Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
Кафедра «Компьютерные технологии»

С. В. Казаков

Разработка алгоритма упрощения графа перекрытий при сборке геномных последовательностей

Магистерская работа

Научный руководитель: д.т.н., проф. А. А. Шалыто

Санкт-Петербург
2013

Оглавление

ВЕДЕНИЕ

В 1958 году Ф. Криком было сформулировано правило, названное впоследствии «Центральной догмой молекулярной биологии». Она постулирует правило передачи информации от ДНК через РНК к белку. В дальнейшем были открыты и иные переходы информации (обратная транскрипция, репликация). Таким образом, было обосновано три принципиально важных уровня хранения и реализации биологической информации: геном, транскиптом и протеом.

Геном представляет собой полный набор генов и иного, некодирующего наследственного материала организма, содержащегося в гаплоидном наборе хромосом и внеядерных элементах. Иными словами, вся генетическая информация организма хранится в геноме. Там же происходят ее изменения.

Многие современные задачи биологии и медицины требуют знания генома живых организмов, который состоит из нескольких нуклеотидных последовательностей ДНК.

При изучении генома живого существа обычно выделяют три основных этапа:

- секвенирование молекул ДНК, содержащих информацию о геноме (выполняется с использованием специальных устройств-секвенаторов);
- сборка геномной последовательности (или коротко – сборка генома, выполняется с использованием компьютеров);
- анализ и сравнение геномов (выполняется с использованием компьютеров).

Изучение генома человека и других живых существ имеет важное прикладное значение. На основании результатов сборки генома конкретного человека возможна реализация персонифицированной медицины – определения предрасположенности человека к различным болезням, создание индивидуальных лекарств и т. д. Кроме этого, на основе результатов

исследования геномов растений и животных с использованием методов биоинженерии могут быть выведены новые их виды, обладающие определенными свойствами.

Еще одно важное приложение исследования ДНК — генетические заболевания. У особей, зараженных одним генетическим заболеванием, наблюдаются одинаковые изменения в ДНК, что может быть использовано в медицине как для теоретического исследования заболевания, так и для лечения от него.

Задача разработки методов сборки геномных последовательностей является, в определенном смысле, центральной среди всех задач биоинформатики. Это объясняется тем, что без ее решения нельзя приступить к детальному изучению генома живого существа и его анализу с применением других алгоритмов биоинформатики.

При этом при решении этой задачи возникает большое число трудностей.

Цель работы – улучшить существующие решения одной из них – проблемы запутанности графа перекрытий при сборке контигов.

Работа состоит из введения, четырех глав и заключения.

В первой главе приведен обзор предметной области – рассмотрены необходимые для понимания тематики основы биоинформатики, объяснена актуальность задачи, описаны основные моменты, необходимые для понимания процесса сборки генома. Также в первой главе дается набор терминов, используемых в работе, и их определений.

Во второй главе описана проблема запутанности графа перекрытий, возникающая при сборке генома. Также описаны существующие подходы её решения.

В третьей главе описан предлагаемый метод.

Четвертая глава содержит экспериментальные исследования, приведены результаты работы.

ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

В этой главе рассмотрены необходимые для понимания тематики основы биоинформатики, объяснена актуальность задачи, описаны основные моменты, необходимые для понимания процесса сборки генома. Также в первой главе дается набор терминов, используемых в работе, и их определений.

1.1. Биоинформатика

Современные задачи, возникающие в биологии и медицине, требуют работы с большим объемом данных. Поэтому применение вычислительных устройств и алгоритмов находит все более широкое применение в этих областях науки.

1.1.1. ДНК

ДНК (дезоксирибонуклеиновая кислота) — химическое вещество, биологический полимер, обеспечивающий хранение и передачу из поколения в поколение генетической информации. В клетках эукариотов (например, животных и растений) ДНК находится в ядре каждой клетки организма.

ДНК состоит из двух цепочек (последовательностей нуклеотидов). Каждый нуклеотид состоит из азотистого основания, сахара и фосфатной группы.

В ДНК встречается четыре типа азотистых оснований — аденин, цитозин, гуанин и тимин, которые обозначаются соответственно буквами A, C, G и T. Азотистые основания одной цепочки ДНК соединены с азотистыми основаниями другой водородными связями согласно принципу комплементарности — аденин соединяется только с тимином, а гуанин — с цитозином. Таким образом, одна из двух цепочек ДНК однозначно получается из другой путем разворачивания и замены каждого нуклеотида на комплементарный.

Информация, хранящаяся в ДНК организмов, очень важна, так как отражает важные свойства живых организмов — наследственность и

изменчивость. Наследственная информация расположена не в одной молекуле ДНК, а, вообще говоря, в нескольких, обычно расположенных в хромосомах. Кроме того, у некоторых организмов, к которым, например, относится большинство млекопитающих, набор хромосом удвоен, такие организмы называются диплоидными. В каждой паре находится по одной хромосоме от каждого родителя, причем эти хромосомы, за исключением пары половых хромосом, отличаются в относительно небольшом числе нуклеотидов. Также существуют организмы с утроенным, учетверенным и т.д. наборами хромосом.

1.1.2. Задача секвенирования

Секвенирование ДНК — определение нуклеотидной последовательности по имеющемуся образцу ДНК. В результате этого процесса получается линейная цепочка, отражающая последовательность нуклеотидов в ДНК.

Существующие технологии не позволяют прочитать всю цепочку ДНК (как и слишком большие её части). Для секвенирования применяются другие технологии.

Современными методами секвенирования ДНК являются методы нового поколения (next-generation sequencing)[1]. При секвенировании этими методами ДНК случайным образом дробится на мелкие участки, длиной до 500 нуклеотидов, каждый из которых затем считывается.

Для того чтобы было возможным восстановить исходную последовательность ДНК, обеспечивается многократное покрытие генома чтениями. Пусть имеется набор из N чтений длины l, и пусть геном имеет длину L. Тогда говорят, что данный набор обеспечивает покрытие генома, равное $\frac{N \cdot l}{L}$.

Для еще большего удешевления процесса считывания используются так называемые парные чтения. При прочтении парных чтений секвенатором

выделяется расположенный в случайном месте последовательности ДНК фрагмент, из которого затем считываются префикс и суффикс.

Результатом работы секвенатора в случае использования парных чтений являются пары последовательностей, про которые известно, на каком расстоянии они располагались в исходной последовательности ДНК.

Современные технологии (такие как, например, Illumina GAIIx [2]), позволяют получить длину фрагмента около 500 нуклеотидов, а длину считанных префиксов и суффиксов до 150 нуклеотидов.

Полученные данные собираются в единую последовательность при помощи специального программного обеспечения – сборщика генома. Из-за наличия повторов в геномной последовательности однозначно восстановить саму последовательность практически невозможно, поэтому требуется восстановить как можно более длинные непрерывные фрагменты геномной последовательности.

1.2. Сборка генома

Сборка геномной последовательности (или коротко – сборка генома) – процесс получения больших фрагментов генома, выполняется с использованием компьютеров.

Задача разработки методов сборки геномных последовательностей является, в определенном смысле, центральной среди всех задач биоинформатики. Это объясняется тем, что без ее решения нельзя приступить к детальному изучению генома живого существа и его анализу с применением других алгоритмов биоинформатики.

Обычно решением этой задачи является набор контигов, объединенных в скэффолды. Контигом называется непрерывная последовательность нуклеотидов. Скэффолдом называется последовательность контигов с оценкой

на расстояния между ними. При этом предполагается, что такие (или близкие к ним) контиги и скэффолды действительно присутствуют в геноме.

В рамках данной работы будет рассматриваться так называемая задача *de novo сборки генома* – сборки генома живого существа, для которого геном еще не известен.

Сложность задачи сборки геномной последовательности обусловлена следующими факторами:

- большой объем входных данных;
- сложность структуры генома – наличие в нем повторов и полиморфизмов;
- наличие ошибок в исходных данных, полученных с устройств-секвенаторов.

Для того чтобы уменьшить влияние некоторых факторов, вызывающих ошибки, геномную последовательность покрывают чтениями несколько десятков раз. При этом покрытие геномной последовательности чтениям оказывается достаточно равномерным – все позиции, независимо от их расположения в последовательности, покрыты чтениями примерно одинаковое число раз.

Для решения указанных проблем сборку геномной последовательности обычно разбивают на следующие этапы:

- исправление ошибок в исходных данных;
- восстановление фрагментов генома по парным чтениям, такие фрагменты называются квазиконтигами;
- сборка контигов (расширенных фрагментов) из квазиконтигов;
- построение скэффолдов – упорядоченных контигов с оценкой на расстояния между ними.

Далее подробнее рассмотрим архитектуру и этапы сборки одного из сборщиков генома – сборщика *ITMO Genome Assember* [3-5]. Данный сборщик

хорошо зарекомендовал себя в рамках проектов *de novo Genome Assembly Project* [6] и *Assemblathon 2* [7].

На основе этого сборщика будут тестироваться предлагаемые алгоритмы.

Архитектура сборщика *ITMO Genome Assember* показана на рис. 1.

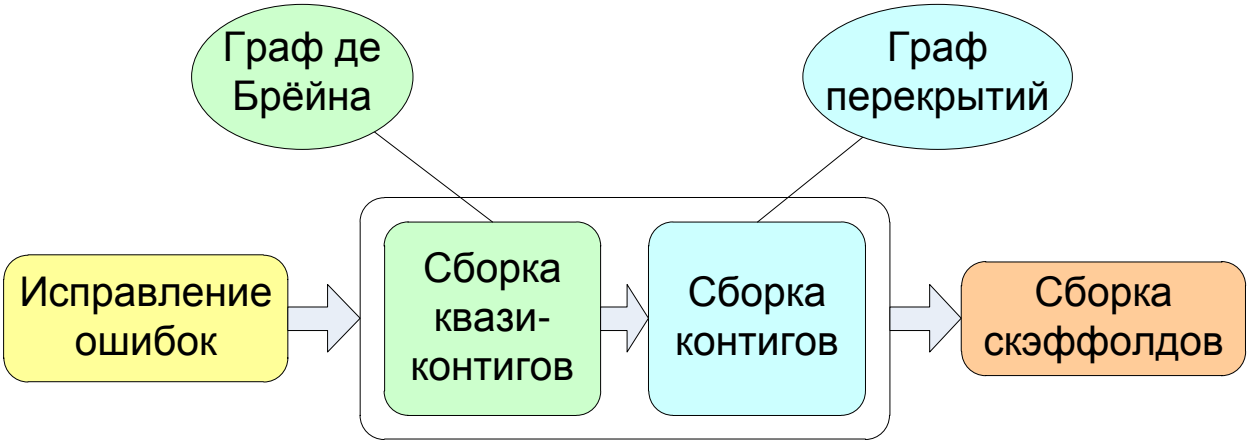


Рис. 1. Архитектура сборщика

1.2.1. Исправление ошибок

Приведем описание алгоритма исправления ошибок в данных секвенирования.

