МОЛЕКУЛЯРНАЯ БИОЛОГИЯ, 2015, том 49, № 3, с. 372–383

МОЛЕКУЛЯРНАЯ ФИЛОГЕНЕТИКА

УДК 575.852

ПЕРЕСТРОЙКА И РЕКОНСТРУКЦИЯ ХРОМОСОМНЫХ СТРУКТУР

© 2015 г. К. Ю. Горбунов, Р. А. Гершгорин, В. А. Любецкий*

Институт проблем передачи информации им. А.А. Харкевича Российской академии наук, Москва 127051

Поступила в редакцию 17.12.2014 г.
Принята к печати 24.12.2014 г.

Под хромосомной структурой понимают набор хромосом, состоящих из генов с указанием их принадлежности одной из цепей ДНК, а также циклический или линейный порядок в хромосоме. Широко исследуемая задача – определение кратчайшей последовательности операций по перестройке хромосом, которая переводит одну структуру в другую. В случае одинаковых цен всех операций и постоянного состава генов решение задачи известно. В нашей работе представлен принципиально новый метод, который позволяет решить как эту задачу, так и ряд ее обобщений. А именно, для постоянного набора генов создан новый точный алгоритм решения задачи для равных и неравных цен, который имеет линейную вычислительную сложность. Также получены как точный, так и эвристический алгоритмы решения новой задачи: реконструкции на внутренние вершины дерева видов хромосомных структур с разными наборами генов, когда исходные структуры заданы только в листьях.

***Ключевые слова*: хромосомная структура, хромосомная перестройка, точный алгоритм, линейная сложность алгоритма, предковая структура, дерево видов, эволюция вдоль дерева, принцип парсимонии.**

REARRANGEMENT AND INFERENCE OF CHROMOSOME STRUCTURES, by *K. Yu. Gorbunov, R. A. Gershgorin, V. A. Lyubetsky** **(Kharkevich Institute for Information Transmission Problems, Russian Academy of Science, Moscow, 127051 Russia; *e-mail: lyubetsk@iitp.ru).** **A chromosome structure is defined as a set of chromosomes with genes assigned to one of the DNA strands and having a cyclic or linear arrangement. A well-known problem is to infer the shortest path of chromosome rearrangements connecting a pair of structures. In cases with equal rearrangement costs the solution was known; our solution is based on a principally novel approach. The study presents an original exact solving algorithm with linear complexity for both equal and unequal costs. We consider the case of structures consisting of the same set of genes. The study also presents the exact and heuristic algorithms to solve another problem, the inference of ancestral chromosome structures given fixed structures in leaves.**

***Keywords*: chromosome structure, chromosome rearrangement, effective algorithm, ancestral structure, species tree, evolution along a species tree, parsimony.**

DOI: 10.7868/S0026898415030076

Ранее в работах [1–3] и в других публикациях рассматривали задачи оценки числа хромосомных перестроек в разных частях геномов у разных видов, определения участков наиболее и наименее подверженных перестройкам, оценки частот перестроек на разных этапах эволюции, выявления этапов их резкого возрастания и т.д. Их решали, в основном, на основе выделения и сравнения участков синтении. Этот метод дает приближенные решения упомянутых задач, позволяет описать наиболее крупные события, связанные с перестройками, приближенно восстановить структуру предковых геномов. Однако он не позволяет строить сценарии эволюции хромосомных перестроек вдоль всего дерева видов. Разработка эффективных алгоритмов вычисления расстояния между хромосомными структурами и на этой основе построение полного сценария хромосомных перестроек в геномах видов – актуальные задачи. Первая из них состоит в определении кратчайшей последовательности операций, переводящей одну хромосомную структуру в другую. Вторая из них позволяет находить наиболее вероятные сценарии хромосомных перестроек, оценивать эволюционную близость видов, выявлять перестройки, связанные с мобильными элементами и характерными белками.

При одинаковых ценах операций и постоянном наборе генов задача решена в работе [4], см. также последующие книги [5, 6]. В работах [1–6] приведены ссылки по истории вопроса вместе с подробными биологическими мотивировками

* Эл. почта: lyubetsk@iitp.ru

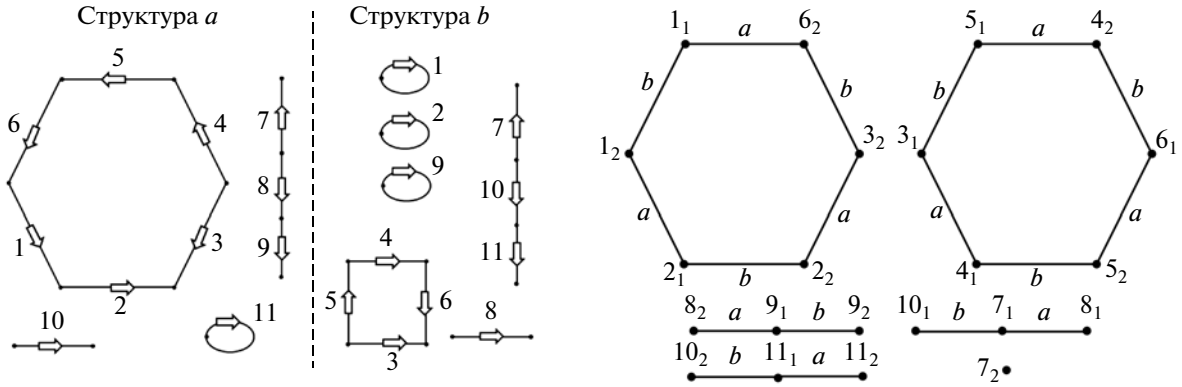

Рис. 1. Две хромосомные структуры *a* и *b* из $n = 11$ генов каждая (слева) и их общий граф $a + b$ (справа).

задач. Отметим большой цикл работ П. Певзнера и его школы, в котором получены алгоритмы хромосомных перестроек с помощью операций инверсии и, частично, трансверсии и транслокации, см. обзорную главу 4 в работе [6]. Насколько авторы могут судить, в этой главе точные полиномиальные алгоритмы обсуждали только для хромосомной структуры, состоящей из одной линейной цепи без паралогов, а операция состоит в инверсии участка цепи. Это – специальный случай двойной переклейки, одной из операций в более широком списке, рассматриваемом ниже.

В данной работе предложен новый точный и линейный по вычислительной сложности алгоритм решения этой задачи, который основан на методе, принципиально отличном от методов, использованных в работах [4–6]. Затем представлен эвристический алгоритм решения более общей задачи: определения кратчайшей последовательности операций по перестройке хромосом при неравных ценах операций, когда минимизируется суммарная цена последовательности. При некоторых ограничениях она может быть решена также и точным линейным алгоритмом, который, однако, представлен в журнал “Проблемы передачи информации”, так как его изложение требует специальных математических средств.

“Точный” алгоритм сопровождается доказательством того, что он всегда выдает (глобальный) минимум соответствующей цены или иного функционала, указанного в постановке задачи.

Затем нами предложен как кубический точный, так и эвристический алгоритмы решения новой задачи: реконструкции на внутренние вершины дерева видов хромосомных структур с разными наборами генов, структуры заданы в листьях. Эти два алгоритма относятся к разным определениям расстояния между парами структур, приписанных концам ребер в дереве видов. Для одного расстояния доказывается точность алгоритма, для второго нет, но, по-видимому, оно более содержательное с эволюционной точки зрения.

