Программа Distance-Circular-GGL
для сведения к ЦЛП задачи вычисления кратчайших расстояния и преобразования в случае кольцевых хромосом

Руководство пользователя

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

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

2019 год

Оглавление

Общие сведения

Программа **Distance-Circular-GGL** предназначена для автоматической конвертации задачи нахождения кратчайшего расстояния между циклическими геномными (синоним: хромосомными) структурами с паралогами в стандартный формат задачи целочисленного линейного программирования (ЦЛП). 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 включают паралоги генов, последовательность операций допускает переменный генный состав. Задача состоит в минимизации числа операций в последовательности, которая преобразует одну структуру в другую. Последовательность, на которой достигается минимум, называется *кратчайшей*. Число операций этой последовательности называется *кратчайшей длиной*. Задача вычисления кратчайшей длины является NP-трудной, поэтому мы предлагаем её сведение к задаче ЦЛП с линейным числом переменных и ограничений.

Описание модели

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.

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

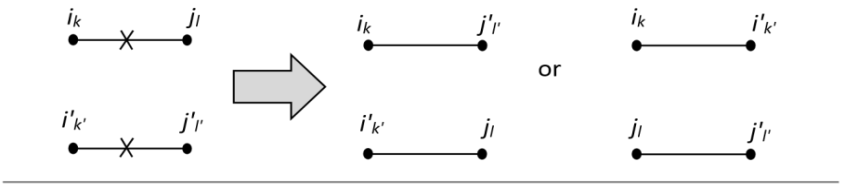
Определение общего графа и его финального вида. В общем графе *a+b* двух структур *a* и *b* вершины – края общих генов и, кроме того, все максимальные участки особых генов. Каждый край берётся один раз; более формально: вместо края пишется имя гена с индексом 1 или 2, указывающим на его начало или конец. 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

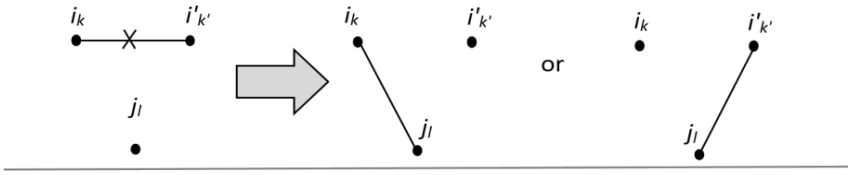
не более одного инцидентного *a*-ребра. (3) Удалить любое ребро. (4) Добавить ребро (скажем, с пометкой *a*) между вершинами, не инцидентными *a*-ребру. Если в результате операции получаются две инцидентные особые вершины, они сливаются в одну вершину, что входит в определение операции; получаемой вершине 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

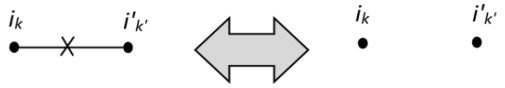
1) Double-cut-and-paste (DP). Исходные рёбра одноимённые.



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.

Сведение к ЦЛП с линейным числом переменных и ограничений

Пусть все структуры состоят только из кольцевых хромосом. Общий граф таких структур состоит только из циклов. Следующая теорема следует из [1].

Теорема 1. Длина кратчайшей последовательности от *a* к *b* равна $B+S_1-S_2$, где *B* – число особых вершин в графе *a*+*b*, S_1 – сумма целых частей половин длин максимальных по включению участков из обычных рёбер, S_2 – число циклов, состоящих из обычных рёбер.

Рассмотрим сведение к ЦЛП, позволяющее вычислить слагаемые *B*, S_1, S_2. Будем называть пару отождествлённых концов рёбер в хромосомной структуре склейкой.

Пусть даны две структуры *a* и *b*. Рассмотрим булеву переменную z_{kij}, равную 1, если паралог *i* гена *k* в структуре *a* соответствует паралогу *j* того же гена *k* в структуре *b*; иначе равную 0. Заметим, что значения переменных z_{kij} задают частичную биекцию паралогов в структурах *a* и *b*.

