ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ АКАДЕМИЧЕСКИЙ УНИВЕРСИТЕТ РОССИЙСКОЙ АКАДЕМИИ НАУК

Диссертация допущена к защите

Зав. кафедрой

_____ А.В. Омельченко

«____» _____ 2015 г.

ДИССЕРТАЦИЯ
НА СОИСКАНИЕ УЧЕНОЙ СТЕПЕНИ
МАГИСТРА

Тема: **Построение скаффолдов на основе анализа сборок нескольких родственных геномов**

Направление: 03.04.01 – Прикладные математика и физика

Выполнил студент Н.К. Ситдыкова

(подпись)

Руководитель:

PhD, доцент М.А. Алексеев

(подпись)

Рецензент:

магистр, м.н.с. М.А. Колмогоров

(подпись)

Санкт-Петербург

2015

Содержание

Введение

Получение полной геномной последовательности является ключевым компонентом любого исследования, посвященного анализу организмов на молекулярном уровне. На получение полного генома человека ушло 13 лет [1]. Технологии секвенирования и сборки геномов с годами разивались, и на данный момент проекты по сборке полного генома завершены или близки к завершению для тысяч организмов [2].

Современные технологии секвенирования позволяют получить короткие подпоследовательности генома (риды), которые используются специальными программами для сборки геномов (ассемблерами). Но ассемблеры чаще всего позволяют восстановить не полный геном, а лишь набор его продолжительных фрагментов (контигов).

Одним из шагов от контигов к полному геному является объдинение контигов в более продолжительные последовательности (скаффолды). Но существующие методы построения скаффолдов либо материально затратны, либо недостаточно точны, что вдохновляет на поиски новых решений.

В данной работе исследуется возможность построения скаффолдов с использованием информации о сборках родственных геномов. В отличие от существующих методов, наш алгоритм опирается на свойства брейкпоинт графа для фрагментированных геномов и информацию о повторах, что позволяет решать поставленную задачу более эффективно.

Диссертация состоит из трёх глав. В первой главе описывается задача построения скаффолдов и существующие методы решения. Вторая глава содержит описание разработанного алгоритма: его базовой версии и расширения. Третья глава посвящена экспериментам по тестированию обеих версий алгоритма, а также сравнению с инструментом Ragout.

Глава 1

Постановка задачи и существующие методы решения

1.1. Задача построения скаффолдов

В каждой клетке любого живого организма содержится генетическая информация, называемая геномом. Геном представляет из себя набор из одной или более хромосом — двухцепочечных молекул ДНК (дизоксирибонуклеиновой кислоты). Каждая цепочка ДНК состоит из последовательности нуклеотидов четырех типов: A, C, G и T. Причем, цепи ДНК комплементарны, и между ними существует взаимно однозначное соответсвие. Напротив каждого нуклеотида одной цепи находится комплементарный ему нуклеотид другой цепи (пары комплементарных нуклеотидов — A и T, C и G). Поэтому часто хромосома рассматривается как последовательность нуклеотидов одной цепи.

Прочитать целиком весь геном на данный момент технически невозможно. Текущие методы секвенирования позволяют считывать только сравнительно короткие фрагменты, называемые *ридами*. Однако, существуют программы, которые решают задачу реконструкции исходной последовательности по перекрывающимся ридам. Такие программы называются *ассемблерами*. К сожалению, ассемблеры в подавляющем большинстве случев не способны восстановить всю последовательность целиком, и результатом их работы является набор продолжительных подпоследовательностей генома, называемых *контигами*. Набор полученных ассемблером контигов называется *сборкой*.

В зависимости от метода секвенирования, покрытия генома ридами и количества продолжительных похожих участков (*повторов*), число полученных контигов, а соответсвенно и их продолжительность, варьируется. Но в большинстве случаев число контигов значительно превосходит число хромосом в

организме, а их продолжительности недостаточно для проведения полноценного анализа организма на молекулярном уровне.

Продолжительность контигов может быть эффективно увеличена с помощью объединения контигов в правильном порядке. Более продолжительная последовательность нуклеотидов, полученная в результате объединения нескольких контигов, называется *скаффолдом*. А задача получения из набора контигов набора скаффолдов, называется *задачей построения скаффолдов*.

1.2. Методы построения скаффолдов

Среди решений задачи построения скаффолдов можно условно выделить два класса:

1. Методы, использующие дополнительные данные секвенирования
2. Методы, использующие информацию о родственных геномах

Под дополнительными данными секвенирования подразумеваются длинные риды PacBio и парные риды с большим расстоянием вставки. Данные технологии появились не так давно, но уже достаточно широко используются как для уточнения сборки контигов ассемблером: [3], ABySS [4], SGA [5], SOAPdenovo2 [6], так и для отдельних программ построения скаффолдов: Bambus2 [7 MIP [8], Opera[9], SCARPA [10], SOPRA [11], SSPACE [12].

Главным минусом методов данного класса являются высокие материальные затраты на получение дополнительных данных секвенирования. При этом на разнообразных датасетах ни один из наиболее популярных методов этого класса не совершил более 90% правильных объединений контигов в скаффолды [13].

Методы второго класса используют один или более родственных геномов (референсов). Для очень близких геномов до сих пор продолжает использоваться метод построения скаффолдов с помощью выравнивания контигов на

референс и соответсвующего их упорядочивания [14]. Однако, при наличии структурных вариаций между геномами, этот метод приводит к многочисленным ошибкам.

В 2006 году была поставлена и решена задача упорядочивания контигов таким образом, чтобы кратчайший сценарий перестроек между референсом и скаффолдами был минимален [15]. При тестировании реализаций данного подхода было обнаружено, что он также склонен к многочисленным ошибкам в случае крупных геномных перестроек [16, 17].

Совсем недавно стали появляться методы, использующее более одного родственного генома для построения скаффолдов, такие как RACA [18] и Ragout [19]. Однако оба инструмента, хотя и позволяют загружать в качестве референсов наборы контигов, осуществляют построение скаффолдов лишь для одого генома-таргета, в то время как при наличии нескольких фрагментированных геномов одновременная сборка более эффективна. Кроме того, ни один из существующих методов не использует топологию брейкпоинт графа и информацию о повторах, полученных от ассемблера, что делает подход, описанный в данной работе, оригинальным.

Глава 2

Алгоритм построения скаффолдов

2.1. Брейкпоинт граф

2.1.1. Представление генома в виде множества знаковых строк

Еще на заре геномики было выявлено, что в геноме существуют консервативные районы, которые сохраняются с небольшими изменениями на протяжении десятков миллионов лет [20]. Но в геномах различных видов живых организмов эти участки имеют разный порядок следования. Зачастую такие участки являются генами — последовательностями ДНК, кодирующими функционально значимую для организма информацию. Именно по причине функциональной значимости генов, точечные мутации в генах и разрезание генов в результате геномных перестроек реже сохраняется внутри популяций. В данной работе, для краткости, будем называть такие консервативные участки генами.

Существование генов позволяет перейти в способе представления хромосомы от строки из нуклеотидов к строке из генов. Но так как ДНК имеет двухцепочечную структуру, и гены могут располагаться на разных цепях, то биологически более верным является представление хромосомы (или контига) в виде знаковой строки. Знак «+» будет соответствовать расположению гена на прямой цепи, а знак «-» — на обратной. Соотвественно, геном состоящий из более чем одной хромосомы, представим в виде множества знаковых строк. В дальнейшем мы будем считать, что геномы нам даны именно в таком виде.

2.1.2. Брейкпоинт граф

Определение брейкпоинт графа было введено Ханненхалли и Певзнером в 1995 году [21].

Хотя классическое определение описывает брейкпоинт граф для кольцевой

хромосомы, данное понятие несложно расширить и для генома, состоящего из линейных хромосом.

Пусть дан геном G в виде знаковой строки над алфовитом генов Γ. Для каждого гена $X \in \Gamma$ определим его концы: голову X^h и хвост X^t. Положим, что u *смежно* v, если концы генов u и v следуют непосредственно друг за другом в какой-либо из хромосом. Тогда *Брейкпоинт граф* $BG(G)$ — граф на множестве вершин $V = \{X^h, X^t | X \in \Gamma\} \cup \{\infty\}$ с ребрами $E = \{(u, v) | u$ *смежно с* $v\} \cup \{(u, \infty) | u$ *совпадает с концом хромосомы*$\}$. Вершину ∞ будем называть *иррегулярной вершиной*, а все остальные — *регулярными вершинами*. Соответсвенно, ребра, соединяющие пары регулярных вершин, будем называть *регулярными ребрами*, а инцедентные иррегулярной вершине — *иррегулярными ребрами*.

