МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра дискретной математики и алгоритмики

ШЕЛЕГ Владислава Михайловна

МЕТОДЫ ПОСТРОЕНИЯ И АНАЛИЗ ФИЛОГЕНЕТИЧЕСКИХ ДЕРЕВЬЕВ ВИРУСОВ

Магистерская диссертация

1-31 80 09 «Прикладная математика и информатика»

Научный руководитель
Баханович Сергей Викторович,
кандидат физико-математических наук

Допущена к защите
«___» _____ 2021 г.
Зав. кафедрой дискретной математики и алгоритмики
_____ В.М. Котов
доктор физико-математических наук, профессор

Минск, 2021

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Магистерская диссертация, 50 с., 17 рис., 1 таблица, 16 источников, 2 приложения.

Ключевые слова: ФИЛОГЕНЕТИЧЕСКОЕ ДЕРЕВО, КОРОНАВИРУС, ФИЛОГЕНИЯ, СОВЕРШЕННАЯ ФИЛОГЕНИЯ, НЕСОВМЕСТИМАЯ МАТРИЦА, ТЕСТ ЧЕТЫРЕХ ГАМЕТ.

Объект исследования: построение и проблемы при построении филогенетических деревьев на основании данных коронавирусных последовательностей. Методы оптимизации построения совершенной филогении.

Цель работы: сформулировать основные проблемы, возникающие при построении филогенетических деревьев, обладающих совершенной филогенией и не обладающих ею; сформулировать, чем наличие совершенной филогении помогает при работе и исследованиях; предложить алгоритмы решения проблем построения филогенетических деревьев; реализовать разработанные алгоритмы.

Методы проведения работы: линейное целочисленное программирование.

Результаты: задачи целочисленного линейного программирования для решения проблем, встречающихся при построении филогенетических деревьев; алгоритмы для построения системы условий и решения задач на языке программирования Python.

Область применения: научные исследования в области филогенетики и эволюционной генетики.

АГУЛЬНАЯ ХАРАКТАРЫСТЫКА ПРАЦЫ

Магістарская дысертацыя,. 50 с., 17 мал., 1 табл., 16 крыніц, 2 дадатка.

Ключавыя словы: ФІЛАГЕНЕТЫЧНАЕ ДРЭВА, КОРОНАВИРУС, ФИЛАГЕНІЯ, ДАСКАНАЛАЯ ФИЛАГЕНИЯ, НЕСУМЯШЧАЛЬНАЯ МАТРЫЦА, ТЕСТ ЧАТЫРОХ ГАМЕТ.

Аб'ект даследавання: пабудова і праблемы пры пабудове філагенетычных дрэў на падставе дадзеных коронавирусных паслядоўнасцяў. Метады аптымізацыі пабудовы дасканалай філагеніі.

Мэта працы: сфармуляваць асноўныя праблемы, якія ўзнікаюць пры пабудове філагенетычных дрэў, якія валодаюць дасканалай філагеніі і якія не валодаюць ёю; сфармуляваць, чым наяўнасць дасканалай філагеніі дапамагае пры працы і даследаваннях; прапанаваць алгарытмы рашэння праблем пабудовы філагенетычных дрэў; рэалізаваць распрацаваныя алгарытмы.

Метады правядзення працы: метады лінейнага цэлалікавага праграмавання.

Вынікі: задачы цэлалікавага лінейнага праграмавання для вырашэння праблем, якія сустракаюцца пры пабудове філагенетычных дрэў; распрацаваныя і рэалізаваныя на мове праграмавання Python алгарытмы для пабудовы сістэмы умоў для задачы і рашэнні задач.

Вобласць прымянення: навуковыя даследаванні ў філагеніі, эвалюцыйнай генетыкі.

ABSTRACT

Master thesis: , 50 p., 17 fig., 1 table, 16 sources, 2 appendix.

Keywords: PHYLOGENETIC TREE, CORONAVIRUS, PHYLOGENY, PERFECT PHYLOGENY, INCOMPATIBLE MATRIX, TEST OF FOUR GAMETS.

Object of research: the building and problems in the construction of phylogenetic trees based on the data of coronavirus sequences. Optimization methods for building perfect phylogeny.

Objective: formulate problem of constructing phylogenetic trees with perfect phylogeny and without it; formulate how the presence of perfect phylogeny helps in work and research; to propose algorithms for solving problems of constructing phylogenetic trees; implement the developed algorithms.

Methods: linear integer programming.

Results: integer linear programming for solving problems encountered in the construction of phylogenetic trees; algorithms developed and implemented in the Python programming language for constructing a system of conditions for a problem and solving problems.

Application area: scientific research in the field of phylogeny, evolutionary genetics.

ОГЛАВЛЕНИЕ

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ДНК – дезоксирибонуклеиновая кислота, молекула, несущая в себе всю необходимую информацию для развития, роста, размножения и дальнейшего функционирования организма.
NGS – next generation sequencing, секвенирование следующего поколения.
ЦМ – цепь Маркова.
ILP – integer linear programming, целочисленное линейное программирование.

ВВЕДЕНИЕ

Вирусные заболевания – одни из самых распространенных видов заболеваний в мире. Способы распространения наибольшего количества вирусов, так называемый воздушный путь передачи вируса, а также высокая восприимчивость человека к данной группе заболеваний определяет их основную эпидемиологическую особенность – очень быстрое и широкое распространение. Количество заболевших по всему миру сезонно и связано с сезонными иммунодефицитом человека. Вирусы есть причина существенного процента болезней и смертности людей.

При возможности, любую проблему такого масштаба стоит пытаться предупреждать, а не лечить уже проявившееся заболевания. Для этого рассматриваются способы построения филогенетических деревьев вирусов, данные из которых позволят в дальнейшем исследовать, что именно повлияло на появление мутации того или иного типа, анализировать процесс лечения людей с похожей картиной болезни для нахождения недочетов и усовершенствования подхода к лечению конкретного типа заболевания. Высокая смертность людей от вирусов в данный момент связана с тем, что данная группа возбудителей склонна к частым мутациям, что приводит к тому, что человеческий организм вынужден каждый год быть готовым к противостоянию новому штамму вируса.

Филогенетические деревья на основе биологических данных вирусов позволяют увидеть, каким именно образом происходят их мутации, возможно, предсказать дальнейшие вероятности отдельных мутаций. Однако перевод биологического материала в цифровые данные имеет погрешности, которые могут существенным образом сказываться на результатах дальнейших исследований, поэтому задача разработки математических методов и программ построения деревьев, учитывающих и по возможности нивелирующих погрешности данных секвенирования, является очень актуальной, особенно в виду сложной эпидемиологической обстановки в мире в связи с распространением вируса SARS-CoV-2.

ГЛАВА 1
ОСОБЕННОСТИ ЗАДАЧИ

1.1. Вирусные заболевания, особенности, мутации

Вирус — это инфекционный агент, который размножается только внутри живых клеток организма. Вирусы имеют способность к жизнедеятельности внутри абсолютно разнообразных форм жизни, начиная от животных и заканчивая любыми микроорганизмами. Выделяются среди вирусов так называемые сателлиты – это вирусы, которые способны к размножению только в присутствии какого-либо другого вида вируса. Вирусы невероятно живучи, что объясняет их нахождение почти во всех системах на земле. Также благодаря своей живучести и способности к размножению количество вирусов на земле превосходит многие остальные живые или псевдо-живые виды организмов и является одним из самых распространённых на Земле.

Вирусы, которые еще не попали в организм или только находятся в процессе заражения какой-либо клетки представляют собой независимые объекты вирионы, которые включают:

1. ДНК/РНК, то есть последовательность аминокислот, описывающих способности вируса к заражению, размножению и т. п.;
2. Оболочки, которая защищает объект и его ДНК/РНК от внешних повреждений и позволяет в целости и сохранности внедрится в клетку для инфицирования;
3. Очень редко, но иногда встречается липидная оболочка.

Вирусы имеют очень разнообразные формы, что позволяет им проживать на огромных территориях и взаимодействовать со многими видами живых существ.

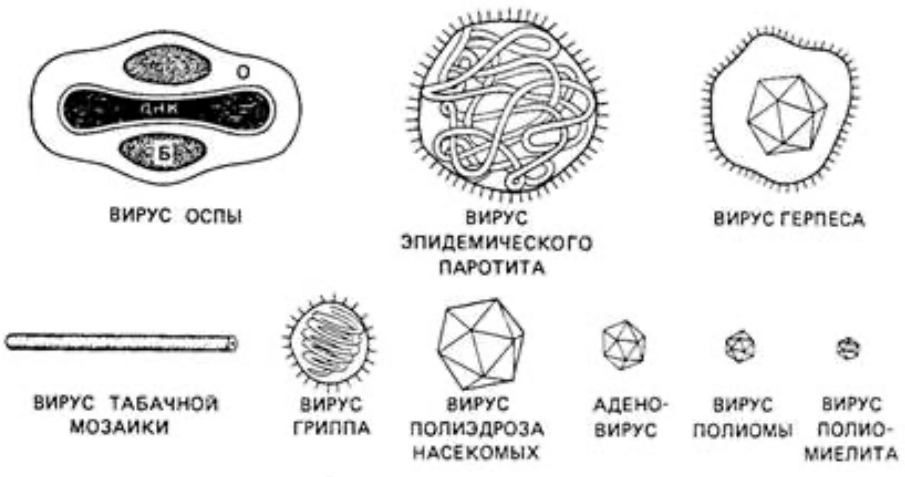

Рисунок 1.1. Форма и относительные размеры вирусов

Способы распространения вирусов не сильно широки, обычно какой-либо живой организм, внутри которого уже присутствует вирус, передает его при взаимодействии с другими живыми организмами. К таким примерам относятся: кровососущие насекомые, которые после укуса одного зараженного существа перемещаются для укуса другого, до этого не бывшего инфицированным; по аналогии с предыдущим пунктом, насекомые, которые питаются растениями и/или их соком, могут употребить в пищу одно зараженное и разнести заболевание на все последующие источники пищи; люди, которые пренебрегают средствами индивидуальной защиты при проявлениях болезни, могут передавать болезнетворные сущности через выделение слюны с кашлем и/или насморком или же передают их через кровь или иные естественные для организма жидкости. Однако, данные способы размножения вируса допустимы не только между одним видом. Для каждого вируса существует так называемый круг хозяев, который показывает, между какими биологическими видами может передаваться данный вирус. К примеру, домашние животные разных видов могут быть распространителями туберкулеза.

Если рассматривать вирус с эволюционной точки зрения, то вирусы представляют значительный интерес для исследователей, ведь на них можно посмотреть с двух противоположных точек зрения: как на объект, подверженный эволюции, и как на один из механизмов эволюции одновременно. С точки зрения объекта эволюции вирусы вызывают много споров из-за того, что являются одновременно и живыми, и не живыми объектами. Они удовлетворяют таким критериям живых организмов как способность нести в себе какую-либо генетическую информацию, способность размножаться, передавая вышеупомянутый генетический материал, и способность к развитию, то есть мутациям и прочее, что позволяет выживать в результате естественного отбора сильнейшим видам вирусов. При этом у них отсутствует, к примеру, структура клетки, без чего организм с точки зрения биологии нельзя назвать живым. Вирусы относят к категории «на пороге жизни», не исключая возможности их дальнейшей эволюции в полноценные живые существа. С точки зрения же механизма эволюции, вирусы являются одним из самых важных способов так называемого горизонтального переноса генов. Горизонтальный перенос генов – способность некоторого организма внедрить свой генетический код в другой организм таким образом, что объект получает новые генетические особенности внутри своего вида без вертикальных мутаций при естественном размножении.

Группа вирусов, состоящая из разных видов, будет включать в себя большее количество геномного разнообразия, чем такие же большие и широко распространённые группы растений и животных. Однако несмотря на то, что вирусов существуют не менее чем несколько миллионов, человечество имеет подробное описание не более 7000 типов и 193 000 описанных последовательностей генома, имеющих очень разнообразные размеры [2], от двух до 2500 белков. Столь сильный разброс размеров вирусов объясняется

тем, что существует два вида вирусов со своими особенностями репликации: РНК- и ДНК-вирусы. Репликация ДНК-вирусов считается намного более надежной и менее подверженной мутациям за счет высокой точности ферментов, которые участвуют в процессе. Надежный механизм позволяет передавать большее количество информации за один раз без потерь, что позволяет иметь большую длину. В отличии от РНК-вирусов, которые имеют намного более частые ошибки и мутации.

