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: **Определение копийности контигов с использованием информации о сборке**

Направление: 010900.68 – Прикладные математика и физика

Выполнил студент Д.А. Мелешко

(подпись)

Руководитель:

PhD Son Pham

(подпись)

Рецензент:

специалист Дмитрий Антипов

(подпись)

Санкт-Петербург

2015

Глава 1

Постановка задачи

1.1. Задача определения изменения копийности

Вплоть до последней четверти двадцатого столетия наше знание о структуре вариаций в геноме ограничивалось огромными по размеру полиморфизмами, которые возможно было заметить в микроскоп, и однонуклеотидными вариациями, которые можно было идентифицировать с помощью различных вариаций метода полимеразно-цепной реакции. Дальнейшее развитие биотехнологий показало также наличие сравнительно небольших участков ДНК, количество которых изменяется от генома к геному одного вида. Подобное явление было названо изменением копийности и изменило взгляды биологов на структуру вариаций в геноме и ход развития многих заболеваний.

Удаления, вставки, дупликации, транслокации - все эти события могут привести к изменению копийности. Несмотря на большое количество исследований, посвящённых изменению копийности, их количество, позиции в геноме, генный контент остаются недостаточно хорошо описанными. Изменения копийности могут быть как спорадическими, так и приобретёнными, и вызывать большое количество заболеваний. Большая по размеру структурная вариация скорее всего приведёт к заболеванию, однако это сильно зависит от того какие гены были включены в участок, подвергшийся вариации. Тем не менее влияние изменения копийности на фенотип человека пока ещё остаётся не до конца исследованным.

Учитывая то, что изменение копийности является причиной многих заболеваний, то задача поиска таких событий достаточно важна и содержательна.

Глава 2

Описание разработанного метода

CNVera принимает на вход множество парных ридов, сгенерированных с некоторого неизвестного генома (целевого генома), и референсный геном, последовательность которого должна быть схожа с последовательностью целевого генома. Сперва алгоритм использует String Graph Assembler (SGA) для сборки. Итогом сборки является множество контигов. Для этого множества CNVera решает задачу *Определения копийности контигов с использованием референсного генома* и для каждого контига получает натуральное число, которое является предположением того, какое количество раз этот контиг встречается в целевом геноме. Далее приведено детальное описание алгоритма.

2.1. О сборке генома

Первым шагом алгоритма CNVera является сборка генома из множества парных ридов при помощи SGA. Используя FM-индекс [1], SGA строит из ридов стринг граф, упрощает и сжимает (заменяя неразветвляющиеся пути одной вершиной) его. В результате получается двунаправленный граф, в котором вершины представляют собой контиги, а ребра - достаточно большие перекрытия между последовательностями, содержащимися в вершинах. В дополнение к этому скрипт DistanceEst трансформируют информацию о парных ридах в информацию о парных контигах: пары контигов и расстояние между этими парами, измеренное в нуклеотидах (для подробного описания этой процедуры можно обратиться к [2]). В дальнейшем, в целях упрощения повествования, алгоритмы будут описаны как для обычных направленных графов, однако их достаточно просто расширить на случай двунаправленных.

Используя результаты работы SGA и референсный геном, мы ставим себе целью определение количества раз, которое каждый контиг встречается в целе-

вом геноме. Так как каждый контиг соответствует некоторой вершине графа, то в идеале целевой геном соответствует некоторому неизвестному пути, который проходит через каждую вершину графа хотя бы один раз (вершинно-покрывающий путь). Таким образом задача о нахождении копийности контига сводится к задаче о нахождении количества раз, которое этот путь проходит через вершину, соответствующую данному контигу. Одним из возможных решений этой проблемы могло бы быть нахождение вершинно-покрывающего пути, соответствующего геному, однако это задача чрезвычайно сложна и является по сути основной задачей сборки генома. Несмотря на то, что, даже имея некоторый вершинно-покрывающий путь, мы не можем сказать верен ли он, поскольку не знаем целевого генома, однако с помощью парной информации мы могли бы восстановить множества подпутей, входящих в искомый вершинно-покрывающий путь.