Пример брейкпоинт графа изображен на рисунке 2.1

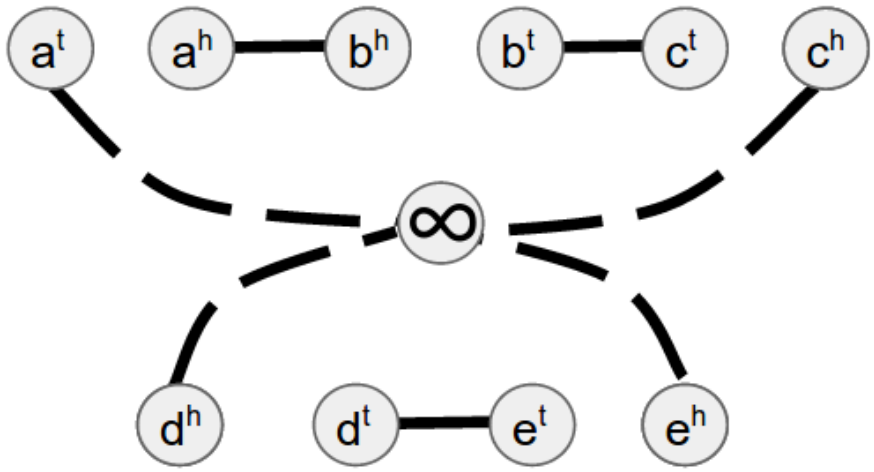


Рис. 2.1. Брейкпоинт граф для генома из двух линейных хромосом $G = \{(+a\ -b\ +c), (-d\ +e)\}$. Регулярные ребра изображены сплошной линией, а иррегулярные — пунктирной.

2.1.3. Множественный брейкпоинт граф

Для изучения геномных престроек, произошедших между несколькими геномами, в работе [22] было введено понятие *множественного брейкпоинт графа*.

Для k геномов $G_1, ..., G_k$ в виде знаковых строк над алфавитом из N различных генов, множественный брейкпоинта граф состоит из $2N$ регулярных вершин, соответствующих концам генов, одной *иррегулярной* вершины ∞ и ребер k цветов. Регулярное ребро цвета C_i проводится между регулярными вершинами u и v, если концы генов u и v смежны в геноме G_i, а иррегулярное ребро цвета $_i$ проводится между регулярной вершиной u и иррегулярной вершиной ∞, если конец гена u также является концом хромосомы или контига в геноме G_i (смотри рисунок 2.2).

Ребра нескольких цветов, соединяющих одни и те же вершины, образуют *мультиребро*, которое характеризуется *мультицветом* Q — множеством цветов, которые присутствуют в этом мультиребре.

Во множественном брейкпоинт графе *компоненту связности* удобно определить как максимальное множество регулярных вершин, такое, что между двумя любыми вершинами существует путь из регулярных ребер.

2.1.4. T-консистентный мультицвет

В работе [23] вводится понятие *T-консистентного мультицвета*.

Пусть T — филогенетическое дерево, листьями которого являются геномы $G_1, ..., G_k$, а мультицвет Q целиком состоит из всех цветов, соответсвующих геномам-листьям какого-либо поддерева T' дерева T. Тогда цвет Q и его дополнение $\overline{Q}$ оба являются T-консистентными (смотри рисунок 2.3).

Заметим, что мультиребро цвета Q из этого определения будет соответствовать смежности в предковом геноме, являющемся корнем поддерева T'. В то время как мультиребро цвета $\overline{Q}$ соответсвует смежности в геноме-предке,

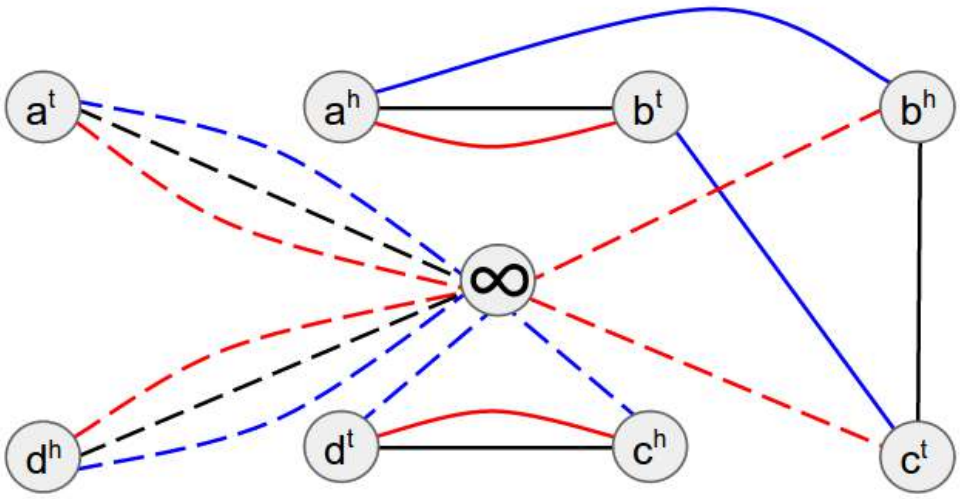


Рис. 2.2. Множественный breakpoint граф для $G_1 = \{(+a\ +b\ +c\ +d)\}$, $G_2 = \{(+a\ +b), (+c\ +d)\}$, $G_3 = \{(+a\ -b\ +c), (+d)\}$. Цвета ребер соответсвуют цветам геномов: C_1 — черный, C_2 — красный, C_3 — синий. Регулярные ребра изображены сплошными линиями, иррегулярные — пунктирными.

являющемся корнем всего дерева T.

2.1.5. Операция объединения контигов

Рассмотрим последствия фрагментации одной хромосомы в геноме G на два контига. Для каких-то смежных концов генов x и y мы теряем информацию о том, что они смежны. В результате, в брейкпоинт графе не хватает ребра цвета i между вершинами u и v, соответсвующими концам генов x, y. При этом данные концы генов оказываются концами двух разных контигов, что приводит к появлению иррегулярных ребер цвета i, инцедентных u и v.

Логично, что операция объединения этих двух контигов должна производить обратные действия.

Определим операцию $scaffold(e_1, e_2, Q)$ для пары иррегулярных мультиребер $e_1 = (u, \infty, Q)$, $e_2 = (v, \infty, Q)$ и мультицвета Q в брейкпоинт графе BG как следующие преобразования:

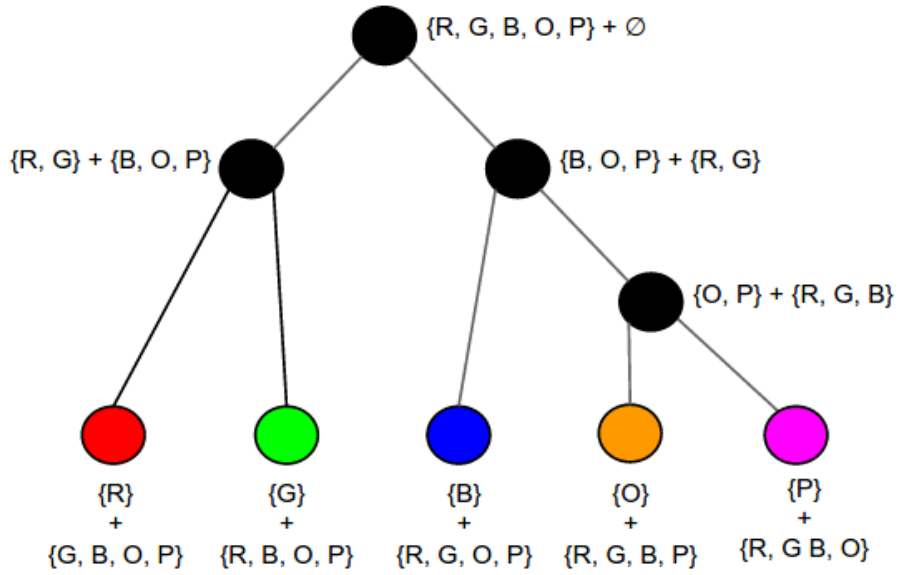


Рис. 2.3. Филогенетическое дерево T для пяти геномв, имеющих цвета: красный (R), зеленый (G), синий (B), оранжевый (O) и розовый (P). Каждая вершина подписана парой комплиментарных T-консистентных мультицветов.

- $E(BG') = E(BG) \cup (u, v, Q)$ — добавление *мультиребра сборки* мультицвета Q между u и v
- $E(BG') = E(BG) \setminus \{e_1\}$ — удаление иррегулярного мультиребра e_1
- $E(BG') = E(BG) \setminus \{e_2\}$ — удаление иррегулярного мультиребра e_2

Заметим, что Q — мультицвет и может состоять из более чем одного цвета, что позволяет с помощью этой операции объединять пары контигов сразу в нескольких геномах. Также отметим, что мультицвет Q должен быть подмножеством мультицвета, состоящего из цветов геномов-сборок. В противном случае, операция $scaffold$ затронет концы хромосом полных геномов. Пример применения операции $scaffold$ изображен на рисунке 2.4.