1) Вычисление *B*. Опишем каждую склейку *s* в структуре *a* булевой переменной x_{as}; она равна 1 если эта склейка расположена на границе *a*-блока, иначе она равна 0. Аналогично введём x_{bs} для структуры *b*. Ограничения таковы, что если край паралога i_1 гена *k* отождествляется с краем паралога i_2 гена *l* в *s*, то $x_{as} \ge \sum_j z_{ki_1j} - \sum_j z_{li_2j}$ и $x_{as} \ge \sum_j z_{li_2j} - \sum_j z_{ki_1j}$; аналогичные ограничения верны для склеек из *b*. Эти ограничения означают, что если $\sum_j z_{ki_1j}$ и $\sum_j z_{ki_2j}$ не равны, то есть края склейки *s* принадлежат общему и особому генам, то $x_{as}=1$. Запишем первую часть минимизируемой функции в виде $F = 0.5\cdot\sum_s \left(x_{as}+x_{bs}\right)+\ldots$, где остальные слагаемые будут определены позже. Если $\sum_j z_{ki_1j}$ и $\sum_j z_{ki_2j}$ равны, то $x_{as}=0$, так как x_{as} и x_{bs} – слагаемые в *F* с положительным коэффициентом.

Теперь вычислим количество петель, т.е. циклических блоков (будем далее называть их *особыми хромосомами*). Для этого введём для каждой хромосомы *h* в исходных структурах булеву переменную o_h, которая должна быть равна 1, если *h* особая, и 0 иначе. Накладываем ограничения: $o_h \ge 1-\sum_{k.i}\sum_j z_{kij}$, если *h* лежит в *a* или $o_h \ge 1-\sum_{k.j}\sum_i z_{kij}$, если *h* лежит в *b*, где первая сумма берётся по всем генам из *h*. Если *h* особая, то двойная сумма равна 0 и o_h=1, а иначе равенство o_h=0 следует из того, что переменные o_h входят в

минимизируемую функцию F с положительными коэффициентами: определяем её как $F = \ldots + \sum_h o_h + \ldots$, где другие слагаемые приведены выше и ниже.

Каждому нециклическому блоку соответствуют две граничные переменные x, а каждому циклическому – одна переменная o, что обеспечивает корректность вычисления величины B.

2) Вычисление s_1. Каждую склейку s в структуре a опишем булевой переменной y_{as}; она равна 0, если эта склейка расположена на границе или внутри блока. Аналогично введём переменную y_{bs}. Для склеек общих генов переменные y_{as} и y_{bs} принимают чередующиеся значения 0 и 1 внутри каждого участка, состоящего из обычных рёбер, это чередование начинается с 0 на одной из границ участка. Опишем ограничения. Две склейки будем называть *потенциально соседними* (как рёбра в общем графе), если они принадлежат различным структурам и обе содержат один и тот же край паралога одного гена. Следующие ограничения накладываются на пару s_1 (из a) и s_2 (из b) потенциально соседних склеек: $y_{as_1} \leqslant 4 - z_{kij} - \sum_j z_{k_1 i_1 j} - \sum_j z_{k_2 j i_2} - y_{bs_2}$ и $y_{bs_2} \geqslant z_{kij} + \sum_j z_{k_1 i_1 j} + \sum_j z_{k_2 j i_2} - 2 - y_{as_1}$, где край паралога i гена k и паралога i_1 гена k_1 являются отождествлёнными в s_1, а также края паралога j гена k и паралога i_2 гена k_2 в s_2. Эти неравенства означают, что значения y_{as} и y_{bs} чередуются на каждом участке обычных рёбер. Продолжим определять минимизируемый функционал: $F = \ldots + \sum_s (y_{as} + y_{bs}) + \ldots$, где остальные слагаемые будут определены ниже. На границах участков, состоящих из нечётного числа обычных рёбер, а также внутри или на границе блоков, переменные y_{as} и y_{bs}. равны 0, так как они входят в F с положительным коэффициентом. Таким образом, сумма переменных y_{as} и y_{bs} равна s_1.