2.2. О преобразовании пар контигов в подпути

Для каждой пары контигов алгоритм находит в графе путь, который начинается в первом контиге и заканчивается во втором, а длина этого пути в нуклеотидах близка к той, которую предсказал SGA. Если существует единственный такой путь, то он, как правило, является правильным подпутём пути, соответствующему целевому геному (смотрите рисунок 2.1). Однако различные копии одного повтора могут иметь не абсолютно идентичные последовательности либо целевой геном может представлять диплоидный или полиплоидный организм, и это является большой проблемой для преобразования, поскольку в таких регионах может существовать несколько путей примерно равной длины, соединяющих первый и второй контиги (смотрите рисунок 2.2). Тем не менее многие ассемблеры содержат процедуры для удаления балджей, но они чаще используются для исправления оставшихся в ридах ошибок. Чтобы справиться с проблемой множественных путей в работе алгоритма SGA настроен таким

образом, чтобы максимально агрессивно удалять балджи: возможная разница между двумя путями балджа значительно увеличена, также разрешено удалять балджи, чьи длины незначительно различаются. После применения этих процедур практически все балджи в графе сборки схлопываются и мы получаем для большинства повторных регионов консенсусную последовательность, которая, несмотря на то, что часть информации в ней теряется, тем не менее имеет высокую ценность. После этого мы можем максимально успешно применять необходимую процедуру.

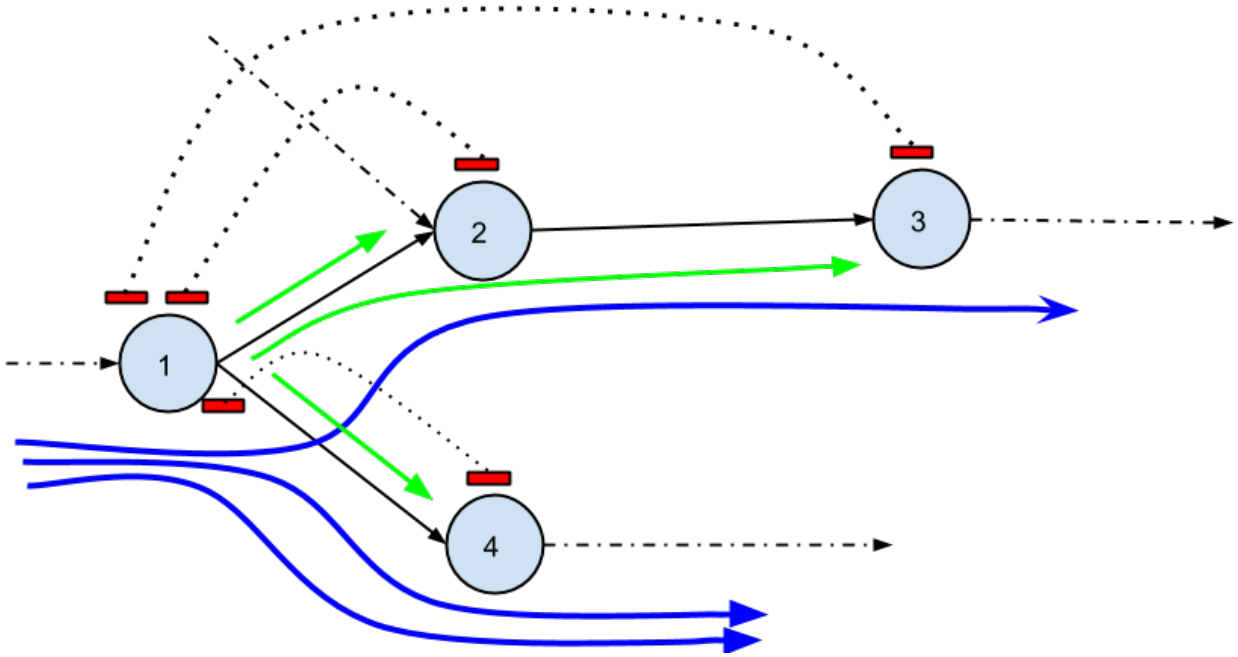

Рис. 2.1. Парные прочтения (красные прямоугольники, соединённые линиями) трансформируются в геномные пути (зеленые линии). Парные прочтения (1,2) и (1,3) трансформируются, соответственно, в геномные пути (1-2) и (1-2-3). Путь (1-2) является подпутём (1-2-3). Синей линией показано то, как целевой геном проходит по графу сборки (этот путь нам неизвестен).