Все алгоритмы компьютерно реализованы, примеры применения соответствующих компьютерных программ приведены ниже и подробнее на сайте http://lab6.iitp.ru/ru/pr_chromo/: две программы chromo и chrom_reconstruction. Программирование, счет, подготовка и анализ данных выполнены Р. Гершгориным. Понятия и результаты, приведенные в разделе “Алгоритм решения задачи и его обоснование” и в пунктах “Специальное расстояние” и “Случай разных цен операций” получены двумя другими авторами.

Данная статья – расширенный вариант двух пленарных докладов авторов на конференциях [7, 8].

ПОСТАНОВКА ЗАДАЧИ О КРАТЧАЙШЕЙ ПОСЛЕДОВАТЕЛЬНОСТИ ХРОМОСОМНЫХ ПЕРЕСТРОЕК

Хромосомная структура – набор линейных и циклических хромосом, каждый ген в хромосоме задан началом и концом; длина гена, нуклеотидный состав и межгенные участки хромосомы не учтены. В этой модели гены в хромосоме считают расположенными “линейно” в виде цепи или цикла и примыкающими друг к другу. С учетом двух цепей для примыкающих генов возможны варианты: конец одного гена совпадает с началом другого или того же гена, конец – с концом другого гена, начало – с началом другого гена. Хромосомную структуру удобно рассматривать как граф, состоящий из компонент, каждая из которых изображает хромосому. Ребрам приписаны имена (номера) генов – числа от 1 до n без повторений. Таким образом, ген представлен только своим именем (номером). На рис. 1 слева приведен пример двух хромосомных структур *a* и *b*. В вершине, соединяющей два гена, отождествлены начала и/или концы этих генов (в соответствии со стрелками); крайней вершине цепи приписаны начало или конец гена. Пометка i_j означает нача-

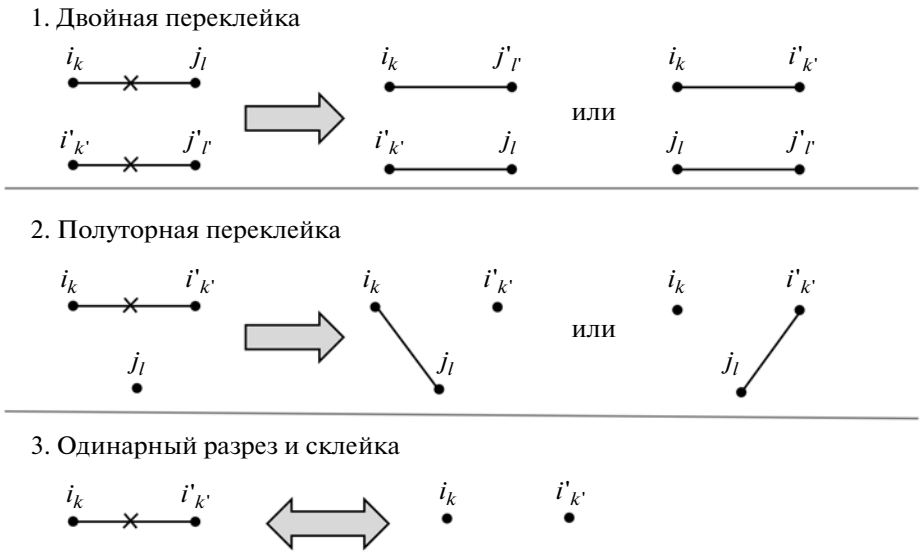

Рис. 2. Операции над хромосомной структурой. Крестик указывает на разрез (расклейку двух склеенных вершин), двойная стрелка – на результат операции. Эти операции предложены в работе [4].

ло гена с номером i (при $j = 1$) или его конец (при $j = 2$); петля в некоторой вершине означает ген, у которого начало совпадает с концом, например ген 11 в структуре a, показанной на рис. 1 слева.

Пусть далее фиксированы две структуры a и b с *одинаковым числом n* генов; например, показанные на рис. 1 слева, где $n = 11$.

Знак ~ указывает на отождествление начал и концов ("краев") генов, соответствующее их соседству на хромосоме; например, для цикла на рис. 1 слева имеем $5_2 \sim 6_1$, $4_2 \sim 5_1$, $3_1 \sim 4_1$, $2_2 \sim 3_2$, $1_2 \sim 2_1$, $1_1 \sim 6_2$.

Новый метод, на котором основан наш подход, состоит в использовании двух понятий: общего графа и его качества, которые обладают нетривиальными свойствами. А именно, *общим графом* структур a и b назовем граф, в котором вершины – это края i_j всех генов из a (они же из b), и ребра соединяют две вершины, если они отождествлены в a или в b. Каждое ребро отмечаем именем структуры, в которой произошло отождествление краев генов, т.е. именами структур a или b. В графе ребра могут быть параллельными: одно ребро из a, другое – из b (циклы длины 2). Общий граф (обозначим его $a + b$) легко описать: это чередующиеся (a, b)-цепи и (a, b)-циклы, рис. 1 справа. Итак, общий граф $a + b$ показывает, какие края генов отождествлены в структурах a и b. Вместо отождествления часто говорят о *склейке* соответствующих краев i_j.

Длина цепи (или цикла) – число ребер в ней (в нем), изолированные вершины считаем цепями длины 0 (будем считать нуль четным числом). *Качество* общего графа – число циклов, сложенное с половиной числа четных цепей в нем (четная цепь содержит четное число ребер, цепи нечетной длины не учитывают).

Исходную задачу можно сформулировать в терминах самих структур a и b или в терминах их общего графа $a + b$. Начнем с первой формулировки. Для двух структур a и b найти последовательность операций, минимальную по их числу, которая переводит одну из них в другую, например, структуру a в структуру b, рис. 1 слева. Саму последовательность назовем *минимальной*, ее длину – минимальной длиной. Удобно сказать это по-другому: найти структуру c, к которой можно привести каждую из структур a и b суммарно минимальным числом операций. Это вытекает из того, что набор операций замкнут относительно перехода к обратной операции.

Этот набор включает следующие операции над хромосомой или парой хромосом (и, тем самым, над хромосомной структурой) [4]. Разрез двух склеек вершин и переклейка четырех краев по-другому ("двойная переклейка"); разрез склейки вершин и новое склеивание одного ее края с каким-то не склеенным краем ("полуторная переклейка"); разрез склейки и обратная к ней операция склейки двух не склеенных краев ("одинарные переклейки"), рис. 2.

Назовем *финальным* (*финального вида*) общий граф двух совпадающих структур, т.е. граф вида $c + c$ для некоторой структуры c. Он содержит лишь циклы длины 2 (одно ребро из структуры a, другое – из структуры b) и изолированные вершины. Такие циклы далее называются 2-циклами.

