурная классификация.

По данному признаку факторы транскрипции классифицируют на основании сходства первичной структуры (что предполагает и сходство третичной структуры) ДНК-связывающих доменов [13,14]. Данная классификация представлена деревом, узлы которого соответствуют некоторой общей структуре ДНК-связывающих доменов, присущей транскрипционным факторам, соответствующие весовые матрицы которых содержатся в данном узле.

Для того чтобы оценить актуальность и применимость введенной нами конструкции геномных профилей, будем использовать структурную классификацию транскрипционных факторов, так как она является наиболее доступной и хранит данные в структурном виде в виде дерева. Проведем семплирование исходной классификации и проанализируем, насколько коррелирует семплированная выборка с полученными геномными профилями. Под семплированием будем понимать процесс получения случайного дерева классификации, по структуре похожего на существующую биологическую классификацию. Для получения семплов будем пользоваться двумя операциями:

1) Обмен значениями — два узла дерева меняются набором весовых матриц, соответствующих данной структуре. Пример операции изображен на рис. 16.

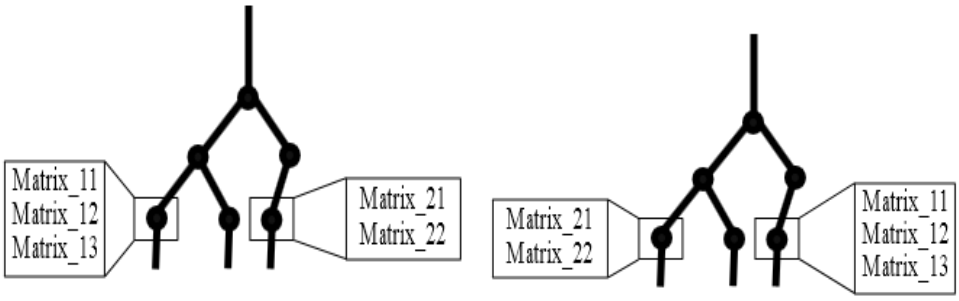

Рис.16. Обмен значениями двух узлов до (слева) и после (справа) операции

2) Обмен дочерней классификацией – два узла меняются местами вместе с оставшейся под ними дочерней классификацией. Пример операции изображен на рис. 17.

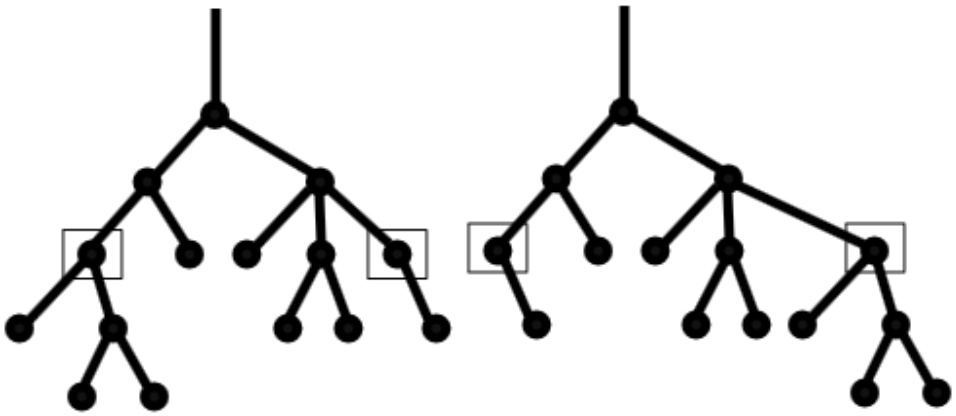

Рис.17. Обмен дочерней классификацией до (слева) и после (справа) операции

Таким образом, применив на каждом уровне заданное количество операций обмена, получим семплированное дерево классификации. Подробнее алгоритм изображен на рис. 18.