В результате такой трансформации, мы получаем из пар контигов пути, которые являются подпутями неизвестного пути целевого генома в графе сборки.

Сейчас мы дадим формальную постановку задачи. Количество раз, которое целевой геном проходит вершину u в графе, назовём целевой копийностью вершины u и обозначим как $T(u)$. Референсный геном, приложенный к графу,

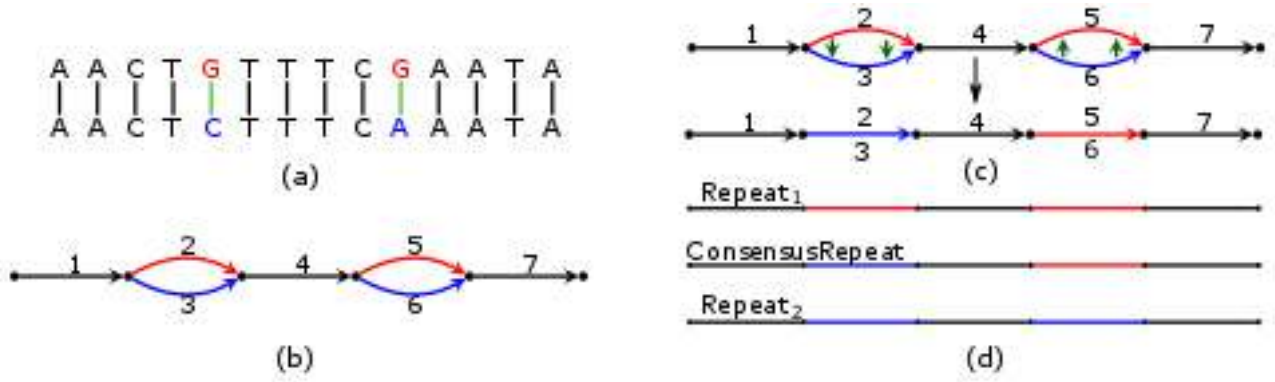

Рис. 2.2. Процедура удаления балджей маскирует различия в разных копиях одного повтора. а) Два повтора с неидентичными нуклеотидными последовательностями. б) Граф ассемблирования этих повторов имеет два балджа. в) Процедура удаления балджей удаляет различия между разными копиями одного повтора. д) Схлопывание балджей позволяет нам получить консенсусную нуклеотидную последовательность повтора

также генерирует путь, который мы будем называть референсным путём. Количество раз, которое референсный путь проходит вершину v в графе, назовём референсной копийностью вершины v и будем обозначать как $R(v)$.

Пусть дан граф сборки $G(V, E)$, множество путей P и функция T, которая назначает положительное целое число каждой вершине $v \in V$. Будем называть тройку (G, P, T) консистентной, если существует функция PD, которая назначает положительное целое число каждому пути $p \in P$ такое, что для любой вершины $v \in V, T(u) = \sum_{start(p)=v \cup p \not\subset \forall p' \neq p \ in \ P} PD(p)$. Если обратиться к интуиции, то $PD(p)$ это количество раз, которое путь p включён в целевой геном.

2.3. Постановка и решение задачи

Теперь мы сформулируем задачу об определении копийности контигов с использованием референсного генома следующим образом:

Определение копийности контигов с использованием референсного генома

Пусть дан граф ассемблирования G, набор путей P, в котором нет пути, который был бы полностью включён в другой путь, и референсный путь R (ко-

