Программа ChromoGGL

для вычисления расстояний между геномными структурами и построения эволюционных сценариев

Руководство пользователя

(http://lab6.iitp.ru/ru/chromoggl)

(авторы: В.А. Любецкий, К.Ю. Горбунов, Р.А. Гершгорин)

2019 год

Оглавление

Общие сведения

Программа ChromoGGL предназначены для решения следующих **двух** задач:

1) Вычисление расстояния между двумя данными хромосомными (синоним: геномными) структурами и построение кратчайшей последовательности операций, преобразующих одну структуру в другую. Рассматривается общее определение структуры как произвольного множества путей и циклов, представляющих линейные и кольцевые хромосомы, вместе с операциями, которые преобразуют одну структуру в другую. Структуры могут включать паралогичные гены, если указано их правильное соответствие. Последовательность операций допускает переменный генный состав. Допускаются произвольные цены отдельных операций. Задача состоит в минимизации суммарной цены операций в последовательности, которая преобразует одну структуру в другую. Последовательность, на которой достигается минимум, называется *кратчайшей*. Среднее арифметическое суммарных цен при переходе от одной структуры к другой и наоборот называется *расстоянием* между ними. Несмотря на столь общую постановку, предложенный нами оригинальный алгоритм является *линейным* по времени работы и по используемой памяти; и также весьма быстрым при практическом счёте на суперкомпьютере. При этом для него доказывается *точность*, если выполняются некоторые условия на цены отдельных операций. Эти условия отличаются от известных условий.

2) Вычисление матрицы попарных расстояний между хромосомными структурами из данного множества и построение дерева эволюции этих структур, наилучшим образом согласованного с полученной матрицей. Исходные структуры соответствуют листьям дерева. Применяется простой линейный алгоритм, основанный на алгоритме решения задачи 1. Построение дерева на основании вычисленной матрицы осуществляется известными филогенетическими алгоритмами.

В следующем разделе описан алгоритм решения задачи 1, который является ключевым и для решения задачи 2.

Алгоритм вычисления расстояния между двумя хромосомными структурами

Модель хромосомной структуры описывается как конечное множество ориентированных цепей и циклов, включая петли. Такое множество можно рассматривать как ориентированный граф, который будем называть *хромосомной структурой*. Ребро графа будем называть *геном*; отдельную цепь или отдельный цикл графа – *хромосомой* или *компонентой*. Каждому гену приписано имя, обычно *номер i* этого гена, который может повторяться (в случае паралогов) и тогда номер принимает вид *i.j*. Такая модель, как обычно,

означает, что не учитываются длины генов и межгенных участков, как и состав генов и межгенных участков; направление ребра показывает, на какой цепи лежит ген. Вершина графа показывает «место» соединения соседних генов, независимо от их цепи, т.е. в вершине *отождествляются* (мы говорим, *склеиваются*) два края соседних генов. Обычно в структуре много цепей и циклов, что приводит к их своеобразному взаимодействию, поэтому ситуация многих хромосом в структуре решительно отличается от ситуации одной хромосомы.

Модель включает следующие *стандартные* операции над хромосомной структурой. *Двойная переклейка* – расклейка двух склеек краёв генов и новая переклейка четырёх краёв; *полуторная переклейка* – расклейка двух склеенных краёв и склеивание одного края с каким-то несклеенным краем, второй край остаётся *свободным*; *разрез* или *склейка* – соответственно расклейка двух склеенных краёв с образованием двух свободных краёв или склейка двух свободных краёв. Пусть даны хромосомные структуры *a* и *b*, *общим* (*особым*) называется ген, который принадлежит обеим структурам (только одной из них); ген из структуры *a* называется *a-геном*, соответственно определяется *b*-ген. Модель включает две *дополнительные* операции (подразумевается преобразование *a* в *b*): *удаление* (связного максимального) участка особых *a*-генов и *вставка* участка особых *b*-генов. При удалении, если участок находился строго внутри цепи или цикла, два образовавшихся свободных конца общих генов склеиваются между собой; если он находился с краю цепи, край общего гена становится свободным; наконец, если он являлся отдельной хромосомой, она удаляется целиком. Если участок вставляется строго внутри цепи или цикла, место вставки предварительно расклеивается; вставка может выполняться с краю цепи или как новая хромосома. Нетрудно доказать, что использование немаксимальных удалений особых генов не приводит к расширению возможностей, как и разрезание участка особых *a*-генов в первых трёх операциях или операции вставки. Поэтому эти возможности не рассматриваются.

Итак, даны шесть операций и каждой из них приписано положительное рациональное число, которое называется её *ценой*. Включение в модель цен операций является её принципиальной особенностью нашей модели.

Напомним, задача состоит в поиске *кратчайшей* последовательности из этих операций, которая переводит структуру *a* в структуру *b.* Здесь «кратчайшая» означает последовательность, у которой суммарная цена составляющих её операций минимальна. В последовательности каждая операция рассматривается вместе с хромосомной структурой, к которой она применяется.

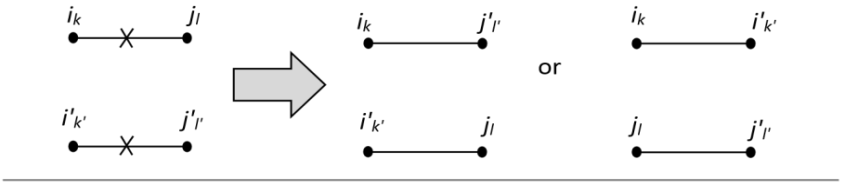
Определение общего графа и его финального вида. В общем графе *a+b* двух структур *a* и *b* вершины – края общих генов и, кроме того, все максимальные участки

особых генов. Каждый край берётся один раз; более формально: вместо края пишется имя гена с индексом 1 или 2, указывающим на его начало или конец. Вершины первого типа называются *обычными*, а второго – *особыми*. Ребро соединяет две обычные вершины, если в одной из структур края склеены, т.е. примыкают друг к другу на хромосоме. Ребро соединяет обычную вершину с особой, если край общего гена склеен с крайним геном участка особых генов. Рёбра из первого случая называются *обычными*, а из второго – *особыми*. В цепи крайнее ребро с особым краем назовём *висячим*. Ребро помечается *a* или *b* в зависимости от того, в какой из них имеется склейка; вершины могут соединяться двумя рёбрами. Особые вершины делятся на *a*- и *b*-вершины. В графе могут быть изолированные вершины – участки из особых генов: если такой участок – цикл, то проводим в нём петлю, которую назовём *особой*. Получается неориентированный граф.

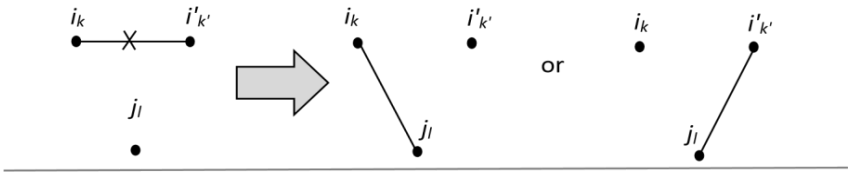
К общему графу *a*+*b* применяются аналоги операций над структурами, которые описываются следующим образом. (1) Удалить два одинаково помеченных ребра и четыре образовавшихся конца соединить двумя новыми неинцидентными рёбрами с той же пометкой. (2) Удалить ребро (скажем, с пометкой *a*) и соединить *a*-ребром один из его концов с обычной вершиной, не инцидентной *a*-ребру или с особой *a*-вершиной, имеющей не более одного инцидентного *a*-ребра. (3) Удалить любое ребро. (4) Добавить ребро (скажем, с пометкой *a*) между вершинами, не инцидентными *a*-ребру. Если в результате операции получаются две инцидентные особые вершины, они сливаются в одну вершину, что входит в определение операции; получаемой вершине приписывается объединение имён исходных вершин. (5) Удаления особой вершины или особой петли; если эта вершина имела две инцидентные ей обычные вершины, они соединяются ребром. Легко определить аналог операция вставки, но оказывается, что без неё можно обойтись без потери общности, что является нетривиальным утверждением.

Стандартные операции

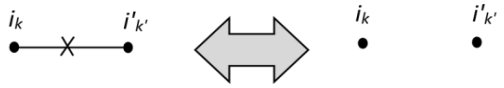
1) Double-cut-and-paste (DP). Исходные рёбра одноимённые.



2) Sesqui-cut-and-paste (SP)



3) Удаление *a*-ребра или добавление *b*-ребра (C) и удаление *b*-ребра или добавление *a*-ребра (J)



Финальный вид общего графа *a*+*b* определяется как общий граф, состоящий из изолированных обычных вершин и *финальных 2-циклов*. Последнее определяется как граф из двух обычных вершин, соединённых обычными рёбрами, одно из *a* и одно из *b.* Легко показать, что *исходная задача эквивалентна* приведению графа *a*+*b* к финальному виду при некоторых ограничениях на цены операций, в том числе, если операции, кроме вставки и удаления, имеют одинаковые цены.