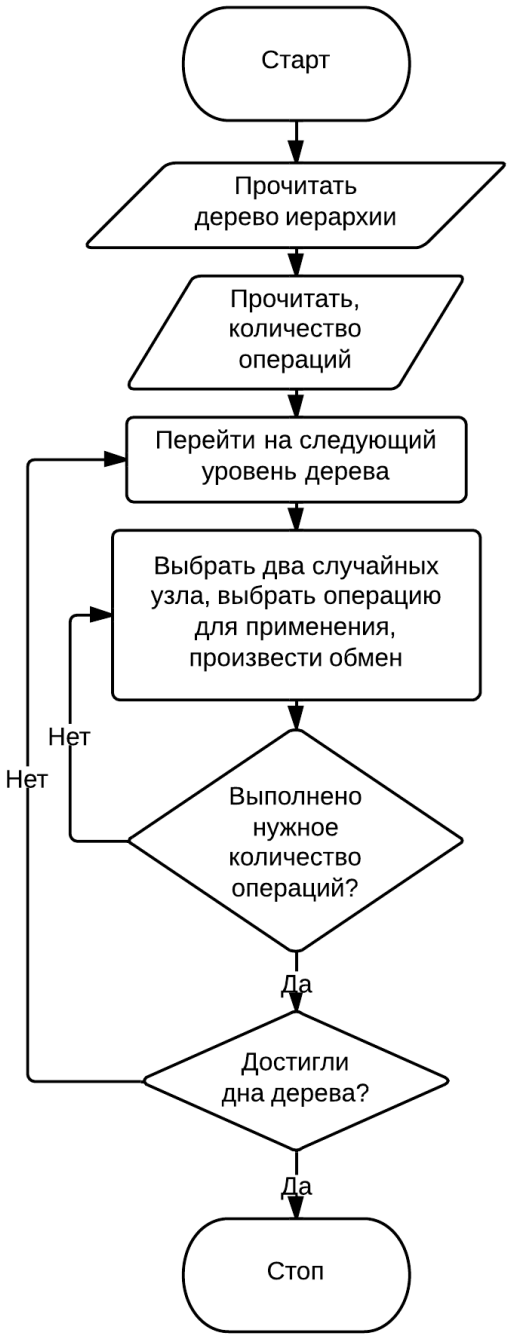

Рис.18. Алгоритм семплирования дерева иерархии

7. Сравнение матрицы расстояний и иерархии. Применив алгоритм семплирования, получим выборку T, T_1, T_2, …, T_N, где T — оригинальное дерево иерархии факторов транскрипции, а T_i — дерево, полученное в результате семплирования. Для каждого дерева построим матрицу расстояний, где номер строки/столбца есть номер весовой матрицы в списке всех весовых матриц для факторов транскрипции, а значения элементов матрицы расстояний суть число рёбер в минимальном пути дерева между вершинами, содержащими выбранные весовые матрицы (так как одна весовая матрица может содержаться в нескольких вершинах классификации). Получим набор матриц расстояний D, D_1, D_2, …, D_N. Для анализа

полученных данных построим матрицу расстояний для геномных профилей. Т.к. геномные профили — это вектора длины 20, мы можем вычислить расстояние между парами профилей с помощью евклидовой метрики. Следовательно, получим матрицу P, элементами которой являются расстояния между соответствующими парами геномных профилей. Для сравнения полученных результатов вычислим линейный коэффициент корреляции между матрицей расстояний геномных профилей и каждой из матриц для деревьев классификации. Коэффициент корреляции рассчитывается по формуле (суммирование проводится по всем элементам матриц):

$$r(X,Y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2}\sqrt{\sum(y_i - \bar{y})^2}},$$

где $\bar{x} = \frac{1}{n}\sum x_i$ и $\bar{y} = \frac{1}{n}\sum y_i$ – средние значения элементов матриц.

Таким образом, мы получили набор r, r_1, r_2, …, r_N значений корреляции, где r = r(P, T), r_i = r(P, T_i). На основании полученных данных мы можем построить гистограмму распределения коэффициентов корреляции случайных деревьев и, проанализировав разницу корреляции случайных деревьев и существующей иерархии, сделать вывод о биологической применимости геномных профилей для факторов транскрипции.

8. Результаты. С помощью созданной авторами программы GenomeSignal были построены геномные профили для весовых матриц библиотеки TRANSFAC. При этом реализация алгоритма поиска потенциальных сайтов связывания для GPU показала **семидесятикратный** прирост производительности на видеокарте GeForce GTX 650 по сравнению с последовательными вычислениями.