Алгоритм основан на частотном анализе содержащихся в чтениях k-меров (подстрок длины k) и не использует граф де Брёйна.

Для эффективного исправления ошибок необходимо, чтобы каждая позиция генома была прочитана несколько раз, так как это единственный способ отличить правильно прочитанный нуклеотид от прочитанного неверно. Это, ввиду невысокой вероятности ошибки, дает основания полагать, что на каждой позиции нуклеотид, считанный наибольшее число раз, является верным. На практике используются наборы чтений, покрывающие геном несколько десятков раз.

Важно отметить, что не только отдельные позиции всего генома были прочитаны несколько десятков раз, но и небольшие его подстроки (не длиннее

самих чтений) встречаются в чтениях несколько раз, причем, чем длиннее подстрока, тем меньше шансов, что несколько различных чтений ее содержат. Последнее соображение вытекает не только из увеличения вероятности попадания чтения на конкретную подстроку при увеличении ее длины, но и из факта наличия в чтениях ошибок и маленькой вероятности того, что одна и та же ошибка была допущена в одной и той же позиции генома более одного раза.

Идея алгоритма состоит в следующем. Для каждого *k*-мера, присутствующего в чтениях (прямых или обратно-комплементарных) хотя бы раз, подсчитаем, сколько раз он встречается в чтениях. На основании этой статистики все *k*-меры можно разделить на две группы – «надежные» *k*-меры и «подозрительные». «Надежные» *k*-меры – это те, которые встречаются в чтениях достаточно большое число раз – не меньше значения некоторого порога *t*, которое является параметром алгоритма. С «надежными» *k*-мерами делать ничего не требуется, предполагается, что в них ошибок нет. «Подозрительные» же *k*-меры вызваны либо плохим покрытием тех частей генома, откуда они были прочитаны, либо, что гораздо более вероятно, наличием в них одной или нескольких ошибок, которые необходимо исправить.

После выделения «подозрительных» *k*-меров для каждого из них необходимо решить, в какой именно позиции могла быть совершена ошибка. Для этого предлагается перебрать все позиции *k*-мера и все возможные нуклеотиды, попробовать заменить имеющийся нуклеотид на перебираемый и проанализировать получившийся *k*-мер. Если новый *k*-мер попадает в группу «надежных», значит, вероятно, рассматриваемый *k*-мер является результатом его ошибочного прочтения. Если в течение перебора был найден только один «надежный» *k*-мер, получающийся из «подозрительного» путем замены одного нуклеотида на другой, полагается, что данный «подозрительный» *k*-мер исправлен, а соответствующее исправление запоминается. Если таких *k*-меров несколько, неясно, какое из исправлений запоминать, поэтому в таких случаях «подозрительные» *k*-меры не исправляются. И наконец, если не было найдено ни

одного способа исправить «подозрительный» *k*-мер, полагается, что в нем было совершено больше одной ошибки, после чего запускается аналогичная процедура, но с исправлением не одного, а сразу двух нуклеотидов. Аналогичным образом можно пытаться изменить не только пары, но и тройки, а также кортежи из большего числа нуклеотидов, однако данное обобщение ощутимо сказывается на быстродействии алгоритма.

1.2.2. Сборка квазиконтигов

Приведем описание алгоритма сборки квазиконтигов.

Для начала дадим определение графа де Брёйна. *Граф де Брёйна* – граф, вершинами которого являются *k*-меры (строки длины *k*), при этом из одной вершины есть ребро в другую, если существует такой (*k*+1)-мер, что *k*-мер, соответствующий первой вершине, является его префиксом, а *k*-мер, соответствующий второй, – суффиксом.

В методе сборки квазиконтигов используется граф де Брёйна, в котором множество ребер состоит только из «надежных» (*k*+1)-меров – тех, которые встречаются в чтениях достаточно большое число раз, не меньшее некоторого порогового значения, чтобы их можно было с очень большой вероятностью считать входящими в геном. Множество вершин состоит из тех вершин графа де Брёйна, которым инцидентно хотя бы одно из выбранных ребер. Если участок нуклеотидной последовательности покрылся достаточно хорошо, то есть все входящие в него (*k*+1)-меры по много раз входят в исходные данные, то в этом подграфе существует путь между первым и последним *k*-мерами участка.

Метод сборки квазиконтигов основан на поиске такого пути для фрагмента, соответствующего парным чтениям. Из всех путей нас интересуют только те, которые укладываются в априорные границы длин фрагментов, поэтому слишком короткие и слишком длинные пути можно отбросить. Из оставшихся путей следует выбрать те, из которых действительно могли получиться имеющиеся парные чтения. Такие пути – хорошие кандидаты на

роль пути, соответствующего фрагменту в действительности. Если нашелся единственный такой путь, то можно с очень большой уверенностью сказать, что он соответствует реальной подстроке геномной последовательности, поэтому этот фрагмент считается восстановленным, а найденный путь выводится.

1.2.3. Сборка контигов

Сборка контигов из квазиконтигов основана на подходе *overlap-layout-consensus* [8] и состоит из нескольких этапов:

- поиск перекрытий;
- удаление транзитивных перекрытий;
- построение графа перекрытий и его упрощение;
- вывод первого приближения контигов;
- микросборка.

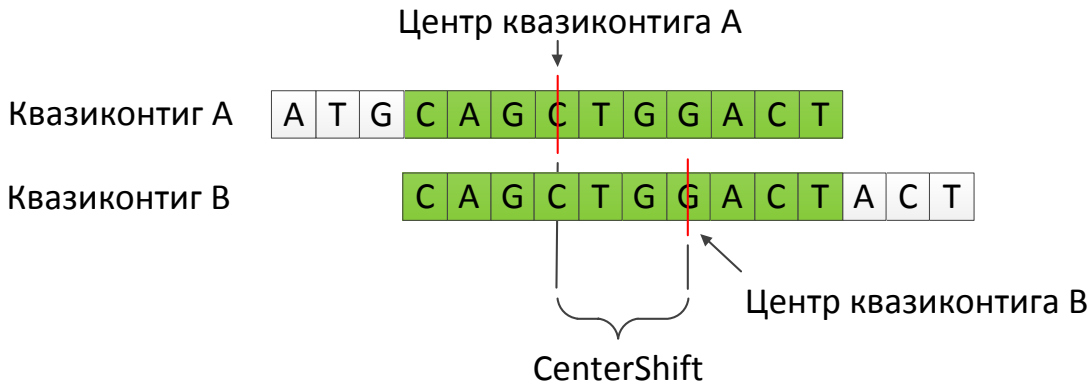
Отметим, что перед сборкой контигов для каждого квазиконтига добавляется его обратно-комплементарная копия, что позволяет упростить алгоритм.

Далее подробнее рассмотрим каждый этап.

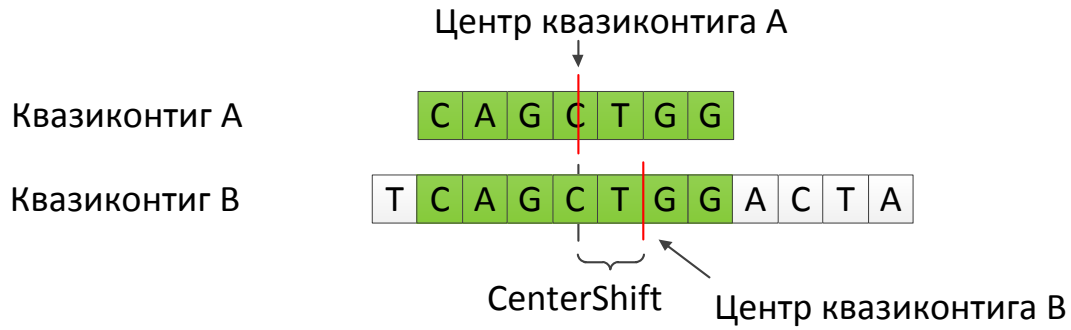
1.2.3.1. Поиск перекрытий

На данном этапе необходимо найти перекрытия между всеми квазиконтигами C_1, C_2, …, C_n.

Для начала дадим определение перекрытию. Пусть имеется квазиконтиг A и квазиконтиг B. Перекрытием называется ситуация, при которой часть квазиконтига A совпадает с частью квазиконтига B при фиксированном их расположении относительно друг друга (см. рис. 2, а-б). При этом строка, по которой перекрываются квазиконтиги, не должна быть маленькой. Минимальный порог длины такой строки записывается в файл конфигурации и является параметром алгоритма.



а)



б)

Рис. 2. Перекрытие двух квазиконтигов: а) случай частичного перекрытия; б) случай покрытия одним квазиконтигом другого

Величина CenterShift – сдвиг между центрами двух перекрывающихся квазиконтигов, принимает только неотрицательные значения. Из-за этого любое перекрытие с ненулевым сдвигом однозначно задается тройкой: <номер первого квазиконтига (левее другого), номер второго квазиконтига, сдвиг CenterShift>. Если центры квазиконтигов совпадают (CenterShift=0), то перекрытие записывается так: <квазиконтиг с наименьшим номером, квазиконтиг с наибольшим номером, 0>.

Теперь рассмотрим подробнее сам процесс поиска перекрытий.

Для эффективного поиска подстрок во всех квазиконтигах C_i используется суффиксный массив этих строк. Его построение происходит следующим образом. Сначала строится строка «$C_1\$C_2\$C_3\$...C_n\$$», где C_i – i-тый квазиконтиг.

После чего создается массив всех суффиксов этой строки, т.е. массив $a[i]$, где при создании устанавливается $a[i]=i$ (i-тый суффикс на i-той позиции). После этого производится сортировка суффиксов в массиве $a[i]$ в лексикографическом порядке.

С помощью построенного суффиксного массива можно быстро искать заданную подстроку во всех квазиконтигах. Это можно сделать, например, с помощью бинарного поиска в суффиксном массиве всех суффиксов, которые начинаются с заданной строки. Они будут располагаться рядом за счет сортировки.

Далее процесс поиска перекрытий выглядит просто. Для каждого квазиконтига *B* рассматриваются его префиксы в порядке увеличения длины, начиная с минимального порога. Для того, чтобы некоторый квазиконтиг *A* частично перекрывался с квазиконтигом *B* по выбранному префиксу (не в случае полного покрытия одним квазиконтигом другого), необходимо, чтобы квазиконтиг *A* оканчивался на него. Таким образом, необходимо найти все суффиксы в суффиксном массиве, которые начинаются со строки «<зафиксированный префикс *B*>$» и понять, какой квазиконтиг завершается на этом месте. Таким способом можно найти все перекрытия кроме случаев, когда один квазиконтиг полностью покрывает другой. Такой случай можно обработать отдельно, для каждого квазиконтига произведя поиск суффиксов, начинающихся с него.