Алгоритм приведения общего графа к финальному виду в случае различных цен операций и дополнительных операций удаления и вставки

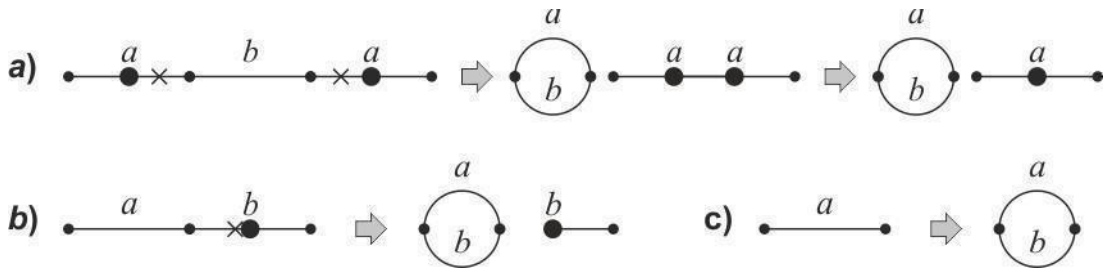
Напомним: в этом разделе соответствие паралогов предполагается известным, но допускаются все операции, неравный генный состав и любые цены операций. По исходным структурам *a* или *b* тривиально строится общий граф *a*+*b*. Цель алгоритма – привести *a*+*b* к финальному виду.

Опишем алгоритм приведения общего графа к финальному виду. При единой общей схеме алгоритм зависит от соотношения цен операций, которые заранее фиксируются.

Шаг 1. Удалить все особые *a*-петли.

Шаг 2. Удалить обычное ребро, не входящее в 2-цикл, и замкнуть его в финальный 2-цикл двойной (если внутреннее) или полуторной (если крайнее) переклейкой или склейкой (если единственное). Повторять действие, пока возможно. Если цена двойной переклейки не

больше цены полуторной, то сначала выполняются все двойные переклейки, иначе – все полуторные.



Перед формальным описанием Шагов 3-4 поясним их смысл. На шаге 3 определённые комбинации от двух до четырёх цепей несколькими операциями заменяются на одну цепь. Каждая применяемая операция уменьшает число особых вершин на 1. Принципиальный момент состоит в определении этих комбинаций. Это делается через понятие типа, которое мы введём чуть ниже. Шаг 4 применяется, если удаление *b*-вершины дороже по цене всех других операций. Полученное к этому шагу множество хромосом (т.е. путей и циклов) разбивается на пары, совместная обработка которых заменяет операцию удаления *b*-вершины на более дешёвую операцию переклейки со слиянием двух *b*-вершин; при этом общее число операций не меняется. Нами доказано, что порядок действий, указанный ниже в шагах 3-4, обеспечивает оптимальность результата.

Итак, перейдём к определению типа цепи и цикла. *Длиной* цепи или цикла назовём число внутренних особых вершин сложенное с числом обычных рёбер. *2-Цикл* – цикл длины 2. *Нечётной* (*чётной*) назовём цепь, у которой длина нечётная (чётная). *a-Цепью* называется изолированная *b*-вершина или нечётная цепь, у которой крайние ребра с обычным краем помечены *a*; симметрично определяется *b-цепь*. Оставшимся после шагов 1-2 цепям и циклам (исключая финальные 2-циклы и изолированные обычные вершины), *приписываем тип*: 2-циклу, содержащему *a*-вершину, но не *b*-вершину, – «*a*-цикл»; симметрично – «*b*-цикл». Циклу, в котором имеются как *a*-, так и *b*-вершины, приписываем тип «цикл». Особой *b*-петле приписываем тип «петля». *a*-Цепи *приписываем тип*: 1*a*, если в ней одно висячее ребро; 2*a*, если в ней два таких ребра; 2*a'*, если она – изолированная *b*-вершина; 3*a*, если в ней нет висячих рёбер, но имеются как *a*-, так и *b*-вершины (тогда длина цепи строго больше 1); 3*a'*, если нет ни висячих рёбер, ни *b*-вершин (тогда длина цепи равна 1). *b*-Цепям тип приписывается аналогично. Чётной цепи *приписываем тип*: 1, если в ней одно висячее ребро и имеются *b*- и *a*-вершины; 1′, если она состоит из одной обычной вершины и инцидентной ей *a*-вершины; 1″, если состоит из одной обычной вершины и инцидентной ей *b*-вершины; 2, если в ней два висячих ребра и имеется невисячее ребро; 2′, если в ней только два висячих ребра; 3, если ребра имеются, но нет висячих рёбер. Цепи типа 1 разделим на цепи типов 1_a, если крайняя особая вершина – *a*-вершина, и 1_b, если она – *b*-вершина.

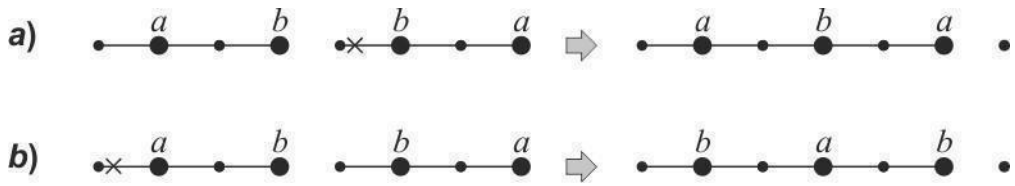
Введём особый тип 1_c: он означает *отложенный выбор между цепями* типов 1_a и 1_b – двумя возможными результатами операции. Алгоритм хранит оба результата до шагов 4.15–4.23, на которых из каждой пары результатов выбирается один и, тем самым, однозначно определяется цепочка операций.

Тип $2a$ означает объединение типов $2a$ и $2a'$, тип $3b$ – типов $3b$ и $3b'$, тип 1_b – типов 1_b и $1''$; тип 2 – типов 2 и $2'$.

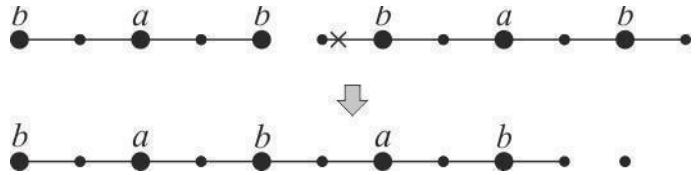
Ниже в описании шага 3 перед знаком «=» перечисляются (разделяемые знаком «+») типы цепей, их комбинация преобразуется в комбинацию цепей с типами, указанными после «=»; в полученной комбинации изолированные обычные вершины и финальные 2-циклы не указываем, им не приписан тип. Если ниже пункт содержит несколько равенств, то описывается действие, относящиеся к первому равенству; другие действия аналогичны.

Шаг 3. В указанном порядке выполняем следующие действия, каждое из них до тех пор, пока оно возможно.

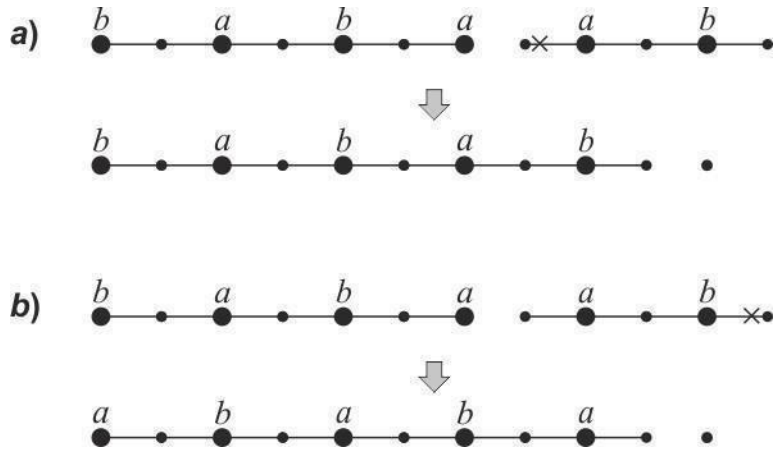
3.1. $1a+1b=1_c$. Расклеим крайнее невисячее ребро (назовём его *внешним*) в одной из двух цепей типа $1a$ или $1b$ и соответствующую особую вершину склеим с крайней особой вершиной другой цепи (полуторная переклейка). Два варианта показаны на рисунке:



3.2. $2a+3b=1_b$, $2b+3a=1_a$, $2b'+3a=1_a$, $2b+3a'=1_a$, $2b'+3a'=1'$. в $3b$-цепи расклеим внешнее ребро и особую вершину склеим с крайней особой вершиной $2a$-цепи:



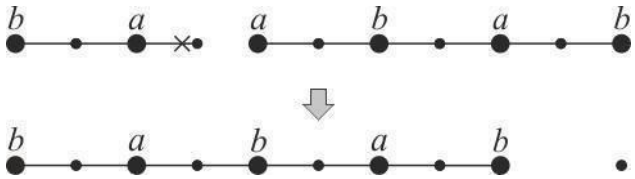
3.3. $2+3=1_c$. В 3-цепи расклеим внешнее ребро и особую вершину склеим с крайней особой вершиной 2-цепи. В зависимости от того, какое из двух внешних ребро расклеивается, в результате получается цепь типа 1_a или 1_b, рис. *a* и *b*.



3.4. $1b+2a+3=2+3=1_c$, $1a+2b+3=2+3=1_c$, $1a+2b'+3=2+3=1_c$. Сначала выполняем $1b+2a=2$ (описание ниже); затем $2+3=1_c$.

3.5. $1a+3b+2=3+2=1_c$, $1b+3a+2=3+2=1_c$, $1b+3a'+2=3+2=1_c$. Сначала $1a+3b=3$ (описание ниже); затем $2+3=1_c$.

3.6. $1a+2=2a$, $1b+2=2b$. В $1a$-цепи расклеим внешнее ребро и особую вершину склеим с крайней особой вершиной 2-цепи:



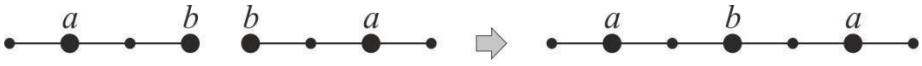
3.7. $1a+3=3a$, $1b+3=3b$. В 3-цепи расклеим крайнее b-ребро и особую вершину склеим с крайней особой вершиной $1a$-цепи:

3.8. $1a+1a+2b+3b=2+3=1_c$, $1a+1a+2b'+3b=2+3=1_c$, $1b+1b+2a+3a=2+3=1_c$, $1b+1b+2a+3a'=2+3=1_c$. Сначала выполняем $1a+2b=2$ и $1a+3b=3$; затем $2+3=1_c$.

