Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет информационных технологий и программирования
Кафедра компьютерных технологий

Мельник Михаил Валерьевич

Применение методов решения задачи о выполнимости булевой формулы для построения минимальной филогенетической сети

Научный руководитель: аспирант кафедры КТ В. И. Ульянцев

Санкт-Петербург
2015

Содержание

Введение

Построение минимальной филогенетической сети является важной задачей филогенетики. Филогенетическая сеть используется для представления эволюционных взаимосвязей между различными биологическими видами в ретикулярной модели эволюции. В ретикулярной (сетчатой) модели эволюции, в отличие от классической (древовидной) модели, присутствует горизонтальный перенос генов, а также гибридизация между видами, что не позволяет использовать более простые структуры, такие как филогенетические деревья. Филогенетические сети активно исследуются в настоящее время [1–3].

Хотя существует несколько формальных определений филогенетических сетей, в данной работе рассматриваются только гибридизационные сети [4, 5]. Исходными данными для построения гибридизационной сети является набор из нескольких филогенетических деревьев, построенных на одном и том же множестве таксонов. Каждое филогенетическое дерево представляет собой эволюционную историю какого-то гена. Деревья могут иметь различную топологию из-за наличия ретикуляций. Задача состоит в том, чтобы построить гибридизационную сеть с минимальным количеством вершин, содержащую в себе каждое из исходных деревьев как подграф.

Большинство алгоритмов для построения гибридизационных сетей основаны на эвристиках и не гарантируют точный результат [6, 7]. Кроме того, многие алгоритмы не могут работать более чем с двумя деревьями. Недавно был представлен точный алгоритм, работающий с многими деревьями [8]. С точной и эвристической версией этого алгоритма, как с наиболее производительной на текущий момент, будет производиться сравнение.

В данной работе представлен новый способ построения минималь-

ной гибридизационной сети, основанный на сведении к задаче о выполнимости булевой формулы (SAT), гарантирующий точный результат. На основе представленного алгоритма была разработана утилита PhyloSAT, решающая поставленную задачу. Исходный код утилиты доступен на GitHub (`https://github.com/ctlab/PhyloSAT`).

Подходы, основывающиеся на сведении к задаче SAT, успешно применяются для решения различных задач, в том числе для задач филогенетики [9], построения конечных автоматов [10] и верификации [11]. Это обусловлено тем, что современные SAT-солверы хорошо оптимизированы, и способны успешно справляться с формулами из десятков тысяч переменных за несколько минут.

В ходе работы над исследованием была написана статья для международной конференции AlCoB 2015 [12].

Глава 1. Обзор предметной области

В данной главе приводятся необходимые определения, постановка задачи, проводится обзор существующих алгоритмов решения поставленной задачи.

1.1. Основные определения

Введем необходимые формальные определения, которые будут использоваться в дальнейшем.

1.1.1. Филогенетика

Филогенетика — область биологической систематики, которая занимается идентификацией и прояснением эволюционных взаимоотношений среди разных видов жизни на Земле [13].

Филогенетическим деревом называется дерево, отражающее эволюционные взаимосвязи между различными видами или другими сущностями, имеющими общего предка [14]. В данной работе будут рассматриваться подвешенные двоичные деревья. Листья филогенетического дерева отображают таксоны, а узлы представляют из себя эволюционные события: разделение предкового вида на два независимых. Примеры филогенетических деревьев показаны на Рис. 1.1.

Гибридизационная сеть — направленный ациклический граф с выделенным корнем. Гибридизационная сеть состоит из трех типов вершин - обычных вершин, ретикулярных вершин и листьев. Обычные вершины имеют одного предка и нескольких потомков. *Ретикулярные вершины* имеют более одного предка и нескольких потомков. Листья имеют одного предка и не имеют потомков. В данной работе, без потери общности, будет рассматриваться упрощенная модель гибридизационной сети, в которой обыч-

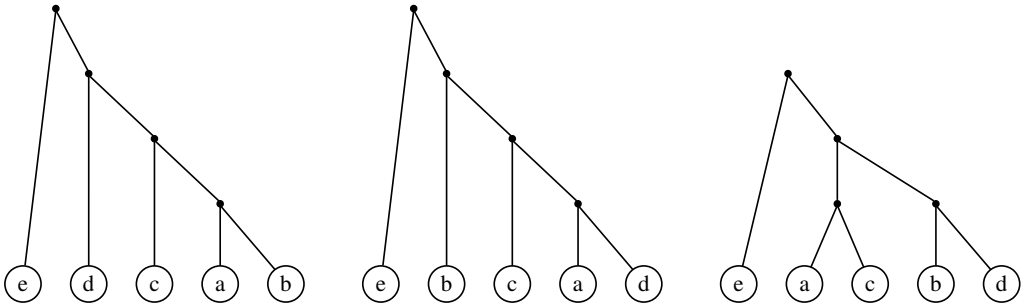

Рис. 1.1: Три филогенетических дерева над множеством таксонов {a, b, c, d, e}.

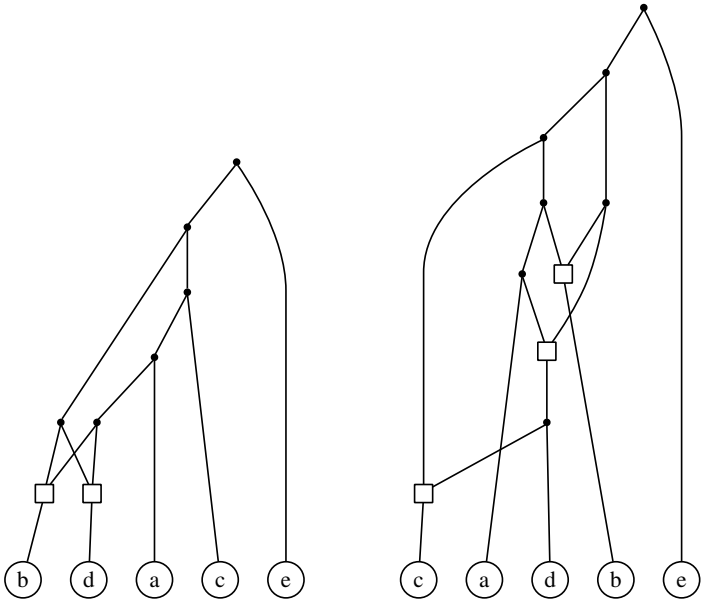

Рис. 1.2: Возможные гибридизационные сети, для деревьев из Рис. 1.1, с двумя и тремя ретикулярными событиями соответственно. Ретикулярные вершины обозначены квадратиками.

ные вершины имеют ровно двух потомков, а ретикулярные вершины имеют ровно двух предков и ровно одного потомка. Заметим, что привести произвольную гибридизационную сеть к упрощенному виду не составляет труда [6]. Примеры гибридизационных сетей показаны на Рис. 1.2.

Гибридизационную сеть можно свести к филогенетическому дереву следующим образом:

- У каждой ретикулярной вершины необходимо выбрать используемого предка, а ребро ведущее к другому предку удалить.
- Если у ретикулярной вершины удалено ребро, ведущее в ее единственного сына, то ретикулярную вершину также следует удалить.

- Стянуть все ребра, имеющие ровно одного предка и ровно одного потомка.

Гибридизационная сеть N *отображает* дерево T, если существует такой способ выбора удаляемых ребер, что после стягивания получится дерево T' изоморфное дереву T. При этом, вершины сети, которые остались после стягивания ребер, будем называть вершинами *используемыми для отображения* дерева t. Гибридизационные сети на Рис. 1.2 содержат в себе все три дерева из Рис. 1.1.

Гибридизационным числом сети N с корнем ρ называется величина $h(N) = \sum_{v \neq \rho} (d^-(v) - 1)$, где за $d^-(v)$ обозначена входящая степень вершины v. Аналогично обозначим за $d^+(v)$ исходящую степень вершины v. Заметим, что в нашей модели значение $h(N)$ равно количеству ретикулярных вершин.

Рассмотрим множество из k филогенетических деревьев $T_1, T_2, \ldots, T_k$, построенных над фиксированным множеством таксонов. Сеть N_{min} называется *минимальной гибридизационной сетью*, если она принадлежит множеству сетей, содержащих в себе все k деревьев, и при этом имеет наименьшее возможное гибридизационное число.

Множество таксонов A называется *кластером* на деревьях $T_1, T_2, \ldots, T_k$, если в каждом дереве T_i существует вершина v_i, такая, что множество листьев в поддереве v_i совпадает с множеством A.

1.1.2. Выполнимость булевой формулы

Задача о выполнимости булевой формулы (SAT) заключается в следующем: можно ли назначить всем переменным, встречающимся в формуле, значения ложь и истина так, чтобы формула стала истинной [15]. Обычно рассматриваются формулы в конъюктивной нормальной форме.

SAT-солвер — программа предназначенная для решения задачи вы-

полнимости булевой формулы.