При таком подходе будут находиться только точные перекрытия, т.е. те перекрытия, при которых часть строки *A* полностью совпадает с частью строки *B*. Однако в квазиконтигах бывают ошибки, и имеет смысл искать неточные перекрытия. Для этого необходимо после фиксирования префикса (или всего квазиконтига), по которому будут перекрываться квазиконтиги, изменить некоторое небольшое число его символов в нем.

Понятно, что имеет смысл изменять только небольшое число символов префикса. Поэтому одним из параметров алгоритма данного этапа является максимальное число ошибок в перекрывающейся строке.

1.2.3.2. Удаление транзитивных перекрытий

Перекрытие <*A*, *C*, CenterShiftAC> между квазиконтигами *A* и *C* будем называть транзитивным, если существуют перекрытия <*A*, *B*, CenterShiftAB> и <*B*, *C*, CenterShiftBC> такие, что:

- CenterShiftAC = CenterShiftAB + CenterShiftBC;
- CenterShiftAB < CenterShiftAC;
- CenterShiftBC < CenterShiftAC.

Пример транзитивного перекрытия <*A*, *C*, CenterShiftAC> изображен на рис. 3.

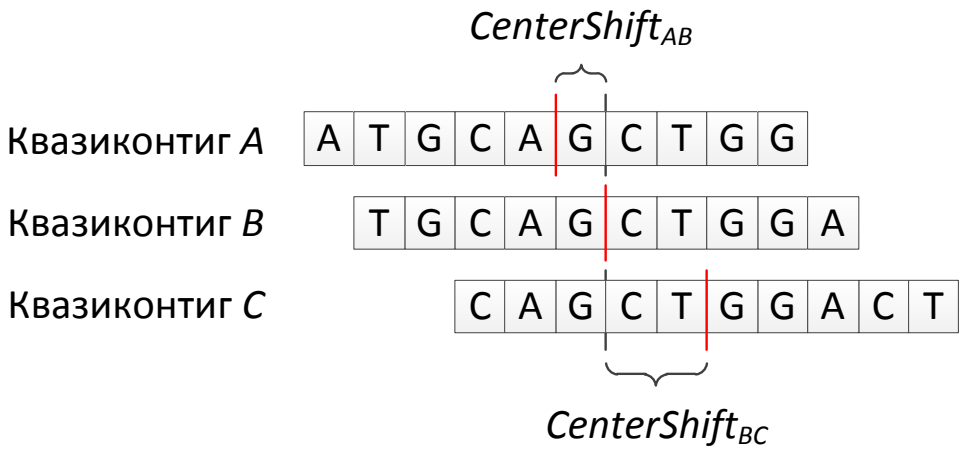


Рис. 3. Транзитивное перекрытие

Заметим, что в транзитивных перекрытиях нет никакой необходимости – мы всегда можем их получить, используя другие меньшие по сдвигу перекрытия. Более того, транзитивные перекрытия увеличивают сложность алгоритмов на следующих шагах. Поэтому необходимо удалить все такие перекрытия.

Поиск транзитивных перекрытий осуществляется следующим образом. Сначала для каждого квазиконтига все его перекрытия упорядочиваются по

сдвигу. После чего для каждого квазиконтига *A* перебираются два соседних квазиконтига *B* и *C*, которые перекрываются с *A*, и также которые перекрываются между собой, и удаляется перекрытие *AB* или *AC* с самым длинным сдвигом.

1.2.3.3. Построение графа перекрытий и его упрощение

Графом перекрытий называется граф, вершины которого – квазиконтиги, а ребра между ними проводятся, если соответствующие квазиконтиги перекрываются. При этом на ребре пишется центральный сдвиг перекрытия и вес ребра. Пример графа перекрытий изображен на рис. 4.

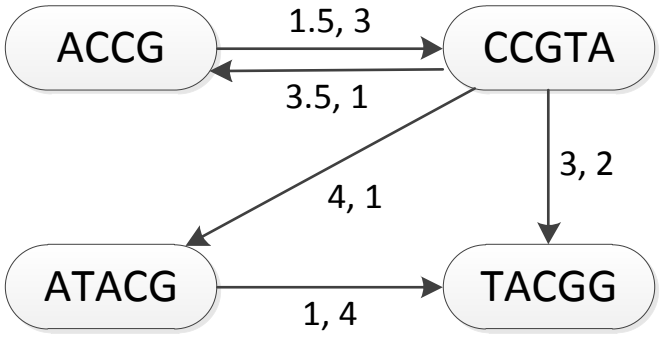


Рис. 4. Граф перекрытий

Вес ребра является числовой характеристикой надежности ребра. Чем она больше, тем больше вероятность того, что данное перекрытие действительно присутствует в геноме, а не образовалось в результате совпадения некоторой части двух квазиконтигов. Вес рассчитывается по длине перекрывающейся части, ввиду того, что чем больше перекрывающаяся часть, тем больше уверенности в том, что это не случайны повтор.

Граф перекрытий довольно легко построить, зная все перекрытия между квазиконтигами. Он очень удобен для последующей работы над перекрытиями и квазиконтигами благодаря наглядному отображению ситуации.

Для лучшего понимания происходящего по графу перекрытий, будем далее писать на ребрах графа только длину соответствующего перекрытия. Пример такого графа изображен на рис. 5.

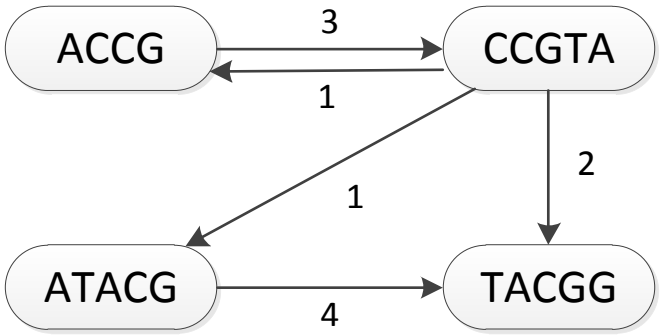


Рис. 5. Граф перекрытий (на ребрах написана длина перекрытия)

После построения графа происходит его упрощение. Под упрощением понимается процесс объединения схожих путей, удаление отростков, удаления не максимальных по весу ребер.

При объединении схожих путей берется два пути, которые одинаковы по длине, имеют одну и ту же начальную и конечную вершину, и производится их объединение, проверяя, что они действительно схожи (отличаются в небольшом числе символов). Объединение происходит путем выстраивания всех вершин в двух путях в одну цепочку, учитывая их сдвиги относительно соседних вершин. Пример изображен на рис. 6.

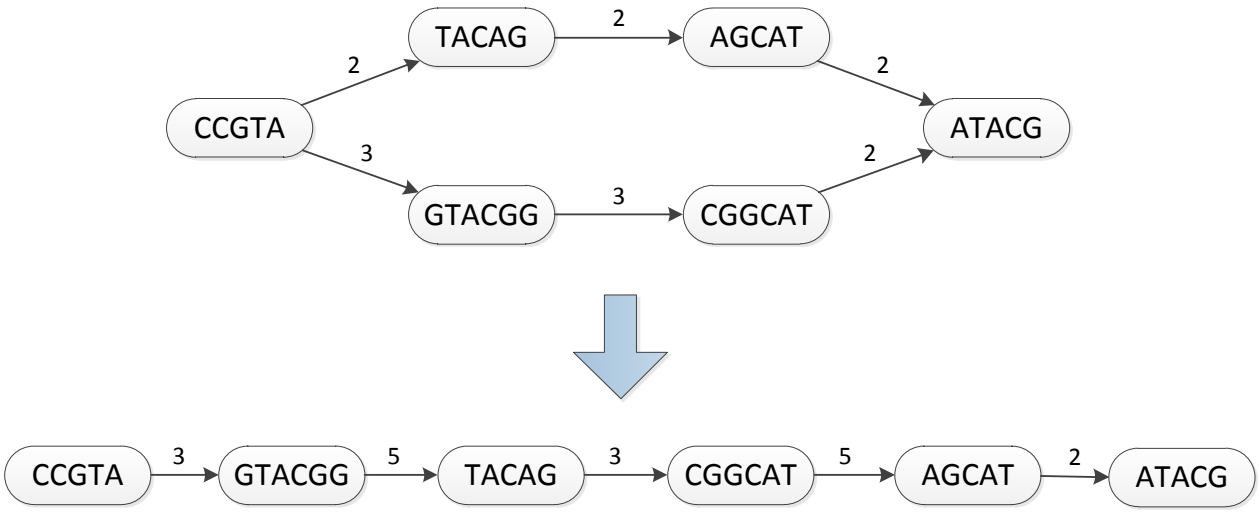


Рис. 6. Объединение путей

Отростком в графе называется небольшая часть графа, наибольший путь из которой содержит не больше некоторой небольшой константы вершин (например, не более пяти вершин), а также ребро в которую ведет из вершины с развилкой. Такие отростки удаляются из графа из-за того, что они, скорее всего, возникли в результате ошибки, и того, что они мешают продолжению контига. Пример удаления отростков изображен на рис. 7.

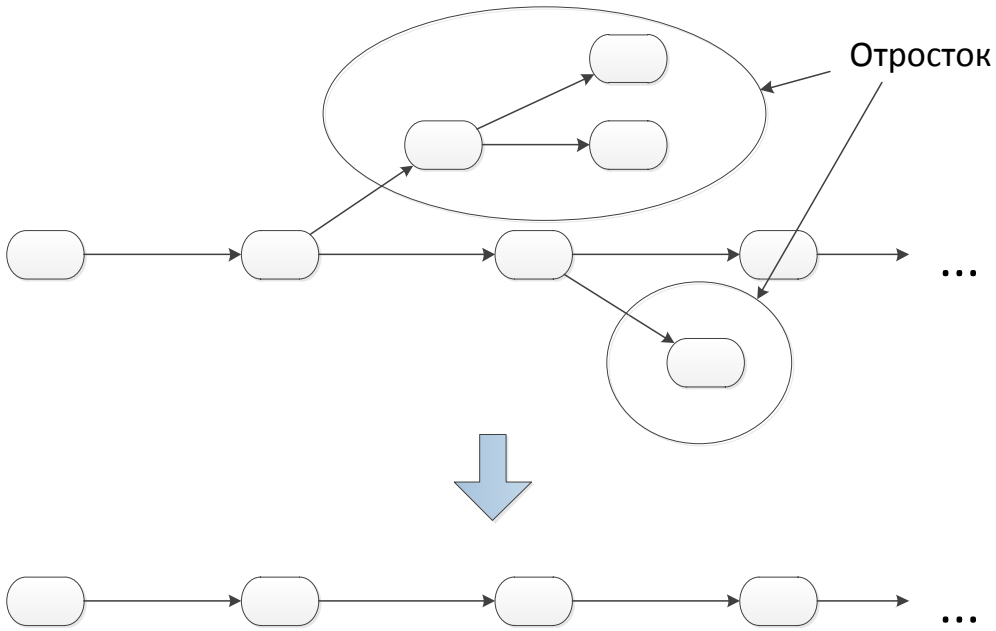


Рис. 7. Удаление отростков

При удалении не максимальных по весу ребер рассматривается каждая вершина и все её исходящие и входящие ребра. Из списка входящих ребер выбирается ребро с наибольшим весом, и из списка исходящих ребер тоже выбирается максимальное по весу ребро, эти ребра помечаются. После рассмотрения всех вершин непомеченные ребра удаляются. Такой процесс помогает избавиться от неправдоподобных ребер в графе перекрытий.