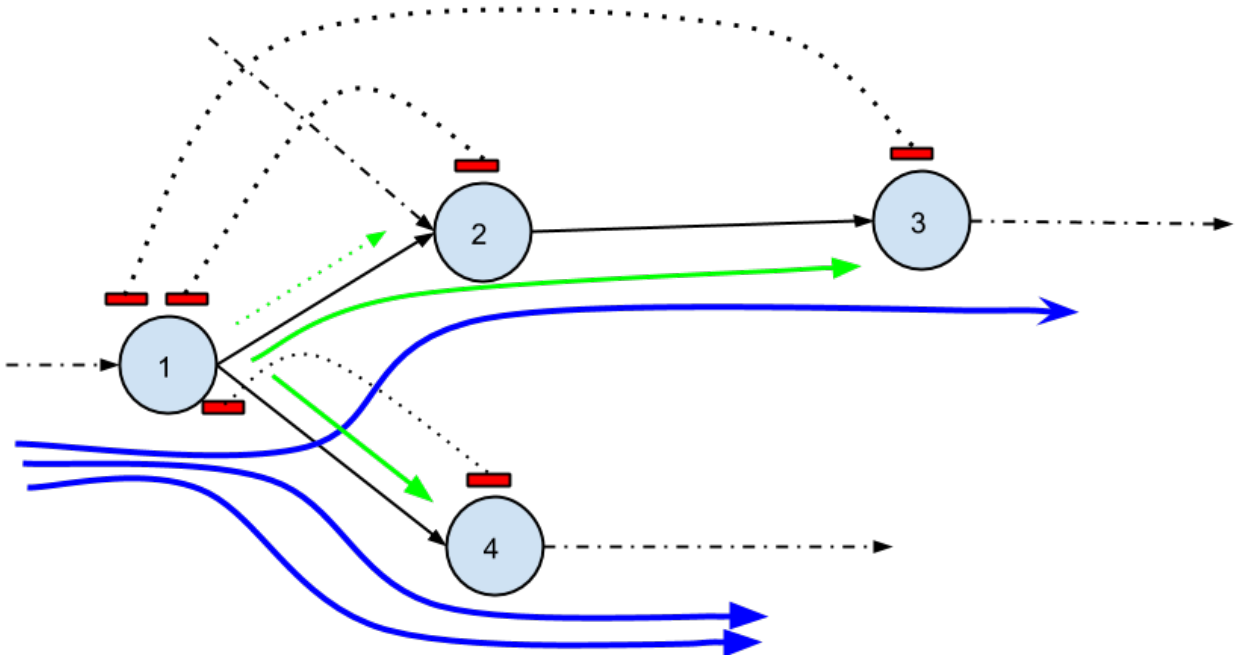

Рис. 2.3. Пути оставшиеся в графе после применения процедуры удаления вложенных путей. Путь (1-2), который показан прерывающейся зелёной линией удалён из $S^1_{forward}$

торый определяет $R(v)$, который определяет $R(v)$ для каждой вершины $v \in V$). Найти функцию T, которая назначает положительное целое число для каждой вершины $v \in V$ такое, что $\sum_{v \in V} |T(v) - R(v)|$ минимально и тройка $(G, S, R(V))$ консистентна.

Так как сложность решения задачи *Определение копийности контигов с использованием референсного генома* неизвестна, ниже предлагается эвристический подход, который для каждой вершины графа восстанавливает ее копийность, опираясь на знание структуры окрестности графа вокруг вершины, информации о путях, входящих в целевой геномный путь и референсного генома.

Для каждой вершины v графа ассемблирования сконструируем множества путей $P^v_{forward}$ и $P^v_{reverse}$. В $P^v_{forward}$ попадут пути из P, которые начинаются в v, а в $P^v_{reverse}$, которые в v заканчиваются. В каждом из множеств проведём следующую процедуру: удалим вложенные пути, поскольку такие пути не несут дополнительной информации и не определяют копийность вер-

шины, в которой они начинаются (смотрите рисунки 2.1 и 2.3). Понятно, что $T(v) \geq max P^v_{forward}, P^v_{reverse}$, так как любой путь из P может входить в целевой геном несколько раз. Чтобы найти точную копийность вершины мы должны определить копийность путей (количество раз, которое целевой геном проходит через вершину), которые начинаются в вершине или заканчиваются в ней.