Для того чтобы проанализировать взаимосвязь геномных профилей и факторов транскрипции, была сгенерирована выборка из N=400 случайных деревьев, основанных на существующем дереве классификации. На каждом уровне исходного дерева проводилось C=5000 операций обмена, вариант обмена выбирался каждый раз случайным образом, так же случайно выбиралась пара узлов для обмена. Таким образом, мы построили 400 случайных деревьев и вычислили соответствующие им матрицы расстояний. Вычислив корреляцию полученных данных с матрицей расстояний геномных профилей, мы получили следующие результаты.

Для того чтобы получить информацию о распределении коэффициентов корреляции, была построена соответствующая гистограмма. Для этого отрезок между минимальным и

максимальным значениями корреляции [-0,00211819; 0,0700183] был разбит на N=20 полуинтервалов одной длины, и был произведен подсчет количества значений, попавших в тот или иной полуинтервал.

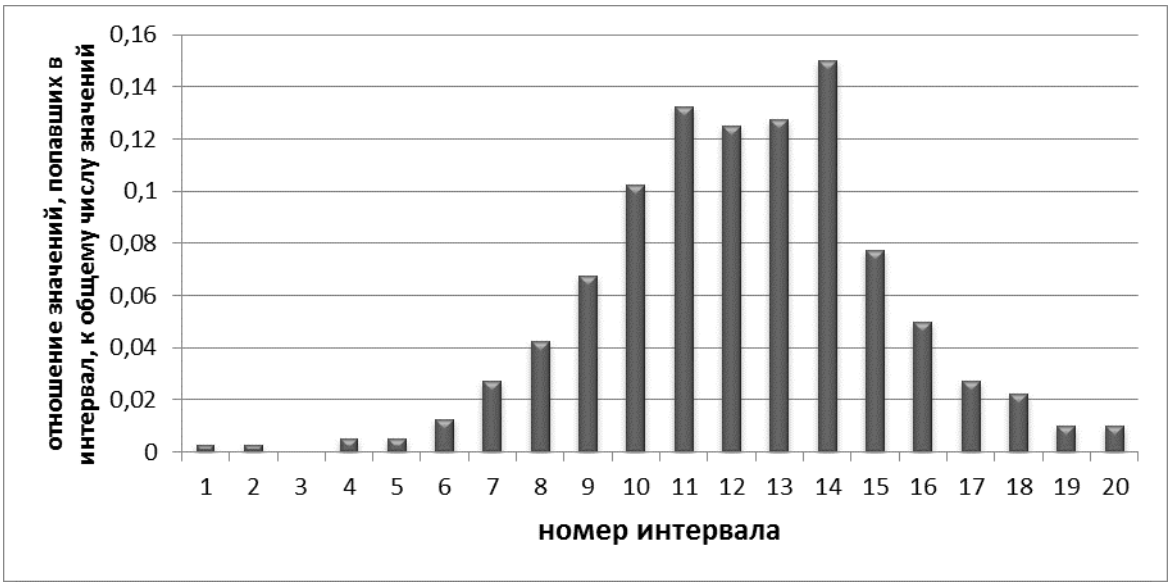

Рис.19. Гистограмма распределения коэффициентов корреляции

Также был построен график значений коэффициентов корреляции.

Рис.20. График полученных значений корреляции для случайно сгенерированных деревьев

Можно видеть, что корреляция с существующими данными в среднем в два раза выше, чем со случайно созданной иерархией транскрипционных факторов. На основе анализа полученных данных был сделан вывод о том, что получение и изучение свойств геномных профилей является оправданным и хорошо соотносится с существующими данными, полученными экспериментальным путем в лабораториях.

9. Заключение. Разработанная программа позволяет строить распределение сайтов на хромосоме, их профили, а также графически отображать результаты в виде гистограмм. Данная программа написана с целью быстрого построения геномных профилей на последовательностях ДНК, необходимых для анализа и сравнения структур весовых матриц. Реализованы несколько алгоритмов кластеризации геномных профилей. Реализованы алгоритмы построения матриц расстояний с помощью новой введенной конструкции «геномные профили весовых матриц» и с помощью существующей иерархии транскрипционных факторов. Реализован алгоритм получения случайных деревьев иерархии транскрипционных факторов, основанный на многократном изменении существующего дерева классификации. Проведен анализ и сделан вывод о целесообразности использования геномных профилей при исследовании влияния сайтов связывания, находящихся в окрестности некоторого данного сайта.