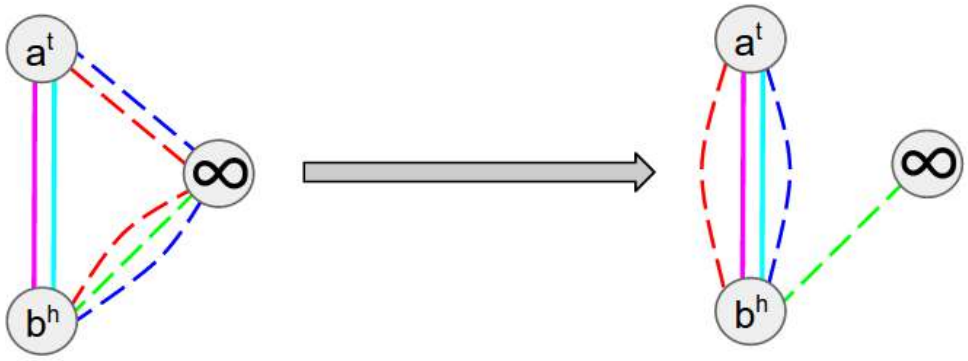


Рис. 2.4. Фрагмент брейкпоинт графа до и после применения операции $scaffold(e_1, e_2, Q)$ для иррегулярных мультиребер $e_1 = (a^t, \infty), e_2 = (b^h, \infty)$ и мультицвета $Q = \{\text{красный, синий}\}$.

2.2. Базовый алгоритм

Данный раздел посвящен базовому алгоритму построения скаффолдов.

2.2.1. Обзор базового алгоритма

Базовый алгоритм принимает на вход филогенетическое дерево T и наборы знаковых строк, представляющие его листья: полные геномы $G_1, \ldots, G_l$ и сборки $A_1, \ldots, A_m$. После того, как для полученных геномов построен множественный брейкпоинт граф, мы генерируем все $2^m - 1$ возможных непустых мультицветов из цветов $C_1, \ldots, C_m$, соответсвующих сборкам $A_1, \ldots, A_m$. Как уже упоминалось ранее, применение операции $scaffold$ имеет смысл на ребрах именно таких мультицветов, которые мы сгенерировали.

Далее, наш алгоритм для каждого мультицвета ищет пары иррегулярных ребер, на которых «выгодно» применить операцию $scaffold$, и применяет ее. Причем мультицвета перебираются в порядке убывания числа цветов в них.

Выбор пар иррегулярных ребер для объединения двух контигов осуществляется в два этапа. Сначала мы сужаем область поисков до нескольких локальных компонент связности, а затем в каждой из них выбираем наилучших

кандидатов с помощью весовой функции. Подробнее эти этапы будут описаны после обзора.

После того, как изначальный брейкпоинт граф был преобразован чередой операций $scaffold$, алгоритм восстанавливает по нему наборы знаковых строк для сборок, содержащие объединенные в скаффолды контиги.

В итоге на выходе мы получаем наборы скаффолдов в виде знаковых строк для собираемых геномов.

Псевдокод алгорима приведен ниже.

$ConstructScaffolds(T, G_1, \ldots, G_k, A_1, \ldots, A_m)$

$BG \leftarrow ConstructBreakpointGraph(G_1, \ldots, G_k, A_1, \ldots, A_m)$
$Mcolors \leftarrow GenerateSubsets(\{C_1, \ldots, C_m\})$
$Sort(Mcolors)$
for $Q \in Mcolors$ **do**
    $CCs \leftarrow Filter(BG)$
    for $CC \in CCs$ **do**
        $ScaffoldPoints \leftarrow GetScaffoldPoints(CC)$
        for $(e_1, e_2) \in ScaffoldPoints$ **do**
            $scaffold(e_1, e_2, Q)$
$A'_1, \ldots, A'_m \leftarrow ConstructSignedStrings(BG)$
return $A'_1, \ldots, A'_m$

2.2.2. Сужение области поисков

Для каждого рассматриваемого мультицвета Q сужение области поисков реализуется с помощью удаления неинтересующих нас мультиребер, поэтому производится над временной копией брейкпоинт графа.

В первую очередь из графа удаляются иррегулярные мультиребра мультицветов, не являющихся надмножествами мультицвета Q. Очевидно, что данные ребра не представляют для нас интереса, так как мы не сможем применить на них операцию $scaffold$ для мультицвета Q.

Также мы удаляем мультиребра мультицветов, содержащих цвета полных

геномов, так как не хотим подвергать объединению контиги, концы которых совпадают с концами хромосом в других геномах.

Далее, для оставшихся иррегулярных ребер осуществляется поиск *мультиребер поддержки*. Мультиребром поддержки для пары иррегулярных мультиребер $e_1 = (u, \infty)$ и $e_2 = (v, \infty)$ называется мультиребро (u, v), присутсвующее в брейкпоинт графе еще до применения на этих мультиребрах операции $scaffold$. Такое ребро гарантирует, что хотя бы в одном геноме есть такая смежность, и тем самым «поддерживает» ребро сборки. Исходя из того, что фрагментация геномов в местах уникальных смежностей хотя и возможна, но менее вероятна, мы ставим необходимым условием для применения операции $scaffold$ наличие ребра поддержки.

После того как найдены мультиребра поддержки, удаляются все остальные регулярные мультиребра. Это позволяет разбить брейкпоинт граф на более мелкие компоненты связности. Очевидно, что пара иррегулярных мультиребер, лежащих в различных компонентах связности, не имеет ребра поддержки. Поэтому дальше мы сможем искать наилучших кандидатов в каждой компоненте связности отдельно.

2.2.3. Выбор лучших кандидатов

Как было заявлено ранее, выбор лучших кандидатов в каждой полученной в результате сужения области поиска компоненте связности осуществляется с помощью весовой функции. Перед тем как ее определить, введем вспомогательную величину.

Определим $Tcons(Q)$ как мощность минимального разбиения мультицвета Q на T-конситсентные мультицвета. При отсутсвии вставок и удалений, если мультиребро e имеет мультицвет Q, то $Tcons(Q)$ определяет число геномов, для которых соответсвующая мультиребру e смежность является приобретенной, а не унаследованной от ближайшего генома-предка (если в геноме-корне филогенетического дерева считать все смежности приобретенными).

То есть, если мультиребро e имеет мультицвет Q, такой что $Tcons(Q) = n, n > 1$, это значит, что соответсвующая мультиребру e смежность независимо возникла в результате перестроек на n различных ветвях филогенетического дерева. И хотя возникновение одинаковых смежностей на нескольких различных ветвях филогенетического дерева возможно, оно является менее вероятным событием. То есть, чем больше величина $Tcons(Q)$ у мультицвета Q мультиребра e, тем менее вероятно его присутствие в настоящем брейкпоинт графе для собранных геномов.

Поэтому в качестве «хорошести» применения функции $scaffold(e_1, e_2, Q)$ именно для иррегулярных мультиребер $e_1 = (u, \infty)$, $e_2 = (v, \infty)$ и мультицвета Q, мы определим для возникающего ребра сборки весовую функцию $Score(u, v, Q)$ как изменение суммы величин $Tcons$ для мультицветов трех мультиребер, которые затрагиваются операцией $scaffold(e_1, e_2, Q)$. А для того, чтобы симулировать ситуацию отсутсвия вставок и удалений, будем дополнять интересующие нас мультицвета недостающими из-за них цветами.

Введем ряд обозначений:

Q_1 — мультицвет иррегулярного мультиребра $e_1 = (u, \infty)$

Q_2 — мультицвет иррегулярного мультиребра $e_2 = (v, \infty)$

Q_s — мультицвет мультиребра поддержки $e_s = (u, v)$

C_1 — мультицвет, состоящий из цветов всех геномов, в которых нет ребер, инцедентных вершине u

C_2 — мультицвет, состоящий из цветов всех геномов, в которых нет ребер, инцедентных вершине v

$C = C_1 \cap C_2$ — мультицвет, которым мы будем дополнять мультицвет ребра поддержки e_s

$C_a = C_1 \setminus C$ — мультицвет, которым мы будем дополнять мультицвет иррегулярного ребра e_1

$C_b = C_2 \setminus C$ — мультицвет, которым мы будем дополнять мультицвет иррегулярного ребра e_2

Тогда до и после применения операции $scaffold(e_1, e_2, Q)$ мы имеем:

$$beforeScaffold(u, v) = Tcons(Q_1 \cup C_a) + Tcons(Q_2 \cup C_b) + Tcons(Q_s \cup C)$$

$$afterScaffold(u, v, Q) = Tcons(Q_1 \cup C_a \setminus Q) + Tcons(Q_2 \cup C_b \setminus Q) + Tcons(Q_s \cup C \cup Q)$$

А весовая функция для возникающего ребра сборки (u, v) и мультицвета Q определяется разницой данных величин:

$$Score(u, v, Q) = beforeScaffold(u, v) - afterScaffold(u, v, Q)$$

Наш алгоритм, работая с мультицветом Q, вычисляет $Score(u, v, Q)$ для всех возможных мультиребер сборки в каждой отдельной компоненте связности. Мы не рассматриваем мультиребра сборки, для которых $Score(u, v, Q) \leq 0$, так как их присутствие в настоящем брейкпоинт графе собранных геномов не является более вероятным, чем их отсутсвие.

Так как мы не можем применить операцию $scaffold$ к одному и тому же иррегулярному ребру в паре с более чем одним другим иррегулярным мультиребром, мультиребра сборки для выбранных пар должны представлять из себя паросочетание.

Процедура выбора наилучших кандидатов строит в отдельной компоненте свзяности наибольшее паросочетание максимального веса из мультиребер сборки и возвращает соответсвующие им пары иррегулярных мультиребер. Тем самым, применяя к ним операцию $scaffold$, мы объединяем как можно больше контигов наиболее вероятным образом.