1.2. Постановка задачи

Задача построения минимальной гибридизационной сети заключается в том, чтобы для множества филогенетических деревьев $T_1, T_2, \ldots, T_k$ построить минимальную гибридизационную сеть.

Было показано, что даже для случая двух деревьев эта задача NP-полна [16].

1.3. Обзор существующих методов

Часто, для решения задачи используются различные ухищрения, например расматриваются сети с различными структурные допущениями, например «galled networks» [17] или k-уровневые сети [18]. Кроме того, часто допускается, что сеть строится только по двум деревьям [19, 20]. Алгоритмы с такими допущениями рассматриваться не будут. Рассмотрим только алгоритмы, решающие поставленную задачу без допущений.

1.3.1. PIRN$_{\mathrm{C}}$

Единственный алгоритм, позволяющий гарантированно находить минимальную гибридизационную сеть для произвольного количества входных деревьев. Будет использоваться для сравнения с точной версией представленного алгоритма.

1.3.2. PIRN$_{\mathrm{CH}}$

Эвристическая версия алгоритма PIRN$_{\mathrm{C}}$. Не гарантирует нахождение точного решения. Будет использоваться для сравнения с неточной версией представленного алгоритма.

1.3.3. MURPAR

Быстрый эвристический алгоритм основанный на сведении к задаче линейного программирования. Не гарантирует нахождения точной минимальной сети, но позиционируется как очень быстрый алгоритм для нахождения точной верхней границы гибридизационного числа.

Глава 2. Описание алгоритма

Основная идея заключается в составлении булевой формулы для набора входных деревьев и гибридизационного числа h, которая выполнима тогда и только тогда, когда существует гибридизационная сеть N_h с гибридизационным числом h, отображающая все входные деревья. Определив диапазон возможных значений гибридизационного числа, можно перебрать их все и составить формулы для фиксированных значений гибридизационного числа. В итоге, ответом на задачу будет та из выполнимых формул, которая соответствует наименьшему гибридизационному числу.

2.1. Препроцессинг

Перед тем как приступать к непосредственному кодированию булевой формулы, применяются несколько эвристик, позволяющих разбить задачу на подзадачи, и следовательно уменьшить ее сложность. Для разбиения применяются следующие правила [9]:

1. **Сокращение поддерева.** Если существует поддерево, содержащееся в каждом из исходных деревьев, значит в этой части эволюционной истории не наблюдалось ретикуляций, и для ее отображения достаточно древовидной структуры. Поэтому во всех исходных деревьях следует заменить это поддерево на лист с новой меткой. После решения задачи, в готовой сети, следует заменить этот лист на исходное поддерево.
2. **Сокращение кластера.** Если существует кластер A, содержащийся в каждом из исходных деревьев, его также следует заменить на лист с новой меткой, а задачу построения минимальной гибридизационной сети решать для этого кластера отдельно. После решения

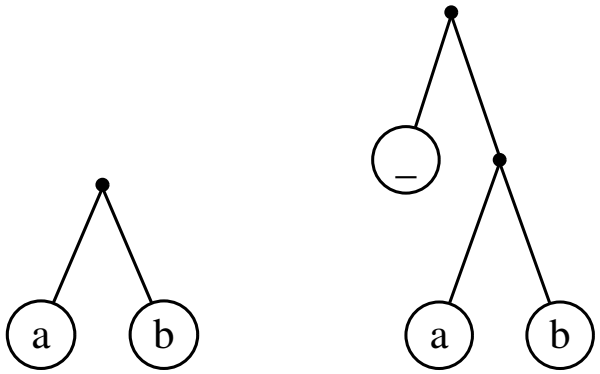

Рис. 2.1: Добавление фиктивного корня к дереву.

задачи, следует заменить этот лист в готовой сети на сеть, являющуюся решением задачи для кластера A.

Предложенный далее алгоритм предполагает, что у всех входных деревьев общий корень, но это предположение не выполняется для построенных подзадач. Поэтому следует добавить фиктивный корень ко всем деревьям в каждой из подзадач. Чтобы сохранить структуру деревьев корень добавляется вместе с новым фиктивным листом. Этот процесс проиллюстрирован на Рис. 2.1. После решения задачи, фиктивный корень и фиктивный лист следует удалить.

2.2. Перебор гибридизационного числа

Для решения задачи требуется найти такое минимальное h, что будет существовать гибридизационная сеть с гибридизационным числом равным h. Существует несколько методик перебора значения h: последовательный перебор от больших значений к маленьким, от маленьких значений к большим и двоичный поиск. В рамках данной работы были реализованы все три методики. Если минимальное гибридизационное число равно $h_{\min}$, то, как правило, наибольшее количество вычислений потребуется для решения формулы при $h = h_{\min} - 1$. Это обусловлено тем фактом, что найти какое-нибудь решение для солвера проще, чем, перебрав все варианты, убе-

диться, что решения не существует. Экспериментальные результаты подтверждают это наблюдение, и поэтому перебор от маленьких значений к большим не представляет интереса, так как его производительность значительно ниже чем у остальных методов. Результаты двоичного поиска практически повторяют результаты перебора от больших значений к маленьким, кроме случаев, в которых выполняется значительное количество проверок для значений меньших чем $h_{\min}$.

В данной работе используется перебор от больших значений к маленьким, как наиболее производительный метод. Кроме того, было экспериментально обнаружено, что у многих из получаемых подзадач гибридизационное число мало, и времени на решение таких подзадач тратится мало, поэтому используется следующая эвристика: перед началом перебора производится попытка быстрого решения со значениями h равными 0, 1, 2 и 3. В каждом из случаев солверу выделяется одна секунда на решение.

Объем вычислений можно очевидным образом сократить, если подобрать более точные границы возможных значений гибридизационного числа. Существует несколько быстрых методов, основанных на различных эвристиках и методах линейного программирования, например $\mathrm{PIRN}_{\mathrm{CH}}$ [6], RIATA-HGT [21] и MURPAR [7].

2.3. Кодирование булевой формулы

Обозначим исходное множество деревьев за T, множество таксонов за A, размер A за n, а предполагаемое гибридизационное число за h. Требуется построить булеву формулу, которая выполнима тогда и только тогда, когда существует сеть, с гибридизационным номером h, которая отображает все деревья из T.

Для начала, заметим, что исходные деревья содержат $2n-1$ вершин, а искомая сеть состоит из $2(n+k)+1$ вершин, т.к. добавляется фиктивный

корень и фиктивный лист. Среди этих вершин $n+1$ лист, k ретикулярных вершин, и $n+k$ обычных вершин.

Введем нумерацию вершин по следующим правилам:

1. Листья будут иметь номера в диапазоне $[0, n]$.
2. Обычные вершины будут иметь номера в диапазоне $[n+1, 2n+k]$.
3. Ретикулярные вершины будут иметь номера в диапазоне $[2n+k+1, 2(n+k)]$.
4. Номер любого листа или обычной вершины меньше номера ее предка.
5. У обычной вершины номер левого сына меньше номера правого сына.
6. У ретикулярной вершины номер левого предка меньше номера правого предка.

Кроме того, для каждой вершины v, введем следующие обозначения:

- $\mathrm{PC}(v)$ — множество возможных детей вершины v
- $\mathrm{PP}(v)$ — множество возможных предков вершины v
- $\mathrm{PU}(v)$ — множество вершин, которые могут находиться выше вершины v в сети

А также обозначим множество листьев за L, множество ретикулярных вершин за R, а множество обычных вершин за V.

2.3.1. Кодирование структуры сети

Чтобы закодировать структуру сети потребуются следующие переменные:

- $l_{v,u}$ и $r_{v,u}$, где $v \in V, u \in PC(v)$.
  $l_{v,u}$ ($r_{v,u}$) истинно тогда и только тогда, когда вершина u является левым (правым) ребенком вершины v.

- $p_{v,u}$, где $v \in L \cup V \backslash \{\rho\}, u \in PP(v)$.

  $p_{v,u}$ истинно тогда и только тогда, когда вершина u является предком вершины v.

- $p^l_{v,u}$ и $p^r_{v,u}$, где $v \in R, u \in PP(v)$.

  $p^l_{v,u}$ ($p^r_{v,u}$) истинно тогда и только тогда, когда вершина u является левым (правым) предком ретикулярной вершины v.

- $c_{v,u}$, где $v \in R, u \in PC(v)$.

  $c_{v,u}$ истинно тогда и только тогда, когда вершина u является ребенком ретикулярной вершины v.

Всего требуется $O((n+k)^2)$ переменных. Заметив, что $k < n$, получаем оценку на количество переменных $O(n^2)$.

Необходимо закодировать уникальность каждой переменной, т.е. что у каждой вершины есть ровно один предок и ровно один левый и правый ребенок, и аналогичные утверждения для остальных переменных. Для этого необходимо разбить утверждение вида «ровно один» на два утверждения «хотя бы один» и «не более чем один». В дальнейшем, для обозначения утверждения «хотя бы один» будет использоваться запись ALO, а для обозначения утверждения «не более чем один» будет использоваться запись AMO.

