Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Магистерская диссертация

Разработка метода сборки транскриптома на основе анализа компонент связности графа де Брёйна

В. О. Долганов, гр. 6538

Научный руководитель: к.т.н. Ф.Н. Царев

Санкт-Петербург

2014

Оглавление

ВВЕДЕНИЕ

С конца XX века все большую популярность приобретает наука бионформатика. Это связано с открытием в середине XX века «Центральной догмы молекулярной биологии», содержащей правила реализации генетической информации, а так же развитием секвенаторов. Секвенаторы позволили получать информацию о последовательностях нуклеотидов. Многие задачи биоинформатики связаны с тремя уровнями хранения и реализации биологической информации: геномом, траскриптомом и протеомом.

Транскриптомом называется совокупность всех транскриптов, синтезируемых клеткой или группой клеток. Он отражает гены, которые экспрессируются в заданный момент времени. Так же транскриптом можно рассматривать, как предшественник протеома.

Знание транскриптома полезно во многих задачах биоинформатики и медицины, таких как:

1. Анализ экспрессии генов.
2. Открытие новых изоформ транскриптов.
3. Ускорение исследования генов и расширение семейства генов.
4. Определение события слияния транскриптов.
5. Построение физической и генетической карты.
6. Анализ аллелей.

Транскрипт можно представить, как последовательность нуклеотидов. Таким образом, транскриптом можно рассматривать, как совокупность множества различных нуклеотидных последовательностей.

Получение этих нуклеотидных последовательностей по образцу можно разбить на два этапа:

1. Секвенирование молекул кДНК, полученных из РНК (производится с помощью специального оборудования — секвенаторов).
2. Сборка транскриптомных последовательностей (производится с помощью специальных алгоритмов на компьютерах).

Задача сборки транскриптома на первый взгляд напоминает задачу сборки генома. К сожалению, в задаче сборки транскриптома присутствуют свои сложности, поэтому сборщики генома не подходят для сборки транскриптома.

В связи с этим разработано множество различных сборщиков транскриптома. Наиболее популярные их них рассчитаны на достижение наилучшего качества сборки. Поэтому данные сборщики могут работать неделями на больших объемах данных.

В данной работе была поставлена цель разработать быстрый сборщик транскриптома, использующий небольшое количество памяти.

Таким сборщиком мог бы воспользоваться исследователь, для предварительного анализа данных, не дожидаясь, пока закончит работу более точный алгоритм.

Данная работа состоит из четырех глав.

В первой главе описаны основные понятия биоинформатики, задача сборки транскриптома, а так же были поставлены задачи данной работы.

Во второй главе описаны существующие сборщики транскриптома. Были рассмотрены оба типа сборщиков: использующиеся и не использующие референс.

В третьей главе приведено описание разработанного метода. Вначале приведена основная идея, затем более подробно описан каждый шаг алгоритма.

В четвертой главе описаны детали реализации, а так же приведены результаты экспериментов. Реализованный метод был сравнен с популярным сборщиком транскриптома Trinity на реальных данных.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе описаны основные понятия в области биоинформатики, рассмотрена задача секвенирования, рассказано о задаче сборки транскриптома и отличии ее от сборки генома. Описаны проблемы, возникающие при решении данной задачи. Обозначены цели, поставленные в данной работе.

1.1. Биоинформатика

В настоящее время одним из популярных направлений в науке является биоинформатика. Основной целью данной науки является помощь в понимании биологических процессов. Для задач биоинформатики характерны большие объемы данных. Поэтому методы для их решения должны быть способны проводить множество вычислений над большим количеством данных за разумное время. Биоинформатика используется во множестве задач, таких как расшифровка генома, нахождение генов, выравнивание последовательностей, построение структуры белка, анализ экспрессии генов.

1.2. ДНК, РНК, транскриптом

Дезоксирибонуклеиновая кислота (ДНК) — биологический биополимер, отвечающий за хранение и передачу генетической информации из поколения в поколение. ДНК состоит из двух цепочек, являющимися последовательностями нуклеотидов. Каждый нуклеотид в последовательности ДНК содержит одно из четырех азотистых оснований: аденин(A), гуанин(G), тимин(T) и цитозин(C). Азотистые основания одной цепочки ДНК соединены с азотистыми основаниями другой цепочки. При этом выполняется принцип комплементарности: аденин соединяется с тимином, а гуанин с цитозином. Таким образом, одна цепочка ДНК однозначно задает вторую.

Хранящаяся в ДНК информация отвечает за важные свойства организмов, таких как наследственность и изменчивость. ДНК потомков

одной особи более схожи, чем ДНК двух случайных особей одного вида, а так же ДНК двух особей одного вида более схожи, чем ДНК разных особей.

Для существования многим организмам необходимы белки. Для создания белков используется информация, хранящаяся на ДНК. Рибонуклеиновая кислота (РНК) играет важную роль в процессе создания белков. Одной из функций РНК является перенос информации с ДНК к месту синтеза белка. РНК состоит из одной цепи нуклеотидов. В состав нуклеотидов РНК входят такие же основания, что и в ДНК за исключением тимина, который заменен на урацил.

Молекула РНК образуется в процессе считывания генетической информации с ДНК, называемого процессом транскрипции. Образованная молекула называется транскриптом. Совокупность всех транскриптов, синтезируемых одной или группой клеток называется транскриптомом.

Знание транскриптома очень важно в биологии и медицине, так как транскриптом отражает экспрессию генов в определенный момент времени. Так же транскриптом является в какой-то степени предшественником протеома.

1.3. Задача секвенирования

Секвенирование биополимеров — определение их нуклеотидных последовательностей по имеющемуся образцу. В результате этого процесса получаются последовательности нуклеотидов, содержащиеся в образце.

Существующие методы секвенирования не позволяют прочитать за раз длинные участки биополимеров. В настоящее время наиболее популярны методы нового поколения (next-generation sequencing). В этих методах длинные последовательности нуклеотидов разбиваются на более мелкие (длинной около 500 нуклеотидов). После этого полученные небольшие последовательности считываются.

Одной из важных характеристик секвенатора является стоимость запуска. Для удешевления процесса секвенирования последовательности считывают не полностью, а только их префиксы и суффиксы. Данные,

полученные таким способом, называются парными чтениями (Рис. 1). Парные чтения представляют собой две последовательности нуклеотидов с разных цепочек ДНК, для которых известно, на каком расстоянии они находятся в исходной последовательности.

Рис. 1. – Парные чтения

Так как РНК является одноцепочной, то для получения её нуклеотидной последовательности использую обратную транскрипцию. Обратная транскрипция – процесс образования ДНК на матрице РНК. После этого, синтезированную молекулу кДНК фрагментируют и считывают. Процесс обратной транскрипции вносит ошибки, что усложняет сборку. В настоящее время ведется разработка по прямому секвенированию РНК.

В настоящее время для секвенирования транскриптома используют технологию секвенирования *RNA-seq* [1], пришедшую на смену микрочипам [2]. Микрочипы предоставляли относительные количественные данные о транскриптах в пробе, в то время как технология *RNA-seq* позволяет получить абсолютную количественную информацию.

Для того чтобы получить исходные последовательности нуклеотидов биополимеров необходимо проанализировать данные секвенатора. Этой задачей занимаются разные сборщики. В частности для получения информации о транскриптоме используются сборщики транскриптома.

1.4. Сборка транскриптома

Задачей сборки транскриптома является восстановление наибольшего числа транскриптов по имеющейся информации, полученной с помощью секвенатора. Далее описаны особенности сборки транскриптома, типы сборщиков транксриптома, а так же структуры, используемые для решения данной задачи.

1.4.1. Особенности сборки транскриптома

Задача сборки транскриптома схожа с задачей сборки генома, а так же имеет схожие входные данные, но при сборке транскриптома имеются отличительные особенности, такие как:

1. Необходимо восстановить не одну длинную последовательность, а множество последовательностей, имеющих разную длину. В отличие от генома, где присутствует одна длинная молекула ДНК, при сборке транскриптома возникает задача восстановить множество нуклеотидных последовательностей, имеющих различную длину.
2. При секвенирование генома покрытие различных участков генома чтениями равномерно, что облегчает процесс нахождения ошибок секвенирования. При секвенирование транскриптома покрытие двух разных транскриптов отличается. Это связано с тем, что у различных генов разный уровень экспрессии в заданный момент. При этом допускается предположение, что покрытие чтениями участков в рамках одного транскрипта равномерно.
3. Существует процесс, который позволяет одному гену производить несколько мРНК и, как следствие, белков. Данный процесс называется альтернативным сплайсингом (Рис. 2). Многие гены эукариотов содержат экзоны и интроны. Интроны удаляются из пре-мРНК, а экзоны могут как включаться, так и нет в итоговый транскрипт. Таким образом, мультиэкзонные гены могут порождать различные транскрипты.

