Информационные процессы, Том 1, № 2, 2001, стр. 178–187.

О двух алгоритмах поиска альтернативной вторичной структуры РНК

К.Ю.Горбунов, Е.В.Любецкая, В.А.Любецкий

Институт проблем передачи информации РАН,
101447, Москва, Большой Каретный переулок, 19, Россия
E-mail Lyubetsk@iitp.ru, тел. 4916780, 4134643, факс (7095) 2090579

Поступила в редколлегию 22.08.2001

Аннотация—Описаны два алгоритма для быстрого поиска потенциальных вторичных структур в геномах организмов.

§1. В этом параграфе опишем используемую нами в соответствующих биологических расчетах программу примерно квадратичного по времени работы (от объема исходных данных) алгоритма поиска потенциальных вторичных структур в данном фрагменте РНК.

Напомним, что задача состоит в поиске участков генома, потенциально годных для образования альтернативных вторичных структур в РНК. Регуляция многих бактериальных генов осуществляется на уровне трансляции или взаимодействия процессов трансляции и транскрипции. При этом основным регуляторным сигналом является образование вторичной структуры РНК. Часто (хотя и не всегда) в рассматриваемой области могут образовываться две альтернативные структуры РНК, что и служит регуляторным переключателем. Эта программа написана на языке Delphi 5 и реализует алгоритм из [1]. Нуклеотидом называется буква в четырехбуквенном алфавите a,c,g,t. Комплементарными считаются нуклеотиды a и t, g и c (а также, в одной из версий программы g и t, расположенные не на концах отрезков). Отрезком называется последовательность из нескольких попарно комплементарных нуклеотидов. Шпилькой называется совокупность комплементарных отрезков $[a_1, \ldots, b_1], \ldots, [a_n, \ldots, b_n]$ и $[c_n, \ldots, d_n], \ldots, [c_1, \ldots, d_1]$, где отрезок $[a_i, \ldots, b_i]$ комплементарен отрезку $[c_i, \ldots, d_i]$. Отрезок $[b_n, \ldots, c_n]$ называется петлей, a_1 называется началом шпильки и обозначается A; b_n и c_n называются соответственно началом и концом петли и обозначаются B и C; d_1 называется концом шпильки и обозначается D.

Перекрытием называется пара шпилек, удовлетворяющих условиям:
а) Одна из них называется первой шпилькой и имеет параметры A', B', C', D', другая — второй и имеет параметры A, B, C, D. Тогда $[C'D'] \cap [AB]$ называется базой пересечения перекрытия и оно должно быть не короче константы “минимальная длина пересечения” (обычно равная 5).
б) $A > B'$ и $C > D'$. Выпячивание — это промежуток между двумя отрезками. Хотя формально мы будем говорить далее о перекрытиях, предлагаемый алгоритм находит и другие “естественные” конфигурации шпилек.

Алгоритм позволяет варьировать следующие параметры: минимальную длину отрезка (обычно 3); максимальную длину выпячивания (обычно 20); максимальную длину шпильки (обычно 100); минимальную длину пересечения (обычно 5); максимальную и минимальную длины петли (обычно 20 и 3).

В настоящее время на основе этой программы реализованы или находятся в процессе реализации следующие ее версии:
— разрешающая комплементарность нуклеотидов g-t при условии, что они не находятся в начале или в конце отрезка;

— варьирующая веса пар нуклеотидов, в том числе отдельно в разных шпильках, составляющих перекрытие. Таким образом можно искать шпильки, в которых соотношение различных пар нуклеотидов наперед задано;
— отыскивающая перекрытия с мощным непрерывным отрезком в базе пересечения. Необходимым условием таких перекрытий является наличие отрезка, входящего в первую и вторую шпильки;
— позволяющая определить штраф (т.е. уменьшение мощности шпильки) за наличие выпячивания или за длину выпячивания;
— позволяющая рассматривать большее количество шпилек, с целью отбора их по критерию биологической значимости.

Общая идея алгоритма состоит в следующем: глобальная переменная X пробегает последовательность, и для каждого X находятся основные параметры канонического перекрытия (включая мощность). Затем, с помощью этих параметров выбирается и находится заданное количество перекрытий. Заметим, что при таком подходе максимальное число перекрытий, которые можно найти, равно длине последовательности. При фиксированном X каноническое перекрытие ищем в промежутке [начало, . . . , конец], рис. 1.

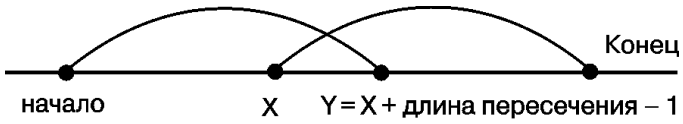

Рис. 1

Программа состоит из трех этапов (рис. 2):