Утверждение ALO кодируется очевидным способом (на примере предков вершины v):

$$\mathrm{ALO}_p(v) = \bigvee_{u \in PP(v)} p_{v,u}$$

Это утверждение означает, что одна из переменных, отвечающих за возможного предка вершины v истинна. Для остальных переменных утверждения «хотя бы один» записываются аналогичным образом. Суммарно потребуется $O(n)$ утверждений ALO для всех переменных, где каждое из утверждений будет состоять из $O(n)$ переменных. Общий размер формулы будет равен $O(n^2)$.

Утверждение AMO можно закодировать различными способами. Самый простой способ — попарное исключение (на примере предков вершины v):

$$\mathrm{AMO}_p(v) = \bigwedge_{i,j \in PP(v)\ :\ i<j} (p_{v,i} \to \neg p_{v,j})$$

Такой метод требует $O(n^2)$ утверждений для каждой переменной. Существуют более оптимальные кодирования. В частности, в данной работе использовалось кодирование Bimander [22], которое для каждой переменной требует $O(\log_2 n)$ дополнительных переменных, но позволяет сократить количество утверждений до $O(n\log_2 n)$. Общий размер утверждений для кодирования AMO составит $O(n^2\log_2 n)$.

Утверждения ALO и AMO для различных переменных указаны в секциях 1–4 Таблицы 2.1.

Чтобы удовлетворить 5 и 6 правило нумерации вершин вводятся утверждения, запрещающие неправильную нумерацию детей обычных вершин и предков ретикулярных вершин. Эти утверждения приведены в секции 5 Таблицы 2.1.

Кроме единственности переменных необходимо связать переменные отвечающие за предков с переменными отвечающими за детей. Это делается очевидным образом: если вершина u является сыном вершины v, то вершина v должна быть предком вершины u, и наоборот. Утверждения, связывающие детей и предков разных типов вершин указаны в секциях 6–9 Таблицы 2.1.

Чтобы удовлетворить 4 правило нумерации вершин, необходимо добавить утверждения, упорядочивающие ребенка ретикулярной вершины относительно предков ретикулярной вершины. Эти утверждения указаны в секции 10 Таблицы 2.1.

Наиболее затратные утверждения — утверждения упорядочивающие детей и предков. Эти утверждения состоят из $O(n^2)$ переменных для

Таблица 2.1: Утверждения для кодирования структуры сети.

	Утверждение	Диапазон
1.1	$\mathrm{ALO}_p(v)$	$v \in V; PP(v)$
1.2	$\mathrm{AMO}_p(v)$	
2.1	$\mathrm{ALO}_l(v)$	$v \in V; PC(v)$
2.2	$\mathrm{AMO}_l(v)$	
2.3	$\mathrm{ALO}_r(v)$	$v \in V; PC(v)$
2.4	$\mathrm{AMO}_r(v)$	
3.1	$\mathrm{ALO}_c(v)$	$v \in R; PC(v)$
3.2	$\mathrm{AMO}_c(v)$	
4.1	$\mathrm{ALO}_{p^l}(v)$	$v \in R; PP(v)$
4.2	$\mathrm{ALO}_{p^r}(v)$	
4.3	$\mathrm{AMO}_{p^l}(v)$	$v \in R; PP(v)$
4.4	$\mathrm{AMO}_{p^r}(v)$	
5.1	$l_{v,u} \to \neg r_{v,w}$	$v \in V; u, w \in PC(v) : u \geq w$
5.2	$p^l_{v,u} \to \neg p^r_{v,w}$	$v \in R; u, w \in PP(v) : u \geq w$
6.1	$l_{v,u} \to p_{u,v}$	$v \in V; u \in V \cap PC(v)$
6.2	$r_{v,u} \to p_{u,v}$	
6.3	$p_{u,v} \to (l_{v,u} \vee r_{v,u})$	
7.1	$l_{v,u} \to (p^l_{u,v} \vee p^r_{u,v})$	$v \in V; u \in R \cap PC(v)$
7.2	$r_{v,u} \to (p^l_{u,v} \vee p^r_{u,v})$	
7.3	$p^l_{u,v} \to (l_{v,u} \vee r_{v,u})$	
7.4	$p^r_{u,v} \to (l_{v,u} \vee r_{v,u})$	
8.1	$c_{v,u} \to p_{u,v}$	$v \in R; u \in V \cap PC(v)$
8.2	$p_{u,v} \to c_{v,u}$	
9.1	$c_{v,u} \to (p^l_{u,v} \vee p^r_{u,v})$	$v \in R; u \in R \cap PC(v)$
9.2	$p^l_{u,v} \to c_{v,u}$	
9.3	$p^r_{u,v} \to c_{v,u}$	
10.1	$c_{v,u} \to \neg p^l_{v,w}$	$v \in R; u \in PC(v); w \in PP(v) : u \geq w$
10.2	$c_{v,u} \to \neg p^r_{v,w}$	

каждой вершины. Суммарный размер всех введенных утверждений составляет $O(n^3)$.

2.3.2. Кодирование отображения вершин деревьев на вершины сети

Осталось позаботиться о том, чтобы сеть отображала все исходные деревья. Для этого введем следующие переменные:

- $x_{t,v_t,v}$, где $t \in T, v_t \in V(t), v \in V$.

  $x_{t,v_t,v}$ истинно тогда и только тогда, когда вершина v соответствует вершине v_t из дерева t, то есть переменные x для каждого дерева t являются инъективным отображением множества вершин этого де-

рева в множество вершин сети. Пример части такого отображения показан на Рис 2.2. Заметим, что листья всех деревьев биективно соответствуют листьям сети т.к. множество таксонов одинаково для всех деревьев и сети. Поэтому, переменные x для листьев не вводятся.

- $d_{t,v}$, где $t \in T, v \in R$.
  $d_{t,v}$ истинно тогда и только тогда, когда для отображения дерева t требуется выбрать левого предка у ретикулярной вершины v.
- $u^r_{t,v}$, где $t \in T, v \in R$.
  $u^r_{t,v}$ истинно тогда и только тогда, когда ребро в ребенка ретикулярной вершины v используется для отображения дерева t. Если ребенок ретикулярной вершины v тоже является ретикулярной вершиной, то у него может быть зафиксировано направление не совпадающее с направлением его предка v, и тогда, при отображении дерева t, вершина v, вместо стягивания в ребро, полностью удалится.
- $u_{t,v}$, где $t \in T, v \in V$.
  $u_{t,v}$ истинно тогда и только тогда, когда вершина v используется для отображения дерева t.
- $a_{t,v,u}$, где $t \in T, v \in V, u \in PU(v)$.
  $a_{t,v,u}$ истинно тогда и только тогда, когда вершина u является первой вершиной на пути от вершины v к корню, которая используется для отображения дерева t. Другими словами, вершина u является предком для вершины v, при отображении дерева t. Заметим, что вершина v может не использоваться для отображения. На Рис. 2.2 вершина u является предком для всех вершин ниже нее, при отображении дерева t.

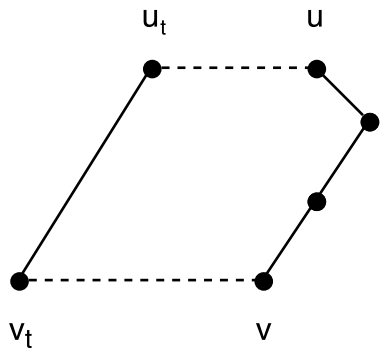

Рис. 2.2: Часть инъективного отображения вершин дерева (слева) на вершины сети (справа). Инъективное отображение изображено пунктирной линией. При отображении дерева, все ребра на пути от u до v стянутся в одно ребро, которое будет соответствовать ребру из u_t в v_t.

Таблица 2.2: Утверждения для кодирования отображения вершин деревьев на вершины сети.