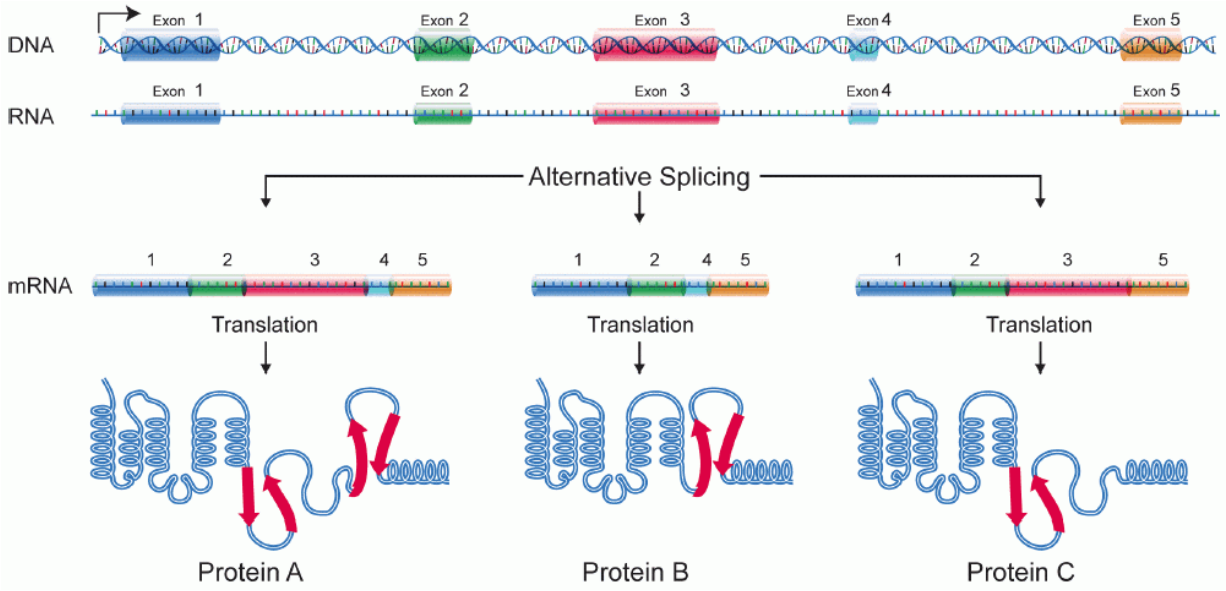

Рис. 2. – Альтернативный сплайсинг

Поэтому сборщики генома имеют малую эффективность при сборке транскриптома и их использование нецелесообразно. В связи с этим возникает необходимость создания новых алгоритмов для сборки транскриптома.

1.4.2. Типы транскриптомных сборщиков

Сборщики транскриптома, как и сборщики генома, делятся на два вида. Сборщики, которые используют референс и сборщики, которые референс не используют (*de novo*). Первый тип сборщиков зачастую обеспечивают лучшее качество сборки, но при этом им необходимо иметь референс, что при исследовании новых организмов не всегда возможно. Поэтому целью данной работы было разработать *de novo* сборщик транскриптома.

1.4.3. Используемы структуры

Существуют несколько наиболее популярных структур, которые используют в задачах сборки генома и транскриптома. Одной из таких структур является *k*-мер. *K*-мером называется последовательность длины *k*. В задачах сборки *k*-меры извлекаются из чтений. Так из чтения длины 40 получается 11 *k*-меров длины 30. Зачастую собирают статистику частоты *k*-меров по всем чтениям. *K*-меры, встречающиеся небольшое количество раз, выкидывают из дальнейшей сборки или пытаются исправить, так как высока вероятность наличия в них ошибки секвенирования. С точки зрения информатики, данная структура весьма эффективна в использовании. Как

при сборке генома, так и при сборке транскриптома существует четыре вида нуклеотидов (по числу различных азотистых оснований). Таким образом, на хранение одного нуклеотида достаточно всего двух битов.

Один из способов дальнейшего использования *k*-меров – построение с их помощью графа де Брёйна [3] (Рис. 3). Вершинами графа де Брёйна являются *k*-меры. Каждому ребру в графе соответствуют (*k*+1)-меры. Направленное ребро соединяет две вершины графа тогда и только тогда, когда *k*-мер начальной вершины является префиксом (*k*+1)-мера ребра, а *k*-мер конечной – суффиксом.

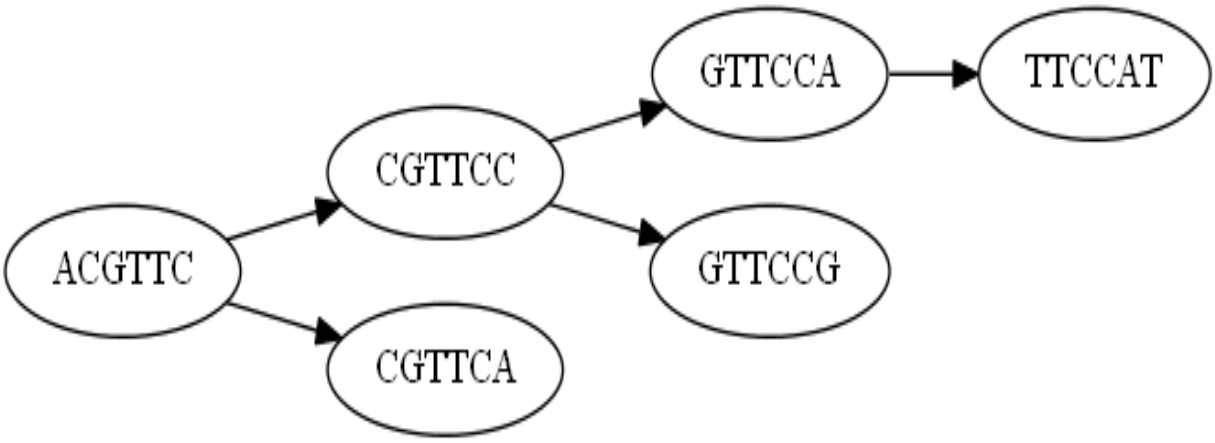

Рис. 3. – Граф де Брёйна

Граф де Брёйна используется для нахождения более длинных, чем изначальных последовательностей. За счет своей структуры он хранит информацию о перекрытии начальных чтений, тем самым позволяет строить длинные последовательности. Для этого осуществляется поиск путей в этом графе между его вершинами. Имея информацию о частоте *k*-меров и начальные чтения, появляется возможность решать неоднозначности в выборе пути, такие как развилки и пузыри, и выкидывать ошибочные *k*-меры, возникшие вследствие ошибок секвенирования. Заметим, что для хранения графа де Брёйна достаточно хранить только его вершины, так как ребра однозначно восстанавливаются по ним. Так же для хранения графа де Брёйна достаточно хранить только половину его вершин. Так как цепочки ДНК обратнокомплементарны друг другу, то наличия одного *k*-мера подразумевает наличие в графе *k*-мера, обратнокомлементарного первому, хранящегося на парной цепочке. Таким образом, достаточно хранить только лексикографически меньший из них.

1.5. Постановка задачи

В настоящее время существует множество сборщиков транскриптома. Многие из них используют сложные алгоритмы, которые работают продолжительное время, вследствие чего время их работы может составить больше недели на больших входных данных. Такое продолжительное время работы может значительно замедлить исследования.

Поэтому возникает задача реализации сборщика, который работал быстрее. Так же существующие сборщики используют значительные количества оперативной памяти. Поэтому дополнительным требованием к быстрому сборщику является использование меньшего количества оперативной памяти. Такой сборщик возможно запустить не на основном сервере, который будет использоваться для более точного сборщика, а на обладающим меньшими ресурсами дополнительном сервере. При этом получить предварительные результаты намного быстрее, что позволит приступить к предварительному анализу данных до завершения работы основного сборщика. Такой сборщик не будет обладать таким же качеством сборки, но сможет предоставить предварительную картину. Например, возможно намного раньше узнать, что экспрессируется интересующий ген, и предпринять в связи с этим соответствующие действия в направлении исследования.

В данной работе была поставлена цель разработки сборщика, удовлетворяющего описанным выше требованиям. Были поставлены следующие задачи:

1. Произвести обзор существующих сборщиков транскриптома.
2. Придумать алгоритм сборки транскриптома, работающий быстрее, чем существующие аналоги.
3. Реализовать придуманный алгоритм на одном из языков программирования.
4. Произвести сравнение реализованного сборщика с одним из распространенных на данный момент сборщиков.

1.6. Выводы по главе 1

1. Описана наука биоинформатика.
2. Описаны биологические термины, такие как ДНК, РНК, транскриптом.