3.9. $1a+1a+2b=3a+2b=1_a$, $1a+1a+2b'=3a+2b'=1_a$, $1b+1b+2a=3b+2a=1_b$. Сначала $1a+1a=3a$ (описание ниже); затем $2b+3a=1_a$.

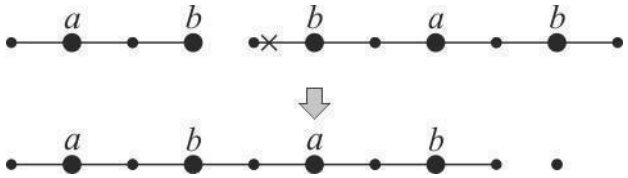
3.10. $1a+1a+3b=1a+3=3a$, $1b+1b+3a=1b+3=3b$, $1b+1b+3a'=1b+3=3b$. Сначала $1a+3b=3$; затем $1a+3=3a$.

3.11. $1a+1a=3a$, $1b+1b=3b$. Склеим крайние особые вершины двух $1a$-цепей:



3.12. $1a+2b=2$, $1a+2b'=2$, $1b+2a=2$. В $1a$-цепи расклеим внешнее ребро и особую вершину склеим с крайней особой вершиной $2b$-цепи:

3.13. 1a+3b=3, 1b+3a=3, 1b+3a'=3. В 3b-цепи расклеим внешнее ребро и особую вершину склеим с крайней особой вершиной 1a-цепи:



3.14. 2a+2b+3+3=2+3=1$_c$, 2a+2b'+3+3=2+3=1$_c$. Сначала 2a+2b+3=2 (описание ниже), затем 2+3=1$_c$.

3.15. 3a+3b+2+2=3+2=1$_c$, 3a'+3b+2+2=3+2=1$_c$. Сначала 3a+3b+2=3 (описание ниже), затем 2+3=1$_c$.

3.16. 2a+3+3=1a+3=3a, 2b+3+3=1b+3=3b, 2b'+3+3=1b+3=3b. Сначала 2a+3=1a (описание ниже в шаге 4), затем 1a+3=3a.

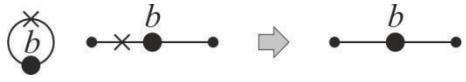
3.17. 3b+2+2=1b+2=2b, 3a+2+2=1a+2=2a, 3a'+2+2=1a+2=2a. Сначала 3b+2=1b (описание ниже в шаге 4); затем 1b+2=2b.

3.18. 2a+2b+3=2a+1b=2, 2a+2b'+3=2a+1b=2. Сначала 2b+3=1b; затем 1b+2a=2.

3.19. 3a+3b+2=3a+1b=3, 3a'+3b+2=3a'+1b=3. Сначала 3b+2=1b; затем 1b+3a=3.

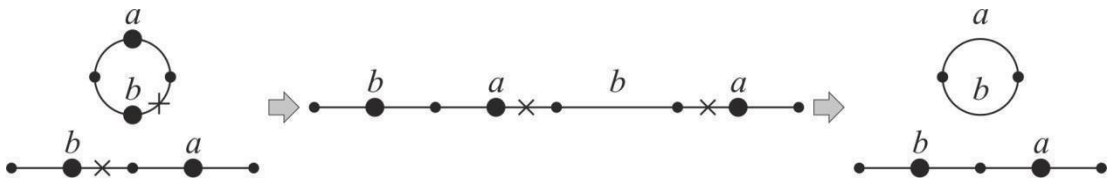
Шаг 4. Если цена двойной переклейки больше цены полуторной, то в следующем порядке применяем каждый указанный пункт 4.1–4.24, пока это возможно; иначе таким же образом применяем пункты 4.1′–4.24′.

4.1. «петля»+любой тип t с b-вершиной = тип t. Объединить b-вершину петли с b-вершиной компоненты типа t двойной переклейкой (если эта цепь не изолированная b-вершина) или полуторной переклейкой (иначе):



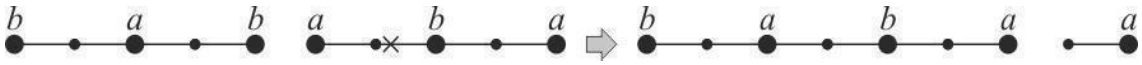
4.1′. То же, что и 4.1.

4.2. «цикл»+любой тип t с b-вершиной и a-вершиной = тип t. Вставить цикл (двойной переклейкой, сливающей две b-вершины) рядом с b-вершиной из компоненты типа t с той стороны, в которой находится a-вершина; образовавшееся обычное ребро вырезать:



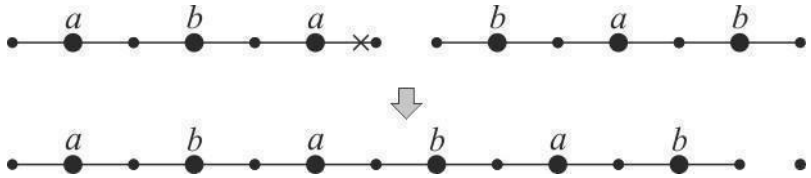
4.2′. То же, что и 4.2.

4.3. $2a+2b=2+1'$. Полуторная переклейка с отрезанием двух вершин $2b$-цепи (крайней a-вершины и соседней обычной вершины) и склейкой образовавшегося края с крайней b-вершиной $2a$-цепи:



4.3′. $2a'+2b=2+1'$.

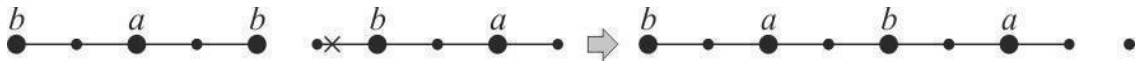
4.4. $3a+3b=3$. В $3a$-цепи расклеить внешнее ребро и особую вершину склеить с крайней обычной вершиной $3b$-цепи:



Ниже рассматривается первый случай, остальные аналогично.

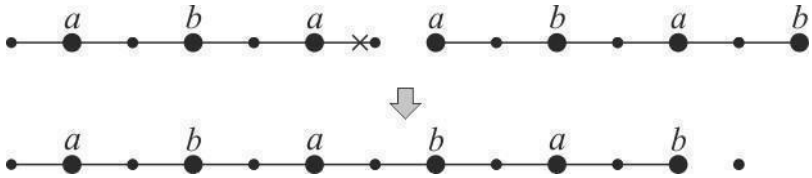
4.4′. $3a+3b'=3$.

4.5. $2a+3=1a$, $2b+3=1b$. В 3-цепи расклеить внешнее b-ребро и особую вершину склеить с крайней особой вершиной $2a$-цепи:



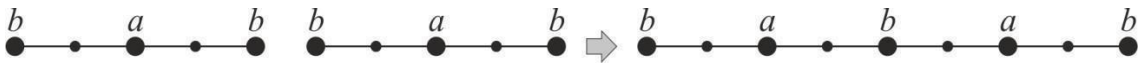
4.5′. $2a'+3=1a$.

4.6. $3a+2=1a$, $3b+2=1b$. В $3a$-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной 2-цепи:



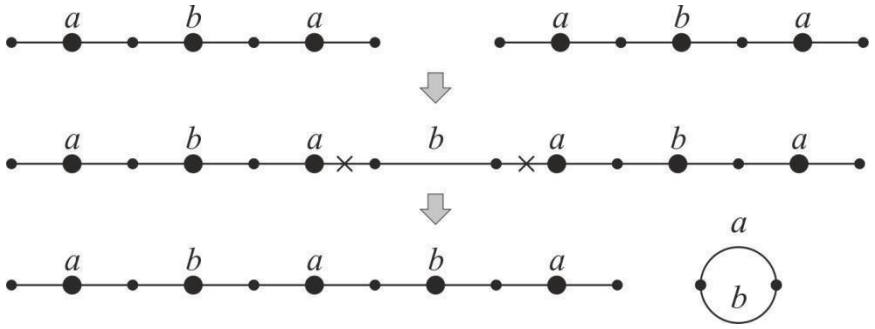
4.6′. $3a+2'=1a$, $3b'+2=1b$.

4.7. $2a+2a=2a$, $2b+2b=2b$. Склеить крайние особые вершины двух цепей:



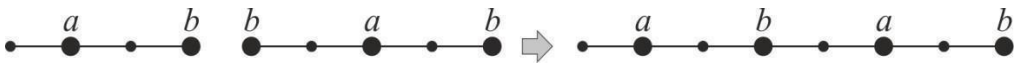
4.7′. $2a'+2a=2a$.

4.8. $3a+3a=3a$, $3b+3b=3b$. Две крайние обычные вершины цепей соединить обычным ребром с последующим его вырезанием:



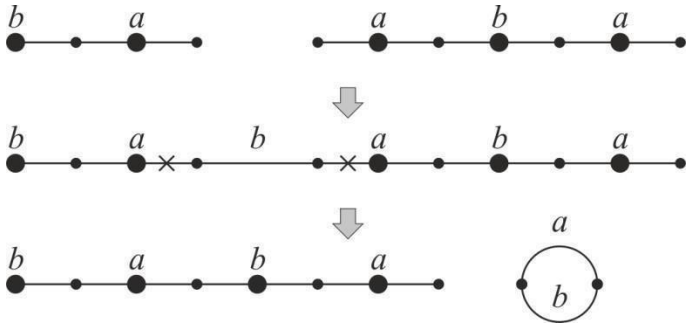
4.8′. $3b'+3b=3b$.