Вторая формулировка исходной задачи такова: образуем общий граф $a + b$. К нему разрешено применять естественные аналоги перечисленных

выше операций. А именно, удаление двух одинаково помеченных ребер и соединение четырех их концов двумя новыми не инцидентными ребрами с той же пометкой; удаление ребра и соединение (ребром с той же пометкой) одного из его концов с вершиной, не инцидентной никакому ребру с этой пометкой; удаление любого ребра; добавление ребра (допустим, с пометкой a) между двумя вершинами, не инцидентными никакому ребру с пометкой a. Найти последовательность операций, минимальную по их числу, которая приводит $a + b$ к *финальному виду*, т.е. к виду $c + c$ для некоторой структуры c. Такую последовательность назовем также *минимальной*. Качество финального графа равно n, а длина минимальной последовательности операций равна $n - k$, где k – качество исходного общего графа $a + b$, а n – число генов в исходной структуре a (оно же в b). Таким образом, нахождение минимальной длины можно свести к нахождению качества $a + b$.

Заметим, что операция над парой структур (когда операцию применяют к одной из них, другая неизменна) соответствует одноименной операции над общим графом.

Эквивалентность двух сформулированных задач вытекает из эквивалентности каждой из них задаче: найти структуру c и две последовательности операций, одна приводит структуру a к c, другая – структуру b к c так, чтобы суммарное число операций в этих последовательностях было минимальным.

Применение любой операции не меняет качество общего графа или изменяет его ровно на 1, это можно доказать рассмотрением всех возникающих пар: операция вместе с типом компонент общего графа, к которым она применяется. Суть нашего подхода: качество финального графа равно n: пока граф не финальный, можно применить одну из операций, увеличивая его качество; для финального графа нет операции, увеличивающей его качество. Иными словами, качество общего графа двух несовпадающих структур строго меньше n; а наибольшее качество равно n и достигается на совпадающих структурах, финальном графе.

Замечание. Общий граф $a + b$ удобно хранить в массиве M, индексы которого меняются от $-n$ до n; отрицательные индексы соответствуют началам одноименных генов, положительные – их концам в структурах a и b. Значение $M[i]$ – пара индексов краев, с которыми край i склеен в структурах a или b (если не склеен, то значение 0). Такой способ хранения обеспечивает линейные время и память алгоритма построения графа $a + b$ по структурам a и b, а также быстрый просмотр его компонент и переход от операции над $a + b$ к операции над a или b.

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ И ЕГО ОБОСНОВАНИЕ

Опишем *точный* алгоритм с *линейной* вычислительной сложностью, который решает задачу в ее второй формулировке. А именно, он приводит общий граф $a + b$ двух структур к финальному виду.

Алгоритм (пункты 1–4 ниже) преобразует $a + b$ следующим образом.

1) Если есть цепь длины, строго большей двух, то укорачиваем ее, выделяя из нее цикл длины 2 операцией двойной переклейки, при этом образуется цикл и цепь, рис. 3*а*. Точнее, в цепи рассмотрим участок вида *..aba..*, из цепи удаляем *ba* (остается *..a..*) и в структуру добавляем цикл *ab*, рис. 3*а*. Аналогично поступаем с участком вида *..bab...*

2) Цепь длины 2: разбиваем ее на цикл и цепь длины 0 полуторной переклейкой, рис. 3*б*.

3) Цепь длины 1: разбиваем ее на две нулевых цепи одинарным разрезом, или одинарной склейкой замыкаем ее в цикл, рис. 3*в*.

4) Цикл с длиной строго большей двух: разбиваем его на два цикла двойной переклейкой, рис. 3*г*. Точнее, выбираем участок вида *..aba..*, от него оставляем ребро a, которое вместе с оставшейся частью цикла образует более короткий цикл, и добавляем в структуру 2-цикл, рис. 3*г*. Аналогично для участка вида *..bab...*

Если в любом порядке применять эти операции до тех пор, пока это возможно, то получится искомая минимальная последовательность операций.

Существенно, что при построении последовательности алгоритм использует лишь информацию о *типе графа*: сколько в нем циклов и какой они длины, сколько в нем цепей и какой длины.

Сложность и корректность алгоритма. Алгоритм, очевидно, имеет линейную вычислительную сложность. При каждом применении операции, указанной в алгоритме, качество общего графа увеличивается ровно на 1. Если общий граф еще не имеет финального вида, то одну из операций можно применить снова.

Отметим новую операцию, которая также увеличивает качество общего графа на 1. Это – разрезание цепи нечетной длины на две цепи четной длины одинарным разрезом по ребру, имеющему ту же пометку, что и крайнее ребро цепи, a или b. Точнее, такое ребро удаляется, а остальные ребра входят в две новые цепи; или вместо ребра образуется изолированная вершина. Можно добавить эту операцию и одновременно ограничить применение первой операции только четными цепями. Более того, в нашем алгоритме годится любой набор операций, которые строго увеличивают качество общего графа, и одна из них применима до

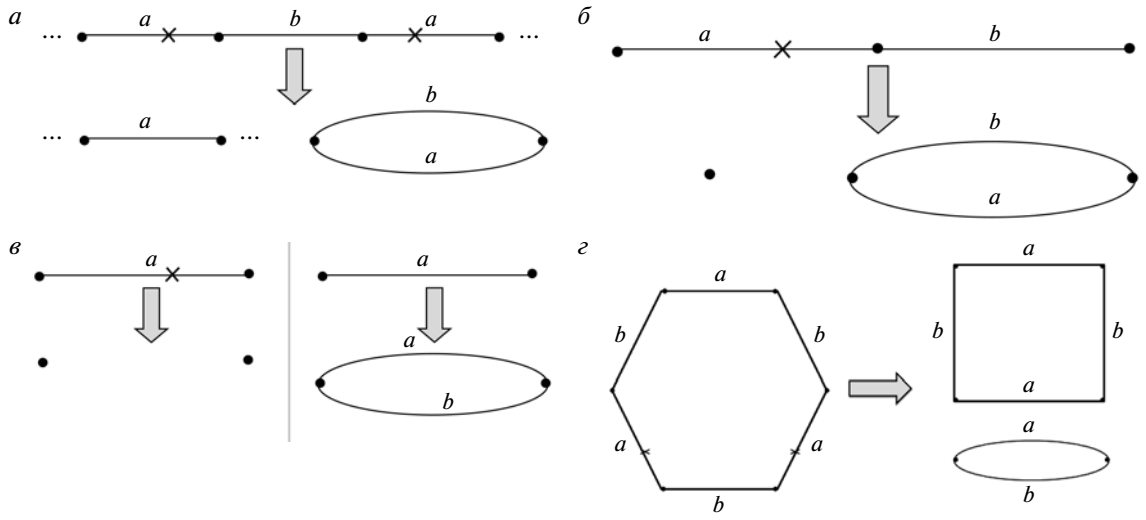

Рис. 3. *а* – Выделение цикла из цепи с длиной большей двух. “Начало” (до крестика) левого ребра структуры *a* склеилось с “концом” (от крестика) правого ребра *a*. “Конец” (от крестика) левого ребра *a* склеился с “началом” (до крестика) правого ребра *a*, ребро *b* не изменилось. *б* – Выделение цикла из цепи с длиной два. *в* – Операции по устранению цепи с длиной 1: получаются две изолированные точки (слева) или 2-цикл (справа). *г* – Разбиение цикла с длиной большей двух на два цикла. В результате: “начала” (до крестиков) ребер структуры *a* соединяются до “квадрата”; “концы” (от крестиков) ребер *a* соединяются с *b* в цикл.

тех пор, пока очередной граф не принимает финального вида.