В результате естественного отбора вирусы постоянно изменяются. Они претерпевают генетические изменения в результате мутации и серьезные генетические изменения в результате рекомбинации. Мутация происходит, когда в вирусный геном вносится ошибка. Рекомбинация происходит, когда сопутствующие вирусы обмениваются генетической информацией, создавая новый вирус. Частота мутаций ДНК-вирусов приближается к скорости мутаций эукариотических клеток, теоретически давая один мутантный вирус в нескольких сотнях или даже тысячах копий генома. РНК-вирусы имеют гораздо более высокую частоту мутаций, предположительно, это может быть одна мутация на копию вирусного генома.

Мутации могут быть вредными, нейтральными, иногда благоприятными. Только мутации, которые не мешают основным функциям вируса, могут сохраняться в вирусной популяции. Мутации могут производить вирусы с новыми антигенными детерминантами. Появление антигенно нового вируса в результате мутации называется антигенным дрейфом. Антигенно измененные вирусы могут вызывать заболевание у ранее резистентных или иммунных хозяев. Мутации могут производить вирусы с пониженной патогенностью, измененным кругом хозяев или измененной специфичностью клеток-мишеней, но с неизменной антигенностью.

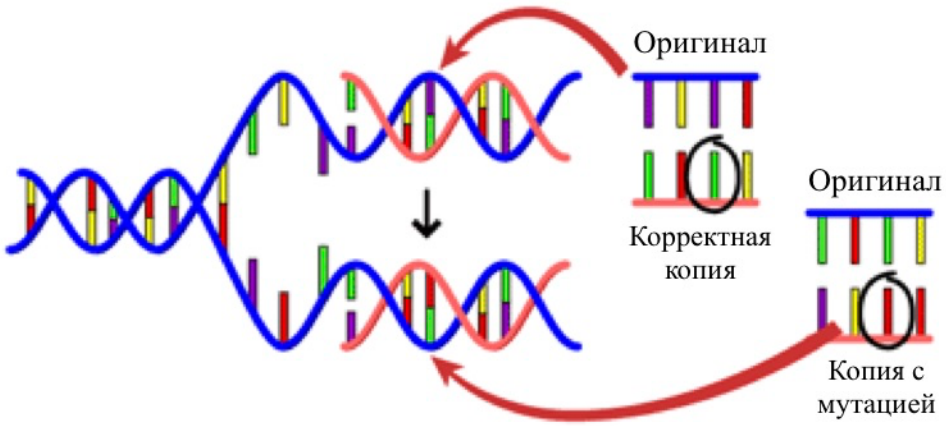

Рисунок 1.2. Пример мутации гена

Рекомбинация включает обмен генетическим материалом между двумя родственными вирусами во время коинфекции клетки-хозяина.

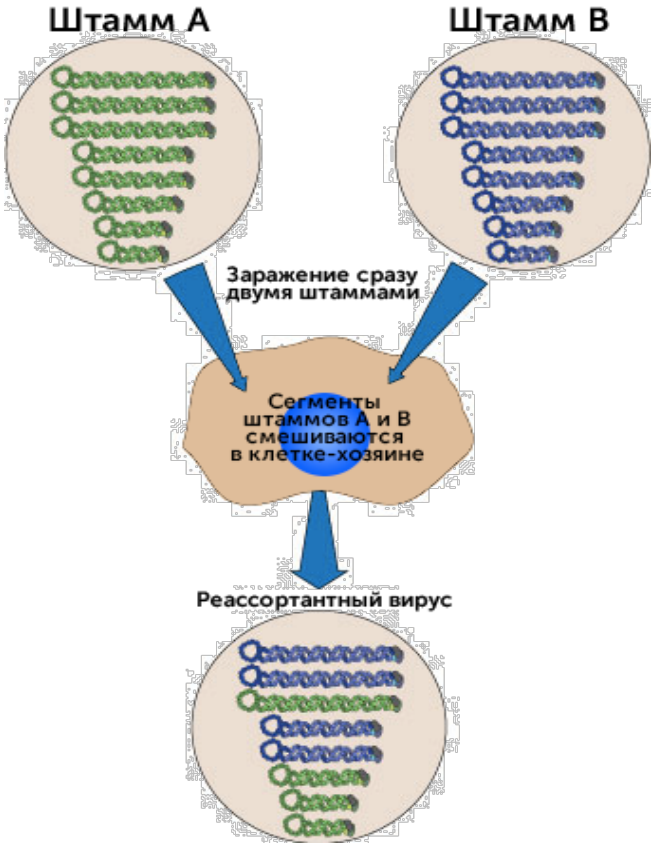

Рисунок 1.3. Пример рекомбинации вирусов

Рекомбинация бывает следующего вида:

- Рекомбинация по независимому ассортименту Рекомбинация путем независимого отбора может происходить среди вирусов с сегментированным геномом. Гены, расположенные на разных частях нуклеиновой кислоты, сортируются случайным образом. Это может привести к генерации вирусов с новыми антигенными детерминантами и новым кругом хозяев. Разработка вирусов с новыми антигенными детерминантами посредством независимого ассортимента называется антигенным сдвигом.
- Рекомбинация не полностью связанных генов Гены, расположенные на одной и той же части нуклеиновой кислоты, могут подвергаться рекомбинации. Чем ближе два гена, тем реже происходит рекомбинация между ними (частичное сцепление).

1.2. Секвенирование

Для того, чтобы исследовать эволюцию клетки надо каким-то образом научиться извлекать последовательности ДНК из исследуемых клеток. В генетике и биохимии секвенирование означает определение первичной структуры, определение аминокислотной или нуклеотидной последовательности. Секвенирование следующего поколения (NGS), массивно-параллельное или глубокое секвенирование [3] – технология секвенирования ДНК, благодаря которой человечество совершило своеобразную революцию в данной отрасли исследований, сделав возможным то, о чем раньше даже не смели мечтать самые смелые ученые. Используя NGS, весь человеческий геном может быть секвенирован в течение одного дня. Предыдущая технология секвенирования – Sanger – используемая для расшифровки генома человека, требовала более десяти лет для получения окончательного варианта. Существует несколько видов секвенирования следующего поколения, отличающиеся в каких-то аспектах реализации, однако все из них содержат три ключевых этапа:

1. Подготовка образца:
Для всех платформ секвенирования следующего поколения требуется библиотека, полученная путем амплификации (увеличение числа копий ДНК) пользовательских последовательностей. Эти последовательности позволяют гибридизировать библиотеку для секвенирования и обеспечивают универсальность.

2. Секвенирование:
На основе каждой из компонент полученной библиотеки создаются отдельные кластеры ДНК, каждый из которых будет действовать как индивидуальная реакция секвенирования. Последовательность каждого кластера считывается (посредством генерации светового или флуоресцентного сигнала) из повторных циклов включения нуклеотидов.

3. Вывод данных:
Каждая машина предоставляет необработанные данные в конце последовательности. Эти необработанные данные представляют собой набор последовательностей ДНК, которые были сгенерированы в каждом кластере.

1.3. Филогенетические деревья

Извлеченные данные необходимо каким-либо образом использовать для дальнейшего анализа. Одним из таких способов является построение филогенетических деревьев. Филогенетическое дерево (эволюционное дерево) [4] – это ориентированно дерево, которое показывает, каким образом происходила эволюция между объектами анализа. Показывает, какое

отношение связывает объекты, являются ли они предком и потомком или же просто имеют давнего общего предка. Объектами может быть что угодно: живые существа, такие как люди и обезьяны, допустим, различные виды растений и т. д. Также можно проанализировать просто отдельно взятые белки в надежде, что может быть обнаружен какой-либо общий предок.

Дерево состоит из множества вершин и множества ребер. Листьями называется конечный объект пути из корня, из которого не выходит ребер и в который входит только одно ребро. В филогенетическом дереве каждый лист есть один из объектов, на которых проводится анализ. Корень есть вершина, в которую не входит ни одного ребра, но выходит не ограниченное количество ребер. В филогенетическом дерево корень есть некий общий предок всех анализируемых сущностей. Нет никаких гарантий, что этот предок когда-либо существовал как вид. Внутренние вершины дерева соответствуют мутациям, которые произошли с предком на тот момент, что привело к появлению всех дочерних последовательностей, располагающихся на путях из этой внутренней вершины. Последовательность мутаций, которая приводит от корня до листа, называется веткой и это и есть эволюционная история объекта. Опять же, нет никакой гарантии, что данная последовательность существовала на самом деле, а не является некоторой приближенной оценкой настоящего эволюционного пути. Построение филогенетического дерева лишь предполагает возможное развитие событий вовремя.

Филогенетические деревья можно разделить на две группы, исходя из наличия корня: укорененные и неукорененные деревья [4]. Укорененные деревья можно отобразить в ориентированные графы, так как каждая листовая вершина имеет как минимум один фиксированный путь до нее от корня. Каждый узел такого дерева соответствует последнему общему предку узлов, которые находятся под ним. Неукорененные деревья не содержат корня, т.е. какого-то предположительного общего предка исследуемых сущностей. Используемость неукорененных деревьев легко объясняется тем, что не всегда понятно, какая именно сущность являлась предком, чтобы установить ход дальнейшей эволюции, тогда как установить какую-либо связь между узлами в каких-то случаях может быть намного проще.

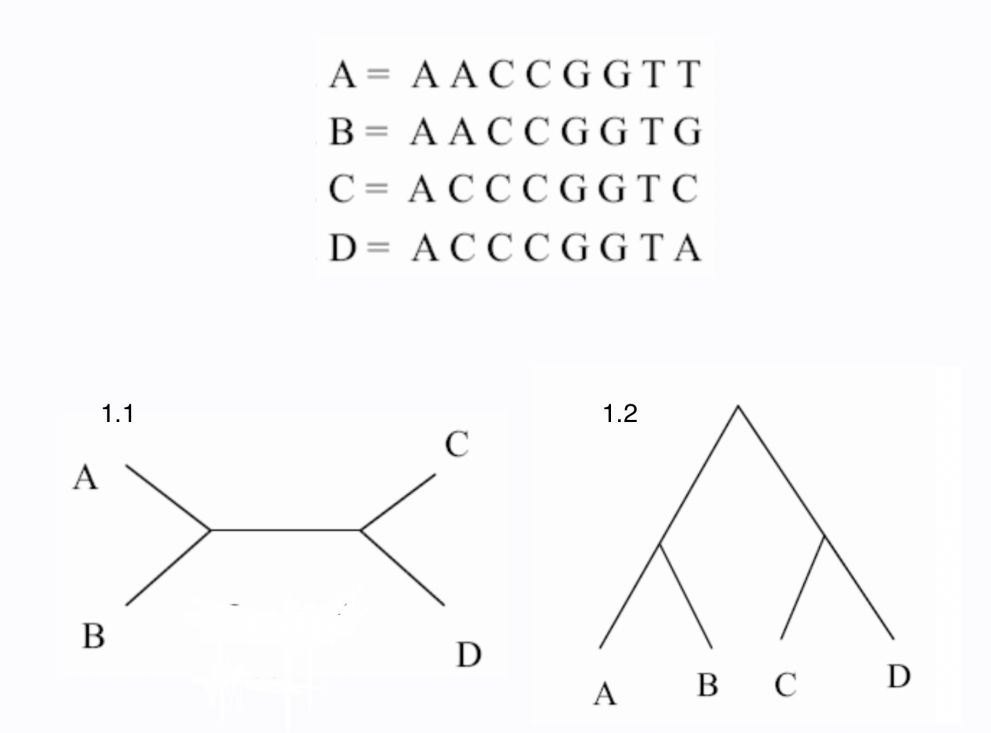

Рисунок 1.4. Пример «неукорененного» (1.1) и «укорененного» (1.2) деревьев

Однако несмотря на то, что в некоторых случаях построенное филогенетическое дерево может дать всю необходимую для дальнейшего анализа информацию, они имеют довольно серьезные ограничения. Как и было уточнено выше, построенное филогенетическое дерево не гарантирует историческую достоверность полученных эволюционных сценариев. Данные, полученные в результате секвенирования, определенно имеют некоторую погрешность в связи с особенностями текущих технологий [3]. Из-за таких погрешностей деревья, построенные на основе одних и тех же начальных данных, но с разной обработкой, могут значительно разниться.