4.9. $1a+2a=1a$, $1b+2b=1b$. Склеить крайние особые вершины двух цепей:



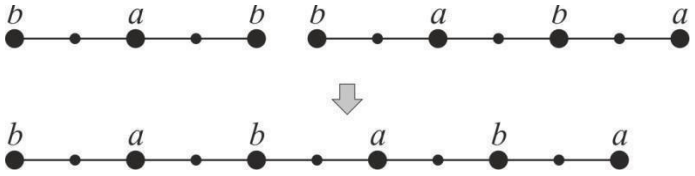
4.9′. $1a+2a'=1a$.

4.10. $1a+3a=1a$, $1b+3b=1b$. Две крайние обычные вершины цепей соединить обычным ребром с последующим его вырезанием:



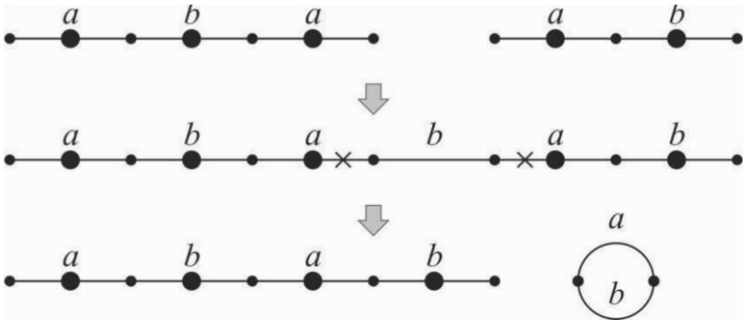
4.10′. $1b+3b'=1b$.

4.11. $2a+2=2$, $2b+2=2$. Склеить крайние особые вершины двух цепей:



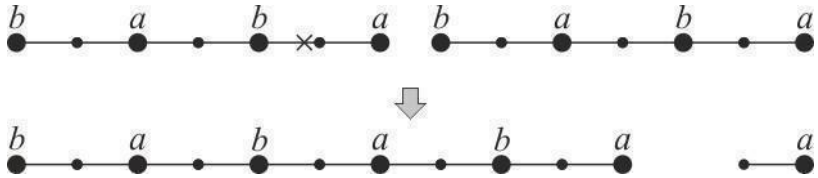
4.11′. $2a'+2=2$, $2a+2'=2$, $2b+2'=2$.

4.12. $3a+3=3$, $3b+3=3$. Две крайние обычные вершины цепей соединить обычным ребром с последующим его вырезанием:



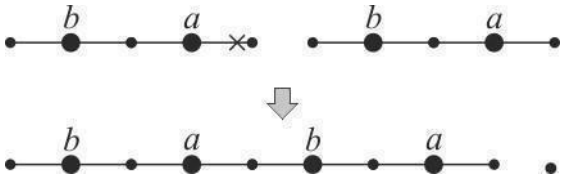
4.12′. $3b'+3=3$.

4.13. $2+2=2+1'$. Полуторная переклейка с отрезанием двух вершин 2-цепи (крайней особой a-вершины и соседней обычной) и склейкой образовавшегося края с крайней b-вершиной другой 2-цепи:



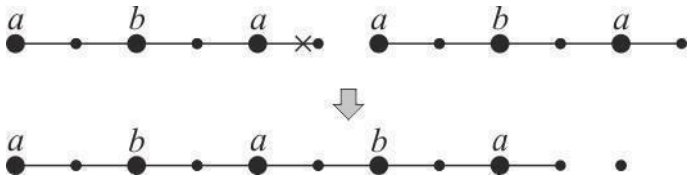
4.13′. $2'+2=2+1'$.

4.14. $3+3=3$. В 3-цепи расклеить внешнее a-ребро и образовавшийся край этой цепи склеить с b-краем другой 3-цепи:



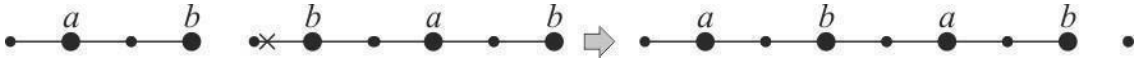
4.14′. Пустое действие.

4.15. $1_a+1_a=1_a$, $1_b+1_b=1_b$, $1_b+1_c=1_b$ (определяется $c=b$). В 1_a-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной другой 1_a-цепи:



4.15′. $1''+1_b=1_b$, $1''+1_c=1_b$ (определяется $c=b$).

4.16. $1a+1_b=1a$, $1b+1_a=1b$, $1a+1_c=1a$ (определяется $c=b$). В 1_b-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной $1a$-цепи:

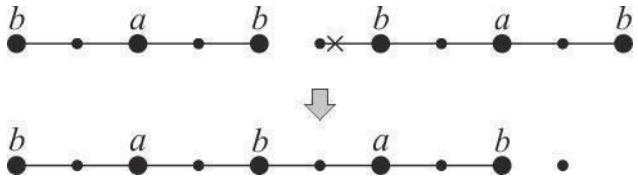


4.16′. $1a+1''=1a$.

4.17. $1a+1_a=1a$, $1b+1_b=1b$, $1b+1_c=1b$ (определяется $c=b$). В $1a$-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной $1a$-цепи:

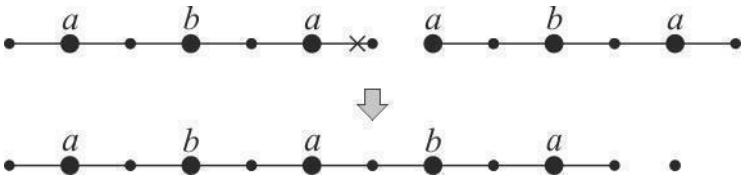
4.17′. $1b+1''=1b$.

4.18. $2a+1_b=2a$, $2b+1_a=2b$, $2a+1_c=2a$ (определяется $c=b$). В 1_b-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной $2a$-цепи:



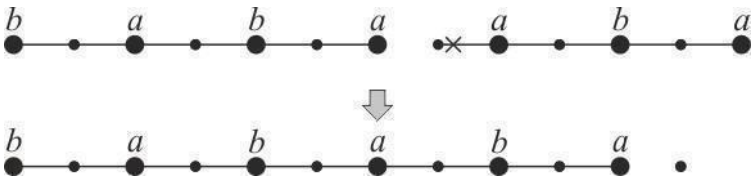
4.18′. $2a'+1_b=2a$, $2a+1''=2a$, $2a'+1_c=2a$ (определяется $c=b$).

4.19. $3a+1_a=3a$, $3b+1_b=3b$, $3b+1_c=3b$ (определяется $c=b$). В $3a$-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной 1_a-цепи:



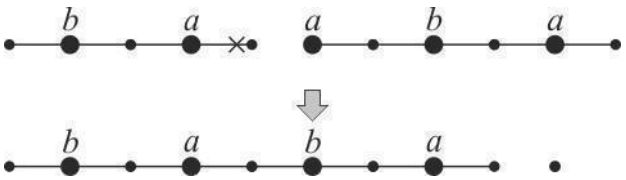
4.19′. $3b'+1_b=3b$, $3b+1''=3b$, $3b'+1_c=3b$ (определяется $c=b$).

4.20. $2+1_a=2$, $2+1_b=2$, $2+1_c=2$ (определяется $c=b$). В 1_a-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной 2-цепи:



4.20′. $2'+1_a=2$, $2'+1_b=2$, $2+1''=2$, $2'+1_c=2$ (определяется $c=b$).

4.21. $3+1_a=3$, $3+1_b=3$, $3+1_c=3$ (определяется $c=b$). В 3-цепи расклеить внешнее ребро и особую вершину склеить с крайней особой вершиной 1_a-цепи:



4.21′. $3+1''=3$.

4.22. $1_a+1_c=1_a$ (определяется $c=a$), $1b+1_c=1b$ (определяется $c=a$), $1a+1_c=1a$ (определяется $c=a$), $2b+1_c=2b$ (определяется $c=a$), $3a+1_c=3a$ (определяется $c=a$).

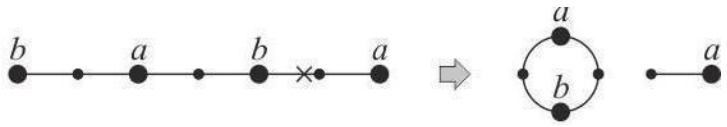
4.22′. Пустое действие.

4.23. Для оставшихся цепей типа 1_c определяем $c=b$ и выполняем $1_b+1_b=1_b$.

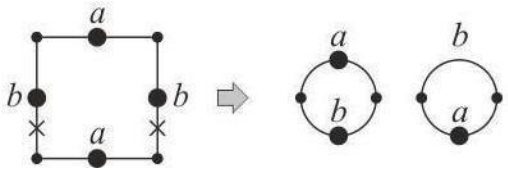
4.23′. Пустое действие.

4.24. Цепи, имеющие невисячее ребро, замыкаем в циклы склейкой (цепи типа $2a$, $2b$, $3a$, $3b$), полуторной переклейкой со слиянием особых вершин (цепи типа 1_a, 1_b, 1_c, 2) или без слияния (цепи типа $1a$, $1b$, 3). При замыкании цепи типа 1_c определяем $c=b$. При замыкании цепи типа 2 выбираем вариант со слиянием двух b-вершин, из образовавшейся цепи типа 1' удаляем a-вершину, см. рисунок ниже. Из циклов, получившихся при замыкании цепей типа $3a$ или $3b$, вырезаем обычные ребра. Затем снова применяем шаг 4.2.