Чтобы улучшить точность оценки копийности вершины мы разработали Split-Extend алгоритм, который использует референсный геном и структуру графа сборки, чтобы предсказать копийность каждого пути. Для каждого пути p из $P^v_{forward}$ и $P^v_{reverse}$ Split-Extend алгоритм рекурсивно рассматривает все возможные продления p и смотрит на количество вхождений этого пути во множество путей, генерируемых референсным геномом. Когда продлённый путь ровно один раз содержится в пути, то процедура завершается с расширенным множеством путей, которое консистентно с референсным геномом и также имеет поддержку со стороны структуры графа (смотрите рисунок 2.4). Мы применяем Split-Extend алгоритм как для $P^v_{forward}$ так и для $P^v_{reverse}$ и в качестве оценки копийности вершины v берём максимум размера двух расширенных множеств, так как сам по себе Split-Extend алгоритм в случае наличия структурных вариаций в интересующей нас области может стремиться к занижению оценки копийности вершины. В таком случае мы надеемся, что либо за вершиной либо перед ней таких событий не возникало. Алгоритм Split-Extend представлен в псевдокоде в Алг.1.

Однако парная информация может быть недостаточно полной в случае коротких контигов, так как количество ридов, прикладывающихся на эти контиги, недостаточно для подтверждения связи между контигами. Таким образом, пути, заканчивающиеся и начинающиеся в таких вершинах, отсутствуют в P. Такая ситуация чаще всего возникает при длине контигов меньшей двух длин рида. Для таких случаев мы используем пути, полученные алгоритмом для окружающих вершин. Для всех путей, содержащих такую вершину v произведём следующую процедуру. Разрежем данные пути по этой вершине и поместим

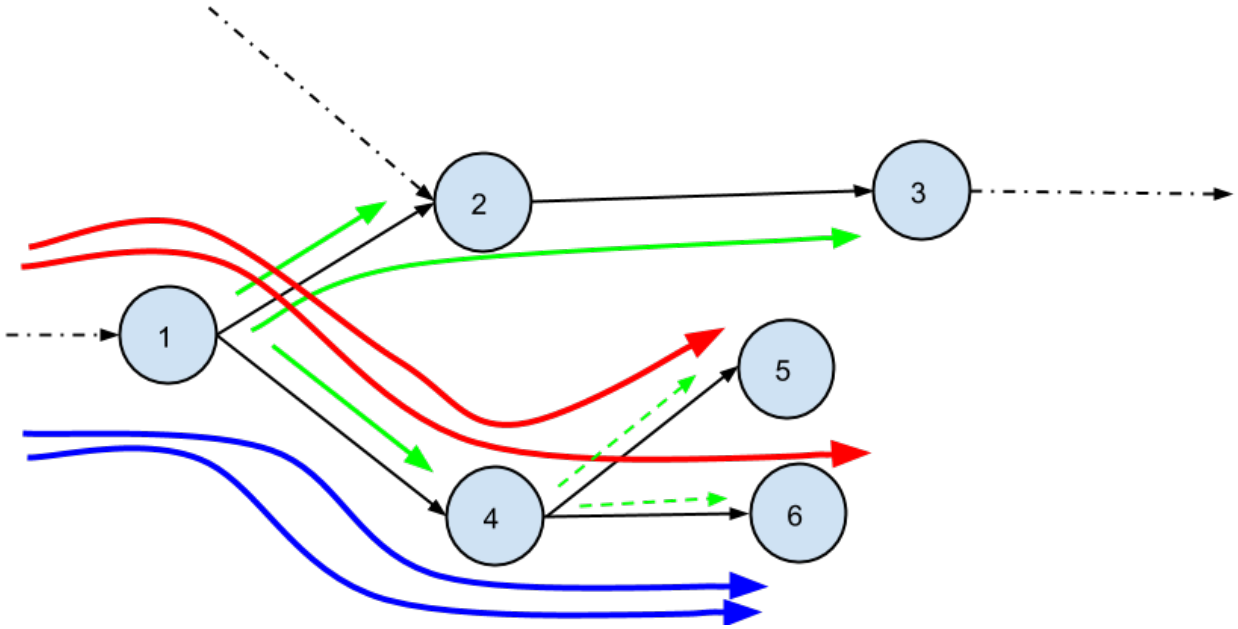

Рис. 2.4. Пример того, как Split-Extend алгоритм успешно предсказывает копийность пути (1-4). Референсный геном проходит сквозь различных соседей контига 4. Используя граф ассемблирования и информацию о прохождении через него референсного генома мы можем заключить, что путь (1-4) имеет копийность два. Это позволяет правильно определить копийность вершины 1, равную трём.