2.3. Расширенный алгоритм

2.3.1. Информация о повторах

Одной из основных причин фрагментации геномов на контиги в сборках являются продолжительные похожие участки. Повторы приводят к тому, что в графе перекрытий или графе де Брюйна (в зависимости от ассемблера) степени

некоторых вершин превосходят двойку. Такие вершины называются *ветвящимися*. Из-за них одназначное восстановление пути в графе, которому соответсвует геном, становится затруднительным. Даже несмотря на многочисленные подходы ассемблеров к разрешению повторов, некоторые повторы остаются неразрешенными, и в таких случаях ассемблеры выдают в качестве контигов наборы однозначных путей между вевтящимися вершинами. Однако при выводе контигов информация о повторах, ограничивающих контиги, теряется. Тогда как она могла бы быть эффективно использована для построения скаффолдов. Так, обладая информацией о том, что в графе после контига следует повтор $repX$, можно сузить множество кандидатов для объединения с ним до множества контигов, перед которыми следует повтор $repX$. Учитывая, что информация о повторах на концах контигов без каких-либо дополнительных вычислений может быть получена от ассемблера, мы реализовали расширенную версию нашего алгоритма постреления скаффолдов, использующую ее.

2.3.2. Расширение с помощью информации о повторах

Расширенный алгоритм построения скаффолдов так же, как и базовый, принимает на вход геномы в виде наборов знаковых строк. Однако знаковые строки для некоторых контигов из геномов-сборок будут содержать добавочную информацию. Если от ассемблера известно, что повтор $repX$ предшествует контигу C, то у его начала будет поставлена метка $repX$. А если повтор $repX$ следует после контига C, то метка $repX$ будет поставлена у его конца. Схематичный фрагмент сжатого графа де Брюйна и полученных по нему знаковых строк с метками изображены на рисунке 2.5.

Для геномов в таком виде можно построить *расширенный множественный брейкпоинт граф*. Такой граф будет состоять из таких же регулярных вершин и ребер, как и обычный. Но также в нем будет присутствовать по паре иррегулярных вершин $repX^h$ и $repX^t$ для каждого повтора $repX$. Пометке $repX$ у начала контига будет сопоставляться иррегулряное ребро между вер-

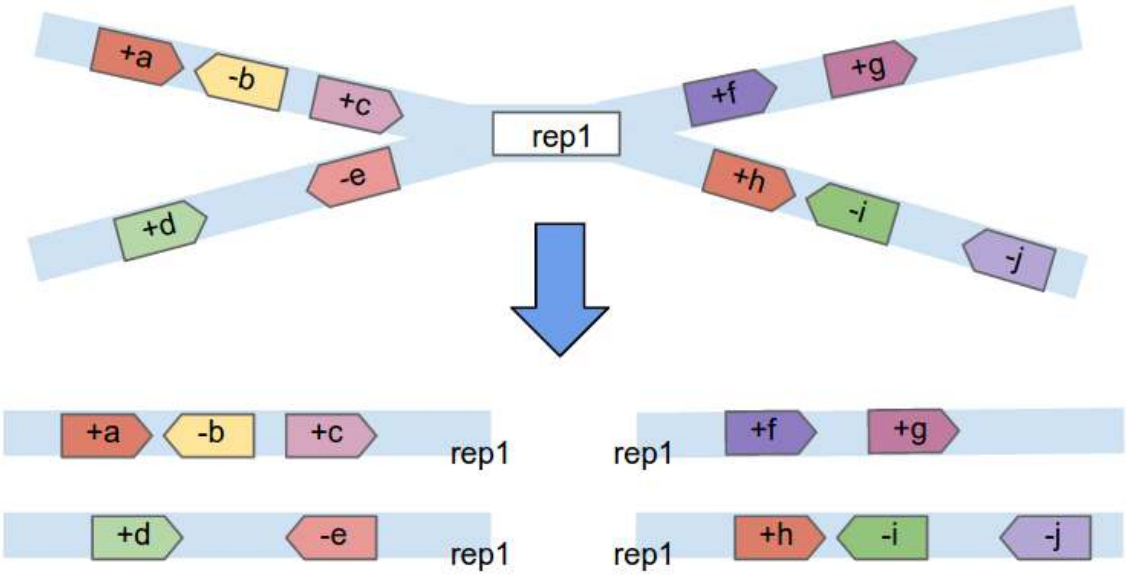


Рис. 2.5. Вверху схематически изображен фрагмент графа де Брюйна для некоторого генома. Из-за повтора $rep1$ ассемблер не может однозначно восстановить путь и потому выдает для этого фрагмента четыре контига, изображенные ниже. Знаковые перестановки с метками для этих контигов будут иметь следующий вид: $(+a\ -b\ +c)_{rep1}$, $(+d\ -e)_{rep1}$, $_{rep1}(+f\ +g)$, $_{rep1}(+h\ -i\ -j)$.

шиной $repX^t$ и вершиной u, соответсвующей концу гена в начале контига, а пометке $repX$ у конца контига будет сопоставляться иррегулярное ребро между вершиной $repX^h$ и вершиной u, соответсвующей концу гена в конце контига. Для концов хромосом и контигов без пометок по-прежнему будет проводиться иррегулярное ребро в вершину ∞. Таким образом, мы по сути расщепляем единственную иррегулярную вершину ∞ обычного множественного брейкпоинт графа на несколько помеченных повторами иррегулярных вершин. Пример расширенного множественного брейкпоинт графа приведен на рисунке 2.6.

В расширенной версии алгоритма для объединения контигов используется та же операция $scaffold(e_1, e_2, Q)$, однако на пару иррегулярных мультиребер e_1, e_2 будет наложено дополнительное условие *совместимости*: либо $e_1 = (u, repX^h), e_2 = (v, repX^t)$, либо $e_1 = (u, repX^t), e_2 = (v, repX^h)$. Это условие позволяет объединять только те контиги, которые разделены одним и тем же повтором $repX$ в графе перекрытий. Соответсвенно этому условию,

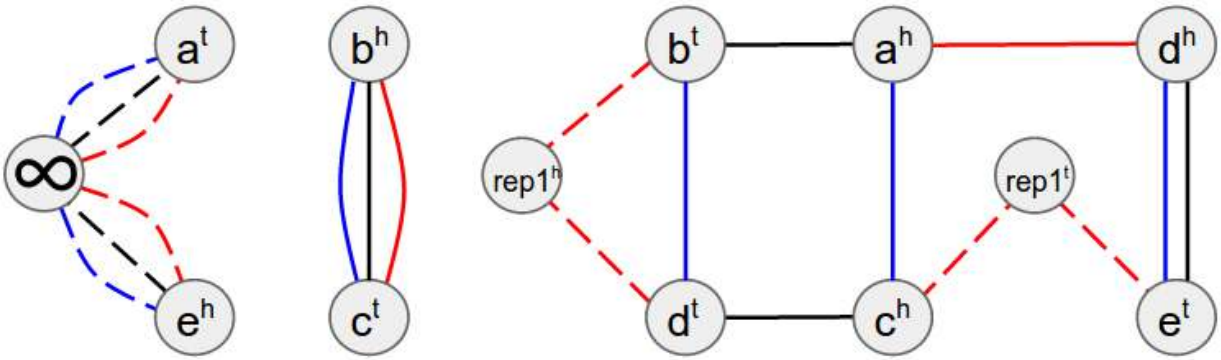


Рис. 2.6. Расширенный множественный брейкпоинт граф для полных геномов $G_1 = \{(+a\ +b\ +c\ +d\ +e)\}$, $G_2 = \{(+a\ -c\ -b\ +d\ +e)\}$ и сборки $G_3 = \{(+a\ -d)_{rep1},\ _{rep1}(-c\ -b)_{rep1},\ _{rep1}(+e)\}$.

понятие мультиребра поддержки тоже будет существовать только для пары совместимых иррегулярных мультиребер.

Все рассуждения для базового алгоритма остаются верными и для расширенного, поэтому сужение области поиска и выбор лучших кандидатов происходит практически так же. Единственное отличие в работе расширенного алгоритма состоит в том, что операция $scaffold$ может быть применена только к паре совместимых иррегулярных ребер. Так, после сужения области поиска, регулярное ребро останется в графе, только если инцедентные его вершинам два иррегулярных ребра будут совместимыми (иначе оно не будет ребром поддержки). И выбор лучших кандидатов будет происходить только среди пар совместимых иррегулярных мультиребер.

Глава 3

Экспериментальные результаты

Первым алгоритмом построения скаффолдов с помощью нескольких референсных геномов был алгоритм RACA. Более поздний инструмент Ragout устраняет многие его недостатки и показывает лучшие результаты [19]. По этой и некоторым техническим причинам в этой главе, помимо теситрования разработанных алгоритмов, проводится сравнение с инструментом Ragout.

Однако стоит отметить, что под использованием инструмента Ragout мы будем подразумевать использование лишь части его функциональсти, решающей задачу построения скаффолдов для геномов в виде наборов знаковых строк. Остальные составляющие Ragout оперируют с геномами в виде нуклеотидных последовательностей, и потому результаты их работы рассматриваться нами не будут.