3) Вычисление s_2. Воспользуемся идеей о подсчёте числа циклов из [2]. Каждую склейку s в структурах a и b опишем целочисленной переменной u_s, ограниченной неравенством $u_s \leqslant m_s$, где m_s принимает значение от 1 до общего числа склеек в a и b. Введём также булеву переменную p_s, ограниченную неравенством $p_s m_s \leqslant u_s$. Данная переменная указывает на то, достигает ли u_s максимально возможное значение m_s. Продолжим определение: $F = \ldots - \sum_s p_s$, где остальные слагаемые были определены ранее. Переменные p_s входят в F с отрицательными коэффициентами, поэтому, если u_s равно m_s, то $p_s = 1$.

Для каждой склейки s, содержащей паралог i гена k в a введём следующее ограничение: $u_s \leqslant m_s \sum_j z_{kij}$. Для b введём аналогичные неравенства. Они обеспечивают равенство $u_s = 0$, если s лежит на границе или внутри блока.

Для двух потенциально соседних склеек s_1 из a и s_2 из b введём следующие ограничения: $u_{s_1} \leqslant u_{s_2} + m_{s_1}(1 - z_{kij})$ и $u_{s_2} \leqslant u_{s_1} + m_{s_2}(1 - z_{kij})$, где s_1 содержит паралог i гена k и s_2 содержит паралог j гена k. Эти неравенства обеспечивают равенство $u_{s_1} = u_{s_2}$ для двух соседних рёбер s_1 и s_2 в общем графе. Соответственно, все переменные u_s принимают одно значение, причем ровно одна из них достигает максимального значения на каждом цикле, состоящем из обычных рёбер. Для циклов, содержащих блоки, эти переменные равны 0 и ни одна из них не достигает своего максимума. Таким образом, количество переменных u_s, достигающих своего максимума (и равное сумме переменных p_s) равно s_2.

Входные данные программы Distance-Circular-GGL

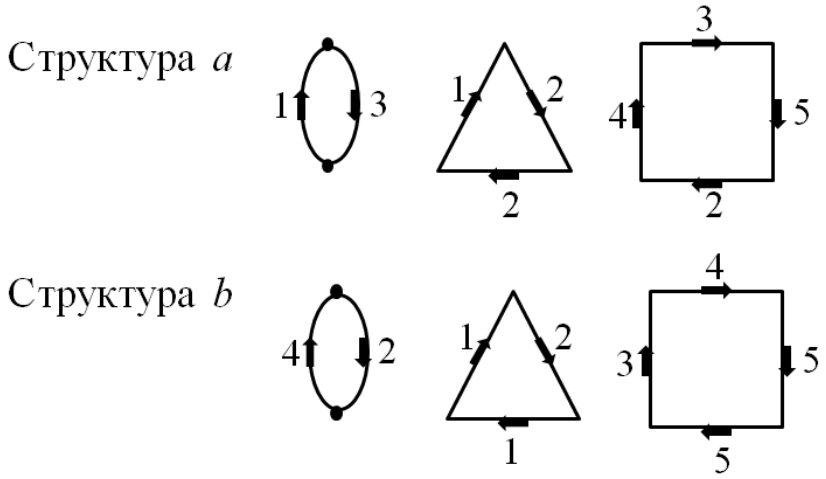


Рис. 1. Пример двух входных структур.

На вход программы подается **файл структур**, содержащий следующие данные.