получившиеся пути в $P^v_{forward}$ и $P^v_{reverse}$, удалим вложенные пути и применим к оставшимся Split-Extend алгоритм. Таким образом, мы повышаем разрешение алгоритма и справляемся с задачей определения копийности лучше, чем алгоритмы основанные на покрытии ридами. Полный псевдокод работы алгоритма CNVera представлен в Алг.2.

input : $P^v_{forward}$ or $P^v_{reverse}$ after contained path removal step

output: Set of pathes P' that goes through v, and $|P'|$ is equal to the copy number of v

initialization;

$P' - \emptyset$

Function *Split-Extend(P: set of paths) : set of paths*

```
foreach path p in P do
    alignments = align p on reference genome
    if |alignments| == 0 then
        continue
    end
    else if |alignments| == 1 then
        P' = P' ∪ p
        continue
    end
    else
        construct P_new as an extensions of p on neighbouring nodes
        Split-Extend(P_new)
        if any extension of p is not in P' then
            P' = P' ∪ p
            continue
        end
    end
    return P'
end
```

Алгоритм 1: Псевдокод алгоритма Split-extend

input : Set of paired reads

output: Set of pairs (contig, copy number)

Construct assembly graph G with SGA from reads

C = set of contigs for G

Transform read pair information to contig pair information

Transform contig pair information to genomic sub-path information

foreach *vertex* v *in* G **do**

construct $P^v_{forward}$ and $P^v_{reverse}$

$P^v_{forward}$ = RemoveContained($P^v_{forward}$)

$P^v_{reverse}$ = RemoveContained($P^v_{reverse}$)

$P'^v_{forward} = \emptyset$ $P'^v_{reverse} = \emptyset$

foreach p *in* $P^v_{forward}$ **do**

$P'^v_{forward} = P'^v_{forward} \cup SplitExtend(p)$

end

foreach p *in* $P^v_{reverse}$ **do**

$P'^v_{reverse} = P'^v_{reverse} \cup SplitExtend(p)$

end

$CopyNumber(v) = max(|P'^v_{forward}|, |P'^v_{reverse}|)$

end

foreach *vertex* v *in* G **do**

if *length(*$v <$ *threshold)* **then**

EstimateCopyNumberByNeighbours(v)

end

end

return all contigs with their copy numbers

Алгоритм 2: Псевдокод алгоритма CNVera

Глава 3

Результаты сравнения с другими инструментами

3.1. Описание моделирования эксперимента

Мы провели сравнение CNVera с другими программными средствами на сгенерированных из референсных E.coli и S. cerevisiae данных. Для каждого референсного генома мы сгенерировали геномы, которые будут являться референсными в эксперименте, путем множественных вставок и удалений участков различной длины, таким образом симулируя структурные вариации по сравнению с изначальными референсными геномами. Для каждого генома мы в итоге получили два референсных - с высоким уровнем структурных вариаций и с низким. Референсный геном с низким уровнем отличий имеет 1000 точечных мутаций и 100 вставок/удалений различной длины, а с высоким 10000 точечных мутаций и 1000 вставок/удалений различной длины.

Чтобы сгенерировать риды, мы использовали ART [3] версии 03.11.14 со следующими параметрами: длина ридов - 100 баз оснований, среднее расстояние вставки - 300, стандартное отклонение - 50, среднее покрытие - 40X, стандартный профиль ошибок Illumina.

Мы сравнили производительность нашего программного обеспечения с единственной программой, которая может работать на таком же подмножестве входных данных - Magnolya. Magnolya была запущена в соответствии с ее руководством пользователя. Чтобы воспроизвести эксперимент так, чтобы он соотносился с реальной сборкой генома, мы произвели предварительную коррекцию ридов с помощью Quake [4]. После этого была проведена сборка с помощью SGA, при этом параметры были настроены таким образом, чтобы активировать более агрессивное упрощение графа по сравнению со стандартными настройками.