Эксперименты проводились на реальных геномах, но сборки были симулированы путем фрагментации полных геномов по повторам (см. раздел 3.1).

Обладая полными геномами, для каждого собираемого генома мы оценивали качество построенных для него скаффолдов с помощью следующих двух характеристик:

$$TP = \frac{correct}{all} \times 100\%$$

$$FP = \frac{incorrect}{all} \times 100\%$$

где

all — число пар контигов, которые смежны в геноме

$correct$ — число пар контигов, которые смежны в построенных скаффолдах и смежны в геноме

$incorrect$ — число пар контигов, которые смежны в построенных скаффолдах, но не смежны в геноме

То есть величина TP по сути описывает насколько мы приблизили сборку к правильному полному геному, а величина FP характеризует ошибочность построенных скаффолдов. Тем самым построенные скаффолды тем лучше, чем больше TP и меньше FP.

3.1. Описание данных для тестирования

Для тестирования базового и расширенного алгоритмов, а также сравнения с инструментом Ragout, мы подготовили два частично пересекающихся датасета: «Млекопитающие» и «Приматы».

Датасет «Млекопитающие» состоит из геномов семи млекопитающих:

1. Человек (*Homo sapiens*)
2. Шимпанзе (*Pan troglodytes*)
3. Крыса (*Rattus norvegicus*)
4. Мышь (*Mus musculus*)
5. Собака (*Canis familiaris*)
6. Кошка (*Felis catus*)
7. Опоссум (*Monodelphis domestica*)

А датасет «Приматы» состоит из геномов шести приматов:

1. Человек (*Homo sapiens*)
2. Шимпанзе (*Pan troglodytes*)
3. Горилла (*Gorilla gorilla*)
4. Орангутанг (*Pongo abelii*)

5. Макака (*Macaca mulatta*)

6. Мартышка (*Callithrix jacchus*)

На рисунках 3.1, 3.2 представлены филогенетические деревья для организмов данных датасетов.

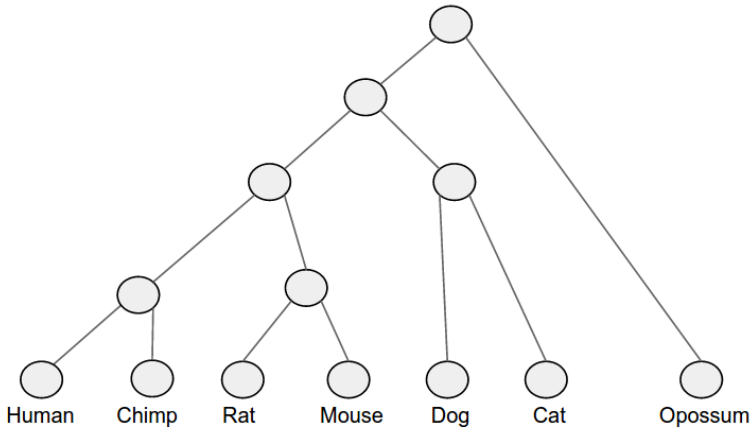


Рис. 3.1. Филогенетическое дерево для датасета «Млекопитающие».

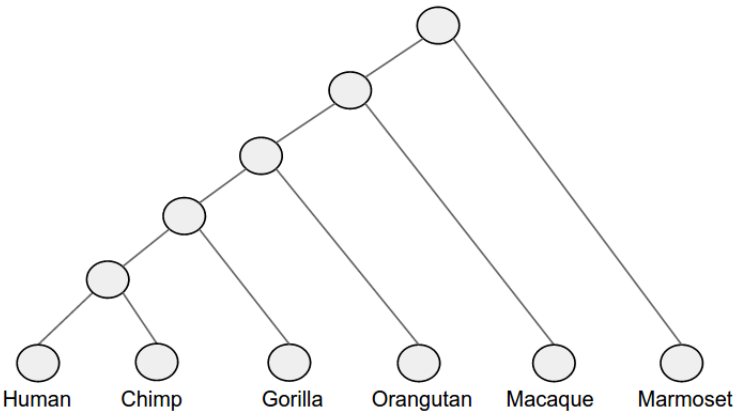


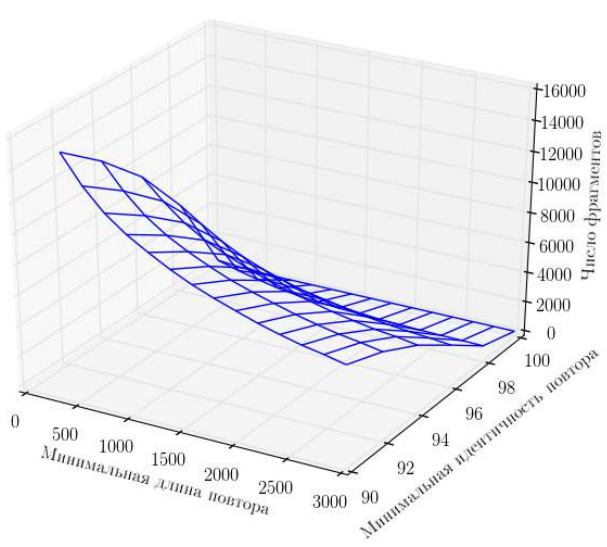
Рис. 3.2. Филогенетическое дерево для датасета «Приматы».

Используя инструмент Ensembl BioMart [24], мы получили для каждого датасета: полные геномы, координаты генов, информацию о гомологичности генов внутри каждого генома (информацию о паралогах), информацию о гомологичных генах для каждой пары геномов (информацию об ортологах). С

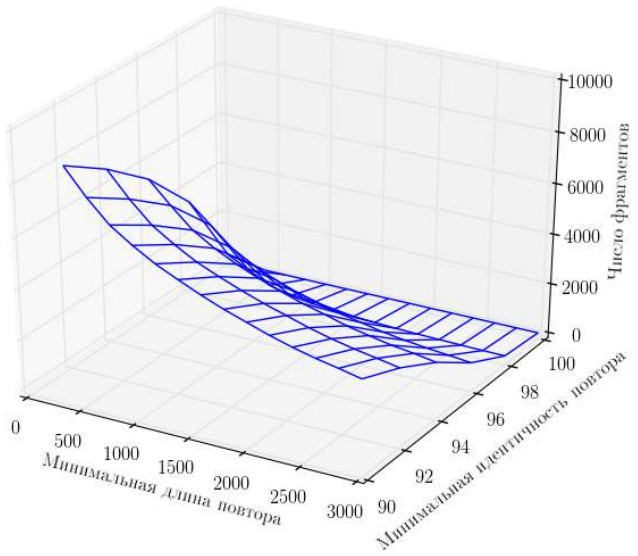
помощью полученной информации мы смогли представить каждый геном в виде набора знаковых строк (по одной знаковой строке для каждой хромосомы). Знаковые строки геномов датасета «Млекопитающие» состояли из 77806 различных генов, а знаковые строки датасета «Приматы» — из 57679.

Как было упомянуто в предыдущей главе, ассемблеры чаще всего фрагментируют геномы в местах повторов. Мы хотим симулировать сборку, фрагментируя геномы таким же образом, что позволит не только приблизить симулированные данные к реальным, но и получить ту информацию о повторах, которая используется расширенным алгоритмом . Для обнаружения повторов, воспользуемся инструментом RepeatMasker [25]. Данный инструмент находит участки входной нуклеотидной последовательности, достаточно похожие на какой-либо из известных повторов, занесенных в такие базы данных, как Dfam и Repbase. Выходные данные представляют собой список найденных участков, для каждого из которых определены координаты в исходной последовательности, идентификатор повтора из базы данных, а также проценты вставок, удалений и замен нуклеотидов. Для каждого участка мы вычисляем его длину l как модуль разности координат и *идентичность ident* как процент совпадающих нуклеодитов.

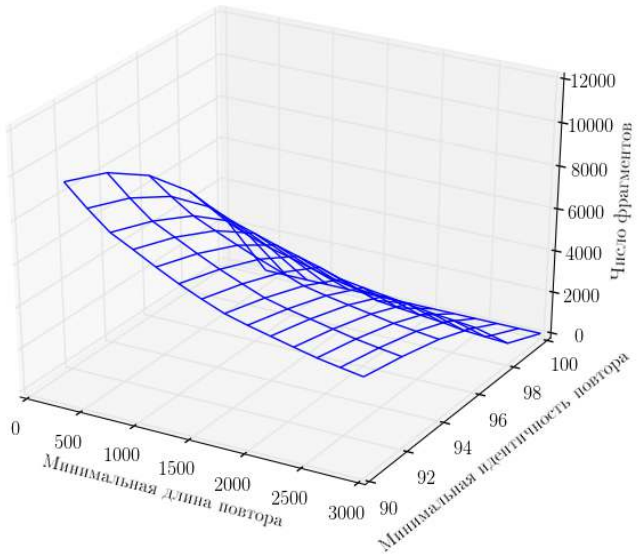
Фрагментация генома будет производиться по двум параметрам: минимальной длине рассматриваемых повторов l_{min} и их минимальной идентичности $ident_{min}$. То есть фрагментация для заданных параметров будет осуществляться по повторам, для которых $l \geq l_{min}$ и $ldent \geq ident_{min}$. В своих экспериментах мы будем рассматривать фрагментацию только с параметрами $l_{min} \geq 400$ и $ident_{min} \geq 90$, так как повторы длины меньше 400 разрешаются обычными парными ридами, а повторы с идентичностью меньше 90% достаточно разные, чтобы быть различенными ассемблером.