1.2.3.4. Вывод первого приближения контигов

На этом этапе выводятся первое приближение контигов, как непересекающихся путей с вершинами, имеющими равные единице входящую и

исходящую степени. Любой контиг сначала выводится в виде последовательности номеров квазиконтигов и смещений между двумя соседними квазиконтигами. После чего строится консенсус всех перечисленных квазиконтигов и выводится одна последовательность нуклеотидов – итоговый контиг.

1.2.3.5. Микросборка

Последним этапом является микросборка, которая позволяет объединить некоторые из контигов, полученных на предыдущем этапе. Этот этап состоит из трех шагов:

- выравнивание чтений на имеющиеся контиги;
- определение связей между парами контигов и определение возможных последовательностей между ними;
- обработка полученного графа связей и вывод более длинных контигов.

На первом шаге все исходные чтения выравниваются, на полученные контиги. Это делается с помощью программы Bowtie [9]. При выравнивании не учитывается парная информация. Для каждого чтения выводятся все позиции в контигах, в которых это чтение встречается, если их число не превышает заданного порога (его значение обычно равно 10).

Далее определяются пары контигов, которые могут идти подряд в геноме на небольшом расстоянии друг от друга, исходя из наличия соединяющих их парных чтений. Пара чтений соединяет два контига, если одно из чтений выравнивается на первый контиг, а другое – на второй. Если было найдено несколько чтений, соединяющих пару контигов, то, скорее всего, в геноме они идут подряд.

Затем для каждой пары соединенных контигов выполняется заполнение промежутка между ними. Для этого выбираются все пары чтений, хотя бы одно из которых было выравнено на один из этих контигов. Из этих чтений строится

граф де Брёйна, который используется, чтобы восстановить квазиконтиги из соединяющих парных чтений. Эти квазиконтиги покрывают промежуток между контигами, тем самым позволяя его заполнить.

После того, как для некоторых пар контигов удалось определить возможную последовательность, заполняющую промежуток между ними, может оказаться, что после одного контига могут следовать несколько других. Возникает задача, сходная с задачей поиска контигов в графе перекрытий. Отличием является то, что вершины, соединенные ребром могут не перекрываться.

ГЛАВА 2. ПРОБЛЕМА ЗАПУТАННОСТИ ГРАФА ПЕРЕКРЫТИЙ. СУЩЕСТВУЮЩИЕ МЕТОДЫ ЕЁ РЕШЕНИЯ

В данной главе рассмотрена проблема запутанности графа перекрытий, рассмотрены существующие методы её решения, для методов приведены их достоинства и недостатки.

2.1. ПРОБЛЕМА ЗАПУТАННОСТИ ГРАФА ПЕРЕКРЫТИЙ

В предыдущей главе было рассказано из каких частей состоит сборщик генома *ITMO Genome Assembler* и для каждой части было дано краткое изложение принципов его работы. Далее в работе будет рассматриваться только этап сборки контигов из квазиконтигов.

В простом понимании, этап сборки контигов из квазиконтигов нужен для того, чтобы расширить квазиконтиги в обоих направлениях настолько, насколько это возможно. Таким образом, из большого числа квазиконтигов с небольшой длиной необходимо собрать небольшое число контигов с большой длиной.

Одним из этапов сборки контигов из квазиконтигов является этап построения графа перекрытий и его упрощение.

При построении графа перекрытий обычно выясняется, что он очень запутан, и даже человеку, смотря на граф, в большинстве случаев непонятно, какие простые пути должны быть оставлены. Одна из маленьких частей реального графа перекрытий изображена на рис. 8.

Такая запутанность графа перекрытий в большей части вызвана сложной структурой генома: в нем находится большое число повторяющихся фрагментов. Из-за этих повторяющихся фрагментов квазиконтиги, находящиеся на самом деле в разных частях генома, могут перекрываться.

Также на запутанность графа перекрытий оказывают влияние другие факторы:

- Наличие в геноме полиморфизмов. Полиморфизм (*Single nucleotide polymorphism, SNP*) — отличие схожих частей генома в одном нуклеотиде.
- Наличие ошибок в квазиконтигах. Такие ошибки могли возникнуть в квазиконтигах из-за, например, наличия ошибок в исходных чтениях.

Уже на следующем этапе – стадии ***Layout*** (вывод первого приближения контигов) – происходит вывод контигов используя простые пути в графе перекрытий (унитиги). Один контиг получается из одного простого пути в графе перекрытий.

Таким образом, наиболее интересная и трудная задача в процессе сборки контигов из квазиконтигов – не просто расширить квазиконтиги настолько, насколько это возможно, а распутать граф перекрытий.

Например, для графа, изображенного на рис. 8, распутанный граф перекрытий должен выглядеть так, как показано на рис. 9.

Традиционно при сборке контигов выделяют приоритеты:

- **Уменьшить число ошибочных контигов.** Контиг называется ошибочным, если он не встречается в геноме.
- **Увеличить среднюю длину собранных контигов** (при неизменной суммарной длине всех собранных контигов).

Ошибочные контиги возникают в большинстве случаев из-за того, что путь, который соответствует ошибочному контигу, содержал ошибочное ребро – ребро, которое соединяло два квазиконтига, которые на самом деле были расположены в разных частях генома. Проходя по такому ребру, мы соединяем две разные несмежные части генома.