	Утверждение	Диапазон
1.1	$\mathrm{ALO}_{a_t}(v)$	$t \in T; v \in V \cup L \cup R; PU(v)$
1.2	$\mathrm{AMO}_{a_t}(v)$	
2.1	$\mathrm{ALO}_{x_t}(v_t)$	$t \in T; v_t \in V(t); V$
2.2	$\mathrm{AMO}_{x_t}(v_t)$	
2.3	$\mathrm{AMO}_{x_t}(v)$	$t \in T; v \in V; V(t)$
3.1	$x_{t,v_t,v} \to u_{t,v}$	$t \in T; v \in V; v_t \in V(t)$
3.2	$x_{t,\rho_t,\rho}$	$t \in T; \rho_t = \rho(t)$
4.1	$x_{t,u_t,u} \to a_{t,v,u}$	$t \in T; v \in L; u \in PP(v); u_t = p(v_t)$
4.2	$a_{t,v,u} \to x_{t,u_t,u}$	
4.3	$(x_{t,v_t,v} \wedge x_{t,u_t,u}) \to a_{t,v,u}$	$t \in T; v \in V; u \in PP(v); v_t \in V(t); u_t = p(v_t)$
4.4	$(x_{t,v_t,v} \wedge a_{t,v,u}) \to x_{t,u_t,u}$	
4.5	$x_{t,v_t,v} \to \neg x_{t,u_t,u}$	$t \in T; v \in V; u \in V; v_t \in V(t); u_t = p(v_t) : u < v$
5.1	$\neg x_{t,v_t,v}$	$t \in T; v \in V; v_t \in V(t) : v_t < \mathrm{size}(\mathrm{subtree}(v_t))$
5.2	$\neg x_{t,v_t,v}$	$t \in T; v \in V; v_t \in V(t) : v_t > \mathrm{size}(t) - \mathrm{depth}(v_t)$
5.3	$\neg x_{t,v_t,v} \vee \neg x_{t',v_{t'},v}$	$t, t' \in T; v \in V; v_t \in V(t); v_{t'} \in V(t')$: множества таксонов в поддеревьях v_t и $v_{t'}$ не пересекаются

Введено $O(tn^2)$ новых переменных. Значит всего требуется $O(tn^2)$ переменных.

Для начала введем ALO и AMO утверждения для переменных x и a, так как необходимо обеспечить их уникальность. Заметим, что для переменных x кроме утверждения «не более чем одна вершина из сети соответствует вершине из дерева» так же верное и утверждение «не более чем одна вершина из дерева соответствует вершине из сети». Эти утверждения указаны в секциях 1–2 Таблицы 2.2.

Благодаря фиктивным корням, известно, что корень сети ρ соответствует корням каждого из деревьев, то есть для любого t утверждение $x_{t,\rho_t,\rho}$ истинно. Кроме того, очевидно, что если вершина v соответствует какой-то вершине v_t в дереве t, то вершина v используется для отображения дерева t, то есть истинно утверждение $x_{t,v_t,v} \rightarrow u_{t,v}$. Переменные u введены для уменьшения размера формул, указанных в следующей секции. Эти утверждения указаны в секции 3 Таблицы 2.2.

Переменные x и a связаны следующим образом: если известно, что вершина v из сети соответствует вершине v_t из дерева t, а предку вершины v_t, вершине u_t, соответствует вершина u, то вершина u является предком вершины v при отображении дерева t. И наоборот, если известно соответствие вершин v и v_t, и известно, что вершина u является предком вершины v при отображении дерева t, то вершина u должна соответствовать предку вершины v_t в дереве t. Заметим, что для листов не требуется явно указывать соответствие нижних вершин, так как они неявно подразумеваются. Кроме того, чтобы удовлетворялось 4 правило нумерации, необходимо чтобы локальный порядок нумерации предков в деревьях сохранялся и в сети. Другими словами, не допускается, чтобы вершина u с меньшим номером соответствовала предку, а вершина v с большим номером соответствовала сыну. Эти утверждения приведены в секции 4 Таблицы 2.2.

Также используются следующие эвристические ограничения, относящиеся к структуре деревьев:

- номер вершины v не может быть меньше, чем количество вершин в ее поддереве.
- номер вершины v не может быть больше чем номер корня минус глубина вершины v.
- если поддеревья вершины v_t и вершины $v_{t'}$ из деревьев t и t' соответственно содержат не пересекающиеся множества таксонов, то

эти вершины не могут соответствовать одной и той же вершине v из сети.

Эти утверждения приведены в секции 5 Таблицы 2.2.

Наиболее затратными утверждениями являются утверждения, связывающие переменные x и a. Эти утверждения состоят из $O(n^2)$ переменных для каждой вершины и каждого дерева. Суммарный размер всех введенных утверждений составляет $O(tn^3)$.

2.3.3. Кодирование трансляции связей «предок–ребенок» деревьев в сеть

Необходимо отобразить явные связи «предок-ребенок» в деревьях на сеть. Рассмотрим вершину сети v и ее предка вершину u. Если известно, что вершина u используется для отображения дерева t, то вершина u является предком вершины v, при отображении дерева t. И наоборот, если известно что вершина u является предком вершины v, при отображении дерева t, то вершина u используется для отображения дерева t. Если известно, что вершина u не используется для отображения дерева, то вершины u и v имеют одного и того же предка, при отображении дерева t. Это можно выразить следующим образом: если известно, что вершина w является предком u, при отображении дерева t, то вершина w является так же и предком v, при отображении дерева t. И наоборот, если известно, что вершина w является предком v при отображении дерева t, то вершина w является так же и предком u, при отображении дерева t. Эти утверждения указаны в секции 1 Таблицы 2.3.

Теперь рассмотрим ретикулярную вершину v и ее предка u.

Если u тоже является ретикулярной вершиной, и направление u совпадает с выбранным направлением для дерева t, то вершины v и u имеют одного и того же предка, при отображении дерева t. Это выражается аналогично предыдущему утверждению. Если u является обычной вершиной, ее

Таблица 2.3: Утверждения для трансляции отношений «предок–ребенок» исходных деревьев в сеть.

	Утверждение	Диапазон
1.1	$(p_{v,u} \wedge u_{t,u}) \rightarrow a_{t,v,u}$	$t \in T; v \in V \cup L; u \in V \cap PP(v)$
1.2	$(p_{v,u} \wedge a_{t,v,u}) \rightarrow u_{t,u}$	
1.3	$(p_{v,u} \wedge \neg u_{t,u} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	$t \in T; v \in V \cup L; u \in V \cap PP(v); w \in PP(u)$
1.4	$(p_{v,u} \wedge \neg u_{t,u} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	
2.1	$(p^l_{v,u} \wedge d_{t,v} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	$t \in T; v \in R; u \in R \cap PP(v); w \in PU(u)$
2.2	$(p^l_{v,u} \wedge d_{t,v} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	
2.3	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	
2.4	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	
2.5	$(p^l_{v,u} \wedge d_{t,v} \wedge u_{t,u}) \rightarrow a_{t,v,u}$	$t \in T; v \in R; u \in V \cap PP(v)$
2.6	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge u_{t,u}) \rightarrow a_{t,v,u}$	
2.7	$(p^l_{v,u} \wedge d_{t,v} \wedge \neg u_{t,u} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	$t \in T; v \in R; u \in V \cap PP(v); w \in PU(u)$
2.8	$(p^l_{v,u} \wedge d_{t,v} \wedge \neg u_{t,u} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	
2.9	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge \neg u_{t,u} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	
2.10	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge \neg u_{t,u} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	
3.1	$(p^l_{v,u} \wedge \neg d_{t,v}) \rightarrow \neg u^r_{t,u}$	$t \in T; v \in R; u \in R \cap PP(v)$
3.2	$(p^r_{v,u} \wedge d_{t,v}) \rightarrow \neg u^r_{t,u}$	
3.3	$(p^l_{v,u} \wedge \neg d_{t,v}) \rightarrow \neg u_{t,u}$	$t \in T; v \in R; u \in V \cap PP(v)$
3.4	$(p^r_{v,u} \wedge d_{t,v}) \rightarrow \neg u_{t,u}$	
4.1	$(p^l_{v,u} \wedge d_{t,v} \wedge u^r_{t,v}) \rightarrow u^r_{t,u}$	$t \in T; v \in R; u \in R \cap PP(v)$
4.2	$(p^r_{v,u} \wedge \neg d_{t,v} \wedge u^r_{t,v}) \rightarrow u^r_{t,u}$	
4.3	$(c_{u,v} \wedge \neg u^r_{t,v}) \rightarrow \neg u^r_{t,u}$	
4.4	$(p^l_{v,u} \wedge \neg u^r_{t,v}) \rightarrow \neg u_{t,u}$	$t \in T; v \in R; u \in V \cap PP(v)$
4.5	$(p^r_{v,u} \wedge \neg u^r_{t,v}) \rightarrow \neg u_{t,u}$	
4.6	$c_{v,u} \rightarrow u^r_{t,v}$	$t \in T; v \in R; u \in (V \cup L) \cap PC(v)$
5.1	$p_{v,u} \rightarrow \neg a_{t,u,w}$	$t \in T; v \in V \cup L; u \in R \cap PP(v);$ $w \in PU(u) : w \leq v$
5.2	$(p_{v,u} \wedge a_{t,u,w}) \rightarrow a_{t,v,w}$	$t \in T; v \in V \cup L; u \in R \cap PP(v);$ $w \in PU(u) : w > v$
5.3	$(p_{v,u} \wedge a_{t,v,w}) \rightarrow a_{t,u,w}$	$t \in T; v \in V \cup L; u \in R \cap PP(v);$ $w \in PU(u) : w > v$

направление совпадает с выбранным направлением для дерева t и при этом используется для отображения дерева t, то вершина u является предком вершины v при отображении дерева t. Если же вершина u не используется, то вершины v и u имеют одного и того же предка, при отображении дерева t. Эти утверждения указаны в секции 2 Таблицы 2.3.