3. Описана задача секвенирования, технология секвенирования РНК, а так же описан формат выходных данных – парные чтения.
4. Описана задача сборки транскриптома, ее особенности, виды сборщиков, а так же популярны структуры, используемые в сборщиках транскриптома.
5. Поставлены цель и задачи данной работы.

ГЛАВА 2. ОБЗОР ОБСУЩЕСТВУЮЩИХ СБОРЩИКОВ

В данной главе описаны существующие программные средства для сборки транскриптома. Рассмотрено несколько *de novo* сборщиков и один сборщик, использующий референс.

2.1. Существующие программные средства

Далее описаны следующие сборщики транскриптома: *Trinity*, *Oases*, *Trans-ABySS*, *Multiple-k*, *Cufflinks*.

2.1.1. Программное средство Trinity

В статье [4] изложен подход к сборке транскриптов в программном средстве *Trinity*. Этот подход состоит из трех этапов (Рис. 4):

1. Этап *Inchworm* – первоначальная сборка контигов на основе *k*-меров, полученных из чтений.
2. Этап *Chrysalis* – разбиение полученных на первом этапе контигов на группы и построение для каждой группы графа де Брёйна.
3. Этап *Butterfly* – анализ графов де Брёйна, с последующим выводом всех возможных последовательностей транскриптов.

Этап *Inchworm* состоит из шести шагов:

1. Вначале составляется словарь *k*-меров из всех чтений.
2. Затем из полученного словаря удаляются *k*-меры, в которых с высокой вероятностью содержится ошибка.
3. После этого из словаря выбирается наиболее часто встречаемый *k*-мер. Этот *k*-мер является исходной последовательностью для наращивания.
4. Затем из словаря выбирается наиболее часто встречаемый *k*-мер, имеющий подстроку длины *k*-1, равную суффиксу или префиксу наращиваемой последовательности, и данная последовательность расширяется с помощью оставшегося символа данного *k*-мера. После этого этот *k*-мер удаляется из

словаря. Этот шаг повторяется до тех пор, пока это возможно. Полученная в итоге последовательность сохраняется.

5. Затем происходит повторение шагов 3-4, пока словарь не истощится.

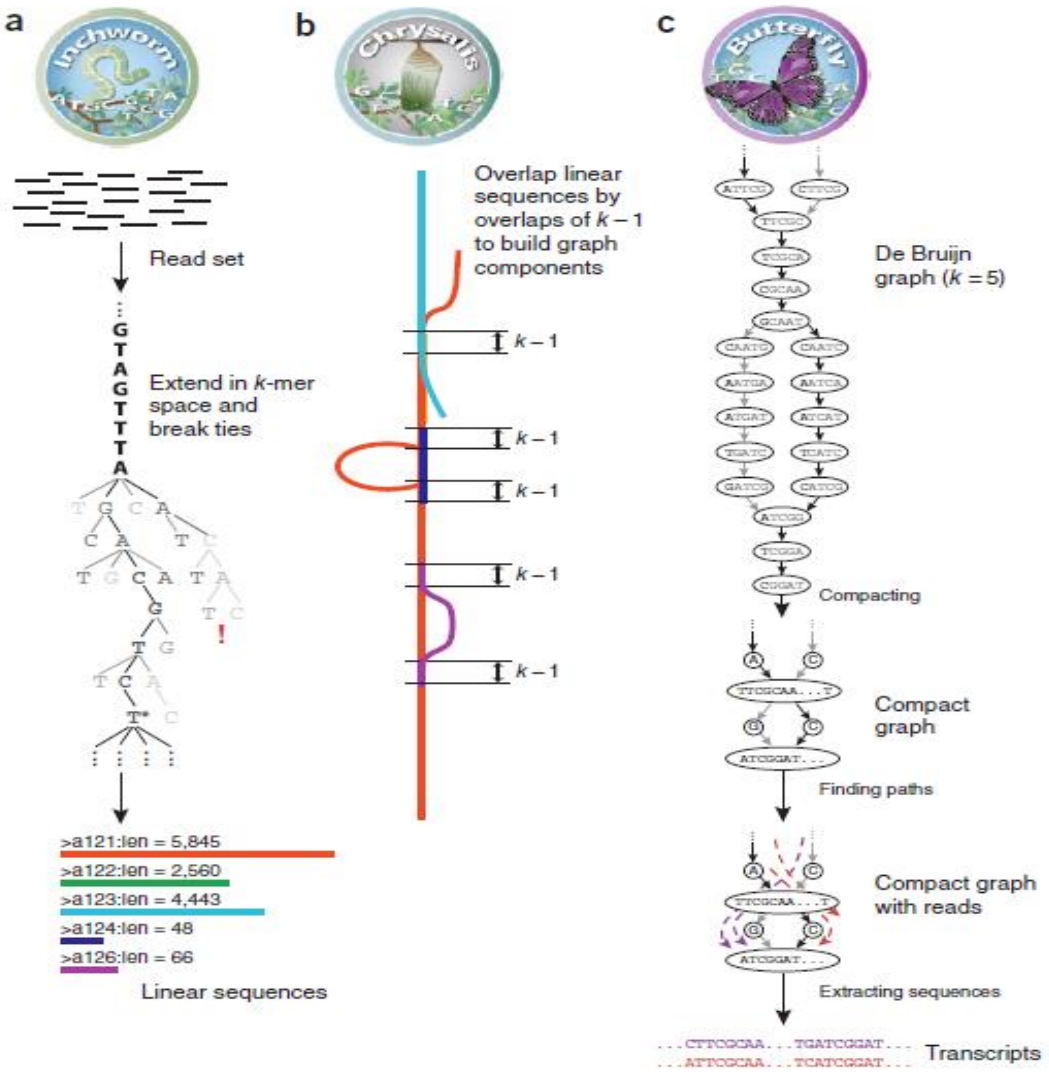

Рис. 4. – Схема работы программного средства *Trinity*

Этап *Chrysalis* состоит из трех шагов:

1. Рекурсивно группируются контиги, полученные на этапе *Inchworm*, в компоненты. Контиги группируются, если они имеют хотя бы одну общую подстроку длины *k*-1, и есть минимальное число таких чтений, что они охватывают переход между контигами и имеют перекрытие длины (*k*-1)/2 с общей подстрокой с обеих сторон.

2. Затем для каждой компоненты строится граф де Брёйна с последовательностями длины *k*-1 для обозначения вершин и последовательностями длины *k* для ребер. Каждому ребру в данных графах присваивается вес, основанный на числе *k*-меров из оригинального набора чтений, которые поддерживают данное ребро.
3. После этого, каждому чтению назначается компонента, содержащая наибольшее число *k*-меров данного чтения, и определяют регионы внутри каждого чтения, которые присутствуют в компоненте.

Этап *Butterfly* состоит из двух шагов:

1. На первом шаге происходит объединение вершин в однозначных путях в графе де Брёйна для образования вершин, которые представляют более длинные последовательности, и удаляются ребра, представляющие незначительные отклонения (поддерживаемые сравнительно небольшим числом чтений).
2. На втором шаге происходит поиск путей в полученном графе, используя чтения для разрешения неясностей.

2.1.2. Программное средство Oases

В статье [5] изложен подход к сборке транскриптов в программном средстве *Oases* (Рис. 5).

Вначале выполняется первая стадия сборщика *Velvet*. Полученные на этом этапе контиги исправляются с помощью нескольких алгоритмов исправления ошибок.

Первый алгоритм исправления ошибок является немного модифицированным алгоритмом исправления ошибок сборщика *Velvet*, называемым *TourBus*. Алгоритм *TourBus* ищет в графе параллельные пути, имеющие одинаковые стартовые и конечные вершины. Если данные пути достаточно схожи, то путь с меньшим покрытием сливается с путем с большим покрытием.

Другим алгоритмом является локальное удаление ребер (*local edge removal)*. Для каждой вершины удаляются те исходящие ребра, которые имеют уровень покрытия меньше десяти процентов от суммы покрытий всех исходящих ребер этой вершины. Так же все контиги с покрытием меньше заданного (трехкратного) удаляются из сборки.

После фильтрации контигов, алгоритм *Oases* строит множества оценок на расстояния между контигами, называемые скэффолдами. Соединение между двумя контигами может быть поддержано охватывающими их одиночными и парными чтениями. Число таких чтений для соединения называется поддержкой (*support)*. Если соединение поддерживается хотя бы одним одиночным чтением, оно называется прямым (*direct)*, иначе непрямым (*indirect)*. Так же соединениям присваиваются веса. За каждое поддерживающее одиночное чтение вес увеличивается на единицу. За парное чтение этот вес увеличивается на величину, основанную на вероятности нахождения данных чтений на местах, где они находятся.