В следующих разделах рассмотрены обобщения сформулированной выше *исходной* задачи.

ЗАДАЧА В СЛУЧАЕ РАЗНЫХ ЦЕН ОПЕРАЦИЙ

Пусть каждой операции из указанного выше списка и каждой обратной к ней операции присвоена цена, строго положительное число. А структуры имеют одинаковый набор генов. Редким в эволюционном процессе операциям назначают большую цену; а частым – меньшую. Задача: найти последовательность операций, которая переводит хромосомную структуру *a* в *b* и имеет минимальную суммарную цену. Такую последовательность назовем *кратчайшей*, а ее цену – *минимальной*. Исходная задача – частный случай, когда цены всех операций одинаковы. Вопрос о существовании линейного или хотя бы полиномиального алгоритма решения такого обобщения остается открытым. Мы предлагаем *эвристический* алгоритм ее решения (см. замечание в конце этого раздела).

Определим ориентированный граф, вершинам которого однозначно приписаны общие графы; назовем его *орграфом*. Точнее, в его вершине хранится лишь информация о типе общего графа этой вершины, т.е. информация, которая используется в изложенном выше алгоритме. В орграфе ребро проводится из вершины v_1 в вершину v_2, если граф v_1 преобразуется одной из операций в граф v_2, при этом ребро помечается ценой операции, если она применяется к структуре *a*, или ценой обратной операции – к структуре *b*. Корню орграфа приписан исходный общий граф $a + b$ данных структур *a* и *b*. Каждый путь из корня продолжается до листа, которому приписан финальный граф, получаемый на этом пути (с качеством *n*). Таким образом, задача состоит в нахождении кратчайшего пути в орграфе из корня в какой-нибудь лист.

Для любого общего графа v известна минимальная последовательность, которая строится изложенным выше алгоритмом и приводит v к финальному виду; обозначим $t(v)$ цену самой дорогой операции, умноженную на минимальную длину. Аналогично обозначим $l(v)$ цену самой дешевой операции, умноженную на минимальную длину.

К орграфу применим следующий алгоритм, который дает эвристическое решение задачи из этого раздела. В нем использована небольшая часть орграфа, так что его не требуется строить целиком.

Пусть получено множество O, включающее корень орграфа и состоящее из вершин v, для которых уже найден кратчайший путь в орграфе из корня в v, целиком лежащий в O; каждой вершине из O приписана кратчайшая цена $c(v)$. *Обозначим* α минимум по всем вершинам v из O величины $c(v) + t(v)$. В O пометим все вершины v, для которых $c(v) + l(v) > \alpha$. Расширим O одной новой вершиной. Для этого рассмотрим множество O_1 – вершин не из O, в которые из непомеченной

вершины в O ведет ребро. Каждой вершине v из O_1 припишем число $c(v)$ – кратчайшую цену пути из корня в v среди всех путей, у которых вершины лежат в O кроме самой v. Выберем вершину w из O_1 с минимальным значением $c(v)$ и получим новую окрестность $O \cup \{w\}$.

Окрестности вложены друг в друга и расширяются до тех пор, пока возможно. Среди листьев, которые вошли в последнюю окрестность, выбирается вершина с минимальным значением $c(v)$. Обратным ходом алгоритма строится минимальный путь в нее из корня.

Замечание. Обозначим цены событий: разреза c_1, склейки c_1', полуторной переклейки $c_{1.5}$, двойной переклейки c_2. Если наложить условие поиска кратчайшей последовательности среди минимальных ("условно кратчайшая"), то получим точный и линейный по вычислительной сложности алгоритм решения задачи из этого раздела, по крайней мере, для соотношений цен: $c_2 \le c_1 \le c_1' \le c_{1.5}$ и $c_1 \le c_1' \le c_{1.5} \le c_2$. Если цены не слишком отличаются друг от друга, то условно кратчайшая последовательность совпадает с кратчайшей. Например, если цены заключены в интервале от δ до ε и $\delta/\varepsilon > n/(n+1)$, то условно кратчайшая последовательность совпадает с кратчайшей. В общем случае они могут различаться.

РЕКОНСТРУКЦИЯ ХРОМОСОМНЫХ СТРУКТУР ВДОЛЬ ДЕРЕВА: ПОСТАНОВКА ЗАДАЧИ, АЛГОРИТМЫ И ИСКУССТВЕННЫЕ ПРИМЕРЫ

Рассмотрим *новую задачу* восстановления хромосомной структуры вдоль дерева, для простоты будем считать цены операций одинаковыми. Дано эволюционное дерево видов (не обязательно бинарное), и в каждом его листе задана своя структура со своим набором генов. Нужно реконструировать структуры во внутренних вершинах дерева (*предковые* структуры) так, чтобы минимизировать *функционал*: сумму по всем ребрам "расстояний" между структурами на концах ребра. Структуры во внутренних вершинах могут включать только гены, представленные в листьях; множество всех таких генов обозначим S. Таким образом, реконструкция структур, заданных в листьях, ищется на основе принципа парсимонии.

Вопрос о выборе этого расстояния нетривиален и нуждается в дальнейшем обсуждении, так что речь идет о семействе задач в зависимости от определения расстояния. Рассмотрим здесь два таких определения. В первом из них *расстояние* между структурами a и b (приписанными концам ребра дерева) определяется как число пар различных краев генов, которые в одной структуре склеены, а в другой – отсутствуют (один или оба) или не склеены, сложенное с числом генов, присутствующих в одной структуре и отсутствующих в другой. Это расстояние назовем *специальным*. Второй вариант, биологически более естественный: *расстояние* – минимальная длина между a и b, определяемая в исходной задаче. Аргумент указанных функционалов – *расстановка* структур по всем внутренним вершинам дерева. Поскольку в листьях структуры заданы, любая расстановка приписывает каждой вершине дерева видов одну структуру. Значение функционала на данной расстановке назовем ее *ценой*.

В случае первого расстояния задача решена точным алгоритмом, сложность которого имеет порядок произведения $|S|^2$ на число листьев, т.е. весьма низкая. Алгоритм может работать и с деревом, которое разбито на временные слои [9–11]. Алгоритм распространяется на функционал, который содержит перед каждым слагаемым свой коэффициент, отражающий эволюционную длину ребра. Со вторым расстоянием, которое назовем *биологическим*, задача решена эвристическим алгоритмом, но алгоритм по-прежнему имеет практически кубическую сложность.

Специальное расстояние. Для данной расстановки и каждой вершины дерева, и каждой пары различных краев генов из S введем переменную, равную 1, если эти края склеены в структуре, соответствующей вершине, и 0 – в противном случае. Кроме того, для каждого гена из S введем переменную, равную 1, если ген отсутствует в этой структуре, и 0 – в противном случае. Вершину и ее структуру не различаем.

В листьях дерева значения всех переменных заданы. Для каждой переменной найдем значения во внутренних вершинах так, чтобы минимизировать указанный функционал, который можно переформулировать как число ребер, на концах которых значения какой-то переменной отличаются; каждое ребро считается столько раз, сколько имеется таких переменных. Для минимизации функционала мы применяли следующий алгоритм. Сначала происходит поиск минимума отдельно по каждой переменной, этот шаг, очевидно, имеет линейную вычислительную сложность. Полученный набор значений во всех вершинах всех переменных назовем *разметкой* дерева. Полученная разметка в какой-то вершине может указывать на склейку одного края с двумя различными краями ("противоречие первого рода") или указывать на склейку края гена, отсутствующего в вершине ("противоречие второго рода").