Если выбранное направление для дерева t не соответствует направлению u, то вершина u не используется для отображения дерева t. Соответствующие утверждения указаны в секции 3 Таблицы 2.3

Рассмотрим теперь связанные ретикулярные вершины, вершину–ребенка v и вершину–предка u. Если направление u совпадает с направ-

лением выбранным в вершине v, и v используется для отображения дерева t, то u тоже используется для отображения дерева t. В случае, когда ретикулярная вершина v не используется, то оба ее предка тоже не используются. Если ребенок ретикулярной вершины v является обычной вершиной, то вершина v обязательно используется, так как нет другого способа попасть в ее ребенка. Эти утверждения указаны в секции 4 Таблицы 2.3.

В секции 5 Таблицы 2.3 указаны утверждения, отвечающие за корректность нумерации вершин для предков, при отображении дерева t.

Наиболее затратными утверждениями являются утверждения из секции 2 Таблицы 2.3. Эти утверждения состоят из $O(n^2)$ переменных для каждой вершины и каждого дерева. Суммарный размер всех введенных утверждений составляет $O(tn^3)$.

2.4. Решение булевой формулы и постпроцессинг

Выбор подходящего солвера является темой для отдельного исследования. Некоторые эксперименты уже проводились [9], и результаты отдельных солверов значительно превосходят результаты других. Не смотря на это, в рамках данной работы такие эксперименты не проводились. Для решения булевой формулы используется солвер CryptoMiniSat 4.2.0 [23].

После того, как решена формула соответствующая минимальной гибридизационной сети, по полученным значениям переменных восстанавливается структура сети, удаляются фиктивный корень и фиктивный лист. Затем, сети, полученные в результате решения подзадач, объединяются в одну сеть, являющуюся решением исходной задачи. Итоговая сеть сохраняется в формате Graphviz [24].

2.5. Альтернативный способ кодирования структуры сети

Предложенный способ кодирования структуры сети можно улучшить, если воспользоваться наблюдением, что можно последовательно добавлять в одно из исходных деревьев ретикулярную и обычную вершину, получая каждый раз корректную сеть. Таким образом, проведя такую операцию k раз, получится сеть с гибридизационным числом k. После этого необходимо добавить утверждения из разделов 2.3.2 и 2.3.3, связав с оставшиеся деревья с сетью. Потенциальные плюсы такого подхода в том, что сеть задается меньшим количеством переменных и утверждений, что, в теории, позволит ускорить работу солвера.

Для начала условимся, что добавление ретикулярного события в дерево описывается следующим образом:

1. Выбираются два ребра, которые будут подразбиты для вставки в них новых вершин.
2. В одно из ребер вставляется обычная вершина, в другое вставляется ретикулярная вершина.
3. Вторым предком новой ретикулярной вершины становится соответственно новая обычная вершина.

Чтобы нумеровать ребра воспользуемся следующей схемой: рассмотрим обычную вершину v; ребро соединяющее вершину v с ее предком будет иметь тот же номер, что и вершина v. В случае ретикулярной вершины u, для описания ребер ведущих в предков понадобится два номера, поэтому номера ребер ведущих к предкам ретикулярной вершины u будут иметь номера $2n-1+k+i$ и $2n+k+i$, если u является i–й по порядку ретикулярной вершиной.

Для того, чтобы закодировать эти события, понадобятся следующие

переменные:

- $h_{v,e}$, где $v \in R, e \in E$.

  $h_{v,e}$ истинно тогда и только тогда, когда ретикулярная вершина v вставлена в ребро e.

- $r_{v,e}$, где $v \in V, e \in E$.

  $r_{v,e}$ истинно тогда и только тогда, когда обычная вершина v вставлена в ребро e.

Наиболее простой способ кодирования заключается в том, чтобы ввести следствие из новых утверждений, в утверждения, представленные в разделе 2.3.1. Такой подход не уменьшит общее количество утверждений, но позволит уменьшить количество реально значимых, независимых утверждений, что вместе с выбором подходящего солвера, позволит значительно ускорить перебор.

Для того, чтобы это осуществить, необходимо лишь указать, какие из переменных, отвечающих за предков будут истинны, после введения всех ретикулярных событий.

Для начала заметим, что нет никакого смысла добавлять одновременно и ретикулярную и обычную вершину в одно и то же ребро. Кроме того, понадобятся уже известные ALO и AMO утверждения для того, чтобы обеспечить уникальность переменных h и r. Эти утверждения приведены в секции 1 Таблицы 2.4.

Для удобства, введем формулы, означающие, что вершина u_i (v_i) была добавлена последней в ребро e:

$$\mathrm{LAST}_h(i,e) = h_{u_i,e} \wedge \neg h_{u_{i+1},e} \wedge \neg r_{v_{i+1},e} \ldots \neg h_{u_k,e} \wedge \neg r_{v_k,e}$$

$$\mathrm{LAST}_r(i,e) = r_{v_i,e} \wedge \neg h_{u_{i+1},e} \wedge \neg r_{v_{i+1},e} \ldots \neg h_{u_k,e} \wedge \neg r_{v_k,e}$$

Теперь рассмотрим обычную вершину v. Если в ребро над этой вершиной не добавлялось никаких новых вершин, то ее предок не изменился

Таблица 2.4: Утверждения для кодирования структуры сети.

	Утверждение	Диапазон
1.1	$\neg h_{u_i,e} \vee \neg r_{v_i,e}$	$v_i \in V; u_i \in R; e \in E$
1.2	$\mathrm{ALO}_h(v)$	$v \in R; E(v)$
1.3	$\mathrm{AMO}_h(v)$	
1.4	$\mathrm{ALO}_r(v)$	$v \in V; E(v)$
1.5	$\mathrm{AMO}_r(v)$	
2.1	$(\neg h_{u_0,e} \wedge \neg r_{v_0,e} \ldots \neg h_{u_k,e} \wedge \neg r_{v_k,e}) \to p_{v_e,u_e}$	$u \in R; v \in V; e \in E; v_e \in e; u_e = p(v_e)$
2.2	$\mathrm{LAST}_h(i,e) \to p_{v_e,u_i}$	$u \in R; v \in V; e \in E; v_e \in e$
2.3	$\mathrm{LAST}_r(i,e) \to p_{v_e,v_i}$	
3.1	$\mathrm{LAST}_h(i,e) \to (lp_{u_i,u_e} \vee lp_{u_i,v_i})$	$u \in R; v \in V; e \in E; u_e \in e$
3.2	$\mathrm{LAST}_h(i,e) \to (rp_{u_i,u_e} \vee rp_{u_i,v_i})$	
3.3	$\mathrm{LAST}_h(i,e) \to lp_{u_e,u_i}$	$u \in R; e \in E; u_e \in e:$
3.4	$\mathrm{LAST}_r(i,e) \to lp_{u_e,v_i}$	e ведет в левого предка
3.5	$\mathrm{LAST}_h(i,e) \to rp_{u_e,u_i}$	$u \in R; e \in E; u_e \in e$
3.6	$\mathrm{LAST}_r(i,e) \to rp_{u_e,v_i}$	e ведет в правого предка

и является ее предком в дереве. Иначе, ее предком будет вершина, добавленная самой последней в ребро над ней. Так как вершины добавляются в сеть в порядке возрастания, то достаточно утверждения, что на i–м шаге вершина была добавлена в ребро e, а на шагах $i+1, i+2, \ldots, k$ в ребро e ничего не было добавлено. Эти утверждения приведены в секции 2 Таблицы 2.4.

Аналогично для ретикулярных вершин. Если в ребра, ведущие в предков ретикулярной вершины v ничего не было добавлено, то эти ребра ведут либо в обычную вершину добавленную на том же шаге, либо в сответствующего предка из дерева. Иначе, соответствующим предком будет вершина, добавленная самой последней в соответствующее ребро. Эти утверждения приведены в секции 3 Таблицы 2.4.

К сожалению, на практике эта модификация не принесла ожидаемого ускорения, поэтому результаты ее тестирования не приведены в следующей главе. Возможно, подбор подходящего солвера принесет какие-то результаты. Кроме того, возможен вариант сведения, не использующий утверждения из раздела 2.3.1, кодирующий всю структуру сети только переменными предложенными в этой главе. Этот вариант сведения будет исследован в дальнейшей работе.

Глава 3. Тестирование и результаты

Для тестирования предложенного подхода была написана программа PhyloSAT, реализующая описанное сведение. Для написания использовался язык программирования Java 7. Программа собирается в JAR-файл, не требует никаких зависимостей и может выполняться на любой операционной системе с установленной средой выполнения Java. Кроме минимально необходимых параметров для указания входных и выходных файлов, поддерживаются также параметры, позволяющие выполнить поиск сети с фиксированным гибридизационным числом. Это, в частности, позволяет использовать PhyloSAT для построения конкретных гибридизационных сетей, или для доказательства того, что таких сетей не существует.