Затем происходит фильтрация скэффолдов. Контиги с длиной больше пороговой (50+*k*-1 нуклеотидов) считаются длинными, остальные короткими. Соединения с небольшим значением поддержки (три или ниже) или с весом меньшим 0,1 удаляются. Два коротких контига рассматриваются только если между ними нет промежутка и они соединены прямым соединением. Короткий и длинный контиги могут быть соединены только прямым соединением, при этом допускается промежуток между ними. Для длинных контигов считается число парных чтений, которые должны их поддерживать. Если действительное число поддерживающих парных чтений для пары меньше десяти процентов от ожидаемой величины, то данное соединение не рассматривается.

Затем контиги собираются в кластеры, называемые локусами (*loci)*. Вначале группируются длинные контиги, имеющие между собой соединения. После этого к каждому кластеру добавляются короткие контиги, имеющие соединения с длинными контигами из этих кластеров. Затем кластеры упрощаются таким образом, чтобы удалить избыточные соединения на дальних расстояниях и оставить только соединения между соседями.

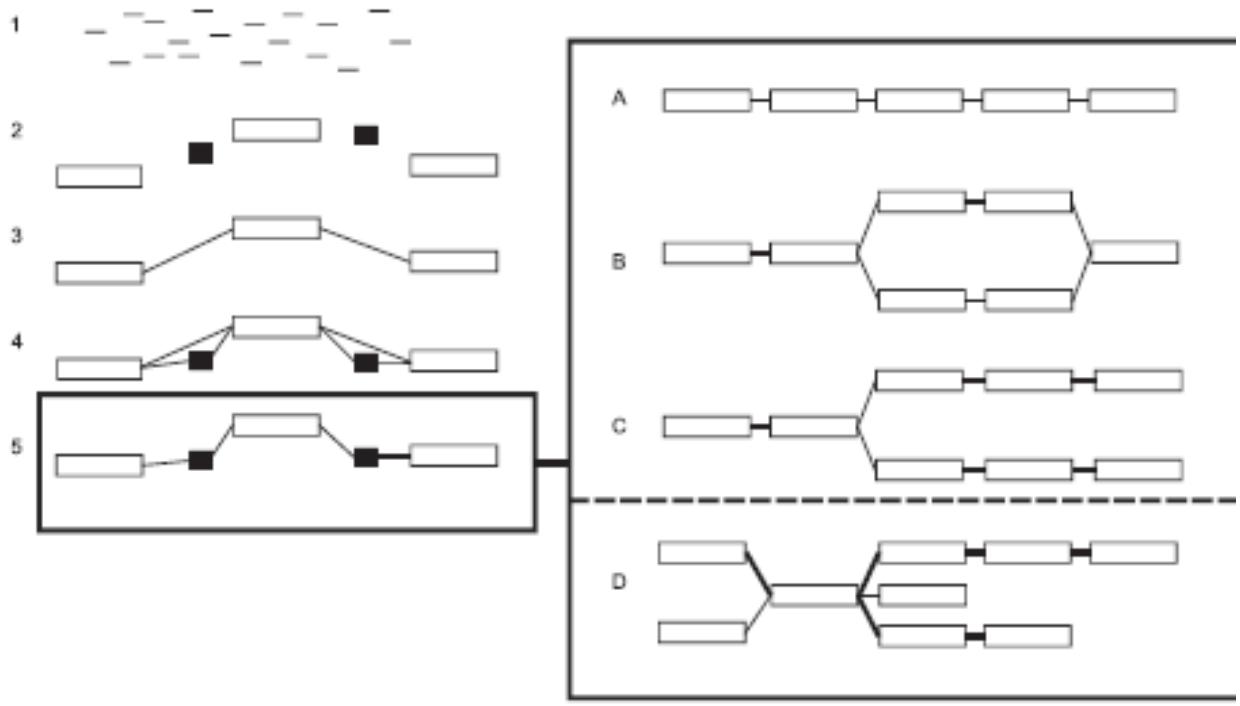

Рис. 5. – Схема работы программного средства *Oases*

На последней стадии транскрипты извлекаются из локусов. В большинстве случаев локус представляет собой простую топологию, которую можно разложить на один или два транскрипта. Выделяют три типа топологий: цепь (*chain)*, вилка (*fork)* и пузырь (*bubble)*. Алгоритм *Oases* определяет локусы с данными топологиями и извлекает из них соответствующие транскрипты.

2.1.3. Программное средство Trans-ABySS

В статье [6] изложен подход к сборке транскриптов в программном средстве *Trans-ABySS*. Основной идеей данного подхода является объединение результатов, полученных при работе сборщика *ABySS* для различных значений *k*.

После сборки первоначальных наборов контигов для различных k (например все k от 26 до 50), происходят объединения соседних наборов: набор контигов для k_i объединяется с набором k_{i+1}, набор k_{i+2} с k_{i+3}. При этом учитываются только контиги с длиной не меньше $(2k\text{-}2)$ нуклеотидов. При объединении наборов, вначале происходит выравнивание контигов из первого множества a на контиги из его парного множества b. Затем происходит выравнивание в обратную сторону: выравнивание контигов из множества b на контиги из множества a. Для выравнивания используется программное средство *BLAT*. Затем, используя полученную информацию, из объединенного набора удаляются те контиги, которые являются

подстроками других контигов в этом наборе. Попарное объединение множеств происходит до тех пор, пока не останется единственное множество.

Для полученных контигов применяются алгоритмы фильтрации. Один из алгоритмов удаляет «островные контиги» - контиги, не имеющие соседних контигов в графе де Брёйна и имеющие длину меньше, чем ограничение равное 150 нуклеотидов.

2.1.4. Программное средство Multiple-k

В статье [7] изложен подход к сборке транскриптов с помощью методов *Multiple-k* и *STM* (*Scaffolding using Translation Mapping method*).

У метода *Multiple-k* есть два алгоритма сборки контигов. В обоих используется сборка транскриптов при помощи сборщика *Velvet* с различными значениями *k* (длинами *k*-меров).

Первым алгоритмом является исключающий алгоритм (*subtractive Multiple-k*). Вначале собираются контиги при большом значении *k*. Гены с высоким уровнем экспрессии собираются наилучшим образом. Использованные чтения удаляются из набора, и сборка повторяется с меньшим значением *k*, что приводит к сборке генов с меньшим уровнем экспрессии. Процедура повторяется несколько раз. Полученные при разных сборках контиги объединяются, образуя финальную сборку.

Вторым алгоритмом является смешивающий алгоритм (*additive Multiple-k*). В данном подходе, чтения, использованные в предыдущих, этапах не удаляются. Таким образом, сборки для разных значений *k* используют весь набор чтений. При таком подходе одинаковые контиги могут образоваться в разных сборках, внося избыточность в финальный набор. Для устранения избыточности используется программное средство *CD-HIT-EST*. Набор контигов выравнивается сам на себя, и короткие избыточные контиги удаляются. Оставшиеся контиги образуют финальную сборку.

Для сборки скэффолдов применяется метод *STM* (Рис. 6). На вход подаются контиги и неиспользованные чтения, длина которых больше заданной величины. Эти данные выравниваются на протеом с помощью программного средства *BLASTX*. Если только один контиг сопоставился данному протеину, то он включается в итоговую сборку. Если таких контигов несколько, то они собираются в скэффолд. При перекрытии

контигов, и образовании неоднозначности в местах перекрытия, используется сборщик *CAP*. Если неоднозначность осталась, то перекрывающиеся контиги добавляются в финальную сборка, как отдельные контиги.