Поэтому следующий шаг алгоритма состоит в устранении всех противоречий, он также имеет линейную вычислительную сложность от произведения числа переменных на размер дерева; важно, что при этом цена разметки не меняется. Так

алгоритм находит решение – разметку, на которой достигается минимум цены.

Шаг алгоритма по устранению противоречий. Упорядочим все края генов из S в каком-то линейном порядке; например, лексикографически. Перебираем вершины дерева в произвольном порядке, а для каждой вершины – края генов согласно этому порядку. Для каждого края удаляем все его склейки в данной вершине, если их четное число, или иначе оставляем одну склейку с краем, наибольшем в этом порядке. В результате устраняются противоречия первого рода. Затем перебираем гены в порядке возрастания их имен и для каждого края отсутствующего гена, если он с кем-то склеен (очевидно, он может быть склеен не более чем с одним краем), удаляем эту склейку и считаем этот ген присутствующим в вершине. В результате устранены противоречия второго рода.

Обоснование этого шага. Очевидно, он устраняет все противоречия. Осталось показать, что при этом цена разметки не увеличивается. Противоречие первого рода в вершине v – пара принимающих единичное значение переменных, которые соответствуют двум парам краев генов из S вида (a, b) и (a, c); такие пары краев назовем *инцидентными*. Порядок на краях генов индуцирует порядок на инцидентных парах краев. Шаг алгоритма по устранению противоречий первого рода эквивалентен следующей процедуре. Перебираем инцидентные пары краев согласно этому порядку. Для каждой пары перебираем вершины дерева и в каждой вершине, если обе соответствующих переменных принимают значения 1, заменяем оба значения на 0. При этом не возникают новые противоречия по другим парам краев. Таким образом, достаточно показать сохранение цены разметки при выполнении этой процедуры для фиксированной пары переменных (x, y).

Назовем *дефектом* разметки разность ее цены и минимальной цены разметки. Ребро дерева будем считать *противоречивым*, если в одном его конце разметка (1, 1), т.е. $x = 1$, $y = 1$, а в другом конце – (0, 0). В листьях дерева противоречий нет по определению. Перебираем внутренние вершины дерева в любом порядке и, если в вершине разметка (1, 1), меняем ее на (0, 0). Пусть дефект текущей разметки равен d. Покажем, что всегда выполнены два свойства: d – четное число (считая 0 четным), и в дереве существует не менее $d/2$ противоречивых ребер. Действительно, сначала дефект разметки нулевой, и указанные свойства выполнены. Рассмотрим очередную вершину v. Если в ней имеется противоречие, т.е. разметка (1, 1), то, как говорили, она заменяется на (0, 0). Рассмотрим вершину u, инцидентную вершине v. Возможны следующие три случая. Разметка в u равна (0, 1) или (1, 0). Тогда цена разметки на ребре (v, u) не изменится, это ребро было и осталось непротиворечивым. Разметка в u равна (0, 0). Тогда цена разметки на ребре (v, u) уменьшается на 2, это ребро было противоречивым, а стало непротиворечивым. Разметка в u равна (1, 1). Тогда цена разметки на ребре (v, u) увеличивается на 2, это ребро было непротиворечивым, а стало противоречивым. Таким образом, число непротиворечивых ребер увеличивается (уменьшается) на c, если, и только если, цена разметки увеличивается (уменьшается) на $2c$.

В конце процедуры противоречий и противоречивых ребер не останется, по свойству 2 дефект разметки станет нулевым. Поскольку мы всегда меняли значения переменных с 1 на 0, новых противоречий (по другим парам переменных) не возникнет. Заметим, что описанная процедура устранения противоречий годится и для произвольного графа.

Противоречием второго рода в вершине v является пара принимающих значения 1 переменных, первая из которых – индикатор отсутствия гена, а вторая соответствует склейке края этого гена; такие пары переменных назовем *инцидентными*. Порядок на краях генов индуцирует порядок на инцидентных парах переменных. Описанная процедура устранения противоречий второго рода эквивалентна следующей процедуре. Перебираем инцидентные пары переменных в этом порядке. Для каждой пары перебираем вершины дерева и в каждой вершине, если обе соответствующих переменных принимают значения 1, заменяем оба значения на 0. Дословно повторяется рассуждение для противоречий первого рода.

Искусственные примеры эволюции хромосомных структур. Биологические примеры приведены ниже, в двух пунктах.

Пример 1. Каждая структура в листьях содержит по три гена с именами 1, 2, 3 (рис. 4*а*). В листьях выполняется: 18 переменных равны 1, три из них соответствуют склейке начала гена с его концом, еще по две переменных в каждом листе соответствуют отношениям остальных краев, рис. 4*а*. Минимальная разметка дерева придает каждой из первых трех переменных значение 1 лишь в одной внутренней вершине дерева, в родителе двух соответствующих листьев, а всем другим переменным – нулевые значения во всех внутренних вершинах. Противоречий не возникает.

Пример 2. Случай неодинаковых наборов генов в листьях. Рис. 4*б* отличается от рис. 4*а* добавлением в трех листьях нового гена с именем 4. Переменная, которая указывает на отсутствие этого гена, в корне и в двух крайних вершинах равна 0, в средней вершине равна 1.

Случай разных цен операций. Алгоритм работает и в случае, если цены четырех событий (склейка двух концов, их расклейка, возникновение гена, потеря) различны. Шаг алгоритма, строящий

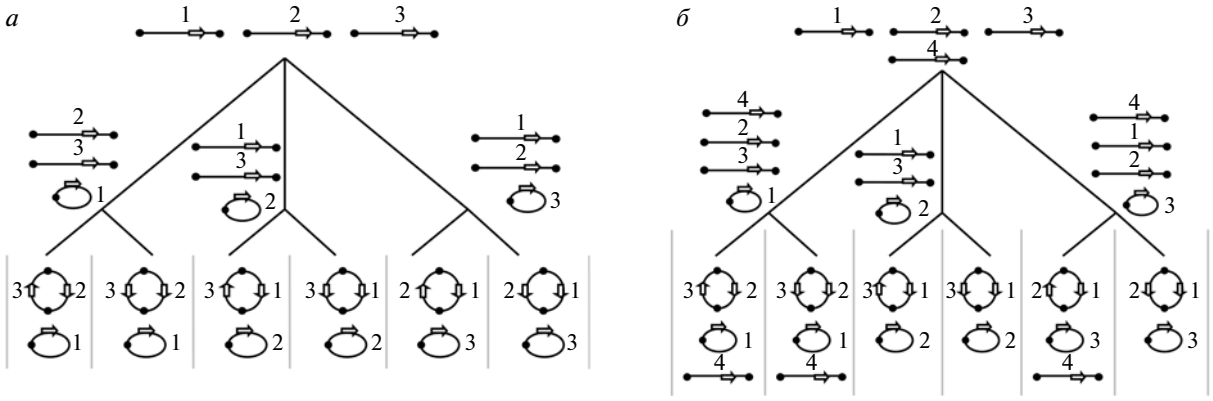

Рис. 4. Дана расстановка структур в листьях, показана найденная алгоритмом расстановка предковых структур вдоль дерева. *а* – Случай одинакового набора генов в листьях. *б* – Случай неодинаковых наборов генов в листьях.