I. Количество N различных имён генов, присутствующих в структурах, и далее N строк, каждая из которых содержит по два числа, первое – номер гена, второе – максимальное число паралогов для гена с данным номером. Например, для рис. 1:

```
5
1 2
2 3
3 2
4 2
5 2
```

II. Количество структур, которое в задаче преобразования равно 2, далее две строки, каждая из которых содержит описание структуры в следующем формате: имя вида/штамма; индекс соответствующего этой структуре листа в дереве; 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 кольцевая; число генов в хромосоме; последовательность генов в хромосоме, записанная как имена генов со знаками «+» или «–», которые указывают на транскрибируемую цепь. Для рис. 1 описание имеет следующий вид:

```
2
Struct_a; 0; 3; C2: +1.1+3.1; C3: +1.2+2.1+2.2; C4: +3.2+5.1+2.3+4.1;
Struct_b; 1; 3; C2: +4.1+2.1; C3: +1.1+2.2+1.2; C4: +4.2+5.1+5.2+3.1;
```

Пример входного файла содержится в составе дистрибутива программы (**data/input_circular.txt**).

Входные параметры программы и её работа

Программа запускается с двумя параметрами командной строки:
-i [имя файла с входными данными] -o [имя файла, содержащего задачу ЦЛП]
Если расширение имени выходного файла не равно .lp, программа автоматически добавит его. Данное расширение используется утилитой cplex.

В процессе работы программы на консоль выводится текущий этап, например, считывание дерева, структур, или вычисление ограничений для переменных.

Выходные данные программы Distance-Circular-GGL

В результате работы программа записывает в выходной файл соответствующую задачу целочисленного линейного программирования в формате LP [3]:

```
Minimize
        + 2 o_a0 + 2 o_a1 + 2 o_a2 + 2 o_b0 + 2 o_b1 + 2 o_b2 - 2 p_a&1.1h3.1t - 2 p_a&1.1t3.1h - 2
p_a&1.2h2.1t - 2 p_a&1.2t2.2h - 2 p_a&2.1h2.2t - 2 p_a&2.3h4.1t - 2 p_a&2.3t5.1h - 2 p_a&3.2h5.1t - 2
p_a&3.2t4.1h - 2 p_b&1.1h2.2t - 2 p_b&1.1t1.2h - 2 p_b&1.2t2.2h - 2 p_b&2.1h4.1t - 2 p_b&2.1t4.1h - 2
p_b&3.1h4.2t - 2 p_b&3.1t5.2h - 2 p_b&4.2h5.1t - 2 p_b&5.1h5.2t + x_a&1.1h3.1t + x_a&1.1t3.1h + ...

Subject To
        z_1.1.1 + z_1.1.2 <= 1
        z_1.2.1 + z_1.2.2 <= 1
        z_2.1.1 + z_2.1.2 <= 1
...
```

```
Bounds
        u_a&1.1h3.1t <= 1
        u_a&1.1t3.1h <= 2
        u_a&1.2h2.1t <= 3
…

Binary
        o_a0
        o_a1
        o_a2
…

General
        u_a&1.1h3.1t
        u_a&1.1t3.1h
        u_a&1.2h2.1t
…
End
```

Полностью файл приведен в контрольном примере в составе дистрибутива программы (**results/output_circular.lp**). Подробное описание всех переменных, участвующих в записи задачи ЦЛП можно найти в [4].

Решение задачи ЦЛП утилитой cplex

Для решения задачи ЦЛП применялись как облачные вычисления, так и вычисления с помощью утилиты IBM на локальном сервере. Для вычисления в облаке IBM необходимо воспользоваться ссылкой [5]. Для вычисления на локальной машине необходимо скачать IBM ILOG CPLEX Optimization Studio [6] и воспользоваться утилитой **cplex.exe**. Утилита запускается в командной строке и далее поддерживает набор команд. Для загрузки файла (например, output_circular.lp) с задачей ЦЛП формата LP необходимо ввести команду

```
> read output_circular.lp
```