1.4. Совершенная филогения

Совершенная филогения – определение, которое используется в филогенетике для обозначения филогенетического дерева, у которого каждый внутренний узел может быть рассмотрен как уникальная вершина пути от корня к листу, то есть каждая последующая вершина есть сын своего родителя, который наследует все его характеристики, не теряя ни одной, только получая новые. В терминах программирования можно обозначить это следующим образом: каждая отсутствующая характеристика у предка помечена как 0, присутствующая, соответственно, 1. Каждая из упомянутых характеристик может менять свое состояние с 0 на 1, но не обратно. Таким образом, любая приобретенная характеристика передается потомкам, не имея возможности ее утратить. К сожалению, в реальных данных такая ситуация встречается крайне редко, поэтому деревья стараются строить таким образом, который позволит

им быть наиболее «совершенными», то есть, иметь наименьшее число отклонений от определения дерева совершенной филогении.

Статистически совершенное филогенетическое дерево можно определить следующим образом: совершенная филогения для матрицы состояний M размером n x m, элементами которой являются 0 и 1 — это корневое дерево T с n листьями, удовлетворяющими следующим условиям:

1. Каждая строка M соответствует ровно одному листу в T
2. Каждый столбец M соответствует ровно одному ребру в T
3. Каждое внутреннее ребро дерева T соответствует хотя бы одному столбцу в матрице M
4. Каждый путь от корня до листа определяется уникальным вектором символов, где каждый символ говорит о том, что соответствующее ребро в матрице M помечено 1 в строке, соответствующей конечному листу в пути.

ГЛАВА 2
ПОДГОТОВКА АНАЛИТИЧЕСКИХ ДАННЫХ

2.1. Обоснование выбора генетических данных короновируса

Как уже было упомянуто, вирусов, исследованных и исследуемых человеком в данный момент времени, очень много (порядка нескольких тысяч хоть сколько-нибудь подробно исследованных). В данный момент времени больший интерес ученых направлен на исследование так называемого коронавируса, SARS-CoV-2, так как с опасностями, представляемыми этим заболеванием, люди еще не научились бороться. Коронавирус и провоцируемая им атипичная пневмония является причиной глобального карантина и закрытия множества границ в течении 2020г.

Данный вид вируса отличается высокой жизнеспособностью, что подтверждается множеством исследований, хоть и не всегда результаты этих исследований похожи [7]: одна группа ученых утверждает, что при нагреве до 70° вирус дезактивируется в течение пяти минут, другая группа утверждает, что при нагреве до 60° живет в течение часа, а при 92° разрушение происходит в течении пятнадцати минут. Пути распространения коронавируса включают в себя: воздушно-капельный, попадание вируса на поверхности с дальнейшим занесением его на слизистые. Из этого следует высокий коэффициент распространяемости коронавирусной инфекции, равный 2-3, что значит, что, в среднем, один больной человек заражает еще 2-3 человек, не болевших ранее. К тому же, до сих пор ученые не установили, с чем связана потенциальная возможность заразиться коронавирусом повторно.

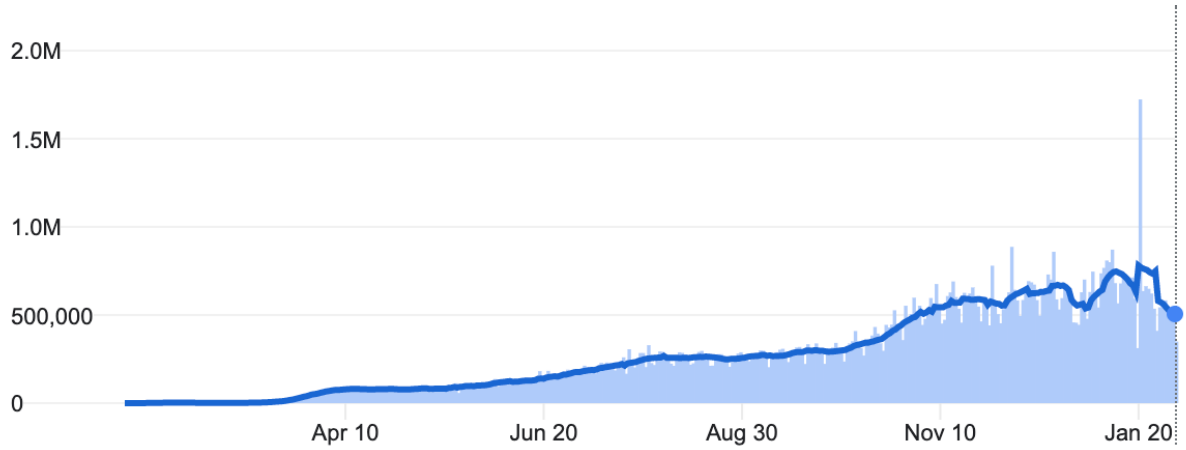

Рисунок 2.1. График обнаружения количества новых случаев коронавируса

Если перейти к описанию течения болезни, то у большинства заболевших болезнь протекает в легкой форме, однако примерно 15% заболевших переносят заболевание в тяжелой форме с необходимость доставки дополнительного кислорода в организм больного. Еще у 5%

пациентов состояние описывается как критическое. Коэффициент смертности от инфекции, порожденной коронавирусом, оценивается примерно в 0.68% [8]. Так же, по состоянию на 28 февраля 2020 года, у 91.1% пациентов была диагностирована пневмония. Естественно, со временем количество случаев с пневмониями стало меньше благодаря расширению возможностей тестирования и обнаружения факта, что большинство людей могут переболеть бессимптомно.

Таким образом, проанализировав актуальность проблемы и опасность текущей эпидемиологической ситуации, а также объем секвенированных последовательностей, находящихся в общем доступе, именно данный вид вируса был выбран для дальнейшего исследования филогенетических деревьев.

2.2. Анализ начальных данных и обработка последовательностей

Благодаря обнародованным в открытом доступе [9] данным секвенированных последовательностей, любой желающий может получить возможность посмотреть на них и провести свои исследования. Коронавирусные последовательности состоят из двух компонент: файл с непосредственным описанием последовательности, т.е. дата взятия, место, возраст и пол пациента, информация о длине последовательности и т.д., и в отдельном файле сами последовательности.

Выбранным методом сортировки было построение деревьев по стране. Так как необходимо учитывать, как же вирус мутирует от страны к стране, но при этом не обрабатывая территории, такие как Великобритания, которая предоставила 60% известных секвенированных последовательностей, необходимо было отобрать список стран, которые секвенировали достаточно последовательностей, но при этом не настолько много, чтобы система обработки не выдержала этого.

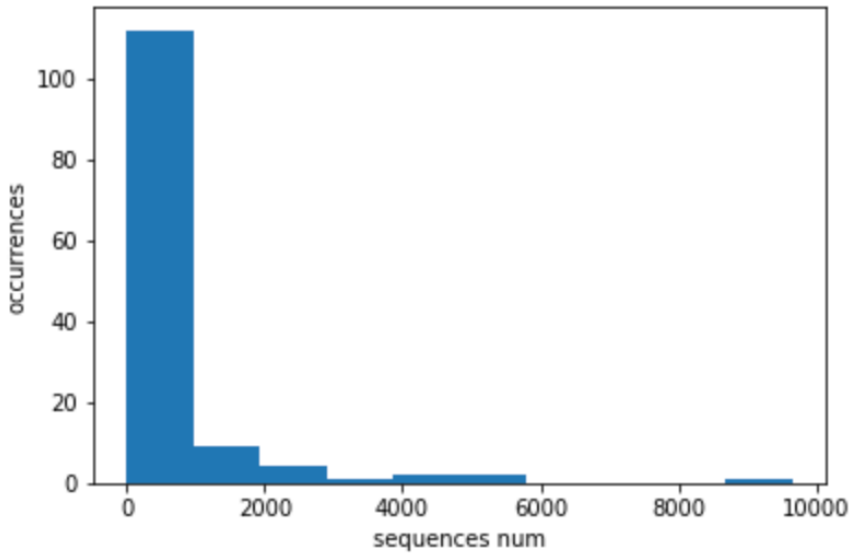

Рисунок 2.2. Гистограмма, отображающее количество стран, которые секвеннировали соответствующее количество последовательностей

Отфильтровав по количеству менее 10000, но более 2000, как наиболее подходящий размер для анализа, получим список стран, которые подходят для дальнейшего исследования: Бельгия, Нидерланды, Дания, Франция, Япония, Португалия, Испания, Швейцария.

Данные, полученные в результате секвенирования, как уже было отмечено в пункте 1.2., не являются идеальными. В полученных образцах последовательностей могут быть пропущены некоторые нуклеотиды, что осложняет дальнейшее исследования, если не провести некоторую обработку. Так как последовательности имеют фиксированную длину, а определить, была ли мутация в зоне, где часто встречаются неопределенные нуклеотиды, невозможно, то необходимо удалить из последовательностей нуклеотиды на соответствующих позициях, чтобы, сократив таким образом длину последовательности, не тратить время и ресурсы на анализ данных, не имеющих смысла.

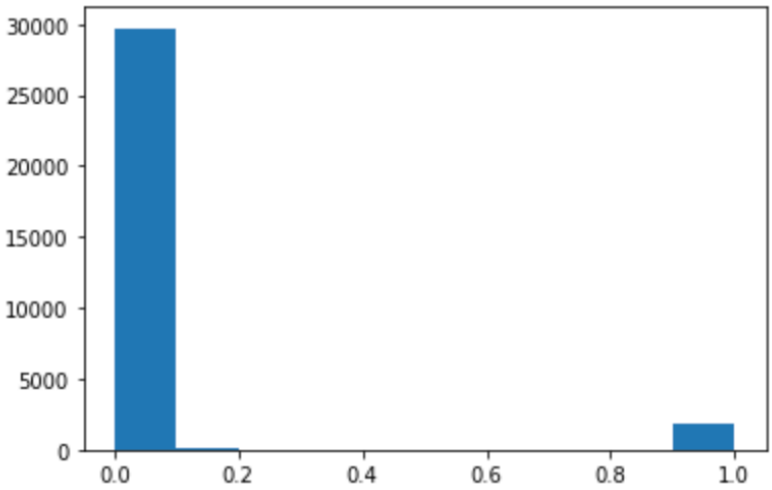

Рисунок 2.3. Гистограмма, отображающая количество неопределенных нуклеотидов относительно всей длины последовательности

Как видно по гистограмме 2.3, наибольший количество последовательностей с неопределенными нуклеотидами выпадает на диапазон, в котором не определено меньше 10% длины. Собственно, эти последовательности в дальнейшем и будут исследованы.

ГЛАВА 3
ПОСТРОЕНИЕ ФИЛОГЕНЕТИЧЕСКИХ ДЕРЕВЬЕВ

3.1. Методы построения филогенетических деревьев

На данный момент существует некоторое множество методов построения филогенетических деревьев [5], которые можно оценивать по следующим критериям:

- Эффективность
- Полезность
- Устойчивость к ошибкам
- Постоянство
- Способность к предупреждениям

Все методы построения деревьев можно разделить на две группы: методы, которые использую оценки генетических дистанций, т.е. предположения о том, насколько велико генетическое сходство между двумя анализируемыми сущностями, и методы, использующие для анализа какие-либо дискретные признаки.

UPMGA(Unweighted Pair Group Method with Arithmetic Mean)

Метод попарного внутригруппового невзвешенного среднего является одним из самых простых методов [4]. Для своего работы требует только одно ограничение, скорость молекулярной эволюции исследуемых сущностей должна быть постоянна. Сам алгоритм состоит из следующих шагов:

- Кладем в матрицу дистанций два таксона с наименьшим генетическим расстоянием. В дальнейшем считаем эти два таксона единой сущностью.
- Пересчитываем матрицу дистанций по правилу $d_{UK} = \frac{d_{U_1K} + d_{U_2K}}{2}$, где d – генетическая дистанция, U – получившийся таксон, U_1 и U_2 – начальные таксоны, K – остальные таксоны.
- Затем выбираем из получившихся таксоны с наименьшим генетическим расстоянием и возвращаемся на первый шаг.