Параллельная реализация алгоритма поиска потенциальных сайтов связывания с транскрипционными факторами позволяет значительно ускорить работу программы на современных видеоадаптерах, поддерживающих технологию CUDA. Эксперименты показали семидесятикратный рост производительности по сравнению с последовательными вычислениями на центральном процессоре. Использование программы на новейших мощных видеоадаптерах позволит получить более высокие показатели скорости работы.

Список литературы

1. Колчанов Н.А., Гончаров С.С., Лихошвай В.А., Иванисенко В.А. Системная компьютерная биология // СО РАН, 2008. С. 10–70.
2. Peter J. Park ChIP–seq: advantages and challenges of a maturing technology // Nature Reviews Genetics. 2009. Vol. 10. P. 669-680.
3. Жимулёв И.Ф. Общая и молекулярная генетика // Новосибирск: Изд-во НГУ, 2002. 458 с.
4. Kozak, M. Initiation of translation in prokaryotes and eukaryotes // Gene. 1999. Vol. 234(2). P. 187–208.
5. Veenstra, G.J., A.P. Wolffe. Gene-selective developmental roles of general transcription factors // Trendsin Biochemical Sciences. 2001. Vol. 26(11). P. 665–671.

6. TRANSFAC: [Электронный ресурс] // Biobase. URL: http://www.gene-regulation.com/pub/databases.html. (Дата обращения: 01.05.2013).
7. Harr, R., M. Häggström, and P. Gustafsson. Search algorithm for pattern match analysis of nucleic acid sequences // Nucleic Acids Research. 1983. Vol. 11(9). P. 2943–2957.
8. Stormo, G.D., et al., Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli // Nucleic Acids Research. 1982. Vol. 10(9). P. 2997–3011.
9. Bulyk, M.L., P.L. Johnson, G.M. Church. Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors // Nucleic Acids Res, 2002. Vol. 30(5). P. 1255–61.
10. Zhou, Q., J.S. Liu. Modeling within-motif dependence for transcription factor binding site predictions // Bioinformatics. 2004. Vol. 20(6). P. 16–909.
11. Jen-Hsun Huang on NVidia GTC 2013 // NVidia GTC 2013. Материалы конференции.
12. CUDA C Programming Guide: [Электронный ресурс] // NVidia. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide. (Дата обращения: 01.05.2013).
13. Matys V, et.al. TRANSFAC and its module TRANSCompel: transcriptional gene regulation in eukaryotes // Nucleic Acids Res. 2006. Vol. 34.
14. Stegmaier P, Kel AE, Wingender E. Systematic DNA-binding domain classification of transcription factors // Genome informatics. International Conference on Genome Informatics. 2004. Vol. 15 (2). P. 276–86.

UDK: 519.688

Title: Program for building and clustering of genomic profiles using GPU

Authors:

Nikitin Sergey Ilyich (Novosibisrk State University),

Cheryomushkin Evgeny Sergeevich (Ershov Institute of Informatics Systems)

Abstract: Before RNA transcription starts, special segments of DNA form a complex of regulatory proteins called transcription factors. This complex allows RNA polymerase to be bound to DNA and to start reading RNA. It is a difficult problem to search binding sites on DNA because of many factors influencing the binding. In particular, other sites in the vicinity of a given site may influence binding. To reveal those dependencies the authors introduce histograms of density distribution of binding sites, called genomic profiles.

The software package developed in the scope of this work allows building genomic profiles using the prediction of binding sites by a weight matrices algorithm for different implementations: for multicore CPU’s or NVidia GPU’s using CUDA. In addition, the software allows clustering

genomic profiles using K-means clustering and hierarchical clustering. The algorithm allows to build samples – random transcription factors hierarchies based on the existing experimental structural classification to estimate the correspondence between genomic profiles construction and the existing classification. The analysis of correspondence of genomic profiles with biological classification of transcription factors was developed using the sampling algorithm.

Keywords: DNA regulation, Genome, Position weight matrices, Transcription factors, transcription factor binding sites, Clustering, Nvidia CUDA, GPU computing