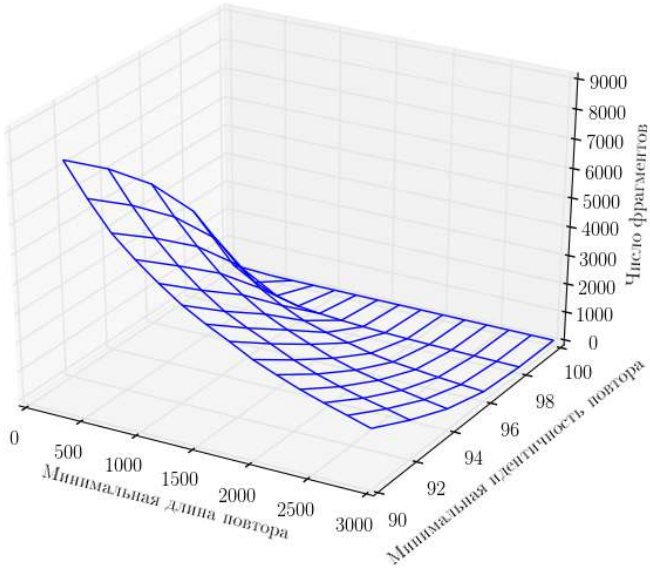
(*а*) Человек



(*б*) Шимпанзе



(*в*) Крыса



(*г*) Горилла

Рис. 3.3. Графики зависимости числа фрагментов в симулированной сборке от минимальной длины повтора (варьируется от 400 до 3000) и минимальной идентичности повтора (варьируется от 90 до 100) для различных организмов.

.

Исследуя зависимость числа фрагментов в симулированной сборке от параметров l_{min} и $ident_{min}$, мы обнаружили существенные различия для разных организмов. На рисунке 3.3 можно заметить, что при минимальных параметрах число фрагментов в сборках для человека и шимпанзе отличается более,

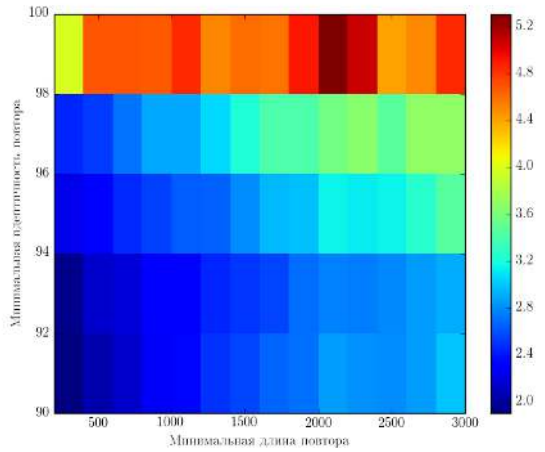
чем в полтора раза. Это скорее всего обусловлено тем, что геном человека более изучен, и не для всех его генов на данный момент были найдены ортологи (полученные нами знаковые строки для человека содержат 35383 генов, не представленных ни в одном из других рассматриваемых геномов, тогда как шимпанзе — только 2893). Однако тенденция уменьшения числа фрагментов с ростом параметров сохраняется для всех организмов.

3.2. Построение скаффолдов для одного генома

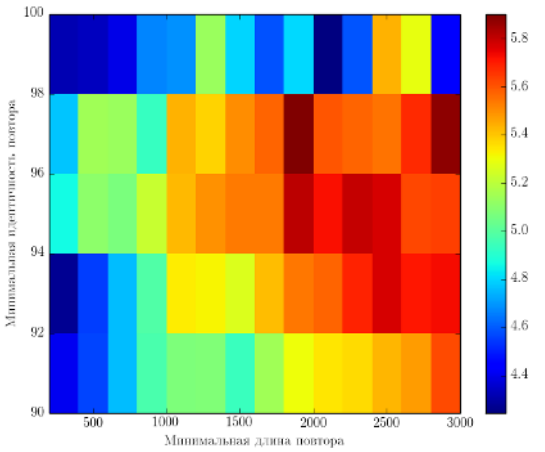
Сначала мы рассматривали случай, когда требуется построить скаффолды только для одного генома. Для датасета «Млекопитающие» мы рассмотрели результаты работы предложенных алгоритмов и Ragout при различных параметрах фрагментации, примененной к *а)* геному человека (см. рисунок 3.4), *б)* геному шимпанзе (см. рисунок 3.5). Все рассматриваемые методы показали для человека результаты гораздо хуже, чем для шимпанзе. Данное различие легко объясняется упоямянутой в предущем разделе разницой в числе симулированных контигов. Так же видно, что величина TP в целом везде выше как у базовового, так и у расширенного алгоритма, чем у Ragout. А пределы, в которых варьируется величина FP, у всех трех алгоритмов примерно одинаковы.

Если всмотреться в результаты построения скаффолдов для шимпанзе, можно отметить еще один важный момент. Чем больше значения параметров фрагментации, а соответственно и меньше число контигов в симулированной сборке, тем сильнее различается величина TP для базового и расширенного алгоритмов. Так, если значение TP у базового алгоритма варьируется от 7.5% до 27.5%, то у расширенного — от 21% до 29%. Из этого можно сделать вывод, что для сильно фрагментированных геномов применение расширенного алгоритма особенно оправдано.

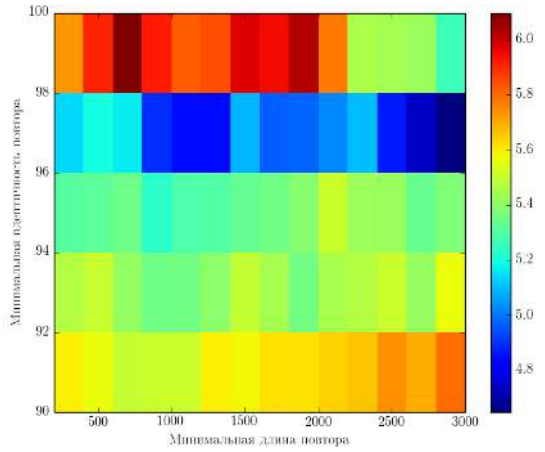
Далее мы исследовали поведение рассматриваемых методов при использовании различных наборов референсных геномов.