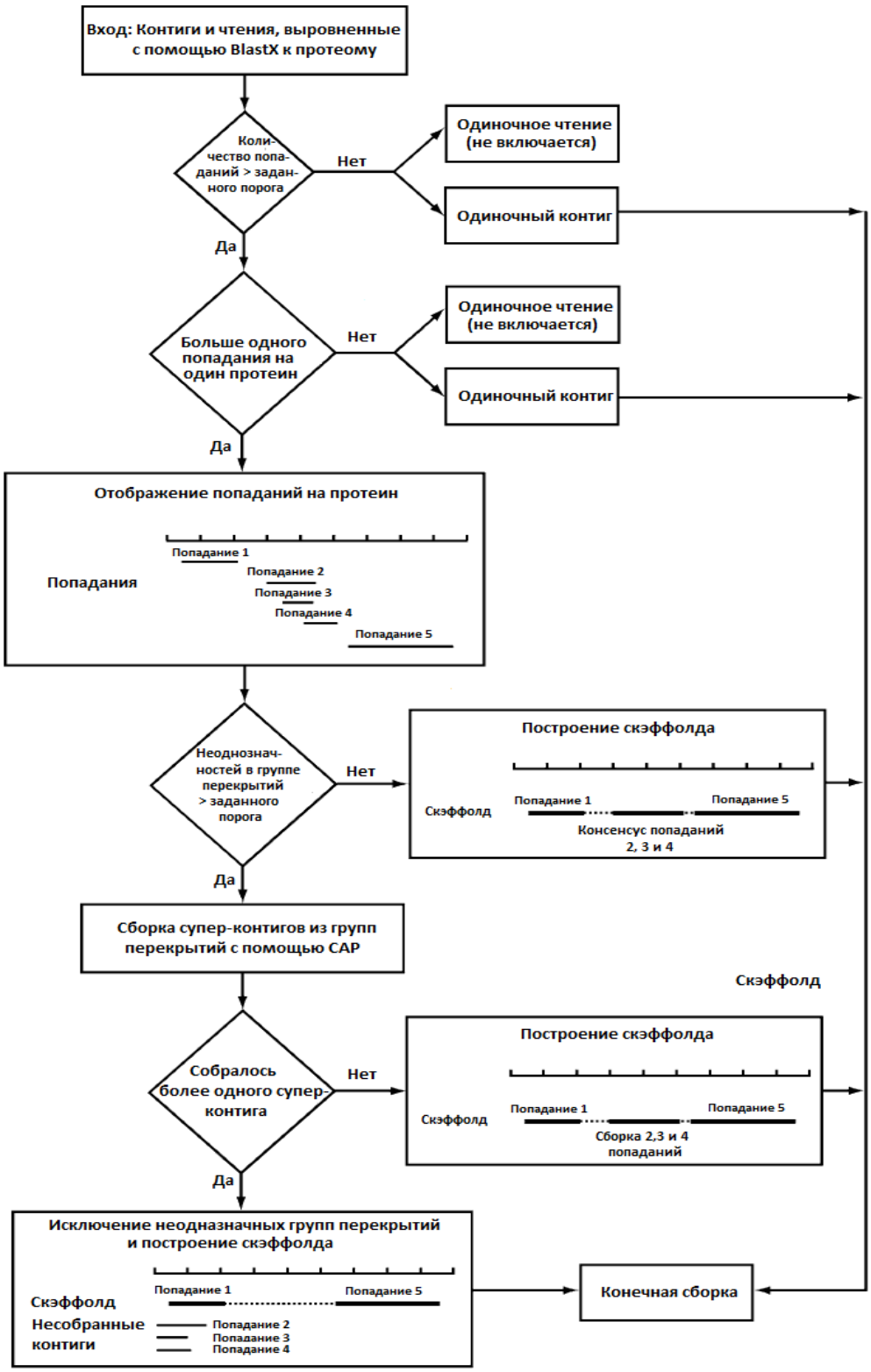

Рис. 6. – Схема работы метода *STM*

2.1.5. Программное средство Cufflinks.

В статье [8] изложен подход к сборке транскриптов в программном средстве *Cufflinks* (Рис. 7).

На вход алгоритм принимает фрагменты последовательностей кДНК, которые были выровнены на геном с помощью программы *TopHat*. Все фрагменты разбиваются на неперекрывающиеся локусы, которые впоследствии независимо обрабатываются с помощью алгоритма Cufflinks. Первым шагом в сборке фрагментов является определение пар несовместимых фрагментов, которые должны были возникнуть из различных изоформ мРНК, полученных в результате сплайсинга. Перекрывающиеся фрагменты являются совместимыми, если их перекрытие содержит полностью одинаковые интроны.

Неперекрывающиеся фрагменты являются совместимыми. Каждому фрагменту соответствует одна вершина в графе. Между фрагментами имеется ребро в графе перекрытий, если они совместимы и имеют пересечение в геноме при выравнивании. Ребро направлено от x к y, если начало фрагмента x имеет значение координаты меньше, чем значение координаты начала y. Затем граф транзитивно сокращается. Таким образом, на множестве фрагментов установлен частичный порядок.

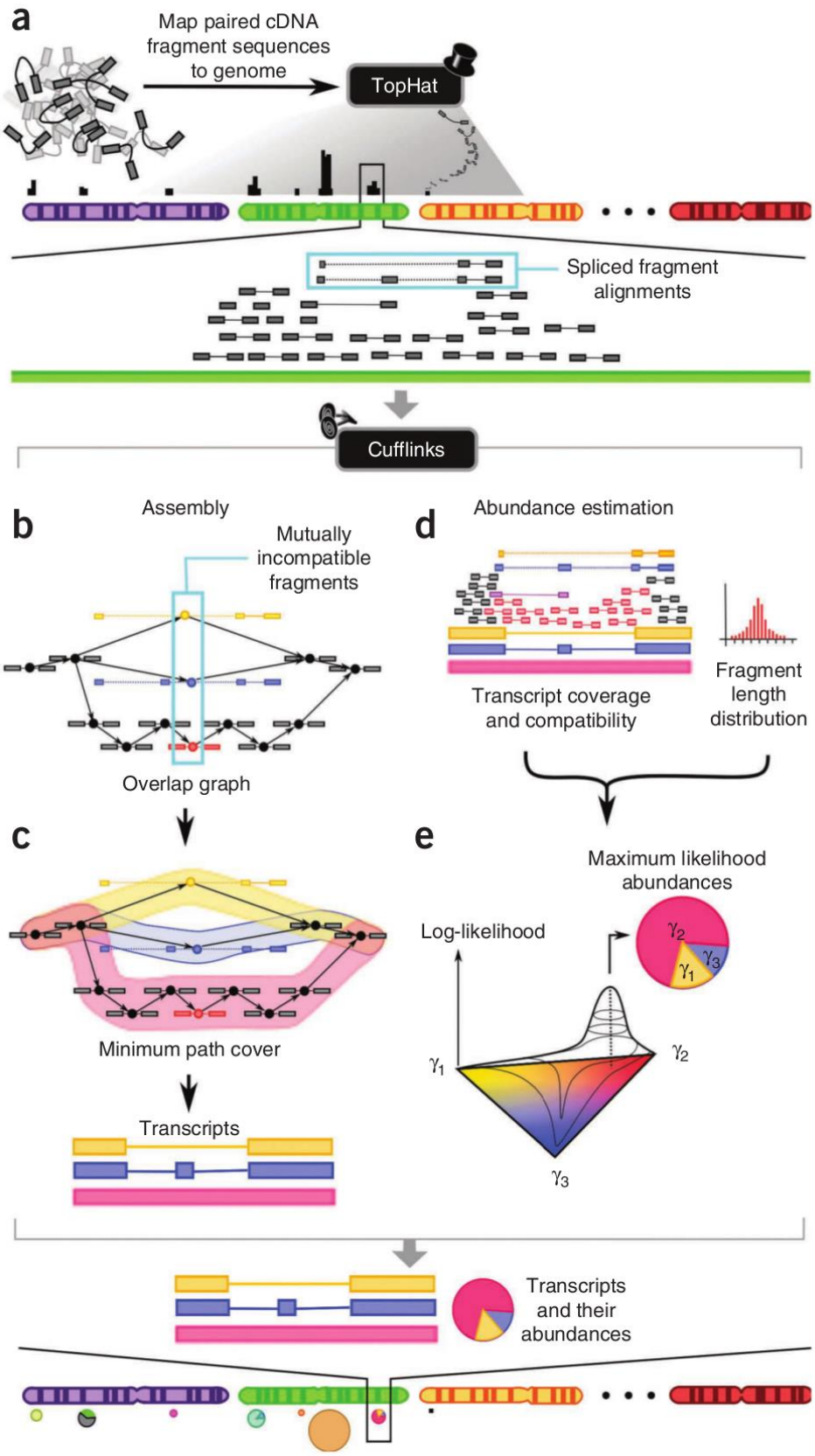

Рис. 7. – Схема работы программного средства *Cufflinks*

Затем алгоритм *Cufflinks* находит минимальное покрытие получившегося графа путями. То есть находит минимальное множество путей такое, что любая вершина принадлежит хотя бы одному пути. По

теореме Дилуорса такое множество путей может быть построено, если вначале найти максимальное множество фрагментов, каждый из которых несовместим с остальными из данного множества (то есть построить антицепь). Каждый фрагмент из антицепи может быть достроен до пути. Достроив каждый фрагмент антицепи до пути, в итоге получается искомое покрытие. Каждый найденный путь представляет из себя набор совместимых фрагментов, соседние из которых перекрываются, и является изоформой. Затем с помощью метода максимального правдоподобия оценивается распространенность каждого транскрипта, распределяя фрагменты по найденным транскриптам.

2.2. Сравнительная таблица

В таблице 1 приведено сравнение программных средств для сборки транскриптома по следующим параметрам:

1. *De novo* – является ли алгоритм сборки de novo.
2. Параллельность – поддерживает ли алгоритм многопоточные вычисления.
3. Парные чтения – поддерживает ли алгоритм работу с парными чтениями.
4. *Stranded reads* – поддерживает ли алгоритм работу с чтениями, для которых известна информация о принадлежности цепочке.