4.24′. То же, что и 4.24.



Шаг 5. Удаляем изолированные особые вершины и петли. Из оставшихся цепей удаляем особые вершины. Из циклов длины больше 2 вырезаем 2-циклы так, чтобы происходило слияние двух b-вершин (соответственно, в 2-цикл включается a-вершина), см. рисунок ниже. Из 2-циклов удаляем особые вершины.



Конец описания алгоритма.

Алгоритм построения дерева эволюции хромосомных структур

Построение эволюционного дерева, оптимально согласованного с полученной матрицей расстояний для данного множества хромосомных структур, выполняется простыми алгоритмами, подобными алгоритмам иерархической кластеризации UPGMA или NJ (neighbor joining).

Установка и запуск в среде Windows программы ChromoGGL

Программа ChromoGGL написана на языке C++ и имеет интерфейс командной строки. Для скачивания доступна 32-битная версия исполняемого модуля. Она не требует специальной установки.

Для использования программы достаточно: скачать архив с программой *chromoggl.zip* и распаковать его в выбранную папку (например, f:/Chromo); запустить командный процессор Windows (Пуск > Выполнить > cmd) и перейти в папку с содержимым архива:

f:

cd f:/Chromo;

выполнить команду: chromoggl –h.

В случае успешного запуска на экране появляется краткая инструкция по запуску программы. В ином случае на экране появляется информация о возникшей ошибке. В случае успешного запуска рекомендуется протестировать программу на имеющихся в архиве примерах с помощью команд run_example_*X*.bat, где *X* пробегает значения 1 – 5.

Входные данные программы ChromoGGL

Файл структур

На вход программы ChromoGGL подается текстовый **файл хромосомных структур**, содержащий одну или более строк со следующими данными: *число* хромосомных структур; *имя* вида/штамма; *число* хромосом, включённых в структуру из этого вида, т.е. число путей и циклов в структуре; *пометка* (*L*), если хромосома линейная, или (*C*), если она циклическая; *число* генов в хромосоме; *последовательность* генов в хромосоме, записанная как имена генов со знаками «+» или «–», которые указывают на транскрибируемую цепь. Например, для следующих двух хромосомных структур их файл приведён ниже.

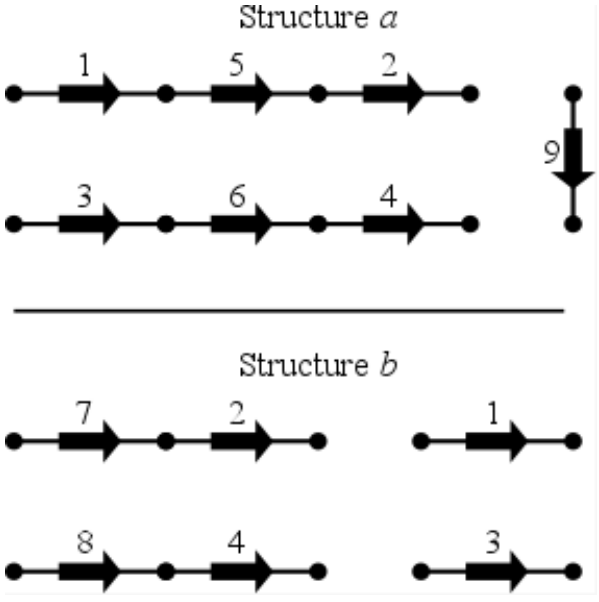


Рис. 1

Файл структур, показанных на рис. 1:

2; Structure_*a*; 3; *L*3: +1+5+2; *L*3: +3+6+4; *L*1: +9; Structure_*b*; 4; *L*2: +7+2; *L*2: +8+4; *L*1: +1; *L*1: +3

Общий граф этих двух структур показан на рис. 2, где точки увеличенного размера отмечают особые вершины, а обычного размера – обычные вершины:

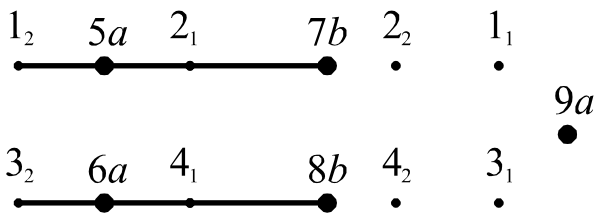


Рис. 2

Напомним: *обычной* называется вершина, соответствующая краю гена, представленного в обеих исходных структурах; *особой* называется вершина, соответствующая максимальному участку генов, представленных лишь в одной из исходных структур (сам участок называется *блоком*). У особой вершины имя структуры относится и к соседним рёбрам. Если особой вершине соответствуют несколько генов, то они выписываются подряд.

Файл с ценами операций

Цены выписываются в строку со следующим форматом, где в качестве их значений могут использоваться любые положительные рациональные числа:

DP = …, SP = …, J = …, C = …, *a*D = …, *b*D = …

Пример файла – строки цен операций:

DP = 1.2, SP = 1.1, J = 1, C = 0.9, *a*D = 0.8, *b*D = 1.5

Файл с весами расстояний разных типов

В этом файле указываются значения весов, с которыми при вычислении расстояния суммируются специальное (т.е. обычное) и брекпойнтовое расстояния между двумя структурами.

Пример файла:

```
[weights]
special=1
breakpoint=0.1
```

Параметры программы ChromoGGL

В командной строке приводятся параметры программы, смысл которых поясняется ниже. Каждый параметр задаётся своим ключом (однобуквенным или длинным) и следующим после него аргументом. Обязательны параметры -c, -o, -m; для прочих параметров есть значение аргумента по умолчанию. Краткая справка о параметрах командной строки выдаётся при запуске программы с ключом -h (--help).

Командная строка

- -c (--chrom) – имя файла хромосомных структур (может включать путь);
- -o (--oper) – имя файла с ценами отдельных операций (может включать путь);
- -m (--mode) – режим запуска, со значением аргумента dist, если решается задача 1, или tree, если решается задача 2;
- -r (--res) – путь к папке с результатами работы программы. По умолчанию используется текущая папка;
- -d (--coeff) – имя файла с ценами отдельных операций (может включать путь); по умолчанию берётся файл distance_weights.ini в текущей папке;
- -p (--method) – алгоритм, применяемый для построения дерева, upgma или nj. По умолчанию используется алгоритм upgma;
- -f – вывод матрицы расстояний дополнительно в формате phylip, если аргумент равен 1. По умолчанию не выполняется.

Выходные данные программы ChromoGGL в задаче 1

В задаче 1 выдаются два файла *joint_graph* – описание общего графа и *shortest_sequence* – описание искомой кратчайшей последовательности.

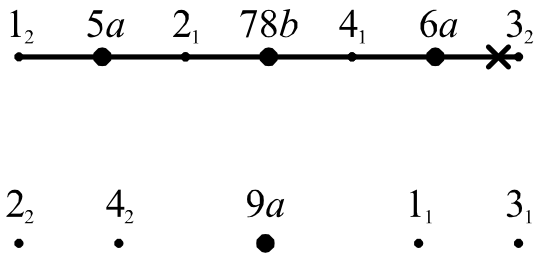
Первый файл имеет вид: заголовок *joint_graph*, заголовок cycles (…), который включает указание числа циклов в общем графе, затем они перечисляются; заголовок paths (…) с указыванием числа путей, которые затем перечисляются. Каждый цикл или путь записывается как линейная последовательность его вершин: общая вершина описывается как имя гена, сопровождаемое цифрой 1, если она – начало гена, и 2, если она – его конец. Особая вершина содержит имена генов, входящих в блок, и в конце имя структуры, к которой относится блок. Если разделение имён генов не является однозначным, то имена генов заключаются в круглые скобки. Ребро между обычными вершинами записывается знаком «_», после него указывается имя структуры, к которой относится ребро. Ребро к особой вершине записывается тем же знаком; имя структуры опускается. В конце файла

указывается число *a*- и *b*-особых вершин в общем графе, сокращенно записываемых как “spnodes”.

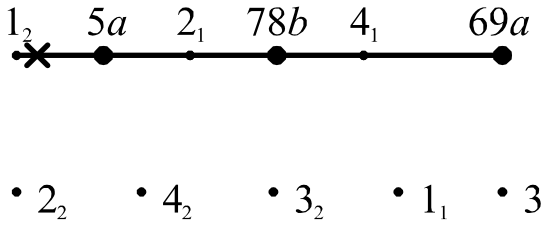
Пример первого файла для общего графа, показанного на рис. 2:
Joint_graph cycles (0): paths (7):1.1; 2.2; 3.1; 4.2; 9*a*; 1.2_5*a*_ 2.1_7*b*;
3.2_6*a*_4.1_8*b* *a*-spnodes: 3, *b*-spnodes: 2

Второй файл содержит кратчайшую последовательность операций, преобразующую общий граф к финальному виду. В его первой строке указаны: *число* операций в последовательности; её *суммарная цена*. Затем перечисляются сами операции. Каждая из них записывается в виде: компоненты, к которым применяются одна или несколько последовательных операций; сокращенные имена операций; полученный результат.