Метод присоединения соседей

Метод присоединения соседей [6] в среднем используется для деревьев, построенных на основе белковых последовательностей. Относится к группе

методов, работающих на основе оценок генетических расстояний. Сам алгоритм основывается на следующих шагах:

- Составляем матрицу попарных расстояний для всех таксонов, вычисляем Q – матрицу по следующему правилу: $Q(i,j) = (n-2)d(i,j) - \sum_{k=1}^{n} d(i,k) - \sum_{k=1}^{n} d(j,k)$, где n – количество таксонов, для которых вычисляются расстояния, а $d(i,j)$ – расстояние между i,j таксонами.
- Ищем пару различных таксонов i,j, таких, что $Q(i,j)$ – наименьшее. Далее эти найденные таксоны присоединяются к новому узлу, а тот соединяется с центральным.
- Рассчитываем расстояние от каждого из присоединенных таксонов до нового узла по следующей формуле(допустим, f и g присоединенные таксоны, а u – новый узел):
$$\delta(f,u) = \frac{1}{2}d(f,g) + \frac{1}{2(n-2)}\left[\sum_{k=1}^{n} d(f,k) - \sum_{k=1}^{n} d(g,k)\right],$$
$$\delta(g,u) = d(f,g) - \delta(f,u)$$
- Рассчитываем расстояния до нового узла от всех остальных таксонов:
$$d(u,k) = \frac{1}{2}[d(f,k) + d(g,k) - d(f,g)]$$
- Формируем новую матрицу попарных расстояний: из матрицы, составленной на первом шага удаляем таксоны, которые были объединены в новую вершину, и добавляем новый узел.
- Повторяем шаги со 2 до тех пор, пока не станут известны длины всех ветвей.

Алгоритм Метрополиса-Гастингса

Реализация алгоритма Метрополиса-Гастингса строится на понятии цепей Маркова, так в процессе строится случайная ЦМ, каждое из состояний которой – филогенетическое дерево [6]. При каждом изменении состояния происходит переход у другому филогенетическому дереву за счет изменения каких-либо параметров модели по определенным правилам. Сам алгоритм основывается на следующих шагах:

- Выбираем случайное начальное дерево T_i.
- Строим новое дерево T_j с, возможно, новой топологией или параметрами модели.
- Вычисляем коэффициент $R = \frac{f(T_j)}{f(T_i)}$, где f некоторая условная вероятность или плотность распределения, зависящая от постановки задачи. Если получившийся коэффициент больше 1, то в качестве нового дерева принимается T_j, если же нет, то генерируется случайная равномерно распределенная на (0,1)

величина, и если значение сгенерированной величины меньше R, то принимаем за новое дерево T_j, иначе оставляет T_i.

- Повторяем процесс с шага 2 до тех пор, пока не будет достигнуто равновесное распределение(состояние наибольшей энтропии)

Стоит отметить, что в случае, если вероятности перехода из дерева T_i в T_j и обратно не равны, то стоит учитывать поправки Гастингса, говорящие о том, что вероятность перехода вычисляется по следующей формуле:
$p_{ij} = \frac{f(T_j)}{f(T_i)} \frac{Q(T_i|T_j)}{Q(T_j|T_i)}$, где функция Q – совместная функция распределения.

3.2. Методы проверки совершенной филогении в деревьях

Алгоритм Гасфилда

Данный метод позволяет не строить филогенетическое дерево, а проверить, будет ли дерево, построенное по его матрице смежности, являться филогенетическим. Алгоритм Гасфилда [10] недавно приобрел популярность при реконструкции филогении опухолей. Этот метод использует вычислительно эффективный сетевой подход, чтобы определить, можно ли восстановить набор бинарных (то есть 0 или 1) признаков из разных групп в действительное филогенетическое дерево. Как и все методы для вывода или реконструкции филогении, алгоритм позволяет нам представить эволюционные отношения между несколькими образцами, людьми или объектами. Алгоритм проверяет возможность построения укорененных деревьев.

Допустим, есть матрица M с n строками и m столбцами, которая показывает наличие некоторой характеристики, где 1 говорит о наличии какого-либо признака, а 0 об отсутствии. Данная матрица и ее ячейки могут говорить, к примеру, о том, является ли некоторый локус гомозиготным или гетерозиготным.

Алгоритм Гасфилда реализует тест, который определяет, может ли такая матрица признаков быть представлена в виде филогенетического дерева, т. е. можно ли построить дерево, которое бы объяснило эволюцию признаков, которые передаются по дереву и не возникают спонтанно? Это называется тестом на «совершенную филогению».

Дерево T определяется следующим образом:
1. каждый объект соответствует ровно одному ребру (и каждое ребро одному объекту)
2. у каждого образца ровно один лист
3. есть уникальный путь из ребер к любому листу

Наличие совершенной филогении означает, что можно построить действительное филогенетическое древо, в котором все признаки

эволюционируют вниз по дереву, а наследственные черты не приобретаются повторно спонтанно. Чтобы реализовать тест на идеальную филогению на матрице бинарных признаков, необходимо выполнить следующие операции:

1. Необходимо отбросить все повторяющиеся столбцы в матрице характеристик M. Каждый столбец оценивается как его двоичное число (так, например, столбец со значением 1000 становится 8) Затем эти значения сортируются в порядке убывания, чтобы определить порядок столбцов. Назовем полученную матрицу M’.
2. Далее надо создать новую матрицу k, где каждая строка соответствует характеристикам, присутствующим для каждого образца. После того, как все характеристики перечислены, строку необходимо закончить знаком «#» и добавить 0 до конце строки.
3. Следующим шагом строится соответствующее дерево из матрицы и удаляются завершающие ребра «#».
4. Чтобы проверить наличие совершенной филогении, необходимо убедиться, что в матрице k каждому листу соответствует уникальный набор характеристик.

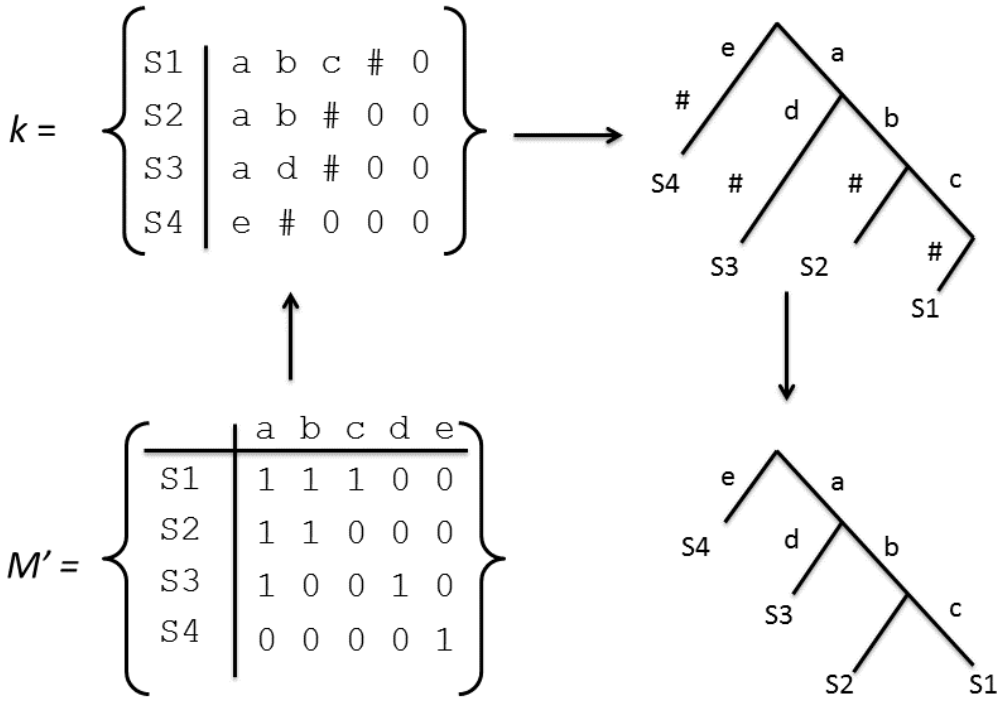

Рисунок 3.1. Пример построения матрицы k и дерева признаков

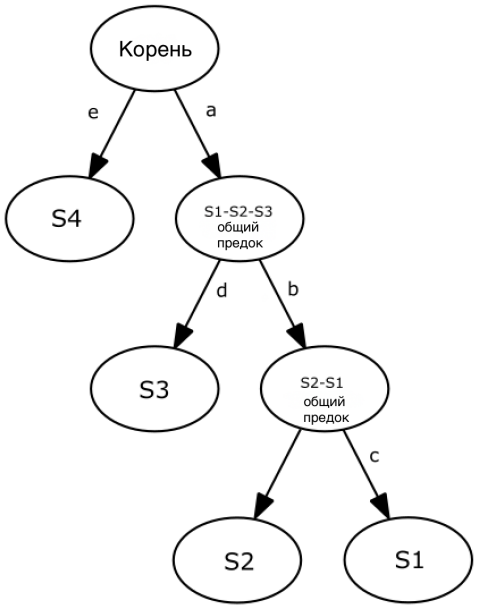

Рисунок 3.2. Результат построения филогенетического дерева по матрице из Рис. 3.1.

3.3. Построение филогенетических деревьев существующими программами

На данный момент существует великое множество средств для построение филогенетических деревьев, которые отличаются алгоритмами, заложенными в них, что, в свою очередь влияет на время работы и на итоговый вид самого дерева. Для начала необходимо построить деревья хотя бы в одной из них, чтобы проанализировать визуально полученный результат. Проанализировав данные об известных программах, была выбрана программа nextstrain, как одна из самых быстрых, представленных сейчас. Для примера приведем деревья, построенные на основании последовательностей стран, упомянутых в пункте 2.2, каждая вершина которого раскрашена в соответствии с полом пациента, у которого была обнаружена последовательность.

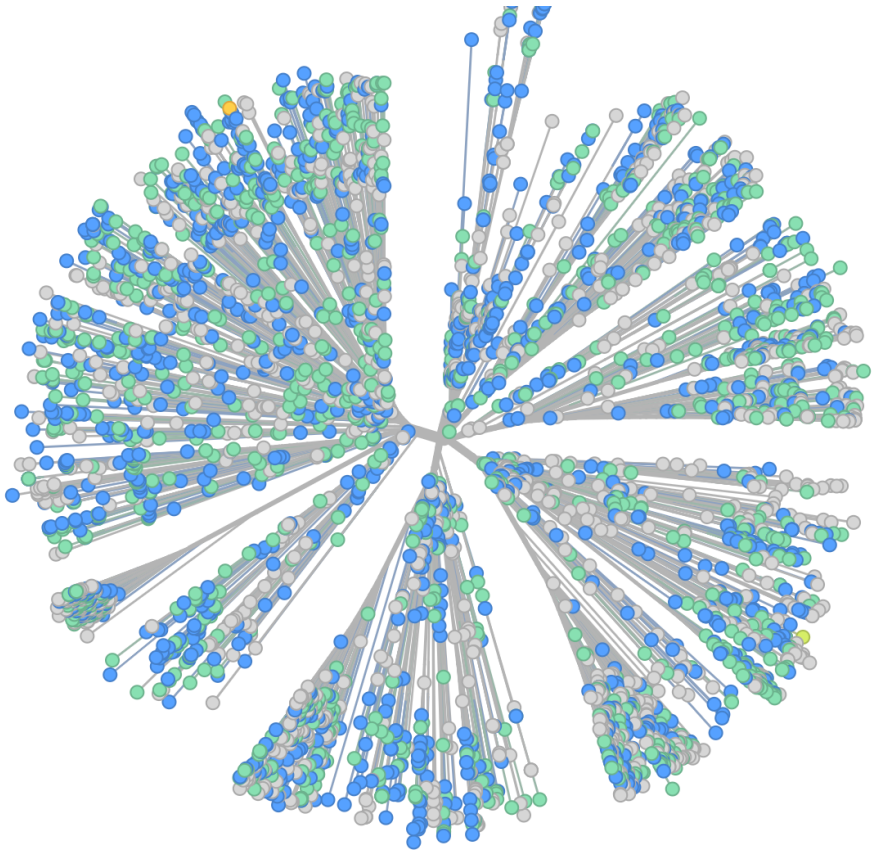

Рисунок 3.3. Филогенетическое дерево, отображенное радиально. Голубым цветом отмечены последовательности, обнаруженные у мужчин, зеленым – у женщин, серым – не известно.