минимальную разметку, не меняется. Теперь при устранении противоречий цена разметки может увеличиться. Однако при соотношении цен, которое кажется биологически возможным: цена склейки двух краев не больше цены их расклейки, и цена потери гена не больше цены его возникновения – цена разметки также не увеличивается, т.е. наш алгоритм остается точным.

Верно более сильное утверждение: если цена перехода $0 \to 1$ строго меньше цены перехода $1 \to 0$, то противоречий не возникнет. Действительно, пусть d – разность цен этих переходов. Фиксируем инцидентную пару переменных. Индукцией по высоте дерева покажем: для любой (не обязательно минимальной) разметки, если в корне дерева есть противоречие, то при его устранении цена разметки уменьшится не менее чем на $2d$; иначе цена не увеличится. Начальный шаг индукции очевиден, опишем индуктивный шаг. Рассмотрим дерево с корнем r и предположим, что для всех "сыновьих" деревьев вершины r утверждение проверено. Если в сыновьей вершине r' разметка равна (1, 1), *назовем* ребро (r, r') и дерево с корнем r' единичными. Переберем случаи разметки (x, y) вершины r. Пусть $(x, y) = (0, 0)$. Очевидно, при устранении противоречий цена разметки не увеличится. Пусть $(x, y) = (0, 1)$ или $(x, y) = (1, 0)$. На каждом единичном ребре цена разметки увеличивается на d, но в каждом единичном дереве цена уменьшается на $2d$. Цена всей разметки не увеличивается.

Пусть $(x, y) = (1, 1)$. На каждом неединичном ребре цена уменьшается как минимум на d, на единичном ребре цена не меняется, но в единичном дереве цена уменьшается на $2d$. Поскольку у вершины не меньше двух сыновей, цена всей разметки уменьшается как минимум на $2d$.

Теперь отсутствие противоречий в разметке доказывается от противного. Рассмотрим максимальное по включению дерево T с корнем v, помеченным парой (1, 1). Если в разметке были противоречия, то устранение их в T (вне T разметку не меняем) уменьшило бы цену разметки в T на $2d$, а на "родительском" ребре вершины v цена могла увеличиться не более чем на d (используем, что выше вершины v нет разметки (1, 1)). Цена всей разметки уменьшается, а это невозможно, так как она уже минимальная.

Биологическое расстояние. В следующем пункте рассмотрены примеры применения этого алгоритма с ценами склейки равными 3, а расклейки – 2, что отличается от рассмотренного в этом пункте соотношения цен. Теперь после устранения противоречий цена разметки обычно незначительно увеличивается.

Применяется следующий эвристический алгоритм. Сначала вышеописанным алгоритмом расставляются предковые структуры с использованием специального расстояния, при этом используются только что указанные значения цен. Для длинных ребер цены могут умножаться на поправочные коэффициенты, отражающие эволюционное время. Так, полученная расстановка служит началом для алгоритма минимизации цены расстановки уже на основе биологического расстояния. На каждом шаге этого алгоритма перебирались все внутренние вершины дерева и все операции. Шаг алгоритма: выбор пары <вершина, операция>, которая наиболее понижает цену расстановки; операцию применяют в этой вершине. Этот алгоритм назовем *спуском*.

Чтобы осуществлять спуск несколько раз, начиная с различных начальных расстановок, в функционал со специальным расстоянием добавляется параметр p – штраф за несклеенность двух краев генов. Вышеописанному алгоритму соответствует значение $p = 0$. Осуществлялся спуск при всех значениях p от 0 до 3 с шагом 0.1, и среди всех так полученных расстановок выбирается наилучшая, итоговая расстановка предковых структур.

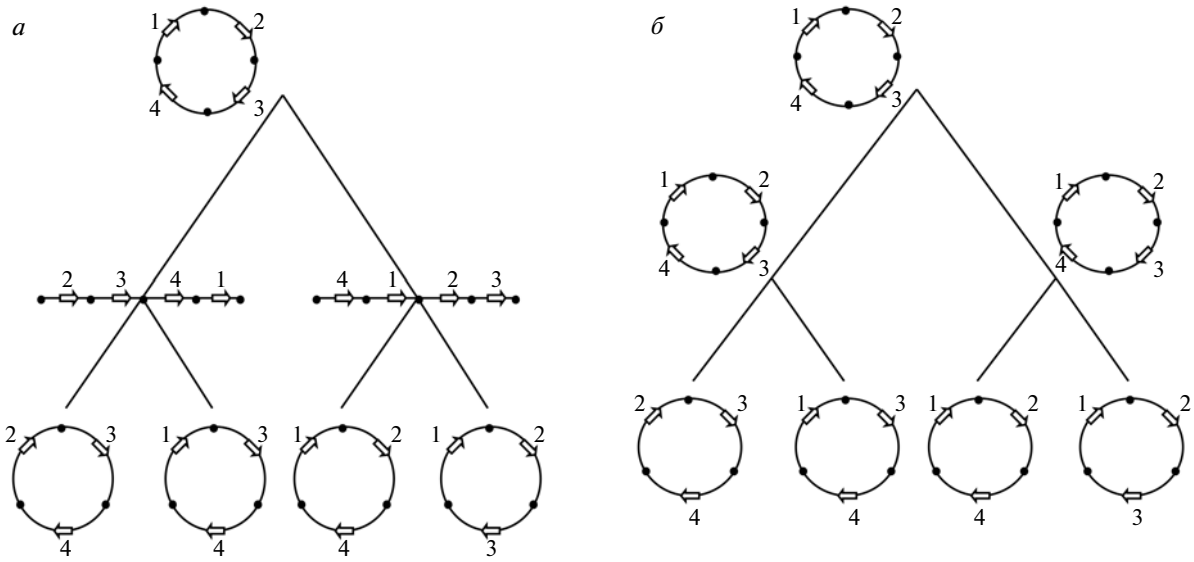

Рис. 5. Дана расстановка структур в листьях, показана найденная расстановка предковых структур вдоль дерева. *а* – Случай специального расстояния. *б* – Случай биологического расстояния.

В приведенное выше определение биологического расстояния можно внести поправку. А именно, если в структуре преобладают циклические хромосомы, то двойные переклейки происходят чаще других операций, поэтому цену c_{double} двойной переклейки брали 0.8, а цена c_{other} любой другой переклейки – 1.2. Обозначим l – число ребер в компоненте общего графа двух исходных структур. Тогда расстояние между двумя структурами приравниваем к сумме по всем циклам величин $0.5c_{\text{double}}(l-2)$, сложенной с суммой по всем нечетным цепям величин $0.5c_{\text{double}}(l-1) + c_{\text{other}}$ и по всем четным цепям величин $0.5c_{\text{double}}(l-2) + c_{\text{other}}$. Нетрудно показать, что это расстояние равно минимальной суммарной цене операций среди всех последовательностей операций, переводящих одну структуру в другую и минимальных по длине.