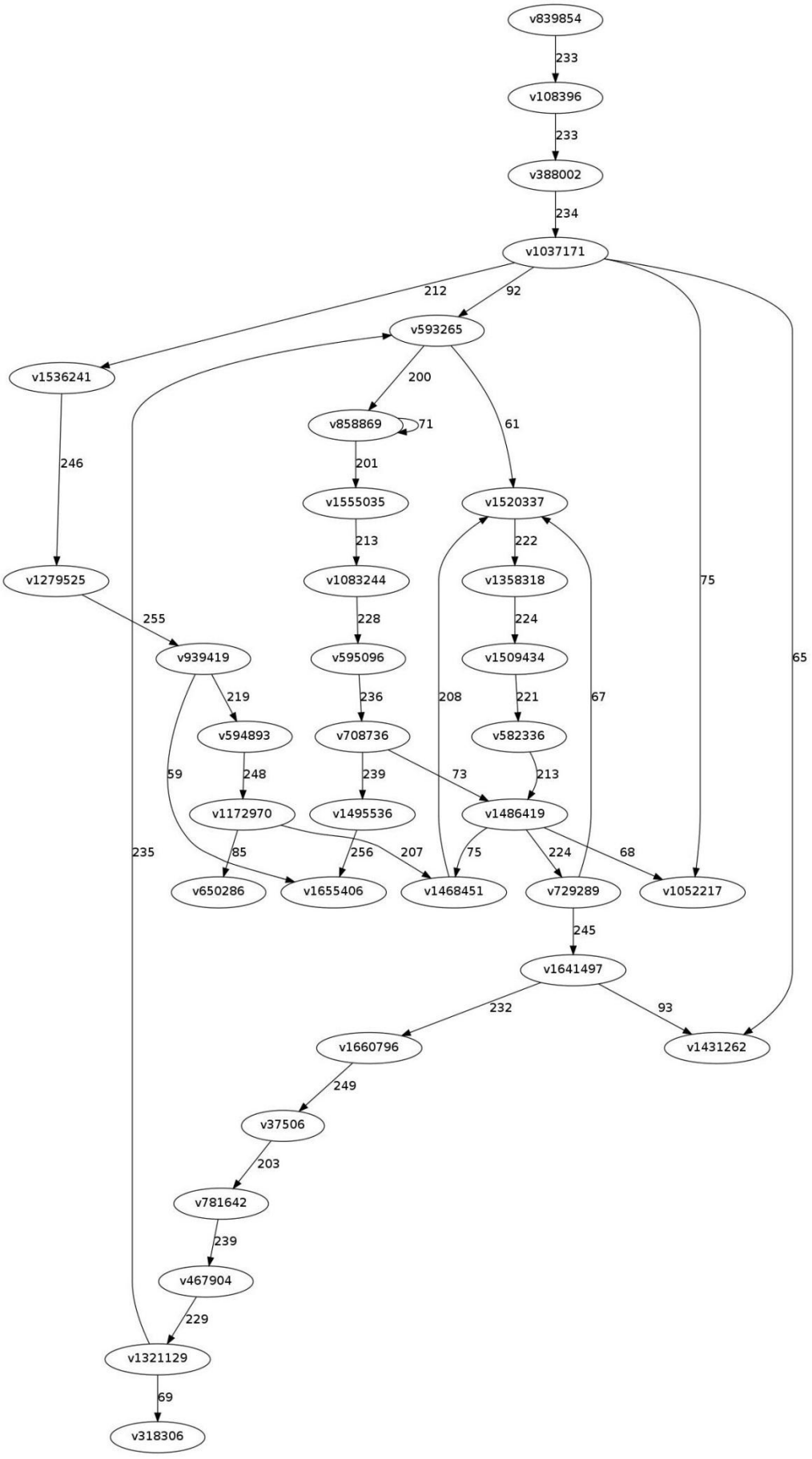


Рис. 8. Часть реального графа перекрытий

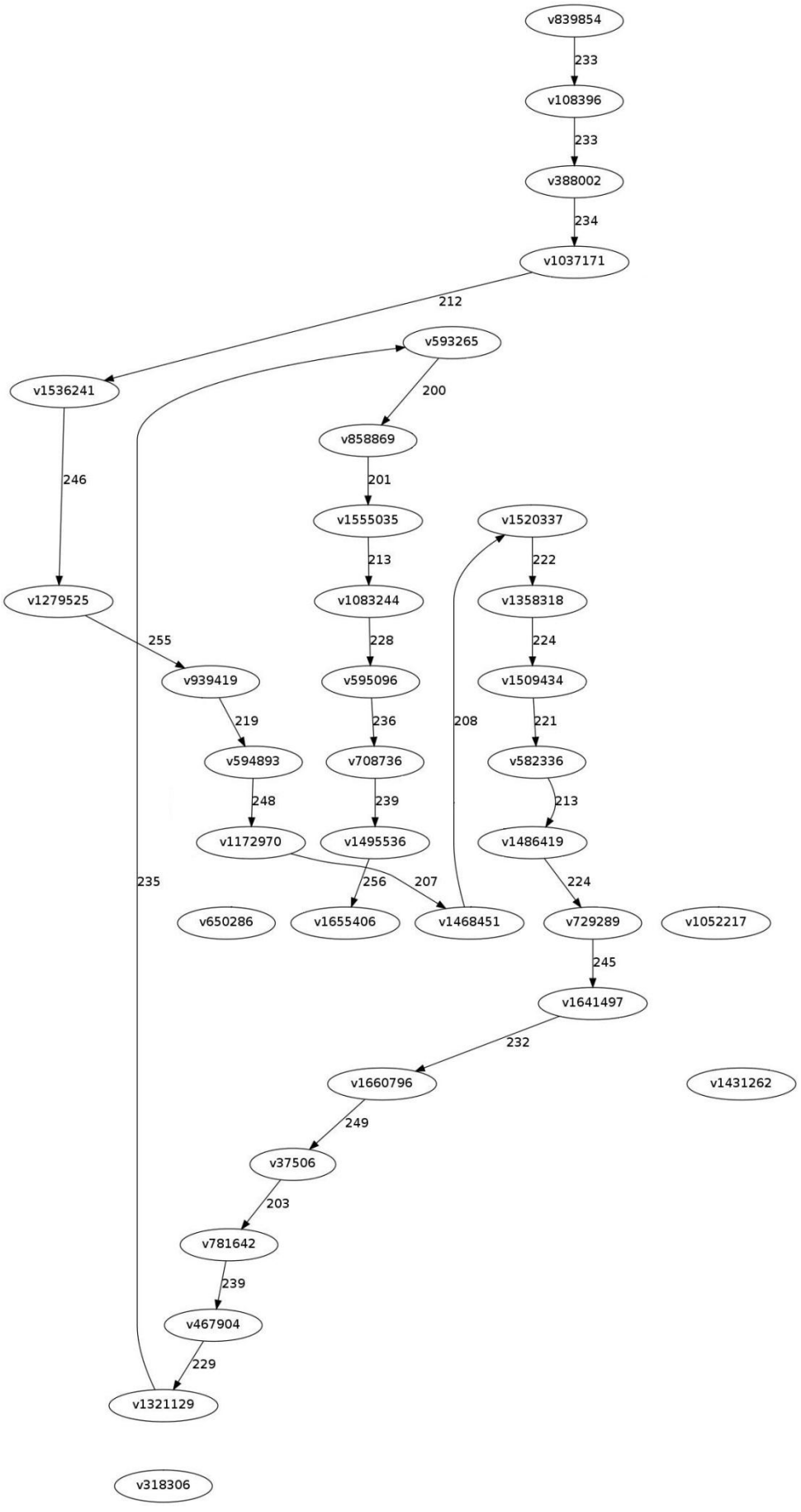


Рис. 9. Распутанная часть реального графа перекрытий

2.2. Существующие решения

Существующие решения для упрощения графа перекрытий так или иначе используют следующие подэтапы:

- Объединение схожих путей.
- Удаление отростков.
- Анализ развилок и их упрощение.

Например, как уже было описано в разделе 1.2.3.3, сборщик генома *ITMO Genome Assembler* использует эти три этапа, а под этапом «анализ развилок и их упрощение» подразумевается этап удаления не максимальных по весу ребер, который и выполняет задачу по анализу и упрощению развилок.

Однако, используя только эти этапы, процесс упрощения графа остается малоэффективным из-за ошибочных ребер. Такие ребра, во-первых, участвуют в создании ошибочных контигов, а во-вторых, мешают всем другим этапам упрощения, т.к. они превращают граф в «запутанный клубок».

Существующие решения так или иначе пытаются бороться с ошибочными ребрами, однако, исключить все ошибочные ребра практически невозможно из-за сложностей при их выявлении.

2.3. Ошибочные ребра

Ошибочные перекрытия возникают в большей части из-за повторяющихся фрагментов, которые в большом количестве присутствуют в любом геноме.

Объясним почему. Предположим у нас есть две разные части одного генома. Пусть квазиконтиг A находится в первой части, а квазиконтиг B – во второй. И пусть в этих частях есть небольшой одинаковый фрагмент, будем называть его «повтор». Поясняющая схема изображена на рис. 10.

Если квазиконтиг *A* заканчивается на этом повторе (не пересекая его полностью), а квазиконтиг *B* начинается на нем (тоже не пересекая его полностью), то между квазиконтигом *A* и квазиконтигом *B* будет найдено перекрытие.

Такое перекрытие (или ребро) будет ошибочным, т.к. оно соединяет два квазиконтига из разных частей генома.

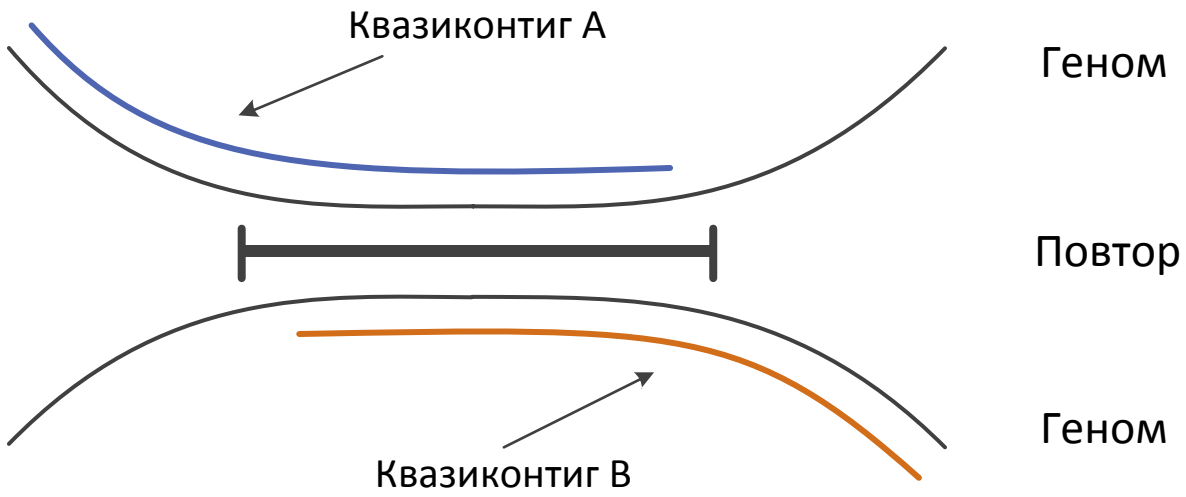


Рис. 10. Ошибочное ребро

На первый взгляд может показаться, что такое случается крайне редко. Однако это не так.

Можно взять для примера распространенный в биоинформатике геном кишечной палочки (*E. coli*, длина генома – 4,6 млн нуклеотидов). Если построить график зависимости числа повторяющихся фрагментов в геноме от размера этого фрагмента (см. рис. 11), то можно заметить, что число повторов в геноме очень велико вне зависимости от длины повтора и они встречаются «на каждом шагу». Причем, чем меньше длина повтора, тем чаще повтор такой длины встречается в геноме.

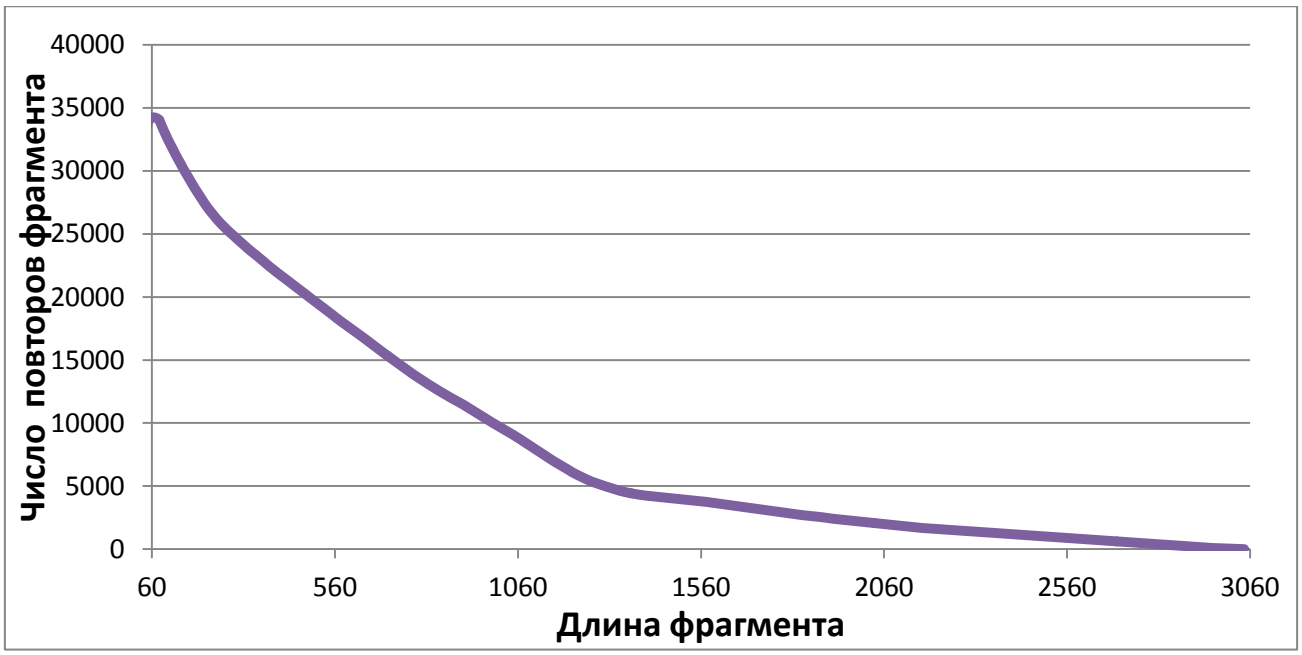


Рис. 11. Зависимость числа повторяющихся фрагментов генома от длины фрагмента

Также с помощью генома *E. coli* было экспериментально подсчитано, что если покрыть его равномерно чтениями (среднее покрытие каждого нуклеотида генома – 40), то процент ошибочных ребер перед этапом упрощения графа перекрытий будет равен 0,5% от всех ребер. Эта величина относительно небольшая, но именно из-за таких ребер качество сборки контигов заметно ухудшается.

Далее поподробнее опишем несколько наиболее распространённых случаев ухудшения качества сборки из-за ошибочных ребер.

Рассмотрим первый простой вариант ситуации в графе перекрытий при появлении ошибочного ребра. Предположим, что в графе перекрытий есть два простых пути, и добавляется ошибочное ребро, которое соединяет вершину в середине первого пути с вершиной в середине второго пути. Такая ситуация изображена на рис. 12, красным цветом показано ошибочное ребро.