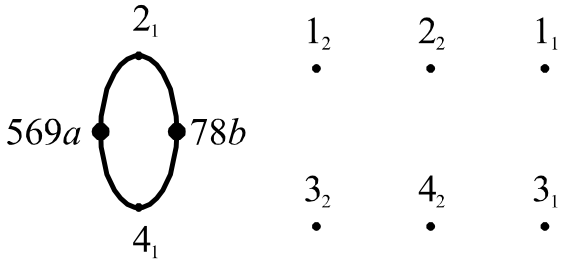
Пример второго файла для общего графа, показанного на рис. 2:
5; 5.4;
[1.2_5*a*_2.1_7*b* (*L*); 3.2_6*a*_4.1_8*b* (*L*)] C [1.2_5*a*_2.1_78*b*_4.1_6*a*_3.2 (*L*)]
Здесь особые вершины 7*b* и 8*b* на двух путях объединяются в особую вершину 78*b* на одном пути. Пометки (*L*) и (*C*) означают «цепь» и «цикл». Ниже изображен результат.



[1.2_5*a*_2.1_78*b*_4.1_6*a*_3.2 (*L*); 9*a* (*L*)] SP [3.2; 1.2_5*a*_2.1_78*b*_4.1_6*a* (*L*)]
Отклейка вершины 3.2 и объединение особых вершин 6*a* и 9*a* в вершину 69*a*. Результат ниже.

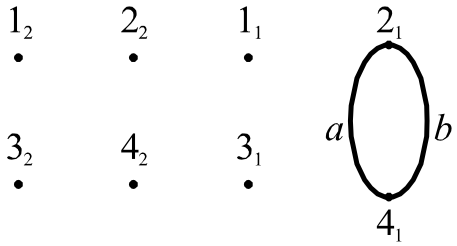


[1.2_5*a*_2.1_78*b*_4.1_6*a* (*L*)] SP [2.1_78*b*_4.1_569*a*_2.1 (*C*); 1.2]
Отклейка вершины 1.2 и объединение особых вершин 5*a* и 69*a* в вершину 569*a*. Ниже результат.



[2.1_78*b*_4.1_569*a*_2.1 (*C*)] *a*D, *b*D [2.1_*a*4.1_*b*2.1 (*C*)]

Удаление особых вершин 569*a* и 78*b*. Ниже результат.



Получен граф финального вида, алгоритм закончил работу.

Примеры решения задачи 1

В архиве с программой находятся несколько примеров решения задачи 1, в том числе искусственный пример №0 – пара структур, показанная на рис. 1, и биологический – пара структур от хромосомы *Salmonella enterica* к хромосоме *Escherichia coli.* Для запуска программы на этих примерах необходимо выполнить соответствующие скрипты *run_example_1.bat* и *run_example_2.bat*, также лежащие в архиве.

Искусственный пример 0

Исходные хромосомные структуры (файл data/artificial_0.txt):

```
2;
Structure_a; 3; L3: +1+5+2; L3: +3+6+4; L1: +9;
Structure_b; 4; L2: +7+2; L2: +8+4; L1: +1; L1: +3;
```

Исходный общий граф для этих структур имеет вид:

```
Joint graph
Cycles (0):
Paths (7): 1.1; b_2.1_a_1.2; 2.2; 3.1; b_4.1_a_3.2; 4.2; a;
a-spnodes: 3, b-spnodes: 2
```

Найденные кратчайшие последовательности для разных наборов цен операций:

- для DP = 0.9, SP = 1.2, J = 1.1, C = 1, aD = 0.8, bD = 1.5 (файл цен config_cir.ini):

```
5; 5.7;
[3.2_6a_4.1_8b(L), 1.2_5a_2.1_7b(L), 9a(L)] C, SP [3.2_6a_4.1_87b_2.1_59a(L), 1.2(L)]
[3.2_6a_4.1_87b_2.1_59a(L)] SP [2.1_659a_4.1_87b_2.1(C), 3.2(L)]
[2.1_659a_4.1_87b_2.1(C)] aD [2.1_a4.1_87b_2.1(C)]
[2.1_a4.1_87b_2.1(C)] bD [2.1_a4.1_b2.1(C)]
```

- для DP = 1, SP = 1, J = 1, C = 1, aD = 1, bD = 1 (файл цен config_equal.ini):

```
5; 5;
[3.2_6a_4.1_8b(L), 1.2_5a_2.1_7b(L), 9a(L)] C, SP [3.2_6a_4.1_87b_2.1_59a(L), 1.2(L)]
[3.2_6a_4.1_87b_2.1_59a(L)] SP [2.1_659a_4.1_87b_2.1(C), 3.2(L)]
[2.1_659a_4.1_87b_2.1(C)] aD [2.1_a4.1_87b_2.1(C)]
[2.1_a4.1_87b_2.1(C)] bD [2.1_a4.1_b2.1(C)]
```

- для DP = 1.2, SP = 1.1, J = 1, C = 0.9, aD = 0.8, bD = 1.5 (файл цен config_lin.ini):

```
5; 5.4;
[3.2_6a_4.1_8b(L), 1.2_5a_2.1_7b(L), 9a(L)] C, SP [3.2_6a_4.1_87b_2.1_59a(L), 1.2(L)]
[3.2_6a_4.1_87b_2.1_59a(L)] SP [2.1_659a_4.1_87b_2.1(C), 3.2(L)]
[2.1_659a_4.1_87b_2.1(C)] aD [2.1_a4.1_87b_2.1(C)]
[2.1_a4.1_87b_2.1(C)] bD [2.1_a4.1_b2.1(C)]
```

Приведенные кратчайшие последовательности лежат вместе с общим графом в папке results/artificial_0 в подпапках cir, equal и lin, соответственно.

Другие искусственные примеры

Также программа тестировалась на наборах искусственных данных №№ 1–8, которые содержатся в дистрибутивном архиве. Для запуска программы на любом из этих примеров скрипт run_example_1.bat запускается с параметром – номером примера. Например, для вычисления примера №4 следует ввести команду:

run_example_1.bat 4.

Исходные хромосомные структуры для примера с номером X содержатся в файле data/artificial_X.txt. Найденные кратчайшие последовательности для тех же трёх наборов цен операций лежат вместе с общим графом в папке results/ artificial_X в подпапках cir, equal и lin, соответственно и здесь для краткости не приводятся.

Биологический пример

Исходные хромосомные структуры (файл data/Salmonella_Escherichia.txt):

```
2;
Salmonella_enterica; 1; C47: -rpl25-rpsV-rplT-rpsA-rplS-rpsP+rpsU-rplU-rps1-rplM-rplQ-rpsD-rps11-rps13-rplO-
rpsE-rplR-rplF-rpsH-rpsN-rplE-rplX-rpl14-rpsQ-rpl29-rplP-rpsC-rplV-rpsS-rplB-rplW-rplD-rpl3-tufA-rps7-
rps12+rpoZ+tufA+rpl11+rplA+rplJ+rplL+rpoB+rpoc1 +rpsF+rpsR+rpl1;
Escherichia_coli; 1; C56: -rpsT+rpsB+rpsA-rpsV-rplT+rplY-rpoE-rplS-rpsP-rpoS+rpsU+rpoD-rpsO-rplU+rpoN+rpoN-
rps1-rplM-rplQ-rpoA-rpsD-rpsK-rpsM-rplO-rpsE-rplR-rplF-rpsH-rpsN-rplE-rplX-rplN-rpsQ-rplP-rpsC-rplV-rpsS-rplB-
rplW-rplD-rplC-rpsJ-tufA-rpsG-rpsL-rpoH-rpoZ+rplK+rplA+rplJ+rplL+rpoB+rpoC+rpsF+rpsR+rpl1;
```

Исходный общий граф для этих структур:

```
Joint graph
Cycles (33): rpsV.1_brplT.2_arpsV.1;
rpsV.2_brpsA.2_arplT.1_rplYrpoEb_rplS.2_arpsA.1_rpsBrpsTb_rpl1.2_rpl25a_rpsV.2; rplS.1_brpsP.2_arplS.1;
rpsP.1_rpoSb_rpsU.1_arpsP.1; rpsU.2_rpoDrpsOb_rplU.2_arpsU.2; rplU.1_rpoNrpoNb_rps1.2_arplU.1;
rps1.1_brplM.2_arps1.1; rplM.1_brplQ.2_arplM.1; rplQ.1_rpoAb_rpsD.2_arplQ.1;
rpsD.1_rpsKrpsMb_rplO.2_rps13rps11a_rpsD.1; rplO.1_brpsE.2_arplO.1; rpsE.1_brplR.2_arpsE.1;
rplR.1_brplF.2_arplR.1; rplF.1_brpsH.2_arplF.1; rpsH.1_brpsN.2_arpsH.1; rpsN.1_brplE.2_arpsN.1;
```

rplE.1_brplX.2_arplE.1; rplX.1_rplNb_rpsQ.2_rpl14a_rplX.1; rpsQ.1_brplP.2_rpl29a_rpsQ.1;
rplP.1_brpsC.2_arplP.1; rpsC.1_brplV.2_arpsC.1; rplV.1_brpsS.2_arplV.1; rpsS.1_brplB.2_arpsS.1;
rplB.1_brplW.2_arplB.1; rplW.1_brplD.2_arplW.1; rplD.1_rplCrpsJb_tufA.2_rpl3a_rplD.1;
tufA.1_rpsGrpsLrpoHb_rpoZ.2_tufArpl11a_rplA.1_rplKb_rpoZ.1_rps12rps7a_tufA.1; rplA.2_brplJ.1_arplA.2;
rplJ.2_brplL.1_arplJ.2; rplL.2_brpoB.1_arplL.2; rpoB.2_rpoCb_rpsF.1_rpoc1a_rpoB.2; rpsF.2_brpsR.1_arpsF.2;
rpsR.2_brpl1.1_arpsR.2;
Paths (0):
a-spnodes: 8, b-spnodes: 12