В примере на рис. 4*а* мы получили другую расстановку предковых структур по биологическому расстоянию по сравнению с вариантом специального расстояния. А именно, в корне цикл (1, 2, 3), слева два цикла (1) и (2, 3), в центре два цикла (2) и (1, 3), справа два цикла (3) и (1, 2); все гены на одной цепи. Действительно, для данных на рис. 4*а*, эволюционный сценарий со специальным расстоянием содержит 15 одинарных склеек (по одной на верхних ребрах и по две на нижних), а он же с биологическим расстоянием – шесть двойных переклеек (по одной на верхних ребрах и по одной на трех нижних).

Пример 3. Наборы генов в листьях, решения задач со специальным и биологическим расстояниями отличны, рис. 5.

Мы видим, что для биологического расстояния на рис. 5*а* сценарий содержит по одному одинарному разрезу на двух верхних ребрах, по одному удалению на четырех нижних ребрах и на них же по одной одинарной склейке. На рис. 5*б* имеем по одному удалению на четырех нижних ребрах.

Примеры реконструкции хромосомных структур на биологических данных. Пример 4. Исходные данные – структуры генов пластид из следующих девяти видов багрянок: NC_021618 *Grateloupia taiwanensis*; NC_021075 *Calliar throntuberculosum*; NC_020795 *Chondrus crispus*; NC_006137 *Gracilaria tenuistipitata*var. liui; NC_023133 *Porphyridium purpureum*; NC_004799 *Cyanidioschyzon merolae* strain 10D; NC_001840 *Cyanidium caldarium*; NC_007932 *Pyropia yezoensis*; NC_000925 *Porphyra purpurea*. Геном каждой из багрянок состоит из одной циклической хромосомы. В каждом отобраны гены, связанные с первой и второй фотосистемами. Согласно таксономии NCBI дерево видов политомическое (т.е. небинарное), рис. 6*а*. Итак, в его листьях заданы следующие хромосомные структуры, каждая из 25 генов. В указанном выше порядке видов перечислим гены пластид, порядок генов отражает их расположение на хромосоме, знак “*” указывает на комплементарную цепь, знак “|C ” указывает на цикличность хромосомы:

psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM psbA *psbV *psaJ *psaF psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL |C;

psbA *psbV *psaJ *psaF psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psaF psaJ psbV *psbA psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

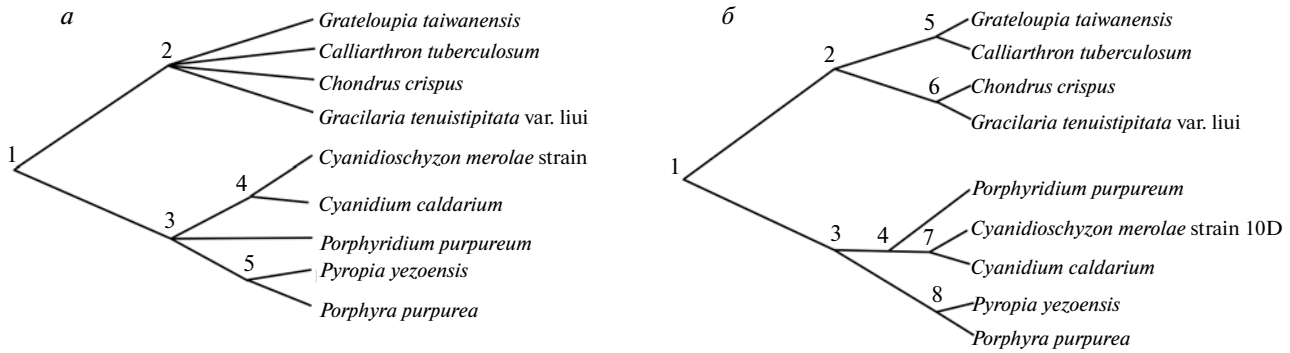

Рис. 6. Дерево перечисленных видов. Внутренние вершины дерева пронумерованы. *а* – Исходное политомическое дерево, *б* – его бинарное разрешение.

psaF psaJ psbV *psbA psbD psbC psaE *psbH psbN *psbT *psbB psbK *psaB *psaA *psaD psbI psaL psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psbD psbC *psbV *psaB *psaA *psbK *psaE psbA *psaJ *psaF psaD *psbJ *psbL *psbF *psbE*psaI *psaL psaM *psbI *psbH psbN *psbT *psbB *psaC *psaK |C;

psaM psbA *psaK *psaC psbD psbC psbB psbT *psbN psbH *psaE psaA psaB *psbK *psaD psaF psaJ psbV psaI *psbJ *psbL *psbF *psbE psaL *psbI |C;

psbD psbC psaI *psbJ *psbL *psbF *psbE psaL *psbI psbA psaK *psaC psaE *psbH psbN *psbT*psbB *psaM psaF psaJ psbV psaD psbK *psaB *psaA |C;

psaF psaJ psbV *psbA psaL *psbI psaD psaA psaB psbK psbB psbT *psbN psbH *psaE *psbC *psbD psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C;

psaF psaJ psbV psbA psaL *psbI psaD psaA psaB psbK psbB psbT *psbN psbH *psaE *psbC *psbD psaK *psaC psaI *psbJ *psbL *psbF *psbE *psaM |C.

Алгоритм применяли в двух вариантах: непосредственно к политомическому дереву и к каждому из его бинарных разрешений. На политомическом дереве полученная минимальная цена решения равна 27.9. Решение приведено в порядке нумерации вершин.

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbVpsbA |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbVpsbA |C;

psaL *psbIpsaDpsbK *psaB *psaApsbDpsbCpsaI *psbJ *psbL *psbF *psbE |C и psbBpsbT *psbNpsbH *psaEpsaC *psaK *psbA *psbV *psaJ *psaFpsaM |C;

psaL *psbIpsaDpsaApsaBpsbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbVpsbA |C.

Минимальная расстановка говорит, что в эволюции можно было поддержать структуру из одной циклической хромосомы кроме вершины 4, в которой возникла вторая циклическая хромосома. Последнее, возможно, связано с необходимостью более быстрой репликации в тот момент, в неблагоприятных условиях внешней среды.

Алгоритм указал для бинарных разрешений исходного дерева наилучшую цену, равную 25.2 для бинарного разрешения, показанного на рис. 6*б*. Решение приведено в порядке нумерации вершин:

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbA |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaL *psbIpsaDpsaApsaB *psbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI*psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbA |C;

psaL *psbIpsaMpsaI *psbJ *psbL *psbF *psbEpsbK *psaB *psaA *psbA *psbV *psaJ *psaFpsaD |C и psbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaC |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsbA *psbV *psaJ *psaFpsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaLpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbApsbDpsbCpsaE *psbHpsbN *psbT *psbBpsbK *psaB *psaA *psaDpsbI |C;

psaL *psbIpsaMpsaI *psbJ *psbL *psbF *psbE |C и psaDpsbK *psaB *psaApsbDpsbCpsbBpsbT *psbNpsbH *psaEpsaC *psaK *psbA *psbV *psaJ *psaF |C;

psaL *psbIpsaDpsaApsaBpsbKpsbBpsbT *psbNpsbH *psaE *psbC *psbDpsaK *psaCpsaI *psbJ *psbL *psbF *psbE *psaMpsaFpsaJpsbV *psbA |C.

Минимальная расстановка структур по внутренним вершинам дерева, показанного на рис. 6*б*, говорит, что в эволюции могла поддерживаться структура из одной циклической хромосомы кроме вершин 4 и 7, в которых по две хромосомы.