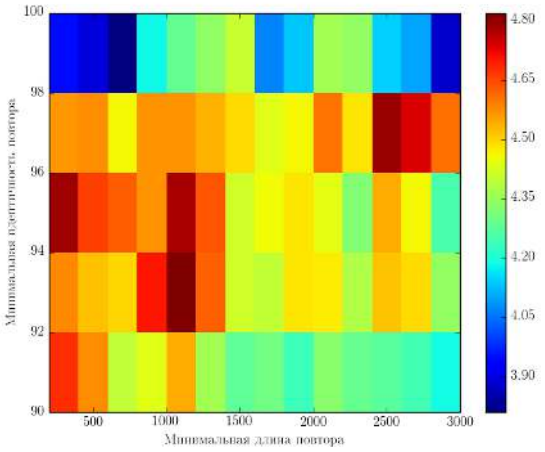
(*а*) Базовый алгоритм: TP



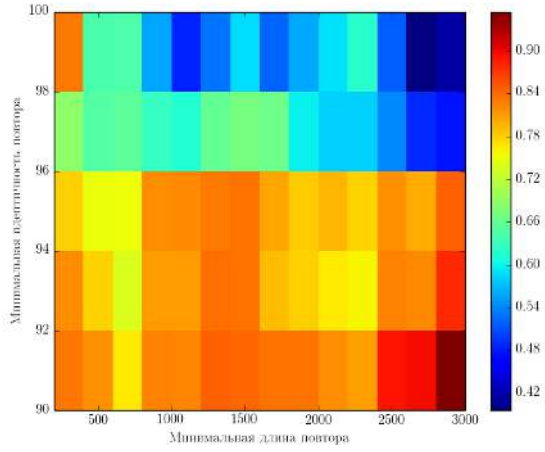
(*б*) Базовый алгоритм: FP



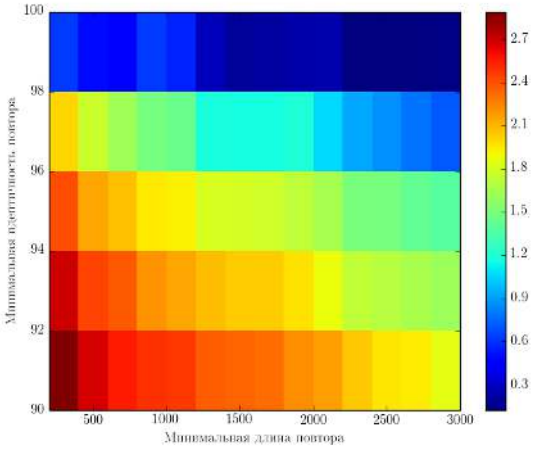
(*в*) Расширенный алгоритм: TP



(*г*) Расширенный алгоритм: FP

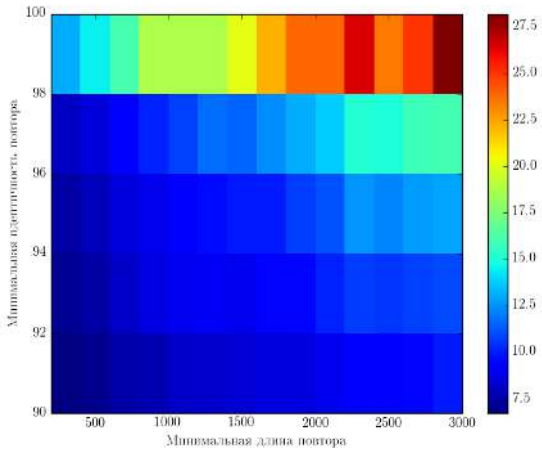


(*д*) Ragout: TP

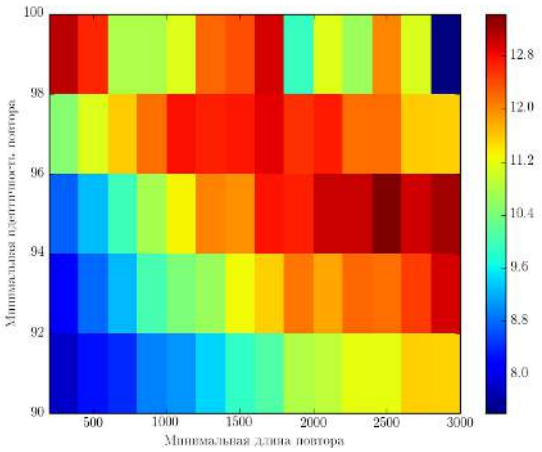


(*е*) Ragout: FP

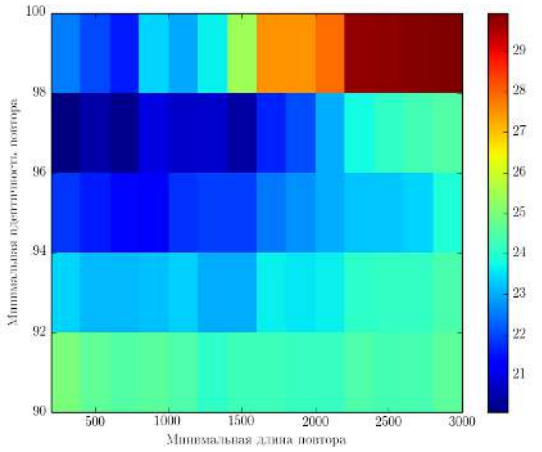
Рис. 3.4. Результаты построения скаффолдов для генома человека на основе контигов, полученных при различных значениях параметров фрагментации. В качестве референсов были использованы остальные геномы датасета «Млекопитающие».



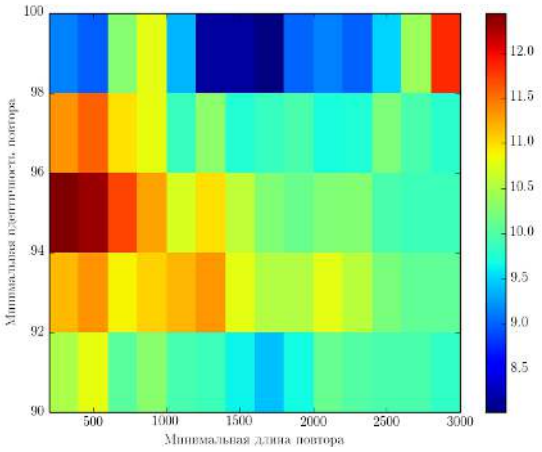
(*а*) Базовый алгоритм: TP



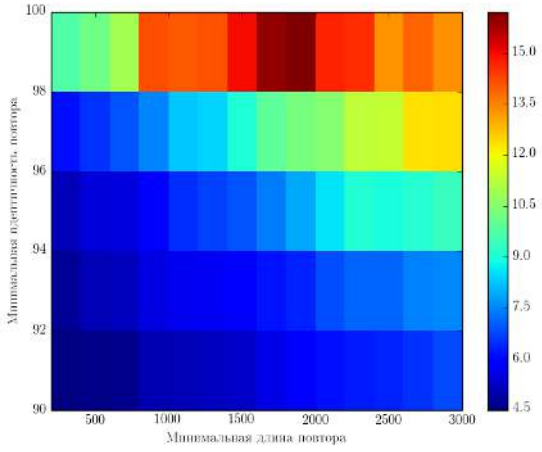
(*б*) Базовый алгоритм: FP



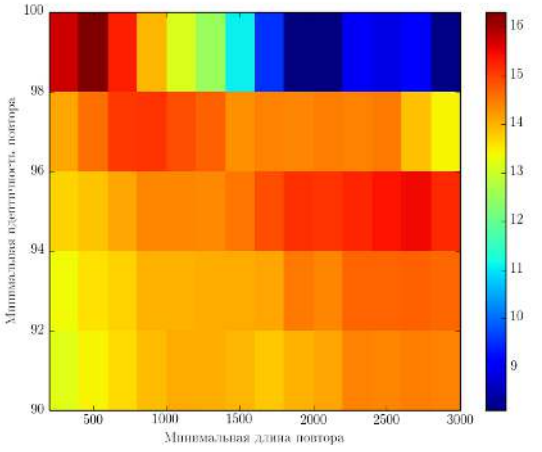
(*в*) Расширенный алгоритм: TP



(*г*) Расширенный алгоритм: FP



(*д*) Ragout: TP



(*е*) Ragout: FP

Рис. 3.5. Результаты построения скаффолдов для генома шимпанзе на основе контигов, полученных при различных значениях параметров фрагментации. В качестве референсов были использованы остальные геномы датасета «Млекопитающие».

Мы симулировали контиги для генома шимпанзе с параметрами фрагментации $l_{min} = 400$, $ident_{min} = 90$, так как такие значения параметров соответствуют самому сложному случаю — фрагментации на наибольшее число контигов. Затем мы запустили рассматриваемые алгоритмы построения скаффолдов для полученных контигов, используя в качестве референсов четыре различных набора геномов:

- 6 млекопитающих — все геномы из датасета «Млекопитающие», кроме шимпанзе
- 4 млекопитающих — все геномы из датасета «Млекопитающие», кроме шимпанзе, кошки и мыши
- 5 приматов — все геномы из датасета «Приматы», кроме шимпанзе
- 3 примата — все геномы из датасета «Приматы», кроме шимпанзе, макаки и мартышки

Результаты данного эксперимента приведены в таблице 3.1.

Сначала опишем наблюдения, касающиеся всех рассматриваемых алгоритмов построения скаффолдов. Как и следовало ожидать, используя в качестве референсов близкородственные геномы из набора «Приматы», удается построить скаффолды лучшего качества (как по TP, так и по FP), нежели используя менее родственные геномы из набора «Млекопитающие». Также был обнаружен менее очевидный факт, что при меньшем числе референсных геномов качество построенных скаффолдов может оказаться, хоть и незначительно, но лучше. Здесь, конечно, большую роль играет то, насколько родственны референсные геномы с собираемым геномом. Мы исключали из наборов референсов геномы не самых близких к шимпанзе организмов, что и могло послужить причиной наблюдаемых результатов. Отсюда можно сделать вывод, что при наличии нескольких полных геномов ближайших родственников собираемого ге-

нома, использование дополнительных геномов менее родственных организмов может пойти во вред качеству скаффолдов.

Что касается различий в результатах для разных методов, то для всех наборов референсов, Ragout проигрывает по обоим характеристикам разработанным нами алгоритмам. Расширенный алгоритм показывает значительное улучшение характеристики TP относительно базового алгоритма, но при этом наблюдается небольшое ухудшение характеристики FP.

Таблица 3.1. Результаты построения скаффолдов для генома шимпанзе при использовании разных наборов геномов в качестве референсных. В качестве сборки шимпанзе были использованы контиги, полученные при параметрах фрагментации $l_{min} = 400$, $ident_{min} = 90$.

	Базовый алгоритм		Расширенный алгоритм		Ragout	
	TP(%)	FP(%)	TP(%)	FP(%)	TP(%)	FP(%)
6 млекопитающих	6.74	7.79	24.98	10.49	4.47	13.21
4 млекопитающих	6.78	7.05	24.58	9.83	4.82	13.52
5 приматов	7.77	5.99	27.91	9.52	6.38	15.86
3 примата	7.96	4.73	28.03	8.23	7.03	16.53

3.3. Построение скаффолдов для нескольких геномов

Также было проведено сравнение разработанных алгоритмов и Ragout для случая, когда требуется построить скаффолды сразу для нескольких геномов.