Чтобы проверить качество результатов, для начала необходимо рассчитать

действительную копийность каждого из получившихся в сборке контигов. Чтобы получить реальную копийность контига v, он был приложен к исходному референсу с помощью BLAST [5]. При этом выравнивания с уровнем точности $< 95\%$ или длиной приложившегося контига $< 95\%$ отбрасывались. Также для каждой позиции в референсе было сохранено лишь одно выравнивание с лучшими параметрами. В итоге реальная копийность контига v определилась как количество его выравниваний среди оставшихся.

3.2. Интерпретация результатов

Обозначим реальную копийность контига v за $T(v)$, а копийность контига, полученную в результате работы одной из программ, за $T'(v)$. Тогда копийность контига считается предсказанной правильно, если выполняется равенство $T(v) = T'(v)$. Ошибка в предсказании контига v рассчитывается как $|T(v) - T'(v)|$. Тогда полную ошибку запуска программы можно записать как $\sum_{vinV} |T(v) - T'(v)|$. В итоге для оценки результатов работы мы использовали две метрики - общее количество контигов, чья копийность была правильно предсказана, и полная ошибка запуска программы. В то время как первая метрика оценивает эффективность подхода в целом, вторая оценивает эффективность подхода в случае коротких и повторных контигов, так как обычно такие контиги имеют высокую копийность и их оценка может содержать крупные по величине ошибки. Для более точной интерпретации мы разделили контиги в сборке на три группы: короче 250 пар оснований, от 251 до 1000 пар оснований и свыше 1000 пар оснований. Метрики были посчитаны для каждой из этих групп в отдельности и для всех контигов сразу.

Результаты сравнения приведены на рисунке 3.1 и таблице 1. Из них можно сделать вывод, что на данных E.Coli и S. cerecisiae в случае длинных контигов оба программных продукта имеют сравнимые результаты, в то время как в случае коротких контигов и контигов средней длины CNVera демонстриру-

ет гораздо более точные результаты. Это показывает эффективность подхода, основанного на сборке в случае коротких контигов.

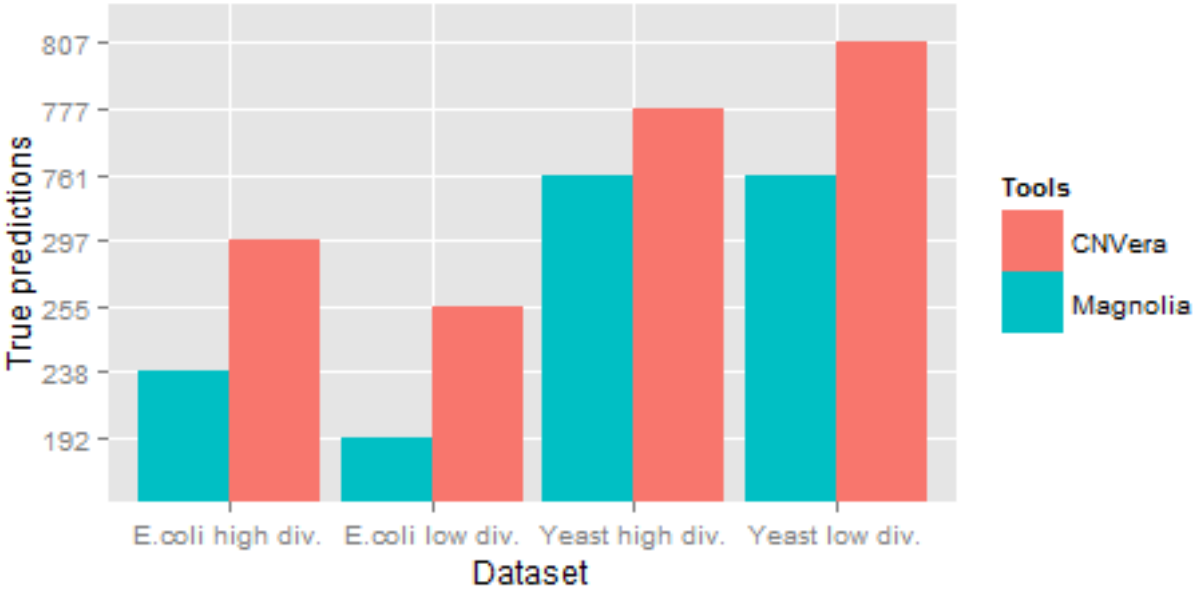

Рис. 3.1. Сравнение производительности алгоритмов CNVera и Magnolya. True prediction - количество контигов, для которых была получена правильная копийность.