Найденные кратчайшие последовательности для тех же трёх наборов цен операций лежат вместе с общим графом в папке results/Salmonella_Escherichia в подпапках cir, equal и lin, соответственно. Например, для первого набора цен (DP = 0.9, SP = 1.2, J = 1.1, C = 1, aD = 0.8, bD = 1.5) построена следующая кратчайшая последовательность:

21; 21.6;
[rpsV.2_brpsA.2_arplT.1_b_rplS.2_arpsA.1_b_rpl1.2_a_rpsV.2(C)] DP [rpsV.2_brpsA.2_arpsV.2(C),
rplT.1_b_rplS.2_arpsA.1_b_rpl1.2_a_rplT.1(C)] , step:
[rplT.1_b_rplS.2_arpsA.1_b_rpl1.2_a_rplT.1(C)] DP [rplS.2_arpsA.1_brplS.2(C), rplT.1_b_rpl1.2_a_rplT.1(C)] , step:
[rpoB.2_a_rpsF.1_b_rpoB.2(C), tufA.1_a_rpoZ.1_b_rplA.1_a_rpoZ.2_b_tufA.1(C)] DP, DP
[tufA.1_a_rpoB.2_b_rplA.1_a_rpoZ.2_b_tufA.1(C), rpoZ.1_brpsF.1_arpoZ.1(C)] , step: 4.2
[tufA.1_a_rpoB.2_b_rplA.1_a_rpoZ.2_b_tufA.1(C), rplD.1_a_tufA.2_b_rplD.1(C)] DP, DP
[rplD.1_a_tufA.1_b_rpoZ.2_a_rplA.1_b_rplD.1(C), tufA.2_brpoB.2_atufA.2(C)] , step:
[rplD.1_a_tufA.1_b_rpoZ.2_a_rplA.1_b_rplD.1(C), rplX.1_a_rpsQ.2_b_rplX.1(C)] DP, DP
[rplX.1_a_rplD.1_b_rplA.1_a_rpoZ.2_b_rplX.1(C), rpsQ.2_btufA.1_arpsQ.2(C)] , step:
[rplX.1_a_rplD.1_b_rplA.1_a_rpoZ.2_b_rplX.1(C), rpsD.1_a_rplO.2_b_rpsD.1(C)] DP, DP
[rpsD.1_a_rplX.1_b_rpoZ.2_a_rplA.1_b_rpsD.1(C), rplO.2_brplD.1_arplO.2(C)] , step:
[rpsD.1_a_rplX.1_b_rpoZ.2_a_rplA.1_b_rpsD.1(C), rplT.1_b_rpl1.2_a_rplT.1(C)] DP, DP
[rpsD.1_b_rplA.1_a_rpoZ.2_b_rpl1.2_a_rpsD.1(C), rplT.1_brplX.1_arplT.1(C)] , step:
[rpsD.1_b_rplA.1_a_rpoZ.2_b_rpl1.2_a_rpsD.1(C)] DP [rpsD.1_b_rpl1.2_a_rpsD.1(C), rplA.1_a_rpoZ.2_brplA.1(C)]
, step:
[rpsP.1_arpsU.1_b_rpsP.1(C)] bD [rpsP.1_arpsU.1_brpsP.1(C)] , step:
[rpsU.2_arplU.2_b_rpsU.2(C)] bD [rpsU.2_arplU.2_brpsU.2(C)] , step:
[rplU.1_arps1.2_b_rplU.1(C)] bD [rplU.1_arps1.2_brplU.1(C)] , step:
[rplQ.1_arpsD.2_b_rplQ.1(C)] bD [rplQ.1_arpsD.2_brplQ.1(C)] , step:
[rpsQ.1_a_rplP.2_brpsQ.1(C)] aD [rpsQ.1_arplP.2_brpsQ.1(C)] , step:
[rplA.1_a_rpoZ.2_brplA.1(C)] aD [rplA.1_arpoZ.2_brplA.1(C)] , step:
[rpsD.1_b_rpl1.2_a_rpsD.1(C)] bD [rpsD.1_brpl1.2_a_rpsD.1(C)] , step:
[rpsD.1_brpl1.2_a_rpsD.1(C)] aD [rpsD.1_brpl1.2_arpsD.1(C)] , step:

Выходные данные программы ChromoGGL в задаче 2

В задаче 2 алгоритм выдает в папку для записи результатов следующие файлы: *distance.tsv* – матрица попарных расстояний между хромосомными структурами, в формате TSV (значения, разделённые символом табуляции) для импорта в Excel; *tree.tre* – дерево эволюции хромосомных структур, оптимально согласованное с матрицей, в скобочном формате Newick, в том числе с указанием длин рёбер (в текущей версии – только при построении дерева методом neighbor joining). Также при значении 1 параметра -f выдается еще два файла: *infile* – матрица попарных расстояний в формате phylip, предназначенная для использования в пакетах phylip, например, построения дерева с помощью программы Neighbor (http://evolution.genetics.washington.edu/phylip/doc/neighbor.html). В этой матрице

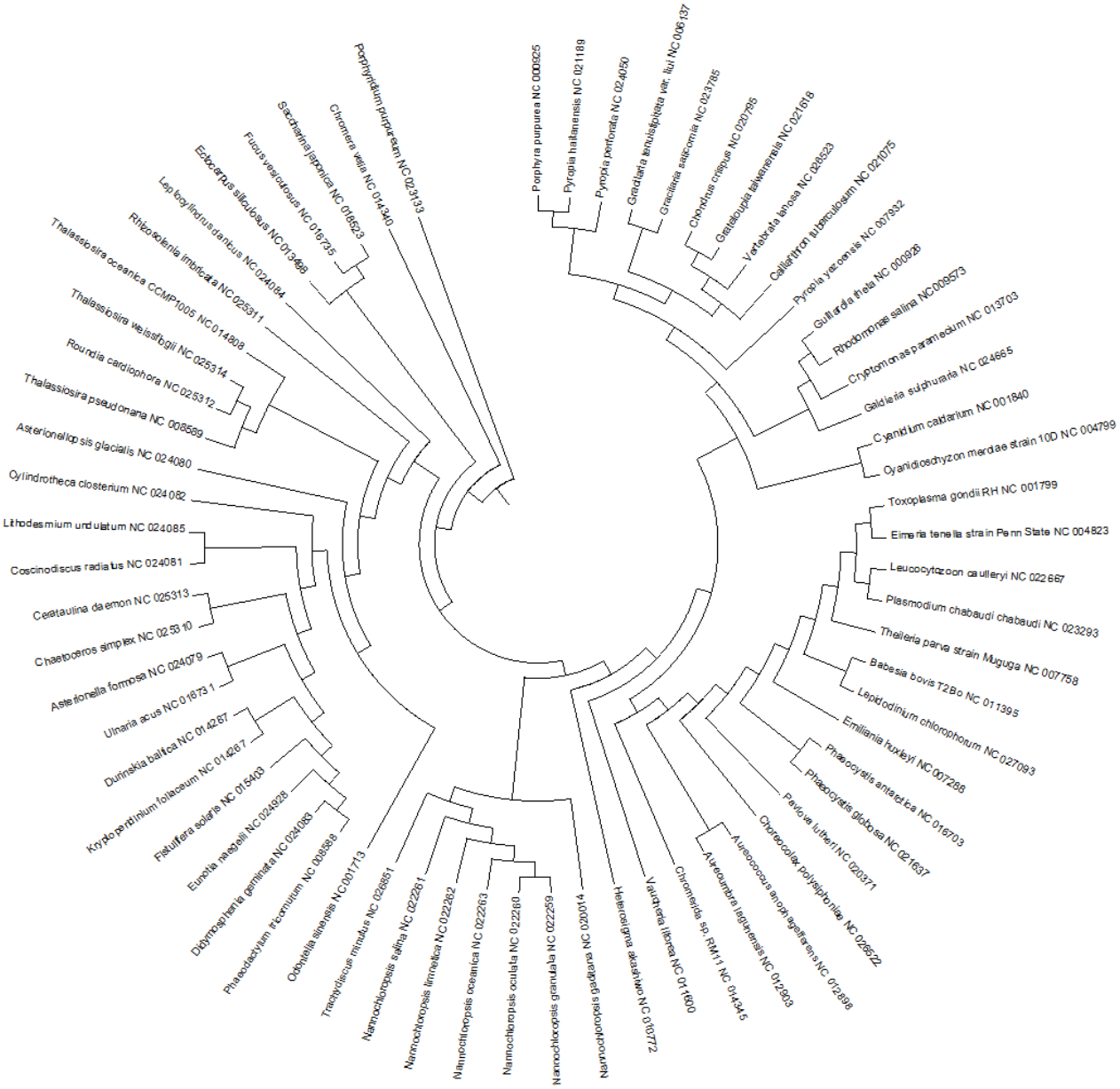
названия видов заменяются номерами из Генбанка, так как phylip поддерживает только короткие имена. Соответствие номеров и имен записывается в файл *map.txt*.

Примеры решения задачи 2

Пластиды родофитной ветви

Первый пример множества хромосомных структур содержит 66 пластид родофитной ветви. Исходные хромосомные структуры содержатся в файле data/rhodophytic_branch.txt. Для запуска программы на этом примере необходимо выполнить скрипт run_example_3.bat.