Таблица 1. Сравнительная таблица

Сборщик	*De novo*	Параллельность	Парные чтения	*Stranded reads*
Trinity	+	+	+	+
Oases	+	+	+	+
Trans-Abyss	+	+	+	-
Multiple-k	+	-	+	+
Cufflinks	-	+	+	+

Одним из наиболее популярных сборщиков транскриптома в настоящее время является сборщик *Trinity*. Данный сборщик был выбран для дальнейшего экспериментального сравнения с разрабатываемым методом.

2.3. Выводы по главе 2

1. Были выбраны и описаны наиболее известные сборщики транскриптома.
2. Предоставлена сравнительная таблица характеристик сборщиков.
3. Был выбран сборщик, для сравнения с разрабатываемым методом.

ГЛАВА 3. ОПИСАНИЕ ПРЕДЛАГАЕМОГО РЕШЕНИЯ

В настоящем разделе приведено описание алгоритма сборки транскриптома по парным чтениям.

Алгоритм сборки транскриптома состоит из трех этапов:

1. Сбор частот *k*-меров и отсев непригодных *k*-меров.
2. Поиск компонент связности графа де Брёйна.
3. Анализ каждой компоненты связности и выделение из нее транскриптов.

3.1. Основная идея метода

В предлагаемом методе используется граф де Брёйна, построенный на основе полученных из чтений *k*-меров. При использовании транскриптомных чтений для построения граф представляет собой большое количество компонент связности. В каждой такой компоненте можно выделить пути, отвечающие транскриптам. Транскрипты, полученные из одной компоненты связности и имеющие общие подпоследовательности с некоторой вероятностью являются изоформами, возникшими в результате альтернативного сплайсинга.

Покрытие каждого отдельного транскрипта чтениями различно, но в рамках одного транскрипта покрытие равномерно. В связи с этим при анализе компонент связности и поиски в них путей учитывалось частота встречи *k*-мера в начальных данных.

3.2. Алгоритм подготовки *k*-меров

На вход алгоритмы подаются парные чтения. Каждое парное чтение обрезается с концов, так как на концах чтений высока вероятность ошибки. Из обрезанных чтений собирается статистика частот встречающихся в них *k*-меров. Считается, что *k*-меры с частотой меньше заданной содержат ошибку, поэтому данные *k*-меры отбрасываются. Оставшиеся после отсева *k*-меры с их частотами передаются следующему алгоритму.

3.3. Алгоритм поиска компонент связности

На вход алгоритму передаются *k*-меры с их частотами. Алгоритм состоит из двух этапов.

В первом этапе происходит поиск простых компонент связности. Для каждого *k*-мера мы храним информацию, был ли он уже использован. Изначально все *k*-меры не использованы. На очередном шаге этапа берется неиспользованный *k*-мер и происходит построение новой компоненты связности, в которой присутствует данный *k*-мер. В новую компоненту связности добавляются *k*-меры, имеющие хотя бы одного соседа из *k*-меров в текущей компоненте связности в графе де Брёйна. Построение компоненты считается оконченным, если невозможно добавить новые *k*-меры в компоненту связности. Все *k*-меры из новой компоненты помечаются использованными. Построение новых компонент происходит до тех пор, пока все *k*-меры не окажутся использованными.

В результате работы первого этапа алгоритма образуется множество небольших компонент пригодных для дальнейшего анализа и одна или небольшое количество больших компонент (Рис. 8). Наличие больших компонент связности объясняется существованием цепочек нуклеотидов длины больше *k*, встречающихся в большом количестве различных транскриптов. За разбиение данных больших компонент на меньшие по размеру и пригодных для анализа, отвечает второй этап алгоритма поиска компонент связности.

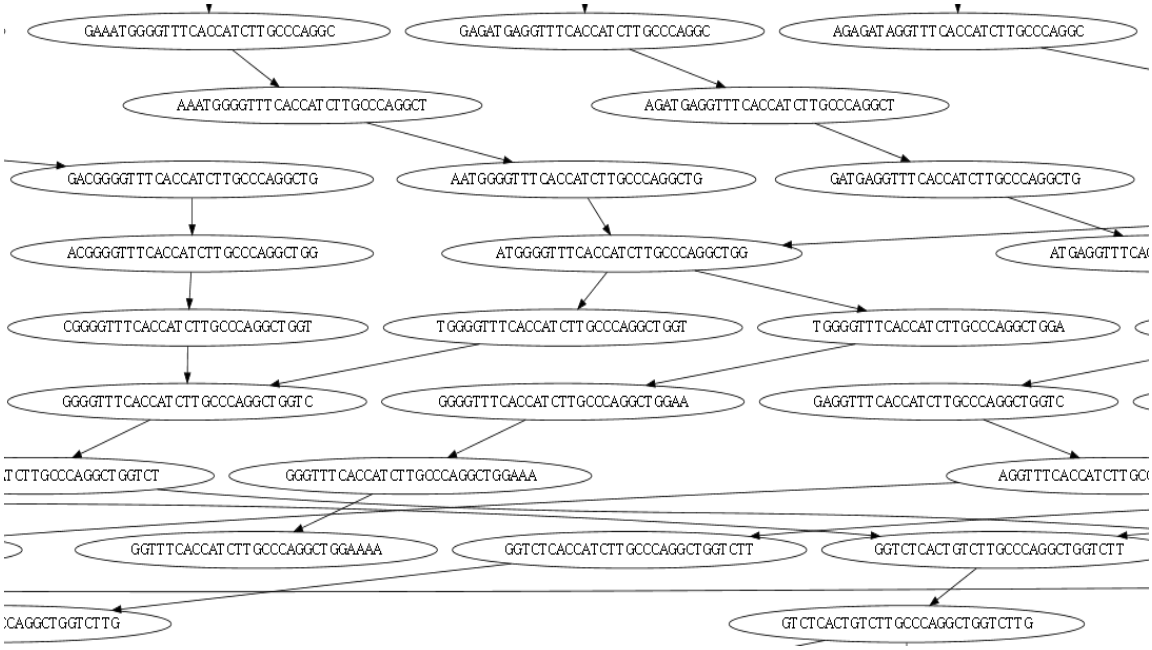

Рис. 8. – Часть большой компоненты связности

На вход второму этапу алгоритма подаются получившиеся в результате работы первого этапа большие компоненты. В каждой такой

компоненте рассматривается каждая вершина. Для вершины рассматриваются все ее ребра и складываются их частоты. Если частота ребра вершины меньше, чем заданное значение, то данное ребро удаляется. После проведения данной операции для всех вершин компоненты, получится новый граф с меньшим числом ребер. Этот граф является несвязным. В нем происходит поиск компонент связности. Образовавшиеся небольшие компоненты связности считаются готовыми к дальнейшему анализу. Если в результате откидывания ребер остались большие компоненты связности, то для них запускается описанный алгоритм еще раз, с менее жесткими условиями на откидывание ребер. Так происходит, пока не останутся только небольшие компоненты связности.

После работы двух выше описанных этапов в результате получаются два набора компонент: набор небольших компонент из исходного графа де Брёйна и набор небольших компонент, возникшие в результате деления большой компоненты связности. Данные наборы и *k*-меры с их частотами передаются следующему алгоритму анализа компонент.

3.4. Алгоритм анализа компонент

На вход алгоритму подаются компоненты, представленные, как наборы *k*-меров, и информация по частотам *k*-меров. Алгоритм анализирует каждую компоненту независимо. В связи с этим, алгоритм распараллелен, что позволяет быстрее обработать все компоненты на многоядерных машинах.