Далее необходимо запустить оптимизатор командой

```
> optimize
```

И, наконец, после окончания вычислений, необходимо записать найденное решение в файл командой

```
> write ilp_solution_circular.sol
```

В результате запишется XML-файл ilp_solution_circular.sol, в котором будет описано найденное решение (оптимальное значение функционала, соответствующие значения всех переменных).

```
<?xml version = "1.0" encoding="UTF-8" standalone="yes"?>
<CPLEXSolution version="1.2">
 <header
  problemName="output_circular.lp"
  ...
  objectiveValue="7.9999999999999991"
  ...
<quality
  epInt="1.0000000000000001e-05"
  epRHS="9.9999999999999995e-07"
  maxIntInfeas="3.5527136788005009e-15"
  maxPrimalInfeas="3.5527136788005009e-15"
  maxX="8"
  maxSlack="25.999999999999996"/>
 ...
<variables>
 <variable name="o_a0" index="0" value="-0"/>
 ...
 <variable name="p_a&amp;amp;1.1h3.1t" index="6" value="-0"/>
 ...
 <variable name="x_a&amp;amp;1.1h3.1t" index="24" value="1"/>
 ...
 <variable name="y_a&amp;amp;1.1h.3.1t" index="42" value="0"/>
 ...
 <variable name="z_1.1.1" index="60" value="1"/>
 ...
 <variable name="u_a&amp;amp;1.1h3.1t" index="76" value="-0"/>
 ...
 </variables>
</CPLEXSolution>
```

Полностью файл решения для рассматриваемого примера приведен в составе дистрибутива программы distance-to-structs-ggl (**data/ilp_solution_circular.sol**).

Утилита cplex.exe доступна в бесплатном варианте, однако она имеет внутренние ограничения на размер задачи. Для получения бесплатной версии без ограничений необходимо запросить академическую лицензию по следующей ссылке.

Преобразование решения в структуры

Для того, чтобы восстановить из решения ЦЛП структуры с оптимальным расстоянием, используется утилита **distance-to-structs-ggl.exe**. Утилита имеет три параметра запуска: -i [имя файла с решением задачи ЦЛП, выданным CPLEX], -s [имя файла с исходными структурами, формат такой же, как в разделе **Входные параметры программы**], -o [имя файла, в который будут записаны структуры с оптимальной расстановкой паралогов, формат в точности соответствует формату входного файла для ChromoGGL]. Для рассматриваемого примера файл структур приведён в составе дистрибутива программы distance-to-structs-ggl (**results/resolved_structures_circular.txt**).

Литература

[1] Горбунов К.Ю., Любецкий В.А. Линейный алгоритм минимальной перестройки структур // Пробл. передачи информации. 2017. Т. 53. Вып. 1. С. 60–78.

[2] Shao M., Lin Y., Moret B. An Exact Algorithm to Compute the DCJ Distance for Genomes with Duplicate Genes. In: Sharan R. (eds) // Research in Computational Molecular Biology, LNCS. 2014. Vol. 8394. P. 280–292. DOI: 10.1007/978-3-319-05269-4_22.

[3] https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.0/ilog.odms.cplex.help/CPLEX/FileFormats/topics/LP.html - описание формата LP задач линейного программирования, поддерживаемого IBM CPLEX Optimizer

[4] Lyubetsky V.A., Gershgorin R.A., Gorbunov K.Yu. Chromosome structures: reduction of certain problems with unequal gene content and gene paralogs to Integer linear programming // BMC Bioinformatics. 2017, Vol. 18, № 537, 18 pages

[5] https://www.ibm.com/us-en/marketplace/decision-optimization-cloud - облачный оптимизатор IBM для решения задач ЦЛП.

[6] https://www.ibm.com/analytics/cplex-optimizer - IBM ILOG CPLEX Optimization Studio, локальная утилита, позволяющая оптимизировать задачи ЦЛП.