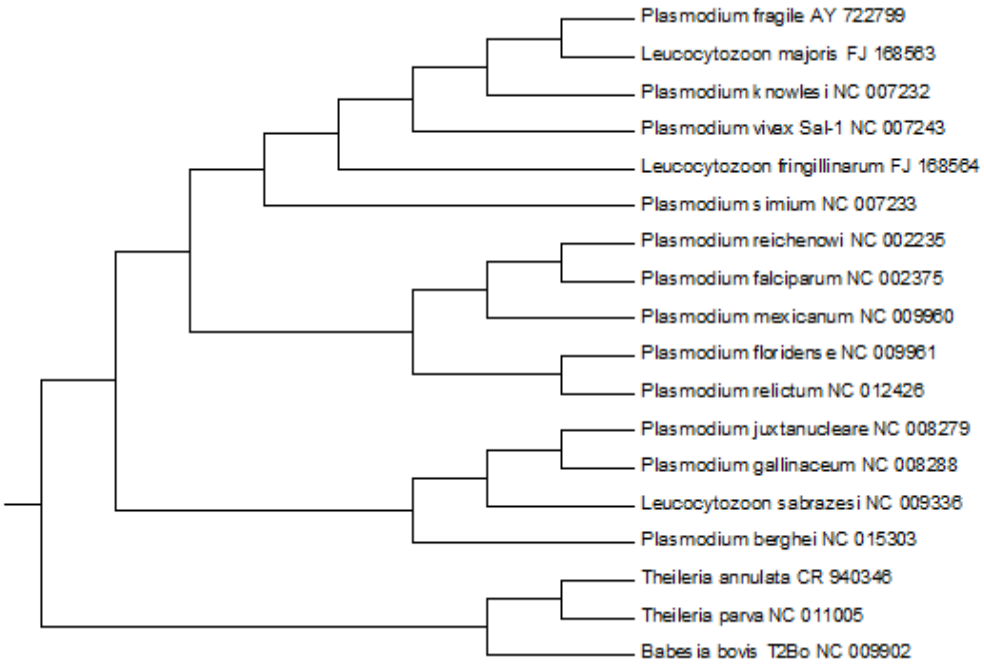
Построенная в результате запуска матрица попарных расстояний содержится в файле results/rhodophytic_branch/distance.tsv, а дерево в формате Newick – в файле results/rhodophytic_branch/tree.tre (ниже приводится его визуализация).



Митохондрии класса Aconoidasida

Второй пример множества хромосомных структур содержит 18 митохондрий класса *Aconoidasida*. В нём представлены как кольцевые, так и линейные хромосомы. Исходные хромосомные структуры содержатся в файле data/mitochondria.txt. Для запуска программы на этом примере необходимо выполнить скрипт run_example_4.bat.

Построенная в результате запуска матрица попарных расстояний содержится в файле results/mitochondria/distance.tsv, а дерево в формате Newick – в файле results/mitochondria/tree.tre (ниже приводится его визуализация).



Вспомогательная утилита gbk_converter

Утилита gbk_converter служит для преобразования генома из формата GenBank (.gbk, .gbff) в формат программы ChromoGGL.

Входные данные gbk_converter:

1) Список файлов: каждый файл в формате gbff содержит фрагмент генома определенного организма из Генбанка. Пример начальной части файла:

```
LOCUS       CP004015             3837060 bp    DNA     circular BCT 31-JAN-2014
DEFINITION  Rhizobium tropici CIAT 899, complete genome.
ACCESSION   CP004015
VERSION     CP004015.1
DBLINK      BioProject: PRJNA42391
            BioSample: SAMN02604105
KEYWORDS    .
SOURCE      Rhizobium tropici CIAT 899
  ORGANISM  Rhizobium tropici CIAT 899
            Bacteria; Proteobacteria; Alphaproteobacteria; Rhizobiales;
            Rhizobiaceae; Rhizobium/Agrobacterium group; Rhizobium.
REFERENCE   1  (bases 1 to 3837060)
  AUTHORS   Ormeno-Orrillo,E., Menna,P., Almeida,L.G., Ollero,F.J.,
            Nicolas,M.F., Pains Rodrigues,E., Shigueyoshi Nakatani,A., Silva
            Batista,J.S., Oliveira Chueire,L.M., Souza,R.C., Ribeiro
            Vasconcelos,A.T., Megias,M., Hungria,M. and Martinez-Romero,E.
```

```
  TITLE     Genomic basis of broad host range and environmental adaptability of
            Rhizobium tropici CIAT 899 and Rhizobium sp. PRF 81 which are used
            in inoculants for common bean (Phaseolus vulgaris L.)
  JOURNAL   BMC Genomics 13 (1), 735 (2012)
   PUBMED   23270491
  REMARK    Publication Status: Online-Only
REFERENCE   2  (bases 1 to 3837060)
  AUTHORS   Ormeno-Orrillo,E. and Martinez-Romero,E.
  TITLE     Direct Submission
  JOURNAL   Submitted (28-DEC-2012) Centro de Ciencias Genomicas, Universidad
            Nacional Autonoma de Mexico, Av. Universidad s/n, Col. Chamilpa,
            Cuernavaca, Morelos 62210, Mexico
COMMENT     Source DNA is available from Esperanza Martinez-Romero
            (emartine@ccg.unam.mx).
FEATURES             Location/Qualifiers
     source          1..3837060
                     /organism="Rhizobium tropici CIAT 899"
                     /mol_type="genomic DNA"
                     /strain="CIAT 899"
                     /isolation_source="Phaseolus vulgaris root nodule"
                     /db_xref="taxon:698761"
                     /country="Colombia"
                     /note="type strain of Rhizobium tropici"
     gene            427..1248
                     /locus_tag="RTCIAT899_CH00005"
     CDS             427..1248
                     /locus_tag="RTCIAT899_CH00005"
                     /codon_start=1
                     /transl_table=11
                     /product="putative phosphotransferase"
                     /protein_id="AGB69434.1"
                     /translation="MENRTSFFHLHLISDSTGETLISAGRAASVQFHASQPIEHVYPL
                     IRNRKQLLPVLEAIDHSPGIVLYTIVDRELADFIAERCREMGVPSVNVLEPVMNVFQT
                     YLGTASRRRVGAQHVMNADYFARIEALNFTMDHDDGQMPDDYDDADVVIIGISRTSKT
                     PTSIYLANRGIKTANIPIVHGVPLPESLARATKPLIVGLVATTDRISQVRENRILGTT
                     PGFDRGGYTDRAAISEELKYARSLCARHNWPIIDVTRRSIEETAAAIVALRPKLR"
     gene            1258..1881
.............................
```

2) файл masks.txt со списком типов генов, которые следует отобрать в структуру. Например:

psa
psb
rpl
rps
rpo

Командная строка: python gbk_converter.py -i [path to directory with gbff files] -m [path to file with masks] -o [path to file with output sequences]

Выходные данные gbk_converter: Файл result.txt с геномными структурами для каждого вида. Например:

Bradyrhizobium japonicum USDA 110 chromosome, complete genome; 1; C53: -rpoH3+rpsU+rplU+rpsP+rplS-rplT+rpoN2+rpsA-rpsO-rpsT+rpoN1-rplI-rpsR-rpsF-rpsB-rpsI-rplM+rpoH1-rplQ-rpoA-rpsK-rpsM-rplO-rpsE-rplR-rplF-rpsH-rpsN-rplE-rplX-rplN-rpsQ-rplP-rpsC-rplV-rpsS-rplB-rplW-rplD-rplC-rpsJ-rpsG-rpsL-rpoC-rpoB-rplL-rplJ-rplA-rplK+rpsD+rpsU+rpoH2-rplY;
Rhizobium tropici CIAT 899, complete genome; 1; C41: -rpsA-rpsO+rplT+rpsT-rpoNch-rplI-rpsR-rpsI-rplM+rplK+rplA+rplL+rpoB+rpoC+rpsL+rpsG+rpsJ+rplC+rplD+rplW+rplB+rplV+rpsC+rplP+rpsQ+rplN+rplX+rpsH+rplF+rplR+rpsE+rplO+rpsM+rpsK+rpoA+rplQ+rpsB-rpsD+rpoH1-rpoH2+rplU;

Рекомендуемые стандартные программы

Для просмотра файлов рекомендуется использовать стандартные программы.
Ссылки удобно открывать в новой вкладке, используя, например, браузер Google Chrome.
Microsoft Excel – для просмотра матриц. Они записываются в формате tsv (tab separated values). В качестве разделителя в числах используется точка. Для отображения матриц нужно в настройках задать соответствующий разделитель, который находится в меню Файл>Параметры>Дополнительно>Использовать системные разделители.
TreeViewX (https://code.google.com/p/treeviewx/) – для просмотра деревьев, которые используют формат Newick. Любой текстовый редактор для просмотра общих графов, кратчайших последовательностей, файлов с описанием исходных листовых хромосомных структур и оптимальной расстановкой структур по дереву.

Литература

[1] К.Ю. Горбунов, Р.А. Гершгорин, В.А. Любецкий. Перестройка и реконструкция хромосомных структур. *Молекулярная биология*, 2015, том. 49, №3, стр. 372–383.
[Англ. перевод: K.Yu. Gorbunov, R.A. Gershgorin, V.A. Lyubetsky. Rearrangement and Inference of Chromosome Structures. *Molecular Biology*, 2015, Vol. 49, No. 3, pp. 327–338.]

[2] V.A. Lyubetsky, R.A. Gershgorin, A.V. Seliverstov, K.Yu. Gorbunov. Algorithms for reconstruction of chromosomal structures // *BMC Bioinformatics*, 2016, Vol. 17, Article 40, 23 pp. DOI: 10.1186/s12859-016-0878-z.

[3] V.A. Lyubetsky, R.A. Gershgorin, K.Yu. Gorbunov. Chromosome structures: reduction of certain problems with unequal gene content and gene paralogs to integer linear programming // *BMC Bioinformatics*, 2017, Vol. 18, Article 537, 18 pp. DOI: 10.1186/s12859-017-1944-x.