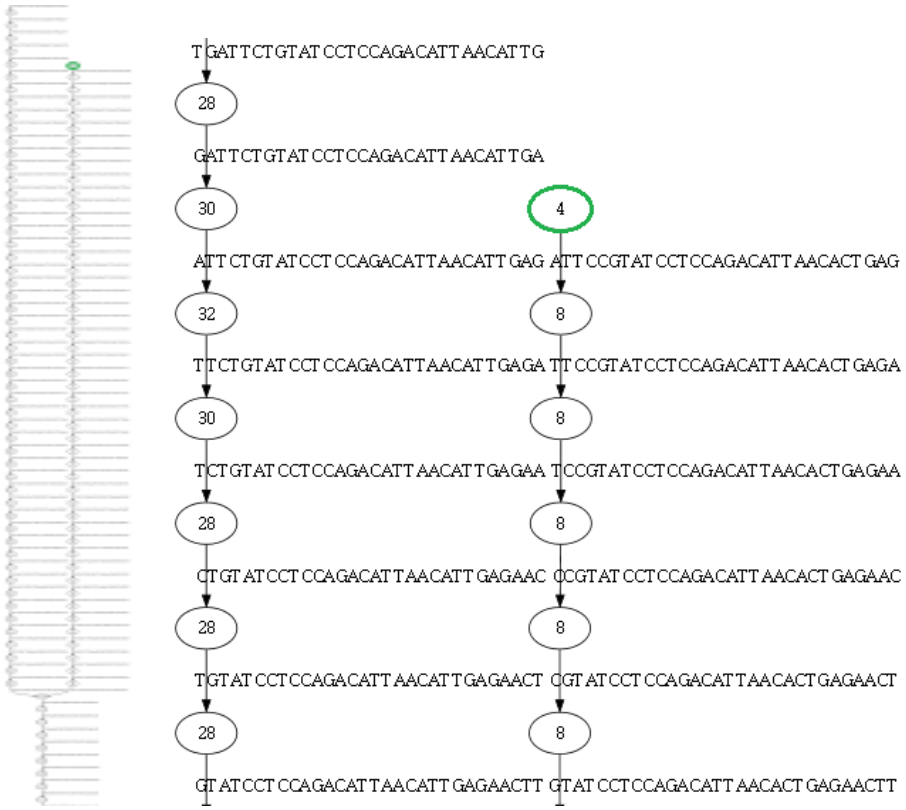

Рис. 9. – Допустимая стартовая вершина

Анализ каждой компоненты начинается с поиска стартовых вершин, из которых будет начинаться алгоритм поиска путей в графе. Стартовая вершина должна удовлетворять двум требованиям. Она должна иметь степень равную единице, и длина пути до ближайшей вершины степени больше двух должна быть больше заданной (таким образом стартовая вершина должна быть концом цепочки-отростка в графе). Для поиска стартовых вершин рассматриваются все вершины степень один, и из каждой ищется путь до ближайшей вершины степени больше двух. Если такой путь не найден или его длина больше заданной, то вершина объявляется стартовой (Рис 9). В противном случае все вершины пути, кроме вершины степени больше двойки удаляются, так как данный отросток графа имеет длину меньше заданной и считается ошибочным (Рис 10).

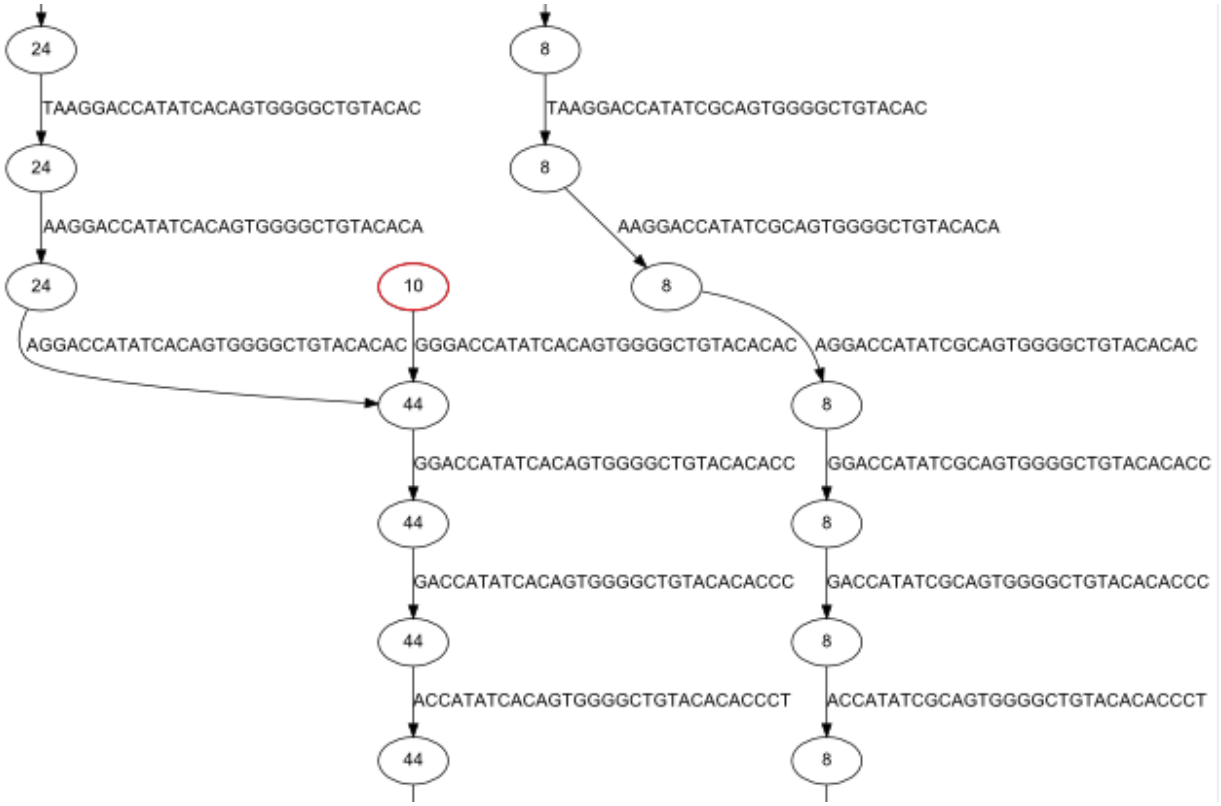

Рис. 10. – Недопустимая стартовая вершина

Далее в каждой компоненте происходит поиск всех возможных путей между стартовыми вершинами. Для этого запускается алгоритм поиска в глубину для каждой стартовой вершины. При поиске путей используется информация о частоте *k*-меров для анализа разветвления пути, так как зачастую в компонентах содержатся неоднозначности, возникшие из-за ошибок секвенирования или ввиду свойств транскриптомных данных (Рис. 11-12).

Рис. 11. – Возможный альтернативный сплайсинг

В пути не могут присутствовать две вершины подряд с отличием частот соответствующих *k*-меров более чем в четыре раза, так как ребро между ними с высокой вероятностью соответствует не альтернативному сплайсингу (Рис. 11), а ошибке в чтениях (Рис 12). В результате поиска образуется набор путей, считающийся набором транскриптов, полученных из данной компоненты.

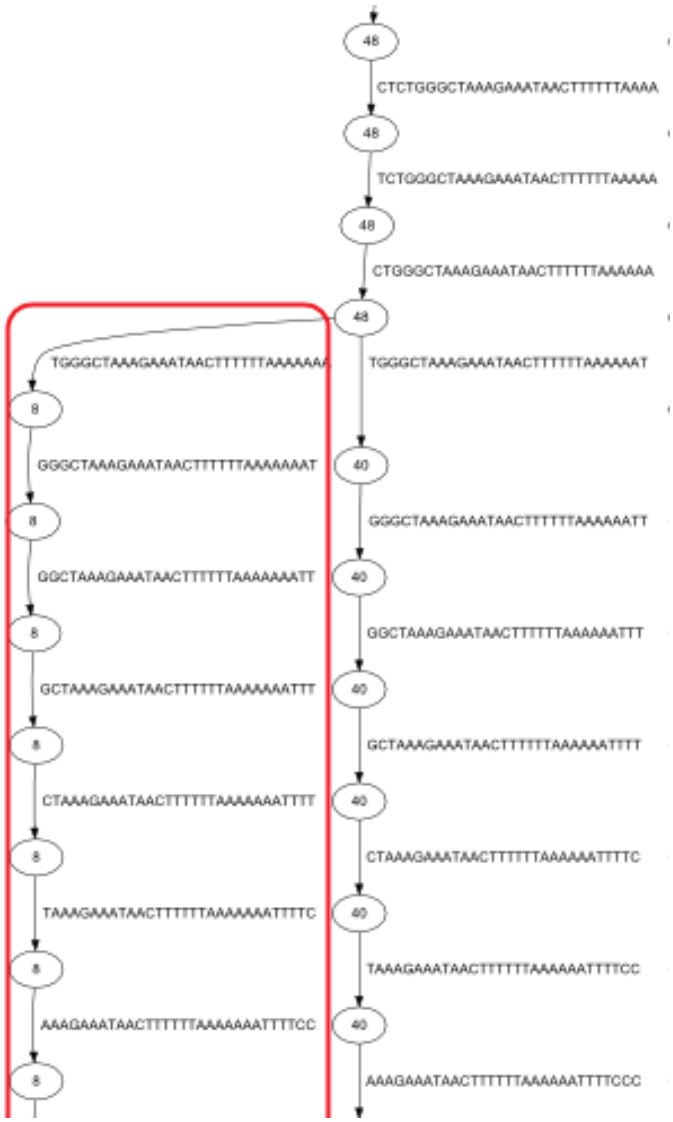

Рис. 12. – Ошибочная ветка

После анализа всех компонент в результате получается набор транскриптов, являющийся собранным транскриптомом. Транскрипты для каждой компоненты хранятся, как по компонентно для возможного исследования альтернативного сплайсинга, так и вместе для получения полного транскриптома.

Для каждой компоненты известно, получена она из начального графа де Брёйна или из разделенной большой компоненты. Эксперименты показали, что количество информации в разных типах компонент не сильно отличается.