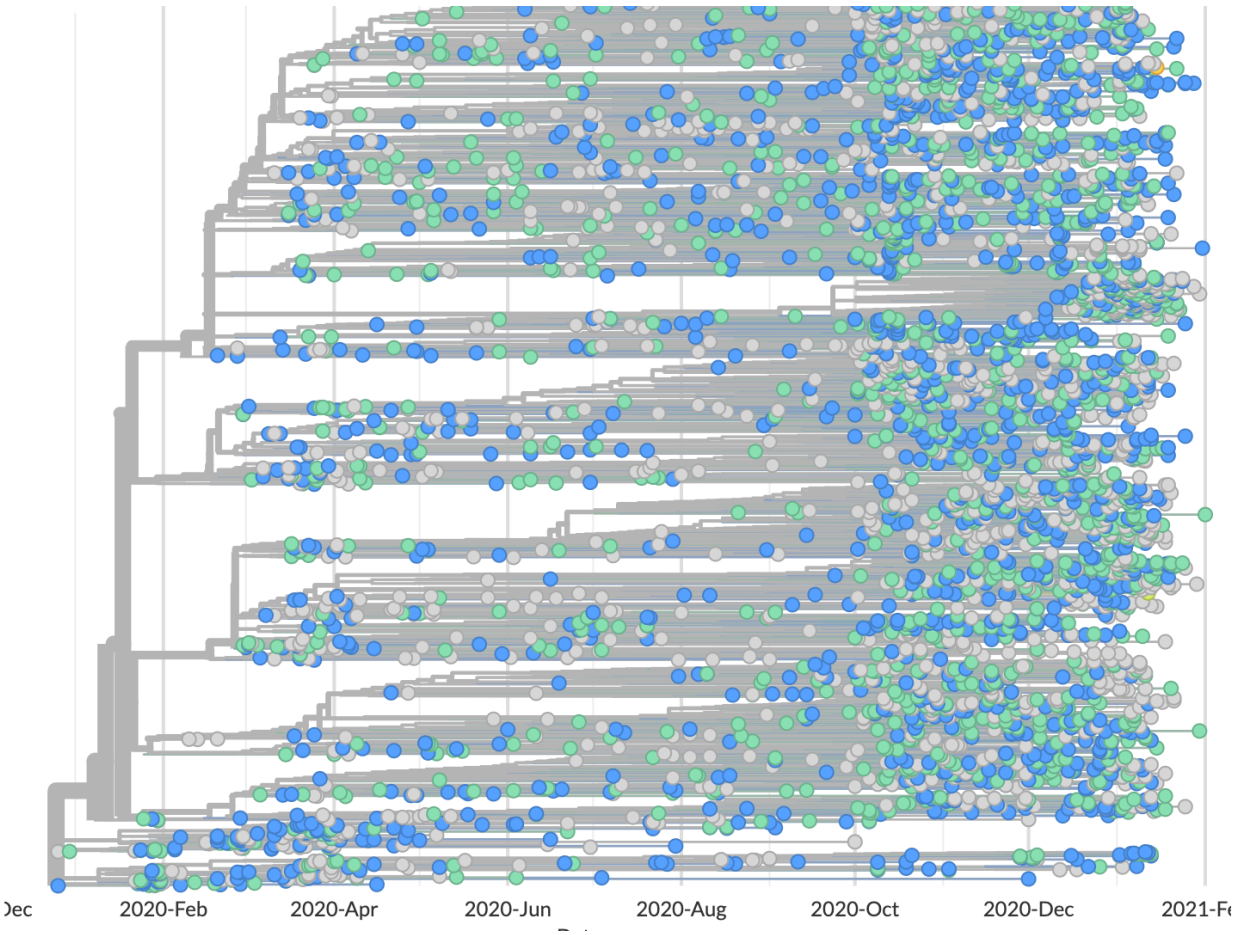

Рисунок 3.4. Филогенетическое дерево. Голубым цветом отмечены последовательности, обнаруженные у мужчин, зеленым – у женщин, серым – не известно.

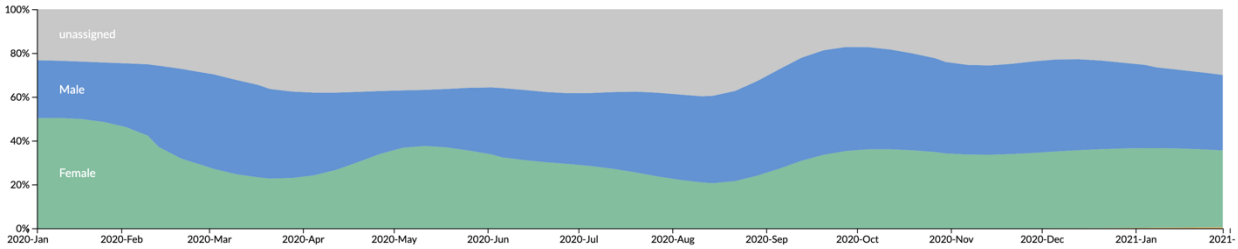

Рисунок 3.5. График, показывающий на долю какого пола приходится какой процент обнаруженных последовательностей

3.4. Проблемы построения филогенетических деревьев

Построение филогенетического дерева – долгий процесс, занимающий иногда вплоть до нескольких часов времени, в зависимости от размера дерева. Было выдвинуто предположение, которое еще только предстоит подтвердить в дальнейшем, что для того, чтобы построить полноценное дерево, удовлетворяющее условиям совершенной филогении, достаточно разбить последовательности по некоторые кластера, построить на основании этих кластеров граф, а в итоге полученные графы, некоторым образом, соединять между собой для того, чтобы получить итоговое дерево.

Однако, уже в результате дальнейшей работы с биологическими данными, было обнаружено, что ошибки, которые приходится обходить в процессе построения филогенетического дерева «с нуля», и методы их решения, представляют собой куда более полезную и разнообразную сферу исследований. В следующей главе представлены результаты работы в попытке упрощения построения филогенетических деревьев не на «чистых» данных. То есть на данных, которые содержат проблемы с физической интерпретацией (пробелы во входных данных, полученные в результате ошибок при секвенировании и погрешностей аппаратов для выполнения этой, без сомнений, сложной процедуры), или же являются повторяющимися, что существенно увеличивает скорость построения дерева в дальнейшем и может привести к некоторым ошибкам.

Большинство проблем в вычислительной филогенетике и популяционной генетике имеют эффективные и элегантные решения в случае, если данные, идущие на вход алгоритму, являются простыми или совершенными, но при наличии хоть немного неточных входных данных предложенные решения становятся не эффективными. Для того, чтобы рассмотреть предложенные обобщения, необходимо ввести определение несовместимых сайтов или столбцов.

Для матрицы M, элементами которой являются 0 и 1, два столбца p и q в M называются несовместимыми тогда и только тогда, когда в M есть четыре строки, в которых столбцы p и q содержат все четыре упорядоченных пары 0,1; 1,0; 1,1; и 0,0. Для матрицы M, элементами которой являются 0 и 1, два узла/сайта p и q в M называются несовместимыми тогда и только тогда, когда в M есть четыре столбца, в которых узлы/сайты p и q содержат все четыре упорядоченных пары 0,1; 1,0; 1,1; и 0,0.

Тест на существование всех четырех пар называется в популяционной генетике «тест четырех гамет». Концепция несовместимости занимает центральное место во многих вопросах, касающихся филогенетических деревьев и истории популяций по следующей причине. Рассматривая каждую строку M как двоичную последовательность, классическая теорема о совершенной филогении гласит, что существует корневое филогенетическое дерево, которое определяет последовательности в M, начиная с некоторой неуказанной корневой последовательности и используя только одну мутацию на узел/сайт, тогда и только тогда, когда нет пары несовместимых узлов в M. Более того, если указана корневая последовательность, то филогенетическое дерево уникально. Предположение об одной мутации на сайт в популяционной генетике называется допущением «бесконечности сайтов».

Проблемы, которые будут в следующем упомянуты в данной работе, являются NP-полными [12, 13, 14], что существенно увеличивает ценность какого-либо хоть сколько-то оптимального решения за полиномиальное время, позволяя использовать его в тех же промышленных вычислениях, например, при исследовании вышеупомянутого короновируса, для определения эволюционного пути того или иного образца, определения общих предков и понимания, в какой именно момент вирус мог мутировать тем или

иным образом. Основываясь на данных, полученных из литературы [14], были сформулированы следующие проблемы:

M1: дана двоичная матрица M, в которой отсутствуют некоторые записи, необходимо дополнить отсутствующие значения, чтобы минимизировать количество несовместимых узлов в получившейся матрице M'.

M2: По всем матрицам, созданным путем вменения пропущенных значений в M, найти такую матрицу M', чтобы минимизировать решение Site Removal проблемы на получившейся в итоге матрице M'.

M3: по всем матрицам, созданным путем подстановки пропущенных значений в M, найти матрицу M′, чтобы минимизировать результирующую нижнюю границу НК для $R_{min}(M')$.

Дополнительные определения будут введены в дальнейшем, при описании полученных решений.

ГЛАВА 4
РЕШЕНИЯ ПРОБЛЕМ ПОСТРОЕНИЯ ФИЛОГЕНЕТИЧЕСКИХ ДЕРЕВЬЕВ

4.1. Задача целочисленного линейного программирования

Линейное программирование [11] – так называемый метод, который позволяет достичь наилучшего условия некоторой задачи и/или математической модели, ограничения в которой задаются системой неравенств. Математически постановка задачи линейного программирования выглядит следующим образом:

$$f(x) = c^T \cdot x \to max$$
$$A \cdot x \leq b$$
$$x \geq 0$$

, где $x = (x_1, \dots, x_n)$ – вектор неизвестных, который только предстоит найти, $c = (c_1, \dots, c_n)$ и $b = (b_1, \dots, b_n)$ – вектора заранее известных коэффициентов, а A – также заранее известная матрица коэффициентов неравенств.

Целочисленное линейное программирование – некоторое сужение линейного программирования, которое добавляет дополнительное условия, ограничивающие пространство решений задачи и/или математической модели, и показывающее, что решение задачи может принадлежать только множеству целых чисел. Таким образом, математическая постановка задачи целочисленного линейного программирования может быть представлена следующим образом в своей канонической форме:

$$f(x) = c^T \cdot x \to max$$
$$A \cdot x \leq b$$
$$x \geq 0$$
$$\mathrm{x} \in \mathbb{Z}^n$$

, переменные заданы таким же образом, как и в задаче линейного программирования.

Так же, задача целочисленного линейного программирования имеет и свою стандартную форму:

$$f(x) = c^T \cdot x \to max$$
$$A \cdot x + s = b$$
$$s \geq 0$$
$$x \geq 0$$
$$x \in \mathbb{Z}^n$$

Переход между стандартной и целочисленной формами осуществляется с помощью введения дополнительных фиктивных переменных в неравенства, что позволяет превратить их в равенства, и накладыванием на фиктивные переменные дополнительных ограничений.

Для решения проблем, сформулированных в пункте 3.4, задачи целочисленного линейного программирования подходят исходя из того, что данные, которые используются при построении филогенетических деревьев бинарны, на основании них можно построить простые линейные неравенства и сформулировать линейную целевую функцию.

Основным недостатком задач целочисленного линейного программирования, при использовании их на практике, является то, что они требуют существенных ресурсов системы, которая выполняет решение. Однако, в случае с филогенетикой, не менее значимые ресурсы нужны также и для построения филогенетических деревьев и для непосредственного определения генетических последовательностей по биологическому материалу.

4.2. Решение проблемы M1

В виду не совершенности существующих методов секвенирования и обработки биологических материалов входные данные не могут быть идеальными, потому что какие-то участки ДНК могут быть потеряны. Если бы данные были идеальны, то определение количества узлов несовместимости и их нахождение было бы легко достижимо за полиномиальное время. Соответственно, проблема восстановления утерянных участков ДНК является одной из самых актуальных.

Напоминая формулировку **проблемы M1**: дана двоичная матрица M, в которой отсутствуют некоторые записи, необходимо дополнить отсутствующие значения, чтобы минимизировать количество несовместимых узлов в получившейся матрице M’.