3.1. Описание экспериментов

Для экспериментального сравнения использовался реальный набор данных, соответствующий злаковым травам [25]. Набор состоит из 57 тестов. Каждый тест содержит от 2 до 6 деревьев и от 5 до 47 таксонов. Все эксперименты проводились на компьютере с процессором AMD Phenom II X6 1090T 3.2 GHz и операционной системой Ubuntu 14.04. Каждый тест запускался с лимитом по времени 1000 секунд. Для сравнения производительности в таких же условиях запускались программы $\mathrm{PIRN_{CH}}$ и $\mathrm{PIRN_{C}}$. Неточная версия алгоритма реализована следующим образом: по истечению заданного лимита времени выводится наилучший найденный ответ.

3.2. Результаты

Тестовый набор состоит из различных по сложности тестов. Так, например, 9 тестов не были решены ни одним из алгоритмов за отведенной

Таблица 3.1: Результаты тестирования точных алгоритмов.

Алгоритм	Решено тестов
PhyloSAT	36
PIRN_C	29

Таблица 3.2: Результаты тестирования неточных алгоритмов.

Алгоритм	Решено тестов
PhyloSAT	48
PIRN_CH	43

время. С другой стороны многие тесты имели тривиальное решение, и все тестируемые алгоритмы выдавали моментальный ответ.

3.2.1. Точный алгоритм

PIRN_C смог решить 29 тестов из 57. Оказалось, что во всех решенных тестах гибридизационное число минимальной сети не превосходит 5. Следовательно, PIRN_C не подходит для построения гибридизационных сетей с большим гибридизационным числом.

PhyloSAT смог решить на 7 тестов больше, чем PIRN_C. Кроме того, PhyloSAT оказался быстрее чем PIRN_C на всех тестах, на которых были получены ответы. Результаты тестирования точных алгоритмов кратко суммированы в Таблице 3.1.

3.2.2. Неточный алгоритм

Из описания реализации понятно, что неточная версия PhyloSAT способна решить все те тесты, что и точная версия, плюс выдает какие-то, возможно не минимальные, гибридизационные сети для остальных тестов. PhyloSAT смог решить 48 тестов, что на 5 тестов больше, чем PIRN_CH. По времени работы явного лидера нет, так как некоторые тесты PIRN_CH выполняет быстрее, чем PhyloSAT. Результаты тестирования неточных алгоритмов суммированы в Таблице 3.2.

Наиболее интересными являются различия между неточными ал-

Таблица 3.3: Сравнение результатов неточных алгоритмов на 12 тяжелых тестах.

Алгоритм	Лучший результат	Лучшее время
PhyloSAT	3	2
PIRN_{CH}	2	5

горитмами на тех тестах, которые не смогли решить точные алгоритмы. Всего таких тестов 12. В трех случаях PhyloSAT нашел более оптимальную сеть, в двух случаях менее оптимальную, а в оставшихся семи случаях алгоритмы представили одинаковые сети. В двух случаях PhyloSAT оказался быстрее и медленнее в пяти случаях. Краткое сравнение этих тестов показано в Таблице 3.3. Подробная таблица с результатами тестирования PhyloSAT и PIRN_{CH} приведена в Приложении 2.

Заметим, что во многих тестах оценка нижней границы гибридизационного числа, которое предоставляет PIRN_{CH} реализуется и является фактическим минимумом. Сравнение гибридизационных чисел найденных сетей с соответствующими оценками нижних границ приведены в Таблице 3.4. В 4 из 12 тестов PhyloSAT нашел оптимальную сеть, но не успел доказать ее минимальность за отведенное время, так как, как упоминалось выше, наибольшее время занимает поиск сети с гибридизационным числом на единицу меньшим минимального. Из этого наблюдения следует, что время работы алгоритма значительно уменьшится в тех случаях, когда нижняя оценка гибридизационного числа совпадает с реальным минимумом, так как не надо будет доказывать его минимальность.

Примеры некоторых входных деревьев и построенных для них минимальных сетей представлены в Приложении 1.

Таблица 3.4: Сравнение полученных гибридизационных чисел с теоретической нижней границей, предоставленной алгоритмом PIRN

Тест	Полученный ответ	Нижняя граница
RbclRpoc	7	7
NdhfPhytIts	13	11
NdhfPhytRpoc	8	6
NdhfRbclRpoc	12	10
NdhfWaxyIts	8	7
PhytRbclIts	9	7
PhytRpocIts	7	7
RbclWaxyIts	6	6
NdhfPhytRbclRpoc	9	7
NdhfPhytRpocIts	10	7
NdhfRbclWaxyIts	6	6
PhytRbclRpocIts	9	6

Заключение

В данной работе предложен алгоритм построения минимальной гибридизационной сети из нескольких филогенетических деревьев, гарантирующий точный результат. Предложенный алгоритм был реализован в программе PhyloSAT, исходный код которой доступен на GitHub (`https://github.com/ctlab/PhyloSAT`). Представленный алгоритм превосходит точный алгоритм $\mathrm{PIRN_C}$ во всех проведенных экспериментах и даже составляет конкуренцию эвристическому алгоритму $\mathrm{PIRN_{CH}}$, который не гарантирует точный результат.

В работе были рассмотрены возможные способы улучшения предложенного алгоритма, которые могут значительно увеличить его производительность. В частности, возможно уточнение перебираемых границ гибридизационного числа и тщательный выбор наиболее подходящего SAT-солвера.

Тем не менее, задача нахождения точного решения минимальной гибридизационной сети, в случае больших гибридизационных чисел, слишком трудоемка, и все еще не может быть решена за разумное время.

Источники

[1] Huson D. H., Rupp R., Scornavacca C. Phylogenetic networks: concepts, algorithms and applications. Cambridge University Press, 2010.

[2] Morrison D. A. Introduction to phylogenetic networks // RJR Productions. 2011.

[3] Nakhleh L. Evolutionary phylogenetic networks: models and issues // Problem solving handbook in computational biology and bioinformatics. Springer, 2011. P. 125–158.

[4] Semple C. Hybridization networks. Department of Mathematics and Statistics, University of Canterbury, 2006.

[5] Chen Z.-Z., Wang L. Hybridnet: a tool for constructing hybridization networks // Bioinformatics. 2010. Vol. 26, no. 22. P. 2912–2913.

[6] Wu Y. Close lower and upper bounds for the minimum reticulate network of multiple phylogenetic trees // Bioinformatics. 2010. Vol. 26, no. 12. P. i140–i148.

[7] Park H. J., Nakhleh L. MURPAR: a fast heuristic for inferring parsimonious phylogenetic networks from multiple gene trees // Bioinformatics Research and Applications. Springer, 2012. P. 213–224.

[8] Wu Y. An algorithm for constructing parsimonious hybridization networks with multiple phylogenetic trees // Journal of Computational Biology. 2013. Vol. 20, no. 10. P. 792–804.

[9] Bonet M. L., John K. S. Efficiently calculating evolutionary tree measures using SAT // Theory and Applications of Satisfiability Testing-SAT 2009. Springer, 2009. P. 4–17.

[10] Heule M. J., Verwer S. Exact DFA identification using SAT solvers // Grammatical Inference: Theoretical Results and Applications. Springer, 2010. P. 66–79.

[11] Bounded model checking / A. Biere, A. Cimatti, E. M. Clarke et al. // Advances in computers. 2003. Vol. 58. P. 117–148.

[12] Ulyantsev V., Melnik M. Constructing Parsimonious Hybridization Networks from Multiple Phylogenetic Trees Using a SAT-solver // To appear in proceeding of the International Conference on Algorithms for Computational Biology. 2015.

[13] Филогенетика. Wikipedia, The Free Encyclopedia. URL: https://ru.wikipedia.org/wiki/Филогенетика.

[14] Филогенетическое дерево. Wikipedia, The Free Encyclopedia. URL: https://ru.wikipedia.org/wiki/Филогенетическое_дерево.

[15] Задача выполнимости булевых формул. Wikipedia, The Free Encyclopedia. URL: https://ru.wikipedia.org/wiki/Задача_выполнимости_булевых_формул.

[16] Bordewich M., Semple C. Computing the minimum number of hybridization events for a consistent evolutionary history // Discrete Applied Mathematics. 2007. Vol. 155, no. 8. P. 914–928.

[17] Huson D. H., Klöpper T. H. Beyond galled trees-decomposition and computation of galled networks // Research in Computational Molecular Biology / Springer. 2007. P. 211–225.

[18] Constructing level-2 phylogenetic networks from triplets / L. van Iersel, J. Keijsper, S. Kelk et al. // IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB). 2009. Vol. 6, no. 4. P. 667–681.

[19] Fast computation of minimum hybridization networks / B. Albrecht, C. Scornavacca, A. Cenci et al. // Bioinformatics. 2012. Vol. 28, no. 2. P. 191–197.