Мы осуществили фрагментацию с параметрами $l_{min} = 400, ident_{min} = 90$ для трех геномов в датасете «Млекопитающие» (человек, шимпанзе, крыса) и трех геномов в датасете «Приматы» (человек, шимпанзе, горилла).

Для каджого датасета мы получили результаты одновременного построения скаффолдов для трех геномов базовым и расширенным алгоритмами. Так как Ragout не позволяет собирать несколько геномов одновременно, для каждо-

го датасета мы собирали каждый из трех фрагментированных геномов отдельно, передавая два остальных под видом референсов.

Результаты представлены в таблицах 3.2, 3.3. Конечно, качество скаффолдов получилось хуже, чем в случаях, когда фрагментирован был только один геном. Однако отношения величин между разными методами сохранились: худшие результаты у Ragout, а величины TP и FP у расширенного алгоритма больше, чем у базового.

Таблица 3.2. Результаты построения скаффолдов для человека, шимпанзе и крысы на основе контигов, полученных при параметрах фрагментации $l_{min} = 400$, $ident_{min} = 90$. В качестве референсов были взяты остальные геномы датасета «Млекопитающие»

	Базовый алгоритм		Расширенный алгоритм		Ragout	
	TP(%)	FP(%)	TP(%)	FP(%)	TP(%)	FP(%)
Человек	1.59	2.33	2.65	1.39	0.76	2.73
Шимпанзе	5.49	4.97	12.65	4.29	4.17	12.60
Крыса	4.91	5.34	18.69	12.70	3.72	10.33

Таблица 3.3. Результаты построения скаффолдов для человека, шимпанзе и гориллы на основе контигов, полученных при параметрах фрагментации $l_{min} = 400$, $ident_{min} = 90$. В качестве референсов были взяты остальные геномы датасета «Приматы»

	Базовый алгоритм		Расширенный алгоритм		Ragout	
	TP(%)	FP(%)	TP(%)	FP(%)	TP(%)	FP(%)
Человек	1.57	1.57	2.45	1.41	0.93	3.21
Шимпанзе	5.75	3.53	12.88	3.35	6.03	15.35
Горилла	5.35	3.45	12.33	5.72	6.65	15.14

Заключение

В данной работе представлен новый алгоритм построения скаффолдов с помощью нескольких референсных геномов, а также его расширение, опирающееся на дополнительную информацию о повторах. Даже базовая версия алгоритма показывает лучшие результаты, чем его сильнейший конкурент Ragout, а расширенная версия настолько эффективна, что может стать достаточно весомой причиной для разработчиков ассемблеров выводить необходимую информацию о повторах.

Дальнейшее развитие данной работы может включать интеграцию с программой для реконструкции предковых геномов MGRA2, что позволит правильно объединять еще больше пар контигов, применяя разработанный алгоритм к множественному брейкпоинт граф после каждого шага MGRA2.

Другое направление развития состоит в добавлении модулей, благодаря которым будет возможно построение скаффолдов для контигов в виде нуклеотидных последовательностей.

Литература

1. The Human Genome Project. URL: http://web.ornl.gov/sci/techresources/Human_Genome/.

2. Pagani I., Liolios K., Jansson J. et al. The Genomes OnLine Database (GOLD) v.4: status of genomic and metagenomic projects and their associated metadata // Nucleic Acids Research. 2011. — dec. Vol. 40, no. D1. P. D571–D579. URL: http://dx.doi.org/10.1093/nar/gkr1100.

3. Bashir A., Klammer A. A., Robins W. P. et al. A hybrid approach for the automated finishing of bacterial genomes // Nat Biotechnol. 2012. — jul. Vol. 30, no. 7. P. 701–707. URL: http://dx.doi.org/10.1038/nbt.2288.

4. Simpson J. T., Wong K., Jackman S. D. et al. ABySS: A parallel assembler for short read sequence data // Genome Research. 2009. — feb. Vol. 19, no. 6. P. 1117–1123. URL: http://dx.doi.org/10.1101/gr.089532.108.

5. Simpson J. T., Durbin R. Efficient de novo assembly of large genomes using compressed data structures // Genome Research. 2011. — dec. Vol. 22, no. 3. P. 549–556. URL: http://dx.doi.org/10.1101/gr.126953.111.

6. Luo R., Liu B., Xie Y. et al. SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler // GigaScience. 2012. Vol. 1, no. 1. P. 18. URL: http://dx.doi.org/10.1186/2047-217X-1-18.

7. Koren S., Treangen T. J., Pop M. Bambus 2: scaffolding metagenomes // Bioinformatics. 2011. — sep. Vol. 27, no. 21. P. 2964–2971. URL: http://dx.doi.org/10.1093/bioinformatics/btr520.

8. Salmela L., Makinen V., Valimaki N. et al. Fast scaffolding with small independent mixed integer programs // Bioinformatics. 2011. — oct. Vol. 27, no. 23. P. 3259–3265. URL: http://dx.doi.org/10.1093/bioinformatics/btr562.

9. Gao S., Sung W.-K., Nagarajan N. Opera: Reconstructing Optimal Genomic Scaffolds with High-Throughput Paired-End Sequences // Journal of Computational Biology. 2011. — nov. Vol. 18, no. 11. P. 1681–1691. URL: http://dx.doi.org/10.1089/cmb.2011.0170.

10. Donmez N., Brudno M. SCARPA: scaffolding reads with practical algorithms // Bioinformatics. 2012. — dec. Vol. 29, no. 4. P. 428–434. URL: http://dx.doi.org/10.1093/bioinformatics/bts716.

11. Dayarian A., Michael T. P., Sengupta A. M. SOPRA: Scaffolding algorithm for paired reads via statistical optimization // BMC Bioinformatics. 2010. Vol. 11, no. 1. P. 345. URL: http://dx.doi.org/10.1186/1471-2105-11-345.

12. Boetzer M., Henkel C. V., Jansen H. J. et al. Scaffolding pre-assembled contigs using SSPACE // Bioinformatics. 2010. — dec. Vol. 27, no. 4. P. 578–579. URL: http://dx.doi.org/10.1093/bioinformatics/btq683.

13. Hunt M., Newbold C., Berriman M., Otto T. D. A comprehensive evaluation of assembly scaffolding tools // Genome Biol. 2014. Vol. 15, no. 3. P. R42. URL: http://dx.doi.org/10.1186/gb-2014-15-3-r42.

14. Dias Z., Dias U., Setubal J. C. SIS: a program to generate draft genome sequence scaffolds for prokaryotes // BMC Bioinformatics. 2012. Vol. 13, no. 1. P. 96. URL: http://dx.doi.org/10.1186/1471-2105-13-96.

15. Gaul É., Blanchette M. Ordering Partially Assembled Genomes Using Gene Arrangements // Comparative Genomics. Springer Science Business Media, 2006. P. 113–128. URL: http://dx.doi.org/10.1007/11864127_10.

16. Richter D. C., Schuster S. C., Huson D. H. OSLay: optimal syntenic layout of unfinished assemblies // Bioinformatics. 2007. — apr. Vol. 23, no. 13. P. 1573–1579. URL: http://dx.doi.org/10.1093/bioinformatics/btm153.

17. Rissman A. I., Mau B., Biehl B. S. et al. Reordering contigs of draft genomes using the Mauve Aligner // Bioinformatics. 2009. — jun. Vol. 25, no. 16. P. 2071–2073. URL: http://dx.doi.org/10.1093/bioinformatics/btp356.

18. Kim J., Larkin D. M., Cai Q. et al. Reference-assisted chromosome assembly // Proceedings of the National Academy of Sciences. 2013. — jan. Vol. 110, no. 5. P. 1785–1790. URL: http://dx.doi.org/10.1073/pnas.1220349110.

19. Kolmogorov M., Raney B., Paten B., Pham S. Ragout–a reference-assisted assembly tool for bacterial genomes // Bioinformatics. 2014. — jun. Vol. 30, no. 12. P. i302–i309. URL: http://dx.doi.org/10.1093/bioinformatics/btu280.

20. Graur D., Li W.-H. Fundamentals of Molecular Evolution. Sinauer Associates, 2000. ISBN: 0878932666.

21. Hannenhalli S., Pevzner P. A. Transforming men into mice (polynomial algorithm for genomic distance problem) // Proceedings of the 36th Annual Symposium on Foundations of Computer Science. FOCS '95. Washington, DC, USA: IEEE Computer Society, 1995. URL: http://portal.acm.org/citation.cfm?id=796277.

22. Caprara A. On the tightness of the alternating-cycle lower bound for sorting by reversals. // J. Comb. Optim. 1999. P. 149–182.

23. Alekseyev M. A., Pevzner P. A. Breakpoint graphs and ancestral genome reconstructions // Genome Research. 2009. — feb. Vol. 19, no. 5. P. 943–957. URL: http://dx.doi.org/10.1101/gr.082784.108.

24. Kasprzyk A. BioMart: driving a paradigm change in biological data management // Database. 2011. — jan. Vol. 2011, no. 0. P. bar049–bar049. URL: http://dx.doi.org/10.1093/database/bar049.

25. Smit H. R. . G. P., AFA. RepeatMasker Open-4.0. 2013-2015. URL: http://www.repeatmasker.org.