Задача M1 формулирует следующий вопрос: могут ли быть сгенерированы потерянные данные таким образом, чтобы на основании итоговой матрицы можно было построить дерево совершенной филогении? Если знать корневую последовательность неизвестной филогении, то имеется эффективное и элегантное решение, однако в случае же отсутствия таких входных данных проблема является NP-полной [12]. В случае если матрицу можно дополнить так, чтобы она не имела несовместимых пар сайтов, то последовательности M согласуются с теорией о том, что они произошли от идеальной филогении. Если же значения не могут быть подобраны так, чтобы несовместимость была нулевой, то решение проблемы M1 должно давать меру отклонения матрицы M от определения совершенной филогении.

Integer Linear Programming (ILP) формулировка для задачи M1 прямая и простая. Ее важность состоит в том, что она обычно разрешает задачу с большой скоростью, вычисляя пропущенные значения с высокой точностью,

и что ее можно использовать для решения более сложных проблем. В более общем плане эти формулировки и вычисления иллюстрируют, что для прикладных задач, диапазон данных которых известен, тот факт, что проблема является NP-сложной, не обязательно означает, что точные решения не могут быть эффективно получены на этих данных.

ILP для задачи M1 состоит из:

- одной двоичной переменной Y (i, j) = {0, 1} для каждой ячейки (i, j) в M, в которой отсутствует значение; значение, присвоенное Y (i, j), тогда является временным подобранным значением для M (i, j).
- далее необходимо идентифицировать все пары столбцов (p, q) в M, которые не обязательно несовместимы на данный, но могут быть сделаны несовместимыми в зависимости от подобранных значений в пропущенных ячейках. Пусть P - множество таких пар столбцов;
- для каждой пары (p, q) в P создается переменная C(p, q), которой будет присвоено значение 1 всякий раз, когда подобранные значения вызывают несовместимость между столбцами p и q.

$$C(p,q) = \begin{cases} 1, p, q - \text{несовместимые} \\ 0, \text{иначе} \end{cases}$$

- Для каждой пары в P также определяется, какая из четырех двоичных комбинаций в настоящее время отсутствует в паре столбцов (p, q); пусть d(p, q) представляет недостающие (неполные) двоичные комбинации.
- создается двоичная переменная B (p, q, a, b) для каждой упорядоченной двоичной комбинации a, b в d(p, q); B (p, q, a, b) будет принудительно установлено в 1, если комбинация a, b была создана (посредством установки переменных Y) в некоторой строке в столбцах (p, q).

$$B(p,q,a,b) = \begin{cases} 1, a, b \text{ в какой} - \text{то строке в } p, q \\ 0, \text{иначе} \end{cases}$$

- Затем создаются неравенства, которые устанавливают двоичную переменную C (p, q) в 1, если B (p, q, a, b) было установлено в 1 для каждой комбинации a, b в d (p, q). Для того, чтобы составить неравенства, необходимо обратить внимание и на a, b, и на строку r в столбцах p, q, в которой потенциально будет выставлено значение a, b. Неравенства составляются по следующим правилам:
    1. В левой части неравенства находятся Y(i, j), в правой B(p, q, a, b).
    2. Если значение a или b равно 0 для неизвестной позиции в p_r или q_r, то переменная Y(r, p) и/или Y(r, q) идет со знаком -, иначе – +.
    3. Если в левой части неравенства находятся две переменные с разными знаками, то нет необходимости добавлять что-либо в правую часть неравенства.

4. Если в левой части неравенства находится две переменные с одинаковыми знаками необходимо добавить 1 к правой части неравенства с соответствующим знаком.
5. Если в левой части неравенства находится одна переменная со знаком - – необходимо вычесть с правой части неравенства 1.
6. Финальное неравенство, учитывающее C(p, q), выглядит следующим образом:

$$C(p,q) \geq \sum_{(a,b)\in d(p,q)} B(p,q,a,b) - |d(p,q)| - 1$$

- Следовательно, C (p, q) устанавливается в 1, если (но не только) вменения отсутствующих значений в столбцах (p, q) приводят к несовместимости этих сайтов.
- Таким образом, общая целевая функция ILP состоит в том, чтобы минимизировать $|F| + \sum_{(p,q)\in P} C(p,q)$, где F - это набор несовместных пар, полученный из начальных элементов 0 и 1 в матрице M.

M	p	q
1	0	0
2	?	1
3	1	0
4	?	?
5	?	0
6	0	?

Рисунок 4.1. Пример матрицы с пропущенными значениями

Чтобы подробней объяснить формулировку задачи, далее будет приведен пример работы алгоритма, показанный на основе данных из рисунка 4.1., где отсутствующая запись обозначена знаком «?».

В матрице представлены пары {(0,0), (1,0)}, тогда d (p, q) равно {(0, 1), (1, 1)}.

Таким образом для образования пары (1, 1) подходят строки 2 и 4, составим для них неравенства по правилам, описанным выше:

1. a = 1, b = 1, таким образом Y(2, p) будет в неравенстве со знаком +. Неизвестная только одна, так что дополнительных 1 в неравенстве добавлять не надо.
2. a = 1, b = 1, таким образом Y(4, p) и Y(4, q) будут в неравенстве со знаком +. Так как неизвестных две, и они имеют одинаковый знак +, то необходимо

добавить дополнительную 1 в неравенство.

$$Y(2,p) \leq B(p,q,1,1)$$

$$Y(4,p) + Y(4,q) \leq B(p,q,1,1) + 1$$

Для образования пары (0, 1) подходят строки 2, 4 и 6, составим для них неравенства по правилам, описанным выше:

1. a = 0, b = 1, таким образом Y(2, p) будет в неравенстве со знаком -. Неизвестная только одна, но она отрицательная, так что необходимо вычесть дополнительную 1 в неравенстве.
2. a = 0, b = 1, в строке 4 две пропущенных переменных, значит в неравенстве будут присутствовать Y(4, p) и Y(4, q). Так 0 в позиции a, а соответственно и в столбце p, то Y(4, p) будет взято со знаком -, тогда как Y(4, q) пойдет в неравенство со знаком +. Неизвестных в неравенстве слева хоть и две, но они имеют разные знаки, следовательно, добавлять дополнительную 1 нет необходимости.
3. a = 0, b = 1, в строке 6 одна пропущенная переменная на позиции b, значит, Y(6, q) будет взята в неравенстве со знаком + и без необходимости добавлять дополнительную 1.

$$-Y(2,p) \leq B(p,q,0,1) - 1$$

$$-Y(4,p) + Y(4,q) \leq B(p,q,0,1)$$

$$Y(6,q) \leq B(p,q,0,1)$$

Финальное неравенство, включающее в себя C(p, q), будет выглядеть следующим образом:

$$C(p,q) \geq B(p,q,1,1) + B(p,q,0,1) - 1$$

Если M является матрицей размером nxm, приведенная выше формулировка ILP для задачи M1 создает не более $n \cdot m$ переменных Y, $2m^2$ переменных B, $\frac{m^2}{2}$ переменных C и имеет оценку O(nm^2), хотя все эти оценки являются наихудшим случаем, и на практике цифры намного меньше. Например, если I - это ожидаемый процент отсутствующих записей (который во многих приложениях составляет всего 3%), то ожидаемое количество переменных Y равно $n \cdot m \cdot I$. Описанная формулировка может создавать избыточные неравенства, необходимо смотреть на конкретную ситуацию, чтобы минимизировать их количество. С другой же стороны, чем больше ограничений стоит на математической задаче – тем меньше множество ее решений, что существенно облегчает ее решение.

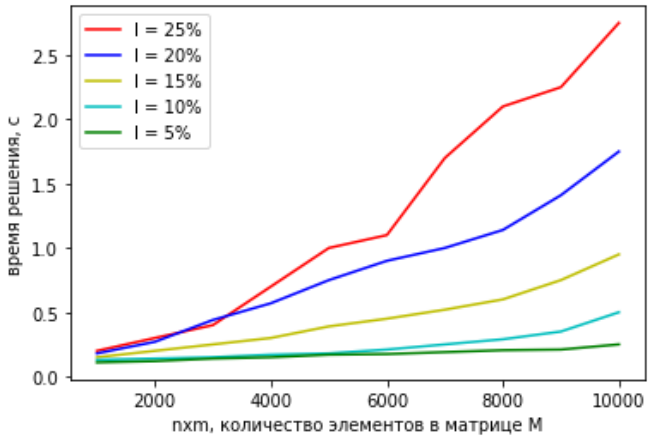

Рисунок 4.2. График зависимости времени выполнения от количества элементов и процента потерянных данных для проблемы M1

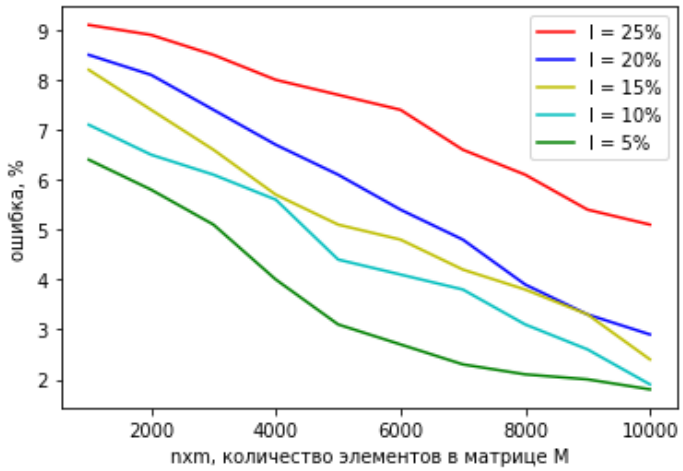

Рисунок 4.3. График зависимости ошибки вычисления от количества элементов и процента потерянных данных для проблемы M1

4.3. Решение проблемы M2

Проблема Site Removal (удаления сайтов). Для двоичной матрицы M без пропущенных записей, но с некоторыми парами несовместимых сайтов, необходимо найти наименьший набор сайтов для удаления из M таким образом, чтобы в итоге ни одна оставшаяся пара сайтов не была несовместимой.

Нахождение небольшого набора сайтов для удаления, чтобы не было несовместимых пар, часто предлагается и используется (особенно в филогенетических исследованиях) в качестве средства очистки данных, которые не полностью соответствуют модели совершенной филогении. Ожидается, что хоть и могут быть некоторые сайты, где наблюдается гомоплазия (из-за повторяющихся или обратных мутаций в этом сайте), идеальная филогения, построенная из оставшихся сайтов, все еще даст достоверную эволюционную информацию о таксонах. Подобные сценарии существуют и в популяционной генетике. Конечно, чтобы получить наиболее информативную филогению, необходимо удалить как можно меньше сайтов.

Проблема удаления сайта является NP-сложной и обычно формулируется как проблема покрытия узла в графе, где каждый узел представляет сайт, а каждое ребро соединяет несовместимую пару сайтов[13, 14].

Когда M имеет отсутствующие записи, проблема удаления сайта обобщается до **проблемы M2**, которая, как было указано в пункте 3.4, звучит следующим образом: По всем матрицам, созданным путем вменения пропущенных значений в M, найти такую матрицу M', чтобы минимизировать решение Site Removal проблемы на получившейся в итоге матрице M'.

ILP для задачи M2 состоит из:

- одной двоичную переменной Y (i, j) для каждой ячейки (i, j) в M, в которой отсутствует значение; значение, присвоенное Y (i, j), тогда является временным подобранным значением для M (i, j).
- далее необходимо идентифицировать все пары столбцов (p, q) в M, которые не обязательно несовместимы на данный, но могут быть сделаны несовместимыми в зависимости от подобранных значений в пропущенных ячейках. Пусть P - множество таких пар столбцов;
- для каждой пары (p, q) в P создается переменная C(p, q), которой будет присвоено значение 1 всякий раз, когда подобранные значения вызывают несовместимость между столбцами p и q. Для каждой пары в P также определяется, какая из четырех двоичных комбинаций в настоящее время отсутствует в паре столбцов (p, q);
- пусть d (p, q) представляет те недостающие (неполные) двоичные комбинации.
- создается двоичная переменная B (p, q, a, b) для каждой упорядоченной двоичной комбинации a, b в d (p, q); B (p, q, a, b) будет принудительно установлено в 1, если комбинация a, b была создана (посредством установки переменных Y) в некоторой строке в столбцах (p, q). Затем программа создает неравенства, которые устанавливают двоичную переменную C (p, q) в 1, если B (p, q, a, b) было установлено в 1 для каждой комбинации a, b в d (p, q). Следовательно, C (p, q) устанавливается в 1, если (но не только если) вменения отсутствующих значений в столбцах (p, q) приводят к несовместимости этих сайтов.
- добавляется переменная бинарная переменная D(i), обозначающая, какой именно сайт должен быть удален или нет, D(i) = 1 в случае необходимости удаления, D(i) = 0 в противном случае.
- для каждой пары $(\mathrm{p}, \mathrm{q}) \in P$ составляем дополнительное ограничивающее условие: $\mathrm{D(p)} + \mathrm{D(q)} - \mathrm{C(p, q)} \geq 0$ которое говорит о том, что если пропущенные значения подставляются так, что пара сайтов (p, q) становится несовместимой, то необходимо удалить либо сайт p, либо сайт q.
- для каждой пары сайтов $(p, q) \in \mathrm{F},$ где F - это набор несовместных пар, полученный из начальных элементов 0 и 1 в матрице M, необходимо проверить следующее условие $\mathrm{D(p)} + \mathrm{D(q)} \geq 1$

- общая целевая функция ILP состоит в том, чтобы минимизировать $\sum_{i=1}^{m} D(i)$.