Таблица 3.1. Сравнение на сгенерированных данных

E.Coli, референс с малой степенью расхождения		
Общее количество контигов - 461, коротких - 292, средних - 46, длинных - 123		
	Количество правильно предсказанных (общее/короткие/средние/длинные)	Сумма ошибок (общее/короткие/средние/длинные)
CNVera	255/111/24/110	449/392/28/19
Magnolia	192/54/18/120	783/716/64/3
E.Coli, референс с высокой степенью расхождения		
Общее количество контигов - 534, коротких - 307, средних - 50, длинных - 168		
	Количество правильно предсказанных (общее/короткие/средние/длинные)	Сумма ошибок (общее/короткие/средние/длинные)
CNVera	297/124/25/158	452/404/38/10
Magnolia	238/57/20/161	880/756/99/8
Дрожжи, референс с малой степенью расхождения		
Общее количество контигов - 2485, коротких - 1607, средних - 574, длинных - 304		
	Количество правильно предсказанных (общее/короткие/средние/длинные)	Сумма ошибок (общее/короткие/средние/длинные)
CNVera	807/383/138/286	21764/17084/4650/286
Magnolia	761/336/135/290	23226/18365/4837/23
Дрожжи, референс с высокой степенью расхождения		
Общее количество контигов - 2485, коротких - 1607, средних - 574, длинных - 304		
	Количество правильно предсказанных (общее/короткие/средние/длинные)	Сумма ошибок (общее/короткие/средние/длинные)
CNVera	777/366/138/273	21784/17086/4661/37
Magnolia	761/336/135/290	23226/18365/4837/23

Заключение

В данной работе представлен новый алгоритм определения копийности контигов в сборке.

Он предназначен для анализа данных секвенирования нового поколения и имеет множество применений в персональной медицине. Также его использование может быть оправдано и в задаче разрешения повторов. Алгоритм основан на локальном восстановлении геномных путей в графе сборки, что представляет собой абсолютно новый подход в решении данной задачи. По сравнению с ранее разработанными методами, полученный алгоритм использует всю информацию, которую возможно получить из процесса сборки, а не только глубину покрытия отдельных контигов, что приводит к улучшению результатов, особенно в случае, когда контиги достаточно коротки.

В настоящее время, тем не менее, проблемой являются участки сборочного графа с большим количеством коротких контигов, разрешение которых является проблемой как с алгоритмической, так и с технической точек зрения. Также недостатком является то, что реализация метода зависит от определённого сборщика, так как в настоящее время невозможно предположить наличие доминирующего.

Данную работу возможно расширить, поскольку информация о глубине покрытия может быть использована в гораздо большем объёме. Это позволило бы использовать сильные стороны как классических методов, так и алгоритма представленного в данной работе.

По результатам работы был принят постер на конференцию HiTSeq 2015.

Литература

1. Ferragina P., Manzini G., Mäkinen V., Navarro G. An alphabet-friendly FM-index // String Processing and Information Retrieval / Springer. 2004. P. 150–160.

2. Simpson J. T., Wong K., Jackman S. D. et al. ABySS: a parallel assembler for short read sequence data // Genome research. 2009. Vol. 19, no. 6. P. 1117–1123.

3. Huang W., Li L., Myers J. R., Marth G. T. ART: a next-generation sequencing read simulator // Bioinformatics. 2012. Vol. 28, no. 4. P. 593–594.

4. Kelley D. R., Schatz M. C., Salzberg S. L. et al. Quake: quality-aware detection and correction of sequencing errors // Genome Biol. 2010. Vol. 11, no. 11. P. R116.

5. Altschul S. F., Gish W., Miller W. et al. Basic local alignment search tool // Journal of molecular biology. 1990. Vol. 215, no. 3. P. 403–410.