3.5. Выводы по главе 3

1. Разработан алгоритм сборки транскриптома.
2. Описана основная идея разработанного метода.
3. Описан каждый этап метода.

ГЛАВА 4. РЕАЛИЗАЦИЯ АЛГОРИТМА И РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

В данной главе кратко описаны детали реализации описанного метода, а так же результаты экспериментов.

4.1. Реализация алгоритма

Описанный в работе метод был реализован на языке программирования *Java*.

При разработке метода были использованы классы, написанные в рамках проекта *ITMO Genome Assembler*.

Реализованный метод был встроен в основной проект, как дополнительный модуль.

4.2. Экспериментальное сравнение

Описанный метод был сравнен с одним из наиболее популярных de novo сборщиков Trinity. Сравнение производилось на реальных транскриптомных данных организма домовая мышь [9].

Для сравнения были выбраны следующие параметры: число контигов (количество восстановленных последовательностей), суммарная длина контигов, максимальная длина контига, минимальная длина контига, средняя длина контига, время работы алгоритма.

Данные содержали около 32 миллионов пар чтений, длиной около 100 нуклеотидов (16 ГБ). Данные были получены с помощью *Illumina Genome Analyzer II* [10]. Сравнение производилось на вычислительном узле НИУ ИТМО, имеющий следующие характеристики: 24-ядерный процессор *AMD Opteron™ Processor* 6234, 128 ГБ оперативной памяти. Результаты тестирования представлены в таблице 2.

Таблица 2. – Результаты сравнения эффективности сборки сборщика транскриптома, разработанного в настоящей НИР, и Trinity

	Разработанный сборщик	*Trinity*
Число контигов	223432	478826
Суммарная длина контигов	191279463	555989630
Максимальная длина контига	23701	24756
Минимальная длина контига	200	201
Средняя длина контига	856	1161
Время работы	10 часов	41 час 17 минут

Как видно из результатов, предложенный сборщик работает в несколько раз быстрее, что и было основным требованием в поставленной задаче. Длина самого длинного транскрипта не сильно отличается в обоих случаях, что свидетельствует о схожести результатов. Как и ожидалось, предложенный метод восстанавливает меньшее число транскриптов. Большая часть транскриптов, собранных предложенным методом, содержится в сборке *Trinity*, что указывает на качество сборки.

Качество и скорость сборки возможно варьировать изменяя длину *k*-меров *k*, а так же параметры по отсечение ветвей и выкидыванию ребер.

4.3. Дополнительные эксперименты

В рамках гранта [11] были проведены эксперименты на других входных данных. Результаты данных экспериментов представлены в таблицах 3-10.

Таблица 3. – Информация о данных организма *booting panicle*

Источник данных	Sequence Read Archive, библиотека SRX017632 http://www.ncbi.nlm.nih.gov/sra/SRX017632
Размер чтений	5 ГБ
Характеристики чтений	9,8 миллиона пар чтений, каждой из которых имеет длину 75 нуклеотидов

Таблица 4. – Сравнение алгоритмов на данных организма *booting panicle*

	Разработанный сборщик	*Trinity*
Время работы	1 час 36 минут	5 часов 46 минут
Число контигов	53827	53981
Суммарная длина	28023656	35668762
N50	721	989
N90	232	276

Таблица 5. – Информация о данных организма *Mus musculus*

Источник данных	Sequence Read Archive, библиотека SRX062280 http://www.ncbi.nlm.nih.gov/sra/SRX062280
Размер чтений	28 ГБ
Характеристики чтений	52,6 миллиона пар чтений, каждой из которых имеет длину 76 нуклеотидов

Таблица 6. – Сравнение алгоритмов на данных организма *Mus musculus*

	Разработанный сборщик	*Trinity*
Время работы	2 час 51 минута	24 часа 49 минут
Число контигов	63110	72325
Суммарная длина	63140855	87152380
N50	2226	2771
N90	370	391

Таблица 7. – Информация о данных организма *Lactuca serriola*

Источник данных	Sequence Read Archive, библиотека SRX098217 http://www.ncbi.nlm.nih.gov/sra/SRX098217
Размер чтений	28 ГБ
Характеристики чтений	46,7 миллиона пар чтений, каждой из которых имеет длину 85 нуклеотидов

Таблица 8. – Сравнение алгоритмов на данных организма *Lactuca serriola*

	Разработанный сборщик	*Trinity*
Время работы	4 часа 35 минут	10 часов 57 минут
Число контигов	184548	145875
Суммарная длина	138735317	147996784
N50	1177	1624
N90	340	423

Таблица 9. – Информация о данных организма *booting panicle*

Источник данных	Sequence Read Archive, библиотека SRX017631 http://www.ncbi.nlm.nih.gov/sra/SRX017631
Размер чтений	5 ГБ
Характеристики чтений	9,8 миллиона пар чтений, каждой из которых имеет длину 75 нуклеотидов

Таблица 10. – Сравнение алгоритмов на данных организма *booting panicle*

	Разработанный сборщик	*Trinity*
Время работы	1 час 38 минут	4 часа 29 минут
Число контигов	57990	54112
Суммарная длина	30873289	35379555
N50	744	968
N90	238	275

Как видно из результатов, предложенный метод работает быстрее сборщика *Trinity* на всех тестовых данных, иногда даже в восемь раз (Таблица 6). При этом восстанавливается значительная часть данных.

Таким образом, реализованный метод быстро работает на больших объемах данных и выдает результат, пригодный для предварительного анализа транскриптома организма.

4.4. Выводы по главе 4

Из представленных результатов экспериментальных исследований можно сделать следующие выводы:

1. Реализованный алгоритм сборки транскриптома работает быстрее наиболее популярного сборщика *Trinity*.
2. Качество полученной предложенным методом сборки ниже, чем качество сборки *Trinity*. При этом восстановлены, как длинные, так и короткие последовательности, а длина наибольших контигов не сильно отличается.

Таким образом, предложенный метод сборки транскриптома работает быстрее и предоставляет сборку, пригодную для предварительного анализа.

ЗАКЛЮЧЕНИЕ

1. В работе были рассмотрены существующие сборщики транскриптома.
2. Был разработан метод быстрой сборки транскриптома, использующий небольшое количество памяти и имеющий возможность работать параллельно.
3. Предложенный метод был реализован на языке программирования *Java* с дополнительными скриптами на *Bash* для удобства запуска.
4. Метод был экспериментально сравнен с наиболее популярным на данный момент сборщиком *Trinity* на различных реальных транскриптомных данных. Результаты экспериментов показали, что предложенный метод работает быстрее, а полученная сборка пригодна для предварительного анализа транскриптома.

   Таким образом, поставленные в работе цели достигнуты.

СПИСОК ИСТОЧНИКОВ

1. Ryan D. Morin, Matthew Bainbridge, Anthony Fejes, Martin Hirst, Martin Krzywinski, Trevor J. Pugh, Helen McDonald, Richard Varhol, Steven J.M. Jones, and Marco A. Marra. Profiling the HeLa S3 transcriptome using randomly primed cDNA and massively parallel short-read sequencing // BioTechniques 2008 45 (1): Pp. 81–94.
2. Patrick Schmid DNA Microarrays CSE 497 Spring 2004.
3. N. G. de Bruijn A combinatorial problem // Koninklijke Nederlandse Akademie v. Wetenschappen. 1946. Vol. 49. Pp. 758–764.
4. Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum1, Kerstin Lindblad-Toh, Nir Friedman and Aviv Regev Full-length transcriptome assembly from RNA-Seq data without a reference genome volume // Nature Biotechnology 7 july 2011 29 number Pp. 644-654.
5. M. H. Schulz, D. R. Zerbino, M. Vingron, E. Birney Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. // Bioinformatics. 2012. Vol 28. No 8. Pp. 1086–1092.
6. G. Robertson, J. Schein, R. Chiu, et al. De novo assembly and analysis of RNA-seq data. // Nature methods. 2010. Vol 7. No 11. Pp. 909–915.
7. Y. Surget-Groba, J. Montoya-Burgos Optimization of de novo transcriptome assembly from next-generation sequencing data. // Genome Res. Oct 2010. Vol. 20, No. 10. Pp. 1432–1440.
8. C. Trapnell, B. A. Williams, G. Pertea, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. // Nature Biotechnology. May 2010. Vol. 28. No. 5. Pp. 511–518.
9. The Sequence Read Archive (SRA) http://sra.dnanexus.com/runs/SRR203276.
10. Illumina, Inc. http://www.illumina.com/systems/genome_analyzer_iix.ilmn.

11. Разработка методов сборки генома, сборки транскриптома и динамического анализа протеома (Соглашение 14.B37.21.0562 от 06.08.2012) http://www.fcpk.ru/catalog.aspx?CatalogId=5300.