Пример 5. Из тех же пластид отобраны гены рибосомных белков и субъединиц РНК-полимеразы. Листьям того же небинарного дерева приписаны хромосомные структуры, в каждой по 45 генов (порядок видов тот же, что в примере 1):

rps4 *rps6 *rpl27 *rpl21 *rpl32 rpl34 rps16 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 |C;

*rps6 *rpl27 *rpl21 *rpl32 rpl34 rps16 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rpl34 rpl32 rpl21 rpl27 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rpl34 rpl32 rpl21 rpl27 rps6 rps16 rpl11 rpl1 rpl12 rpl19 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 *rpl28 rps4 |C;

*rps6 rpl34 *rpl1 *rpl11 rpl12 rpl35 rpl20 *rps4 rpl21 rpl27 *rps18 *rpl33 rps16 rpl19 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 rps10 rps14 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rpl28 rpl32 *rpoBrps2 |C;

rps4 *rpl28 rpl21 rpl27 rpl32 rpl34 rps6 rps16 rpl11 rpl1 rpl12 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoBrps2 |C;

*rps18 *rpl33 rpoBrps2 rpl21 rpl27 rpl32 rpl34 rps6 *rpl19 rpl11 rpl1 rpl12 *rps16 rps4 *rpl28 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rpl20 *rpl35 *rps14 |C;

*rpl34 rpl32 rpl21 rpl27 *rps4 *rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 rpl31 *rps9 *rpl13 *rpoArps11 rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 |C;

*rpl34 rpl32 rpl21 rpl27 *rps4 *rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 |C.

Алгоритм выдает решение с минимальной ценой 31.5. Всем вершинам, кроме 4-й, приписана одна циклическая хромосома, а вершине 4 – две циклических хромосомы. Решение приведено в порядке нумерации вершин:

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps4 |C и rps2 rpl21 rpl27 rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoB |C;

rpl28 rps4 *rpl27 *rpl21 *rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 |C.

Применение алгоритма к различным бинарным разрешениям политомического дерева показало, что наилучший результат достигается на том же бинарном дереве, рис. 6*б*. Полученная минимальная цена равна 29.6. Во всех вершинах, кроме седьмой, по одной циклической хромосоме, в 7-й – две циклических хромосомы. Полученная расстановка структур по внутренним вершинам дерева приводится в порядке их нумерации:

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA*rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4

*rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 *rps10 *rps7 *rps12 *rpl31 *rps9 *rpl13 *rpoA *rps11 *rps13 *rpl36 *rps5 *rpl18 *rpl6 *rps8 *rpl5 *rpl24 *rpl14 *rps17 *rpl29 *rpl16 *rps3 *rpl22 *rpl2 *rpl23 *rpl4 *rpl3 *rps14 *rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl34 *rpl32 rpl21 rpl27 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rpl19 rpl11 rpl1 rpl12 *rps16 *rpl34 rpl32 rpl21 rpl27 rps6 *rps4 |C;

rpl28 *rps2 *rpoBrpl33 rps18 *rpl20 *rpl35 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10*rpl19 rpl11 rpl1 rpl12 *rps16 *rps6 *rpl27 *rpl21 *rpl32 rpl34 *rps4 |C;

rpl28 *rps4 |C и rps2 rpl21 rpl27 rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl35 rpl20 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 *rps18 *rpl33 rpoB |C;

rpl28 rps4 *rpl27 *rpl21 *rpl32 rpl34 rps6 rps16 *rpl12 *rpl1 *rpl11 rpl19 rps14 rpl3 rpl4 rpl23 rpl2 rpl22 rps3 rpl16 rpl29 rps17 rpl14 rpl24 rpl5 rps8 rpl6 rpl18 rps5 rpl36 rps13 rps11 rpoArpl13 rps9 rpl31 rps12 rps7 rps10 rpl35 rpl20 *rps18 *rpl33 rpoBrps2 |C.

Авторы глубоко благодарны А.В. Троицкому и С.А. Спирину за высказанные замечания, которые способствовали улучшению работы.

Исследование выполнено за счет гранта Российского научного фонда (14-50-00150).

СПИСОК ЛИТЕРАТУРЫ

1. Donthu R., Lewin H.A., Larkin D.M. 2009. SyntenyTracker: a tool for defining homologous synteny blocks using radiation hybrid maps and whole-genome sequence. *BMC Res. Notes.* **23**(2), 148, doi: 10.1186/1756-0500-2-148.
2. Romanov M.N., Farré-Belmonte M., Lithgow P.E., O'Connor B., Fowler K.E., Larkin D.M., Griffin D.K. 2014. *In silico* reconstruction of chromosomal rearrangements and an avian ancestral karyotype. In: *International Plant and Animal Genome XXII Conference*, January 11–16, 2014, San Diego, CA, USA.
3. Romanov M.N, Farré M., Lithgow P.E., Fowler K.E., Skinner B.M., O'Connor R., Fonseka G., Backström N., Matsuda Y., Nishida C., Houde P., Jarvis E.D., Ellegren H., Burt D.W., Larkin D.M., Griffin D.K. 2014. Reconstruction of gross avian genome structure, organization and evolution suggests that the chicken lineage most closely resembles the dinosaur avian ancestor. *BMC Genomics.* **15**, 1060, doi: 10.1186/1471-2164-15-1060.
4. Bergeron A., Mixtacki J., Stoye J. 2006. A unifying view of genome rearrangements. *Algorithms Bioinformatics. LNCS.* **4175**, 163–173.
5. Fertin G., Labarre A., Rusu I., Tannier E., Vialette S. 2009. *Combinatorics of Genome Rearrangements*. Cambridge: MIT Press.
6. *Models and Algorithms for Genome Evolution*. 2013. Eds Chauve C., El-Mabrouk N., Tannier E. *Comput. Biol. Series*. London: Springer-Verlag.
7. Любецкий В.А., Горбунов К.Ю. Задачи и алгоритмы, связанные с хромосомными перестройками. *Сборник избранных трудов VIII Международной научно-практической конференции* (Москва, МГУ им. М.В. Ломоносова, 8–10 ноября 2013 г.). 2013. М.: ИНТУИТ.РУ, 764–768.
8. Lyubetsky V.A., Gorbunov K.Yu. 2014. Chromosome structures reconstruction. *Сборник материалов 4-й Московской международной конференции "Молекулярная филогенетика MolPhy-4"* (Москва, МГУ им. М.В. Ломоносова, 23–26 сентября 2014 г.). М.: Торус-Пресс, с. 42.
9. Горбунов К.Ю., Любецкий В.А. 2009. Реконструкция эволюции генов вдоль дерева видов. *Молекуляр. биология.* **43**, 946–958.
10. Lyubetsky V.A., Rubanov L.I., Rusin L.Y., Gorbunov K.Yu. 2012. Cubic time algorithms of amalgamating gene trees and building evolutionary scenarios. *Biol. Direct.* **7**(1), 1–20.
11. Rusin L.Y., Lyubetskaya E.V., Gorbunov K.Yu., Lyubetsky V.A. 2014. Reconciliation of gene and species trees. *Biomed. Res. Int.* 642089, doi: 10.1155/2014/642089.