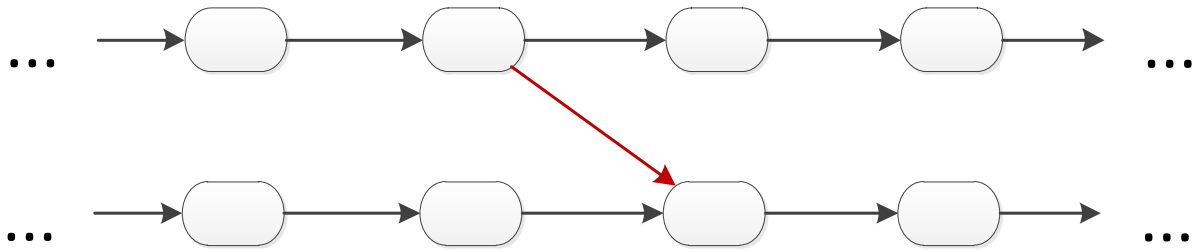


Рис. 12. Часть графа перекрытий с ошибочным ребром. Ситуация 1.

На этапе анализа развилок и их упрощений данная ситуация может легко разрешиться, если вес ошибочного ребра будет сильно меньше весов смежных ребер. Однако на самом деле это не всегда так.

Если же данная ситуация в графе перекрытий не будет решена на этапе анализа и упрощения развилок, то на стадии *Layout* два простых пути, которые были соединены ошибочным ребром, будут разрезаны в местах неопределенностей, уменьшая тем самым среднюю длину получившихся контигов.

Однако предыдущий вариант развития событий не самый плохой. Если же ошибочное ребро соединит два пути так, что не будет конфликтных ребер (2 и более ребер, исходящих из одной вершины, или входящих в одну вершину), то ошибочное ребро с большой вероятностью может быть воспринято как нормально ребро, и получившийся контиг из этого объединенного пути будет ошибочным. Пример такого случая можно увидеть на рис. 13.

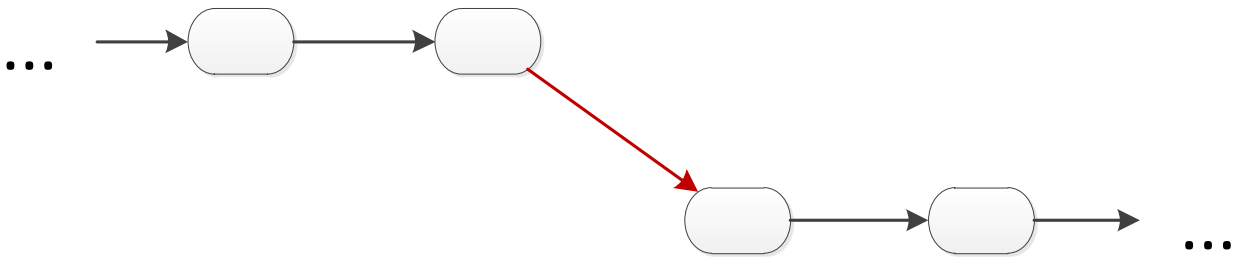


Рис. 13. Часть графа перекрытий с ошибочным ребром. Ситуация 2.

Рассмотренные случаи являются самыми простыми, в чистом виде такие случаи возникают редко. Чаще возникают случаи, которые очень похожи на

первый рассмотренный вариант ситуации в графе перекрытий, но не с одним ошибочным ребром, а с большим количеством таких ребер. Даже после этапа анализа развилок и удаления ребер с малым весом, запутанность графа в некоторых частях остается большой.

Второй рассмотренный вариант встречается намного реже, чем первый. Однако такие случаи наблюдались при рассмотрении графа перекрытий для реальных данных (квазиконтиги, собранные из парных чтений бактерии *E. Coli*).

ГЛАВА 3. ОПИСАНИЕ ПРЕДЛАГАЕМОГО РЕШЕНИЯ

Опишем предлагаемое решение, которое ориентированно на борьбу с ошибочными ребрами.

Для борьбы с такими ребрами предлагается изменить этап упрощения графа перекрытий: добавить этап вычисления правдоподобия каждого ребра, этап удаления малоправдоподобных ребер, а также использовать правдоподобие ребер в других подэтапах упрощения.

Предлагаемый этап упрощения графа перекрытий будет выглядеть следующим образом:

- Вычисление правдоподобия каждого ребра.
- Удаление малоправдоподобных ребер.
- Объединение схожих путей.
- Удаление отростков.
- Анализ развилок и их упрощение (с использованием правдоподобности ребер).

Благодаря этапу удаления малоправдоподобных ребер становиться возможным удалять большую часть ошибочных ребер. Таким образом, кроме непосредственного распутывания графа перекрытий, такое удаление сделает работу следующих этапов более эффективной.

Правдоподобие ребра предлагается вычислять на основе двух оценок:

- На основе теоритической оценки вероятности ребра оказаться ошибочным. Такую вероятность можно оценить используя длину перекрытия и модель распределения повторов в зависимости от длины повтора.
- На основе экспериментальной оценки вероятности ребра оказаться ошибочным. Такая оценка вычисляется на основании покрытия строками небольшой длины перекрывающихся квазиконтигов. Такая

оценка полностью зависит только от исходных данных, и не нуждается ни в какой другой информации о геноме.

Далее более подробно опишем процесс вычисления правдоподобности ребер.

3.1. Вычисление правдоподобности ребра

Правдоподобие ребра вычисляется по следующей формуле:

$$P(e) = 1 - P_w(e)$$

где $P(e)$ – правдоподобие ребра e, $P_w(e)$ – оценочная вероятность ребра оказаться ошибочным.

Оценочная вероятность ребра оказаться ошибочным вычисляется на основе двух оценок – теоритической оценки и экспериментальной оценки:

$$P_w(e) = \alpha P_t(e) + \beta P_{exp}(e)$$

где $P_t(e)$ – теоритическая оценка ребра оказаться ошибочным, $P_{exp}(e)$ – экспериментальная оценка, α и β – коэффициенты учета соответствующих оценок ($\alpha + \beta = 1$).

3.2. Теоритическая оценка

Теоритическая оценка ребра оказаться ошибочным $P_t(e)$ вычисляется с использованием данных о длине перекрытия, а также по модели распределения повторов в зависимости от длины повтора.

Более формально $P_t(e)$ – вероятность того, что существует повтор, который полностью покроет перекрывающуюся часть квазиконтигов A и B. Пример такой ситуации изображен на рис. 14.

Рис. 14. Схема ошибочного перекрытия квазиконтигов

3.3. Экспериментальная оценка

Экспериментальная оценка ребра оказаться ошибочным $P_{exp}(e)$ вычисляется с использованием статистики о покрытии перекрывающихся квазиконтигов строками небольшой длины (k-мерами). Такая оценка полностью зависит только от исходных данных, и не нуждается ни в какой другой информации о геноме.

Например, пусть имеются квазиконтиги A и B, которые перекрываются. Для каждого из них можно подсчитать, сколько раз каждый k-мер в квазиконтиге встречается в исходных парных чтениях. Тогда можно будет построить график зависимости частоты появления k-мера в исходных данных от позиции в соответствующем квазиконтиге.

Заметим, что квазиконтиги A и B перекрываются, что означает, что можно построить объединенный квазиконтиг C = A_u $OV_{A,B}$ B_u, где A_u и B_u – уникальные части квазиконтигов A и B, $OV_{A,B}$ – перекрывающаяся часть квазиконтигов A и B. Поясняющая схема изображена на рис. 15.

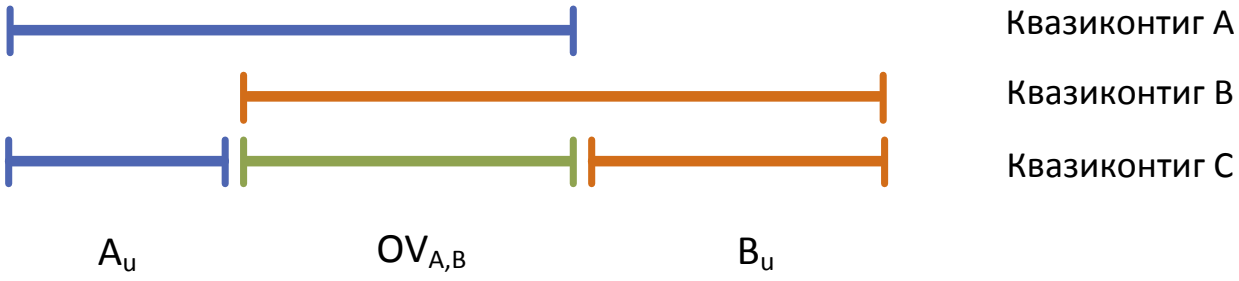


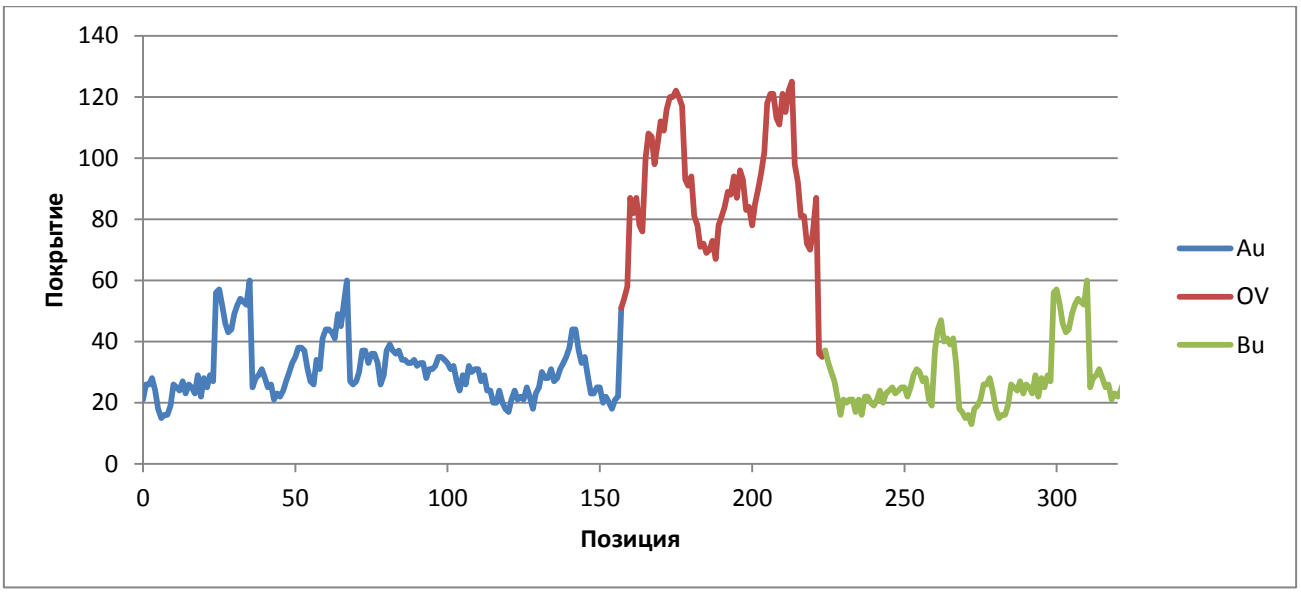
Рис. 15. Схема объединенного квазиконтига

Для объединенного квазиконтига *C* можно тоже построить график зависимости частоты появления *k*-мера в исходных данных от позиции в нем. Такой график будет объединением двух графиков для квазиконтига *A* и *B*.