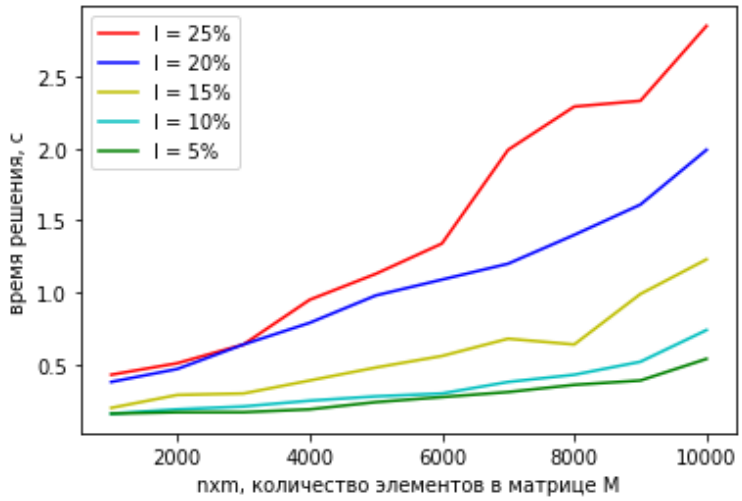

Рисунок 4.4. График зависимости времени выполнения от количества элементов и процента потерянных данных для проблемы M2

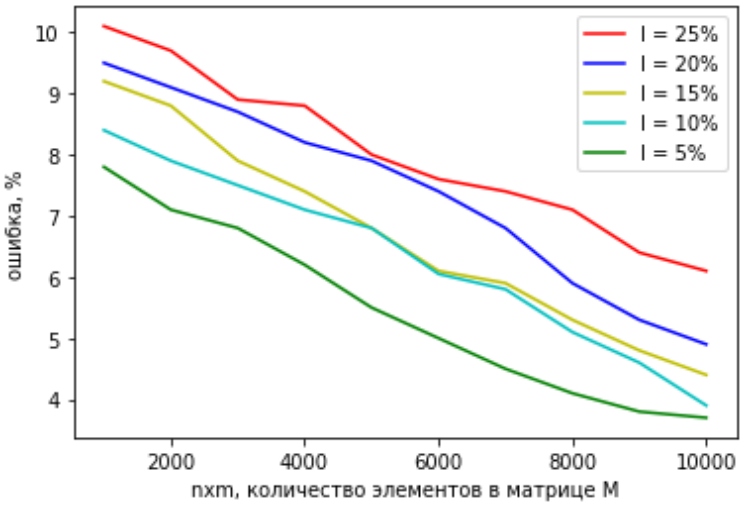

Рисунок 4.5. График зависимости ошибки вычисления от количества элементов и процента потерянных данных для проблемы M2

4.4. Решение проблемы M3

Рекомбинация, как уже было пояснено в пункте 1.1, это фундаментальный молекулярный феномен, когда во время мейоза две последовательности равной длины образуют третью последовательность такой же длины, состоящую из префикса одной из последовательностей, за которым следует суффикс другой последовательности. Основная проблема состоит в том, чтобы определить минимальное количество повторных комбинаций, обозначаемых $R_{min}(M)$, необходимых для генерации набора двоичных последовательностей M из некоторой известной или неизвестной предковой последовательности, когда применимо предположение о бесконечности сайтов [16]. По этой проблеме существует обширная

литература, но нет известного эффективного алгоритма для точного вычисления $R_{min}(M)$. [16] Однако существуют эффективные алгоритмы, которые дают относительно хорошие нижние вычисления границы $R_{min}(M)$, и есть биологические вопросы, касающиеся рекомбинации (например, поиск горячих точек рекомбинации), которые были успешно решены с использованием нижних границ вычисления $R_{min}(M)$, а не с использованием вычисления самого $R_{min}(M)$ [15]. Первая опубликованная и самая основная нижняя граница, называемая границей HK [16], получается следующим образом: необходимо рассмотреть m сайтов M как целые точки 1 ... m на вещественной прямой и выбрать минимальное количество нецелых точек, назвав его множеством R, так, чтобы для каждой пары несовместимых узлов (p, q) в M существует хотя бы одна точка в R строго между p и q. Легко показать, что | R | ≤ $R_{min}(M)$. Когда данные в M заполнены, граница HK – |R| – может быть вычислен за полиномиальное время с помощью жадного алгоритма.

Однако в реальной ситуации, когда некоторые элементы в M отсутствуют, возникает **проблема M3**, которая, как было указано в пункте 3.4, звучит следующим образом: по всем матрицам, созданным путем подстановки пропущенных значений в M, найти матрицу M′, чтобы минимизировать результирующую нижнюю границу HK для $R_{min}(M')$. Причина минимизации заключается в том, что, в случае решения этой задачи, результатом является допустимая нижняя граница количества рекомбинаций, необходимых для генерации истинных базовых данных, когда применяется предположение о бесконечности узлов.

ILP для задачи M3 состоит из:

- одной двоичную переменной Y (i, j) для каждой ячейки (i, j) в M, в которой отсутствует значение; значение, присвоенное Y (i, j), тогда является временным подобранным значением для M (i, j).
- далее необходимо идентифицировать все пары столбцов (p, q) в M, которые не обязательно несовместимы на данный, но могут быть сделаны несовместимыми в зависимости от подобранных значений в пропущенных ячейках. Пусть P - множество таких пар столбцов;
- для каждой пары (p, q) в P создается переменная C(p, q), которой будет присвоено значение 1 всякий раз, когда подобранные значения вызывают несовместимость между столбцами p и q. Для каждой пары в P также определяется, какая из четырех двоичных комбинаций в настоящее время отсутствует в паре столбцов (p, q);
- пусть d (p, q) представляет те недостающие (неполные) двоичные комбинации.
- создается двоичная переменная B (p, q, a, b) для каждой упорядоченной двоичной комбинации a, b в d (p, q); B (p, q, a, b) будет принудительно установлено в 1, если комбинация a, b была создана (посредством установки переменных Y) в некоторой строке в столбцах (p, q). Затем программа создает неравенства, которые устанавливают двоичную

переменную C (p, q) в 1, если B (p, q, a, b) было установлено в 1 для каждой комбинации a, b в d (p, q). Следовательно, C (p, q) устанавливается в 1, если (но не только если) вменения отсутствующих значений в столбцах (p, q) приводят к несовместимости этих сайтов.

- для каждого c из 1 .. m-1 ставится в соответствие переменная R(c). Пусть R(c) будет двоичной переменной, используемой, чтобы указать, следует ли выбирать точку в R в открытом интервале (c, c + 1)
- для каждой пары сайтов $(p, q) \in F \cup P$, ставится дополнительное условие $\sum_{p \le c < q} R(c) \ge C(p, q)$
- общая целевая функция ILP состоит в том, чтобы минимизировать $\sum_{c=1}^{m-1} R(c)$.

4.5. Детали имплементации решений

Как было уточнено в пункте 4.2, задача целочисленного линейного программирования выглядит следующим образом:

$$f(x) = c^T \cdot x \to max$$
$$A \cdot x \le b$$
$$x \ge 0$$
$$x \in \mathbb{Z}^n$$

Для решения задач целочисленного линейного программирования используются такие приложения, как: GLPK, CPLEX, LP Solve, CLP, CVXOPT и т. д. Чтобы выбрать наиболее подходящее программное обеспечение для решения задачи были рассмотрены такие критерии, как наличие обертки, позволяющее использовать ее из других языков программирования, таких как Python, и наличие открытого исходного кода, чтобы в случае неясностей с использованием не опираться только на документацию, а иметь возможность изучить исходный код и разобраться с тем, каким образом используется тот или иной модуль приложения. На основании таких критериев было выбрано приложение CPLEX, точнее его обертка, позволяющая интегрировать использование в свой код с помощью Python 3.7.

Для того, чтобы использовать полноценно возможности GLPK необходимо иметь матрицу коэффициентов неравенств, что представляет некоторую сложность, учитывая, что неравенства генерируются на основании данных и не имеют заранее известного вида.

Таким образом, чтобы построить матрицу коэффициентов неравенства, предлагается следующий подход:

- Неизвестными переменными в задачах, поставленных в пунктах 4.3-4.5, являются не только переменные $Y(i, j)$, обозначающих, какое значение на позицию (i, j) предлагается поставить на конкретном этапе алгоритма, но и переменные

$B(p, q, a, b)$, показывающие, установлена ли хоть одна пара из сайтов p и q в значения a и b и переменные $C(p, q), D(p), R(c)$.

- Чтобы учесть все возможные ограничения, необходимо составить такую матрицу условий, в которой будут учтены все вышеупомянутые переменные. Учитывая, что единственные допустимые свободные коэффициенты в неравенствах будут вынесены в вектор коэффициентов $b = (b_1, \dots, b_n)$ и их значения могут быть -1, 0, 1, то можно добавить в ограничения задачи следующее условие: $b_i \in \{-1, 0, 1\}, i = \overline{1, n}$, где n в данной формулировке не является тем n в определении задачи, задающим размер вектора неизвестных переменных x.

Рассмотрим на примере из пункта 4.3. пример построения такой матрицы. Неравенства, полученные в результате их построения, имеют следующий вид:

$$Y(4, p) + Y(4, q) \leq B(p, q, 1, 1) + 1$$

$$Y(2, p) \leq B(p, q, 1, 1)$$

$$-Y(2, p) \leq B(p, q, 0, 1) - 1$$

$$-Y(4, p) + Y(4, q) \leq B(p, q, 0, 1)$$

$$Y(6, q) \leq B(p, q, 0, 1)$$

$$C(p, q) \geq B(p, q, 1, 1) + B(p, q, 0, 1) - 1$$

Далее необходимо переместить слагаемые, чтобы было абсолютно очевидно, какие именно переменные неизвестны, и подставить точные индексы переменных.

$$Y(4, 1) + Y(4,2) - B(1, 2, 1, 1) \leq 1$$

$$Y(2, 1) - B(1, 2, 1, 1) \leq 0$$

$$-Y(2, 1) - B(1, 2, 0, 1) \leq -1$$

$$-Y(4, 1) + Y(4, 2) - B(1, 2, 0, 1) \leq 0$$

$$Y(6,2) - B(1, 2,0, 1) \leq 0$$

$$-C(1, 2) + B(1, 2, 1, 1) + B(1, 2, 0, 1) \leq 1$$

Таким образом, матрица условий, которые получились в результате построения следующих неравенств, выглядят следующим образом:

$Y(2,1)$	$Y(4,1)$	$Y(4,2)$	$Y(6,2)$	$B(1,2,0,1)$	$B(1,2,1,1)$	$C(1,2)$
0	1	1	0	0	-1	0
1	0	0	0	0	-1	0
-1	0	0	0	-1	0	0
0	-1	1	0	-1	0	0
0	0	0	1	-1	0	0
0	0	0	0	1	1	-1

Таблица 4.1. Матрица условий примера на рис. 4.1

В показанной выше матрице условий не указаны не $Y(i,j)$, которые не являются неуточненными и нет необходимости их исследовать дополнительно, поэтому столбцы, соответствующие им, выставлены в вектор, состоящий из 0, что автоматически исключает их из дальнейшего анализа. Таким образом, чтобы задать целевую функцию, надо задать вид вектора x и вектора стоимости c. Тогда вектор неизвестных в задаче x выглядит следующим образом:

$$x = (Y(1,1), Y(1,2), Y(2,1), Y(2,2), Y(3,1), Y(3,2),$$
$$Y(4,1), Y(4,2), Y(5,1), Y(5,2), Y(6,1), Y(6,2),$$
$$B(1,2,0,1), B(1,2,1,1), C(1,2))$$

В результате получился программный код из приложения В, который состоит из нескольких блоков для построения матрицы условий. Для примера, на рисунке 4.1. программный код строит следующую матрицу условий для неизвестных переменных, что, если сопоставить с таблицей 4.1, соответствует тому, что должно было получиться в результате разрешения задачи постановки условий:

```
for m in constrains_matrix:
    print([int(a) for a in m])
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0]
[0, 0, 0, 0, 0, 0, -1, 1, 0, 0, 0, 0, -1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0]
[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, -1, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, -1]
```

Рисунок 4.6. Пример матрицы условий, полученной программным кодом, для решения задачи, изображенной на рис. 4.1.