[20] Wu Y., Wang J. Fast computation of the exact hybridization number of two phylogenetic trees // Bioinformatics Research and Applications. Springer, 2010. P. 203–214.

[21] Nakhleh L., Ruths D., Wang L.-S. RIATA-HGT: a fast and accurate heuristic for reconstructing horizontal gene transfer // Computing and Combinatorics. Springer, 2005. P. 84–93.

[22] Hölldobler S., Nguyen V. An Efficient Encoding of the At-Most-One Constraint: Tech. Rep.: : Technical report, KRR Group 2013-04, Technische Universität Dresden, 01062 Dresden, Germany, 2013.

[23] Soos M. CryptoMiniSat 4. URL: http://www.msoos.org/cryptominisat4/.

[24] Gansner E. R., North S. C. An open graph visualization system and its applications to software engineering // SOFTWARE - PRACTICE AND EXPERIENCE. 2000. Vol. 30, no. 11. P. 1203–1233. URL: http://www.graphviz.org.

[25] Phylogeny and subfamilial classification of the grasses (Poaceae) / G. P. W. Group, N. P. Barker, L. G. Clark et al. // Annals of the Missouri Botanical Garden. 2001. P. 373–457.

Приложение 1

Примеры полученных решений

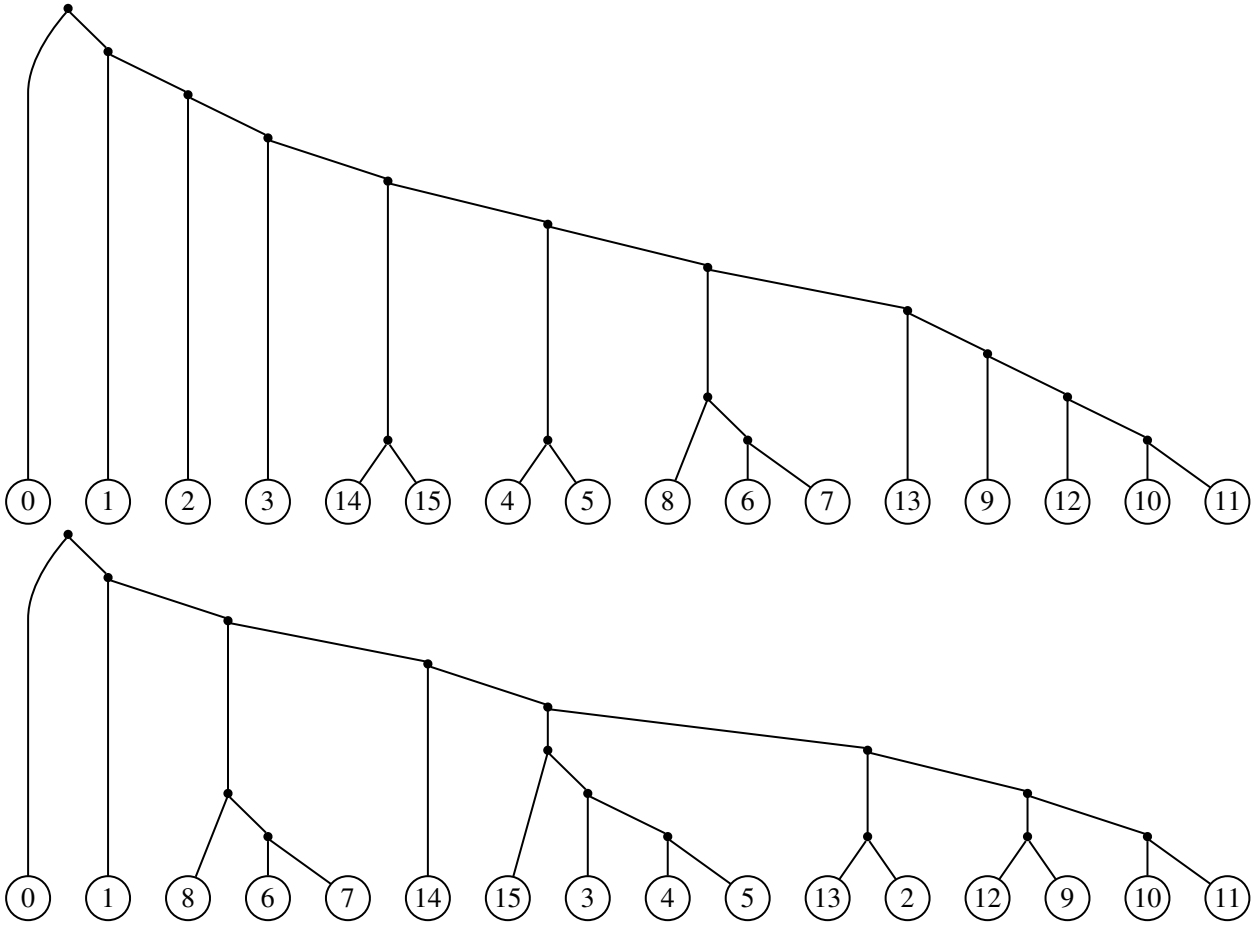

Рис. 1.1: Входные деревья для теста WaxyIts.

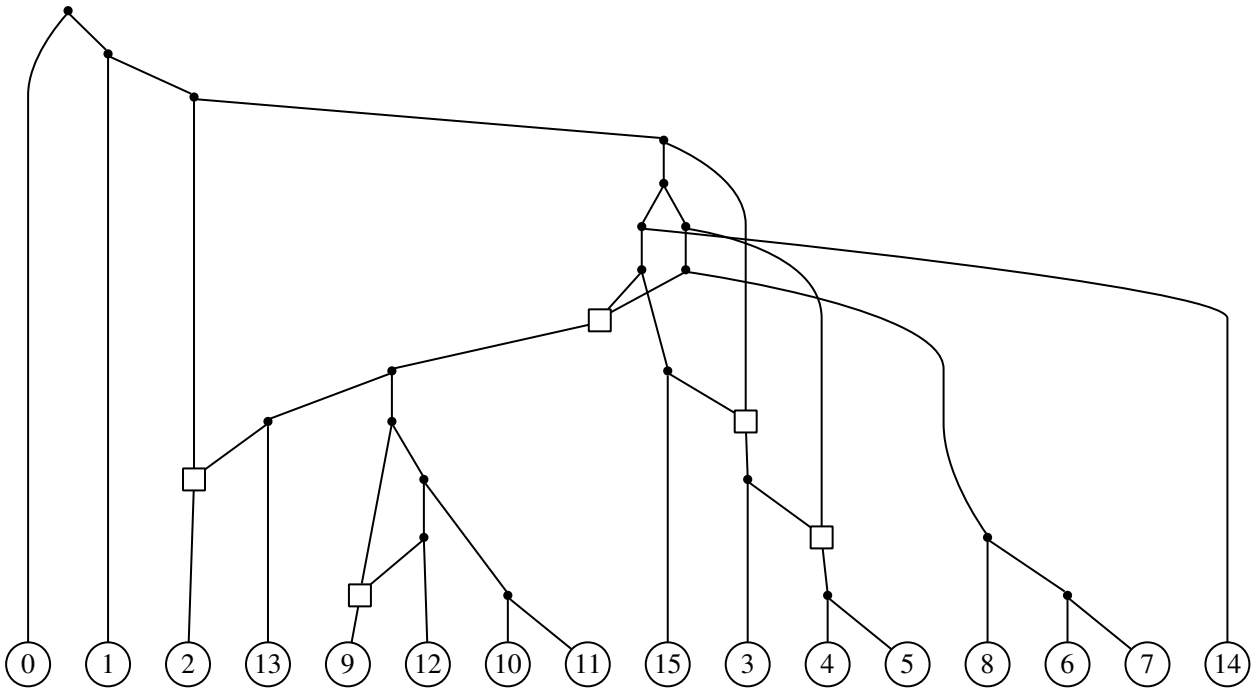

Рис. 1.2: Минимальная гибридизационная сеть для теста WaxyIts.