В идеальной ситуации, когда квазиконтиги перекрываются по части $OV_{A,B}$, которая полностью совпадает с соответствующим суффиксом квазиконтига *A*, а также с соответствующим префиксом квазиконтига *B*, конец графика зависимости частоты *k*-мера от позиции для квазиконтига *A* будет совпадать с началом графика зависимости для квазиконтига *B*. По такому совпадению они и объединяются. Однако, если квазиконтиги *A* и *B* перекрываются с ошибками, то при получении графика для квазиконтига *C*, соответствующие части графиков *A* и *B* усредняются.

Некоторые возможные зависимости частоты появления *k*-мера в исходных данных от позиции в квазиконтиге *C* для реальных данных изображены на рис. 16, а-б. На них синим цветом показана зависимость появления *k*-мера в исходных данных для префикса квазиконтига *A* до перекрывающейся части, красным – для перекрывающейся части и зеленым – для суффикса квазиконтига *B* после перекрывающейся части.

а)



б)

Рис. 16. Графики зависимости частоты появления *k*-мера в исходных данных от его позиции в квазиконтиге.

По этим графикам можно заметить, что их характер поведения различается. Попытаемся это объяснить.

Предположим, что квазиконтиги *A* и *B* действительно находятся рядом и перекрываются как раз из-за этого. Тогда можно предположить, что *k*-меры из объединенного квазиконтига *C* не встречаются в других местах генома (т.е. нет повторов, которые бы совпадали с частью квазиконтига *C* и встречались бы в других местах генома). Такое предположение верно с большой вероятностью, т.к. суммарная длина всех повторов намного меньше длины генома. Тогда (если *k*-меры из квазиконтига *C* не встречаются в других местах генома) с большой вероятностью частота появления этих *k*-меров в исходных данных будет примерно одинаковой, т.к. геном равномерно покрыт квазиконтигами. Такое поведение хорошо наблюдается на рис. 16 а).

Предположим теперь, что квазиконтиги *A* и *B* находятся в разных частях генома и перекрываются только из-за повтора (как изображено на рис. 14). Тогда, ввиду того, что повтор встречается как минимум два раза в геноме, *k*-меры, полученные из него, будут встречаться в исходных данных намного

больше раз, чем *k*-меры из других частей (например, чем *k*-меры из префикса квазиконтига *A* или из суффикса квазиконтига *B*). Такую зависимость можно наблюдать на рис. 16 б).

Экспериментальная оценка ребра оказаться ошибочным как раз ориентируется на такую особенность.

Для выявления такой ситуации делается следующее. Перебираются возможные границы повтора (*R*.begin, *R*.end), такие что:

$$\begin{cases} A.begin < R.begin \bigwedge R.begin \leq B.begin \\ A.end \leq R.end \bigwedge R.end < B.end \end{cases}$$

Поясняющая схема изображена на рис. 17.

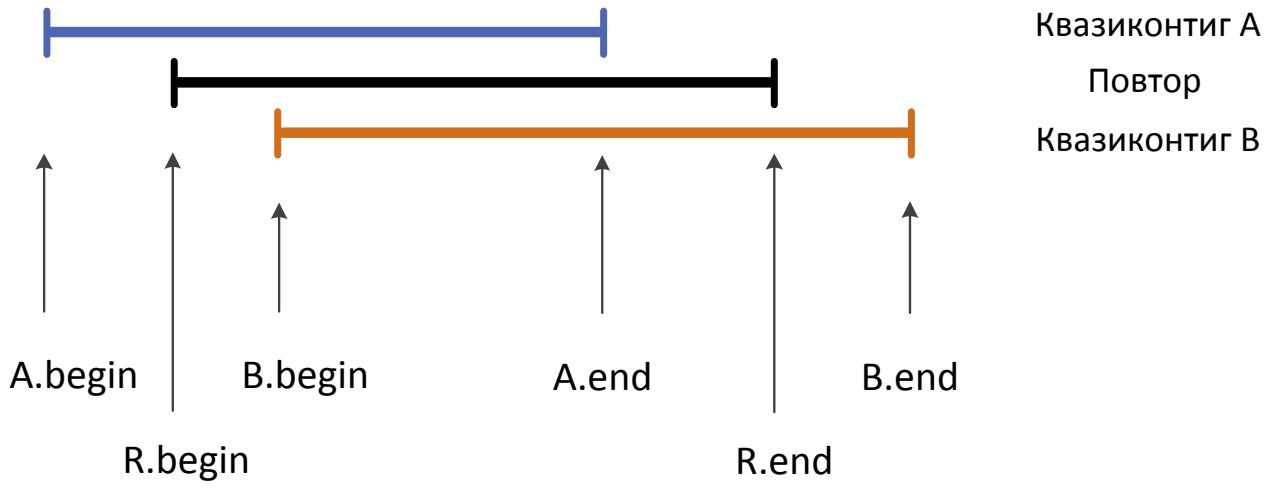


Рис. 17. Схема ошибочного перекрытия квазиконтигов

Далее предполагаем, что повтор начинается именно в этих границах. После этого пытаемся полученный график зависимости появления *k*-мера в исходных данных от позиции аппроксимировать схемой, изображенной на рис. 18.

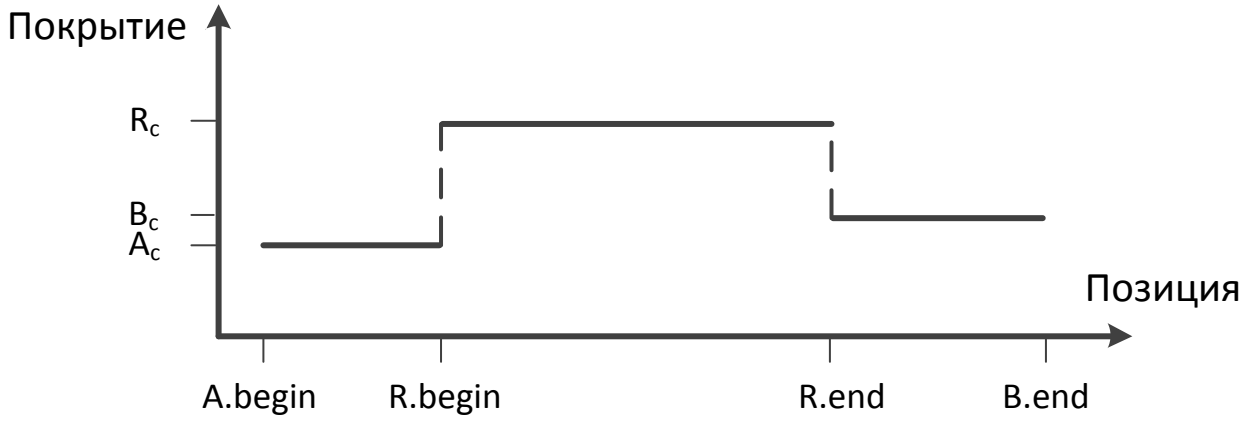


Рис. 18. Зависимость покрытия от позиции в случае, если есть повтор

При этом A_c вычисляется как среднее значение на участке [*A*.begin, *R*.begin), B_c – на участке (*R*.end, *B*.end], R_c – на участке [*R*.begin, *R*.end]. При этом R_c не должно быть меньше величины A_c+B_c, иначе R_c увеличивается до этого значения.

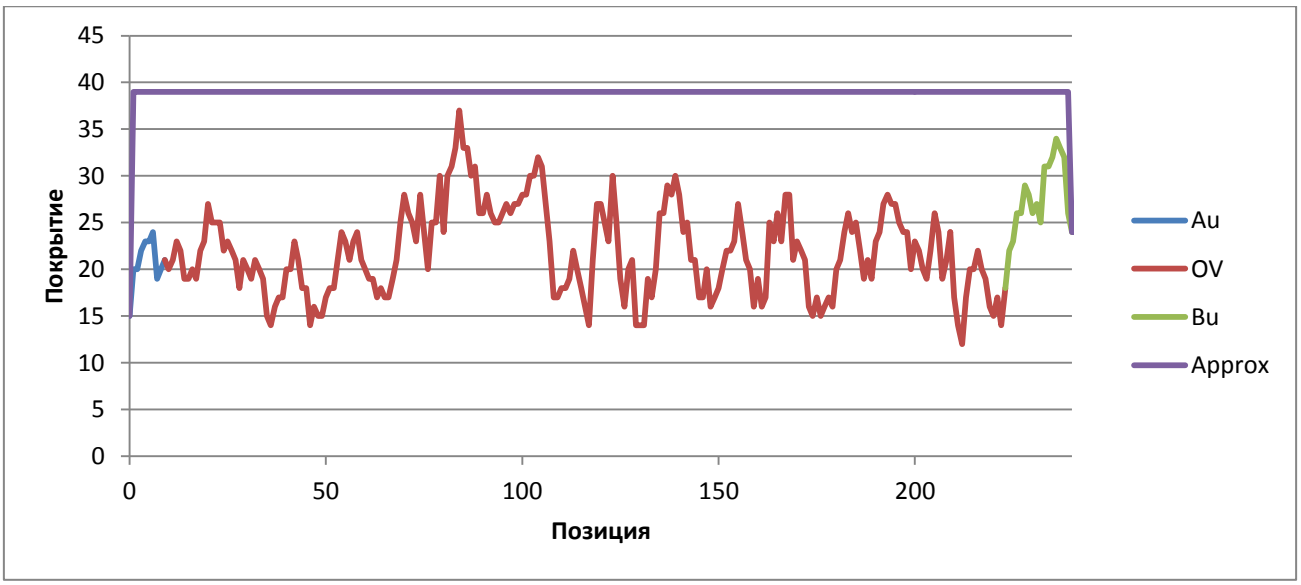
После этого вычисляем коэффициент несоответствия реального графика и аппроксимации. Этот коэффициент вычисляем как отношение площади «ошибочной части» (той части, которая отличает один график от другого) к общей площади реального графика.

Далее выбираем минимальный такой коэффициент для всех возможных границ повтора. Таким образом, мы найдем границы повтора (если повтор есть), которые наиболее хорошо отражаются на графике зависимости частоты появления *k*-мера в исходных данных от позиции.