ЗАКЛЮЧЕНИЕ

В ходе написания работы были рассмотрены основные моменты о жизнедеятельности и мутациях вирусных заболеваний, понятия филогенетических деревьев и секвенирования, способы построения филогенетических деревьев.

В первой главе рассмотрены определения, что есть вирус, секвенирование, филогенетическое дерево, совершенная филогения и где она используется, то есть даны подробные объяснения, что за данные будут использованы в работе, способы их получения и возможные сложности в их обработке.

Во второй главе показаны основные моменты подготовки и обработки данных, используемых для дальнейших попыток построения и анализа.

В третье главе приведены способы построения филогенетических деревьев несколькими алгоритмами, способ проверки наличия совершенной филогении по матрице 0 и 1, задающей дерево, также была проведена попытка оптимизации одного из алгоритмов построения дерева, однако были найдены трудности, решение которых не менее важно, чем оптимизация алгоритма, ведь это позволит с высокой скоростью подготавливать данные для анализа или других исследований.

В четвертой главе подробно разобраны проблемы, возникающие при построении деревьев совершенной филогении, предложены постановки задач целочисленного линейного программирования для каждой из них, реализован программный код, составляющий матрицу условий для задачи, а также использующий программу CPLEX для решения поставленной задачи. Приведены эмпирические результаты решения данных проблем, графики, отображающие скорость работы в зависимости от размера и процента неизвестных данных в начальной матрице. Также предоставлен график, показывающий, насколько ошибается каждый алгоритм.

В дальнейшем планируется продолжить разработку алгоритма оптимизации построения филогенетических деревьев, основанного на разбиение на малые поддеревья, что позволит ускорить и выполнять параллельно данный процесс.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Oxford textbook of Oncology // [Электронный ресурс]. — 2015. Режим доступа : https://books.google.by/books/about/Oxford_Textbook_of_Oncology.html. — Дата доступа : 10.10.2020.
2. Manual of medical oncology // [Электронный ресурс]. — 2006. Режим доступа : https://books.google.by/books/about/The_MD_Anderson_Manual_of_Medical_Oncolo.html. — Дата доступа : 1.11.2020.
3. DNA sequencing // [Электронный ресурс]. — 2017. Режим доступа : https://en.wikipedia.org/wiki/DNA_sequencing. — Дата доступа : 11.11.2019.
4. Лукашов В.В. Молекулярная эволюция и филогенетический анализ / Лукашов В.В. — М.:Бином — 2009 — стр. 18-23.
5. Molecular phylogenetics // [Электронный ресурс]. — 2017. Режим доступа : https://en.wikipedia.org/wiki/Molecular_phylogenetics. — Дата доступа : 15.11.2020.
6. Penny D., Hendy M.D., Steel M.A. Progress with methods for constructing evolutionary trees. — 1992 — стр. 73-79
7. SARS-CoV-2 // [Электронный ресурс]. — 2019. Режим доступа : https://en.wikipedia.org/wiki/Severe_acute_respiratory_syndrome_coronavirus_2 — Дата доступа : 15.11.2020
8. Lethality // [Электронный ресурс]. — 2009. Режим доступа : https://en.wikipedia.org/wiki/Lethality — Дата доступа : 15.11.2020
9. GISAID // [Электронный ресурс]. — 2015. Режим доступа : https://www.gisaid.org/ — Дата доступа : 15.11.2020
10. D.Gusfield. Efficient algorithms for inferring evolutionary trees – 1991 – стр. 2-10.
11. A. Schrijver. Theory of linear and integer programming. – 1986 – стр. 18-43.
12. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees – 1992 – стр. 97–119. .
13. Felsenstein. Inferring Phylogenies. Sinauer, Sunderland, MA., 2004.
14. C. Semple and M. Steel. Phylogenetics. Oxford University Press, UK, 2003.
15. V. Bafna and V. Bansal. Improved recombination lower bounds for haplotype data. Proceedings of RECOMB 2005.
16. R. Hudson and N. Kaplan. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. Genetics – 1985 – стр. 142–169.

Приложение А

```
import pandas as pd
import numpy as np
data = pd.read_csv('COVID data/metadata_2020-12-28_14-16.tsv', sep='\t',
header=0)
data.head()

from collections import Counter
import matplotlib.pyplot as plt
counted = Counter(data['country'])
cnt = list(filter(lambda x: x < 10000, counted.values()))
plt.hist(list(cnt))
plt.xlabel('sequences num')
plt.ylabel('occurrences')
plt.show()

list(filter(lambda x: x[1] > 2000, dict(counted).items()))

from collections import defaultdict
top_countries_dict =
data.loc[data['country'].isin(top_countries)]['country'].to_dict()
grouped_ids = defaultdict(set)
for k, v in top_countries_dict.items():
    grouped_ids[v].add(k)

def create_country_file(country):
with open('COVID data/msa_1229/msa_1229.fasta') as f:
    meta = f.readline()
    v = 0

    while meta:
        sequence = f.readline()
        if country in meta:
            with open(f"COVID data/msa_1229/countries/{country}.fasta", "a+") as cf:
                cf.writelines([meta, sequence])
        v += 1

        meta = f.readline()

metas = 0
v = 0

bel_ids = set()
```

```
with open('COVID data/msa_1229/msa_1229.fasta') as f:
    meta = f.readline()
    while meta:
        sequence = f.readline()
        id_ = meta.split("|")[0][9:]
        if id_ in ids:
            bel_ids.add(id_)

        meta = f.readline()

from tqdm import tqdm

belgium = []
metas = []
common_len = 0
with open("COVID data/Belgium.fasta", "r") as cf:
    meta = cf.readline()
    while meta:
        seq = cf.readline()
        metas.append(meta)
        belgium.append(seq)
        meta = cf.readline()


final_sequences = []
spaces = []
for seq in tqdm(list(zip(*belgium))):
    spaces.append(seq.count('-') * 1.0 / len(seq))
    if seq.count('-') * 1.0 / len(seq) > 0.1:
        continue
    final_sequences.append(seq)
print((final_sequences))

plt.hist(spaces)
plt.show()

def get_unique_columns(matrix):
    m_prime = np.rot90(matrix)

    print("Get unique columns..")
    m_prime = set(tqdm(map(lambda x: '.'.join(map(str,x)), m_prime)))
    m_prime = np.array(list(tqdm(map(lambda x: list(map(int,x.split('.'))),m_prime))))
```

```
    binary_strings = []
    for col in tqdm(m_prime):
        col_string = '0b'+''.join(map(str,col))
        binary_strings.append(int(col_string,2))

    order = np.argsort(binary_strings)[::-1]
    m_prime = m_prime[order]
    m_prime = np.rot90(m_prime[:,:])[::-1]
    print(m_prime)
    return m_prime

def generate_k_matrix(matrix):
    ncol = len(matrix[0])
    k = np.empty([0,ncol])
    features = np.array(range(1, ncol + 1))
    print("Generate k-matrix..")
    for m in tqdm(matrix):
        m = np.array(m)
        row_feats = features[np.where(m != 0)]
        mrow = np.zeros(ncol)
        mrow.fill('0')
        for idx,feature in enumerate(row_feats):
            mrow[idx] = feature
        k = np.append(k,[mrow],axis=0)
    return k, features


def is_not_phylogeny(k, features):
    locations = []
    print("Check phylogeny..")
    for feature in tqdm(features):
        present_at = set([])
        for k_i in k:
            [present_at.add(loc_list) for loc_list in list(np.where(k_i == feature)[0])]
```

```
        locations.append(present_at)
    print(locations)
    loc_test = np.array([len(loc_list)>1 for loc_list in locations])
    return np.any(loc_test)

def check_if_phylogeny_tree_exists(matrix):
    unique = get_unique_columns(matrix)
    k, features = generate_k_matrix(unique)
    print(k)
    if is_not_phylogeny(k, features):
        print('No phylogeny found!')
    else:
        print('Success! Found phylogeny!')
```

Приложение В

```
def generate_p_q_pairs(M):
    indexes = set()
    for index, column in enumerate(M.T):
        if '-' in column:
            indexes.add(index)
    return [(x, y) for x in indexes for y in indexes if y > x]

def generate_missed_pairs(p, q):
    pairs = possible_pairs.copy()
    for r_p, r_q in zip(p, q):
        r_p = int(r_p) if r_p != '-' else r_p
        r_q = int(r_q) if r_q != '-' else r_q
        if (r_p, r_q) in pairs:
            pairs.remove((r_p, r_q))

    return pairs

def generate_constrain_for_row(a, b, r_p, r_q):
    constrains = [0, 0, 0]
    if r_p == '-' and r_q == '-':
        if a == 0 and b == 0:
            constrains[0] = -1
            constrains[1] = -1
            constrains[2] = 1

        if a == 0 and b == 1:
            constrains[0] = -1
            constrains[1] = 1

        if a == 1 and b == 0:
            constrains[0] = 1
            constrains[1] = -1

        if a == 1 and b == 1:
            constrains[0] = 1
            constrains[1] = 1
            constrains[2] = -1

    if r_p == a and r_q == '-':
        if b == 0:
            constrains[1] = -1
            constrains[2] = 1
```

```
        if b == 1:
            constrains[1] = 1

    if r_p == '-' and r_q == b:
        if a == 0:
            constrains[0] = -1
            constrains[2] = 1
        if a == 1:
            constrains[0] = 1

    return constrains


def generate_constrains_for_columns(pair, pair_number, M, p, q,
global_constrains_matrix):
    a, b = pair

    i = 0

    for r_p, r_q in zip(M[p], M[q]):
        r_p = int(r_p) if r_p != '-' else r_p
        r_q = int(r_q) if r_q != '-' else r_q
        if (r_p == a and r_q == '-') or (r_p == '-' and r_q == b) or (r_p == '-' and r_q ==
'-'):
            constrain = generate_constrain_for_row(a, b, r_p, r_q)
            last_row_index = len(global_constrains_matrix)

            global_constrains_matrix = np.vstack([global_constrains_matrix,
np.zeros(len(global_constrains_matrix[0]))])
            global_constrains_matrix[last_row_index][i * len(M) + p] = constrain[0]
            global_constrains_matrix[last_row_index][i * len(M) + q] = constrain[1]
            global_constrains_matrix[last_row_index][len(M[0])* len(M) +
pair_number] = -1
        i += 1
    return global_constrains_matrix


for pair_number, pair in enumerate(P_set):
    p, q = pair
    d_p_q = generate_missed_pairs(M[p], M[q])
    constrains_matrix = np.zeros((1, len(M[0])* len(M) + len(P_set) * len(d_p_q) +
len(P_set)))

    for additional_pair_number, pair in enumerate(d_p_q):
```

```
        constrains_matrix = generate_constrains_for_columns(pair, additional_pair_number, M,  p, q, constrains_matrix)

    last_row_index = len(constrains_matrix)
    constrains_matrix = np.vstack([constrains_matrix, np.zeros(len(constrains_matrix[0]))])

    constrains_matrix[last_row_index][len(M[0])* len(M) + len(P_set) * len(d_p_q) + pair_number] = -1
    for i, pair in enumerate(d_p_q):
        constrains_matrix[last_row_index][len(M[0])* len(M) + pair_number * len(d_p_q) + i] = 1
```