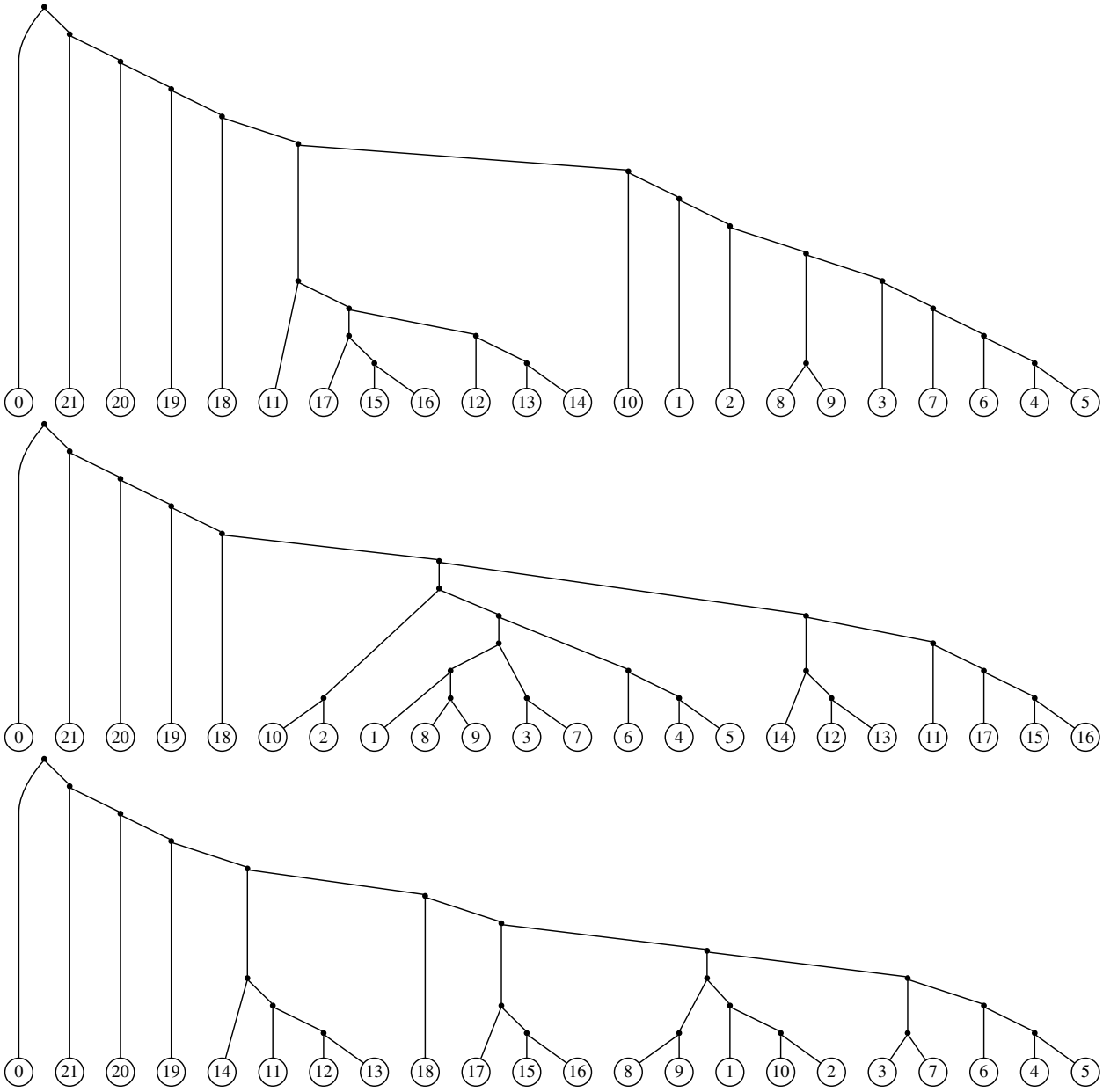

Рис. 1.3: Входные деревья для теста NdhfPhytRbcl.

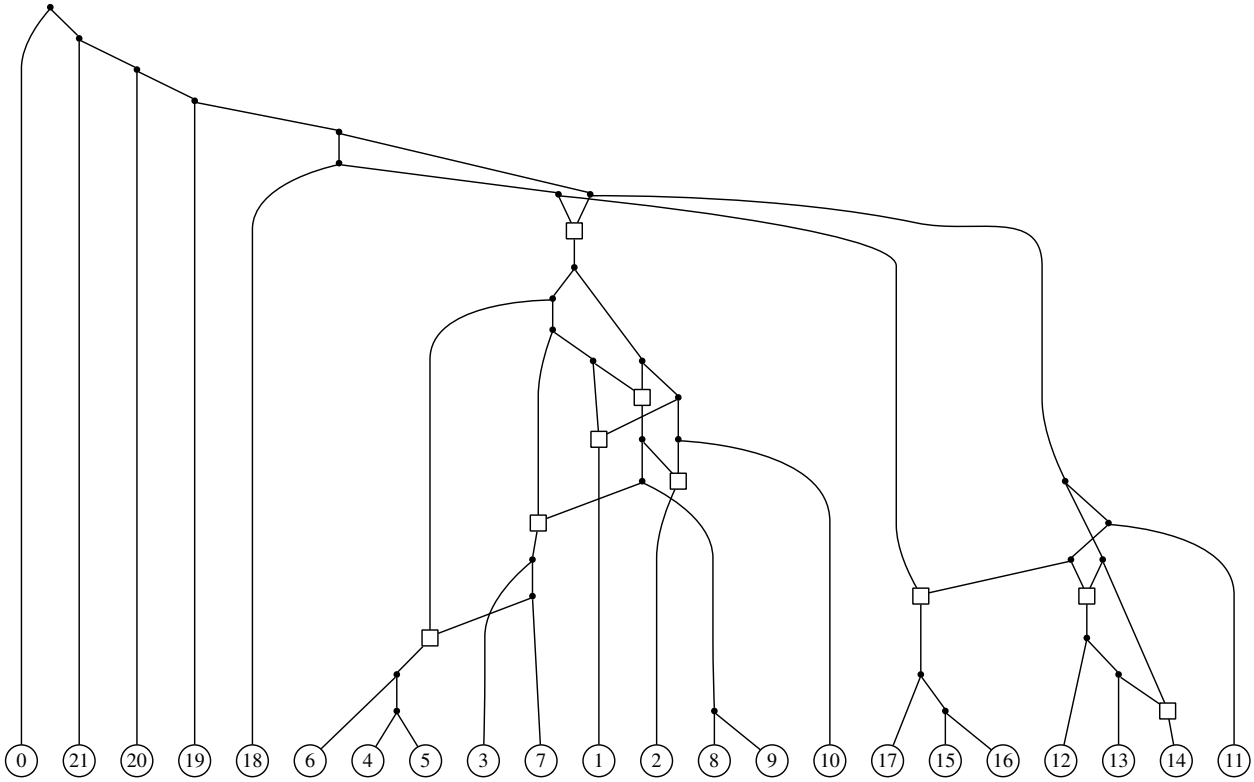

Рис. 1.4: Минимальная гибридизационная сеть для теста NdhfPhytRbcl.

Приложение 2

Подробные результаты тестирования

Таблица 2.1: Подробные результаты тестирования PhyloSAT и PIRN_{CH}. Приведены гибридизационные числа итоговых сетей. -1 означает, что не было получено никакой сети.

	PhyloSat	Время (сек)	PIRN_{CH}	Время (сек)
NdhfIts	-1	1000	-1	1000
NdhfPhyt	8	11	-1	1000
NdhfRbcl	8	1	8	856
NdhfRpoc	9	953	9	485
NdhfWaxy	6	9	6	6
PhytIts	8	45	8	373
PhytRbcl	4	0	4	3
PhytRpoc	4	0	4	1
PhytWaxy	3	0	3	0
RbclIts	-1	1000	-1	1000
RbclRpoc	7	1000	7	43
RbclWaxy	4	4	4	0
RpocIts	-1	1000	-1	1000
RpocWaxy	2	0	2	0
WaxyIts	5	8	5	2
NdhfPhytIts	13	1000	-1	1000
NdhfPhytRbcl	9	100	-1	1000
NdhfPhytRpoc	8	1000	8	28
NdhfPhytWaxy	4	0	4	1
NdhfRbclIts	-1	1000	-1	1000
NdhfRbclRpoc	12	1000	-1	1000
NdhfRbclWaxy	5	67	5	0
NdhfRpocIts	-1	1000	-1	1000
NdhfRpocWaxy	3	0	3	0
NdhfWaxyIts	8	1000	8	89
PhytRbclIts	9	1000	8	116
PhytRbclRpoc	6	12	6	3
PhytRbclWaxy	2	0	2	0
PhytRpocIts	7	1000	7	58
PhytWaxyIts	4	0	4	0
RbclRpocIts	-1	1000	-1	1000
RbclRpocWaxy	3	0	3	0
RbclWaxyIts	6	1000	7	3
RpocWaxyIts	4	2	4	0
NdhfPhytRbclIts	-1	1000	-1	1000
NdhfPhytRbclRpoc	9	1000	10	279
NdhfPhytRbclWaxy	2	0	2	0
NdhfPhytRpocIts	10	1000	-1	1000
NdhfPhytWaxyIts	5	0	5	15
NdhfRbclRpocIts	-1	1000	-1	1000
NdhfRbclRpocWaxy	4	1	4	0
NdhfRbclWaxyIts	6	1000	7	35
NdhfRpocWaxyIts	5	42	5	2
PhytRbclRpocIts	9	1000	8	364
PhytRbclWaxyIts	2	0	2	0
RbclRpocWaxyIts	5	34	5	2
NdhfPhytRbclRpocIts	-1	1000	-1	1000
NdhfPhytRbclWaxyIts	3	0	3	0
NdhfRbclRpocWaxyIts	5	34	5	13