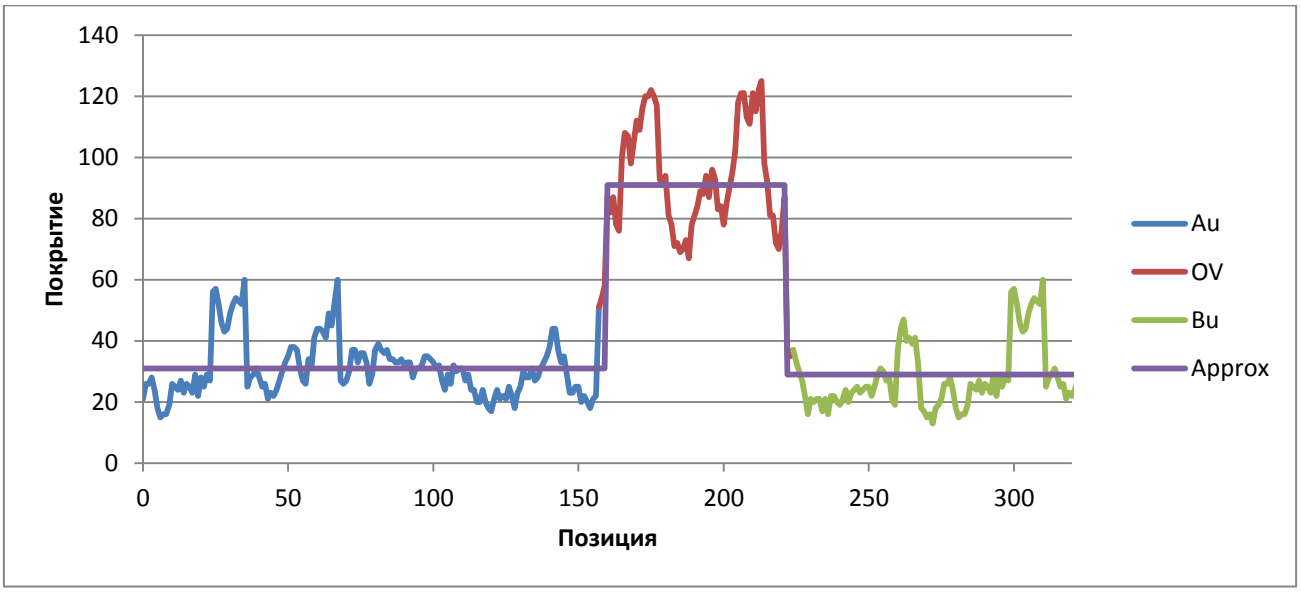
Таким образом, экспериментальная оценка ребра оказаться ошибочным вычисляется следующим образом:

$$P_{exp}(e) = 1 - \min_{\substack{A.begin<R.begin \wedge R.begin \le B.begin \\ A.end \le R.end \wedge R.end < B.end}} \frac{S_{wrong}}{S}$$

Например, для графиков, изображенных на рис. 16, полученная аппроксимация будет выглядеть так, как показано на рис. 19, а-б.



а)



б)

Рис. 19. Графики зависимости частоты появления *k*-мера в исходных данных от его позиции в квазиконтиге.

Коэффициент несоответствия реального графика и аппроксимации для случая а) $k = 0.766$ (т.е. ошибочная площадь занимает 76.6% от площади реального графика), для случая б) $k = 0.224$.

Для случая а) $P_{exp}(e) = 0.234$, для б) $P_{exp}(e) = 0.776$.

ГЛАВА 4. РЕАЛИЗАЦИЯ АЛГОРИТМА И ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

В данной главе приводятся технические характеристики реализации, описываются произведенные эксперименты и полученные результаты. Приводится небольшой анализ полученных результатов, и делаются выводы о предложенном алгоритме.

4.1. Реализация алгоритма

В рамках работы был реализован алгоритм, который описан в данной работе.

Алгоритм был написан на языке программирования *Java*. Были использованы уже написанные классы сборщика генома *ITMO Genome Assembler*, а также сторонние библиотеки, которые использует этот сборщик.

4.2. Экспериментальные исследования

Экспериментальные исследования проводились над разными исходными данными трех бактерий с разными размерами генома. Информация об этих бактериях приведена в таблице 1.

Таблица 1.

Название (англ.)	**Сокращенное название**	**Название (рус.)**	**Длина собранного генома**
Buchnera aphidicola	**Buchnera**	**-**	**641’895**
Haemophilus influenzae	**H. influenzae**	**Гемофильная палочка, палочка Пфейфера**	**1’813’033**
Escherichia coli	**E. coli**	**Кишечная палочка**	**4’639’675**

Каждая из этих бактерии имеет собранный геном, что помогало посчитать число ошибочных контигов, а также сгенерировать наборы исходных данных для экспериментов.

Наборы исходных квазиконтигов были получены следующим способом:

- Безошибочные квазиконтиги (сгенерированы из собранного генома).
- Квазиконтиги с ошибками (сгенерированы из собранного генома).
- Реальные квазиконтиги (с ошибками, собраны сборщиком *ITMO Genome Assembler* из реальных парных чтений).

Для каждой бактерии было использовано два набора исходных квазиконтигов: безошибочные и реальные квазиконтиги. Однако из-за отсутствия подходящих реальных парных чтений для бактерий *Buchnera* и *H. influenzae*, для них были использованы сгенерированные квазиконтиги с ошибками.

Для каждого набора исходных данных для каждой бактерии запускался сборщик *ITMO Genome Assembler* без дополнительных предлагаемых этапов, а также с ними.

Приведем результаты запусков для бактерии *Buchnera* (длина генома – 0,6 млн нуклеотидов). Они показаны в таблице 2 для безошибочных контигов и в таблице 3 для контигов с ошибками.

Таблица 2.

Метод	**Число контигов**	**Суммарная длина контигов**	**Средняя длина контига**	**Ошибочных контигов**
ITMO Genome Assembler без предлагаемых этапов	**33**	**642'561**	**19'472**	**1**
ITMO Genome Assembler с предлагаемыми этапами	**33**	**642'561**	**19'472**	**1**

Таблица 3.

Метод	Число контигов	Суммарная длина контигов	Средняя длина контига	Ошибочных контигов
ITMO Genome Assembler без предлагаемых этапов	**39**	**642'816**	**16'482**	**1**
ITMO Genome Assembler с предлагаемыми этапами	**39**	**642'816**	**16'482**	**1**

Как видно из этих таблиц, результаты сборки не изменились при добавлении предлагаемых дополнительных этапов.

Приведем результаты запусков для бактерии *H. influenzae* (длина генома – 1,8 млн нуклеотидов). Они показаны в таблице 4 для безошибочных контигов и в таблице 5 для контигов с ошибками.

Таблица 4.

Метод	Число контигов	Суммарная длина контигов	Средняя длина контига	Ошибочных контигов
ITMO Genome Assembler без предлагаемых этапов	**71**	**1'789'421**	**25'203**	**1**
ITMO Genome Assembler с предлагаемыми этапами	**62**	**1'788'234**	**28'842**	**0**

Таблица 5.

Метод	Число контигов	Суммарная длина контигов	Средняя длина контига	Ошибочных контигов
ITMO Genome Assembler без предлагаемых этапов	**99**	**1'793'292**	**18'114**	**3**
ITMO Genome Assembler с предлагаемыми этапами	**74**	**1'789'284**	**24'180**	**1**

Как видно из представленных таблиц, результаты сборки значительно улучшились после добавления предлагаемых этапов. При этом улучшения произошли как по характеристике «средняя длина контига» (она увеличилась), так и по характеристике «число ошибочных контигов» (их число уменьшилось).

Результаты запусков для широко распространенного генома кишечной палочки *E. coli* (длина генома – 4,6 млн нуклеотидов) приведены в таблице 6 (для безошибочных квазиконтигов) и таблице 7 (для реальных квазиконтигов).

Таблица 6.

Метод	**Число контигов**	**Суммарная длина контигов**	**Средняя длина контига**	**Ошибочных контигов**
ITMO Genome Assembler без предлагаемых этапов	**160**	**4’599’061**	**28’744**	**2**
ITMO Genome Assembler с предлагаемыми этапами	**140**	**4’598’439**	**32’846**	**1**

Таблица 7.

Метод	**Число контигов**	**Суммарная длина контигов**	**Средняя длина контига**	**Ошибочных контигов**
ITMO Genome Assembler без предлагаемых этапов	**405**	**4’623’885**	**11’417**	**15**
ITMO Genome Assembler с предлагаемыми этапами	**418**	**4’629’350**	**11’075**	**9**

Из представленных таблиц видно, что результаты сборки улучшились после добавления предлагаемых этапов. В первом случае улучшения произошли по обеим характеристикам (средняя длина контига и число ошибочных контигов), во втором случае – улучшения только по числу ошибочных контигов (качество при этом тоже улучшилось виду того, что уменьшение числа

ошибочных контигов – более приоритетная задача, чем увеличение средней длины).

4.3. Выводы

Из представленных результатов экспериментальных исследований можно сделать следующие выводы:

- Для разных геномов дополнительные этапы по-разному изменяют качество итоговой сборки. В большинстве случаев качество сборки улучшается, однако также оно может оставаться без изменений. Случаев ухудшения качества из-за предложенных дополнительных этапов не было замечено.
- Дополнительные этапы могут уменьшать среднюю длину контигов. Это может происходить из-за того, что ребро, находящаяся в центре простого пути, может быть помечено как малоправдоподобное и впоследствии удалено.

Таким образом, предложенный подход упрощения графа перекрытий показал свою эффективность и применимость в процессе сборки геномных данных.

ЗАКЛЮЧЕНИЕ

В работе предложен метод упрощения графа перекрытий, основанный на вычислении правдоподобия ребер графа.

Метод был протестирован на нескольких наборах исходных данных разных бактерий. Результаты экспериментов показали применимость и эффективность предлагаемого подхода.

Таким образом, цели работы достигнуты.

СПИСОК ИСТОЧНИКОВ

1. *Schuster S. C.* Next-generation sequencing transforms today's biology // Nature Methods. 2008. Vol. 5. P. 16–18.
2. Illumina, Inc. http://www.illumina.com/.
3. *Исенбаев В. В., Шалыто А. А.* Разработка системы секвенирования ДНК с использованием paired-end данных. 2010. http://is.ifmo.ru/genom/_isenbaev_thesis.pdf.
4. *Александров А. В., Казаков С. В., Мельников С. В., Сергушичев А. А., Царев Ф. Н., Шалыто А. А.* Метод исправления ошибок в наборе чтений нуклеотидной последовательности // Научно-технический вестник Санкт-Петербургского государственного университета информационных технологий, механики и оптики. 2011. № 5, с. 81–84.
5. *Александров А.В., Казаков С.В., Мельников С.В., Сергушичев А.А., Царев Ф.Н.* Метод сборки контигов геномных последовательностей на основе совместного применения графов де Брюина и графов перекрытий // Научно-технический вестник информационных технологий, механики и оптики. 2012. № 6 (82). с. 93-98.
6. De novo Genome Assembly Assesment Project. http://cnag.bsc.es.
7. Assemblathon 2. http://assemblathon.org.
8. International Human Genome Sequencing Consortium. 2001. Initial sequencing and analysis of the human genome. Nature 409: 860–921.
9. Bowtie: An ultrafast memory-efficient short read aligner. http://bowtie-bio.sourceforge.net/index.shtml.