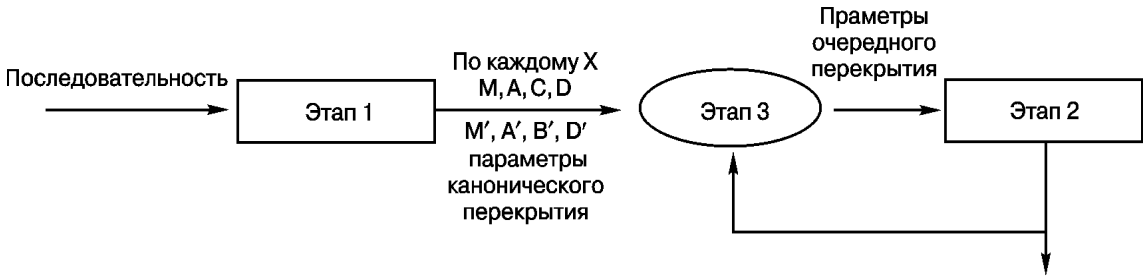

Рис. 2

1. Получает на вход последовательность и выдает по каждому X некоторые параметры канонического перекрытия;
2. Получает на вход параметры нужного перекрытия и выдает само перекрытие;
3. Получает на вход параметры канонических перекрытий по всем X и, вызывая второй этап в качестве процедуры, выделяет лучшие перекрытия.

Схема этапа 1 при фиксированном X приведена на рис. 3.

Максшпилька2 содержит максимальные качества вторых шпилек, а максшпилька1 — первых шпилек. Второй называется шпилька, у которой $B \geq Y$ и $A \leq X$. Первой называется шпилька, у которой $C' \leq X$ и $D' \geq Y$. Очевидно, что в этом случае, если мы составим перекрытие из любых первой и второй шпилек, то его база пересечения будет больше или равна минимальной ДЛИНЫ ПЕРЕСЕЧЕНИЯ, рис. 4.

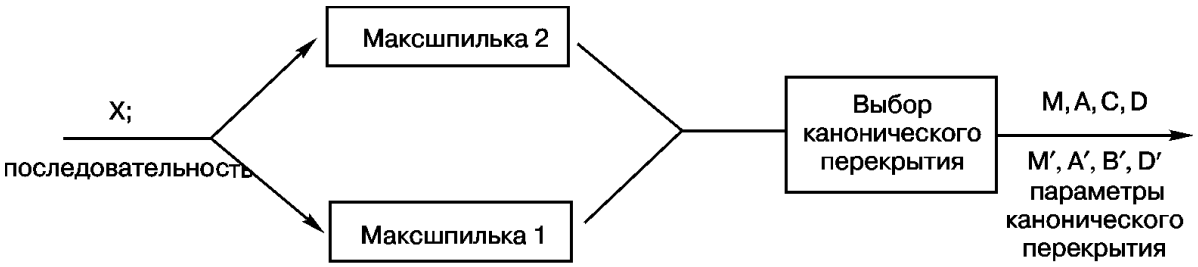

Рис. 3

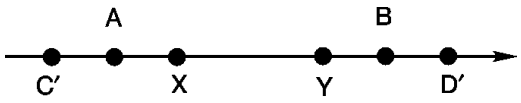

Рис. 4

Массив максшпилька2 организован следующим образом: A является индексом массива; столбец максшпилька2[A] содержит все максимальные несравнимые качества вторых шпилек с началом в A, т.е. имеет вид, показанный на рис. 5.

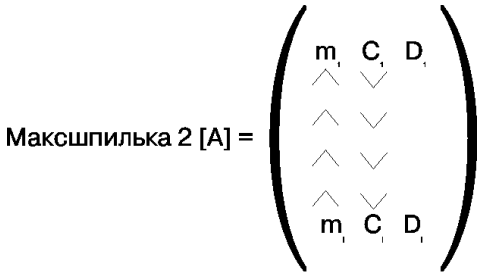

Рис. 5

Массив масшпилька1 организован аналогичным образом.

Для нахождения массива максшпилька2 необходимо определить промежуточные массивы: продолж (индексируется переменной A); продолж$[A]$ хранит максимальные качества шпилек с началом в A и концом, лежащем в отрезке $[K-$максвыпячивание,$\ldots,K]$. Очевидно, что именно такие шпильки могут быть продолжены новыми отрезками при увеличении K, рис. 6.

Продолж$[A]$ организован следующим образом: это очередь длины МАКСВЫПЯЧИВАНИЕ, где i-тый элемент хранит максимальные качества (пары (m, C)) шпилек с началом в A и концом в $K+$МАКСВЫПЯЧИВАНИЕ$-i$. Предшпилька (индексируется по A); предшпилька$[A]$ хранит максимальные качества предшпилек (пары (m, C)) с началом в A и концом в K. Предшпилькой кратности r называется шпилька, у которой первый отрезок (начинающийся в A) содержит r нуклеотидов. Предшпильки бывают кратности $r = 1,\ldots,$МИНПОДРЯД-1. Очевидно, что шпильки образуются из предшпилек. Предшпилька$[A]$ организована следующим образом: это антиочередь длины минподряд-1, где i-ый элемент хранит максимальные качества предшпилек кратности i. Массив максшпилька2 заполняется индукцией по K. На каждом этапе он хранит максимальные качества вторых шпилек, лежащих в отрезке [Начало,K]. На последнем шаге индукции, при $K =$Конец, получим искомый массив. Начальное значение индукции (при $K = Y$) находится очевидно. Опишем индуктивный переход $K \to K + 1$. То есть, необходимо перезаполнить массивы продолж, максшпилька2 и предшпилька при каждом A. При этом возможны два случая.

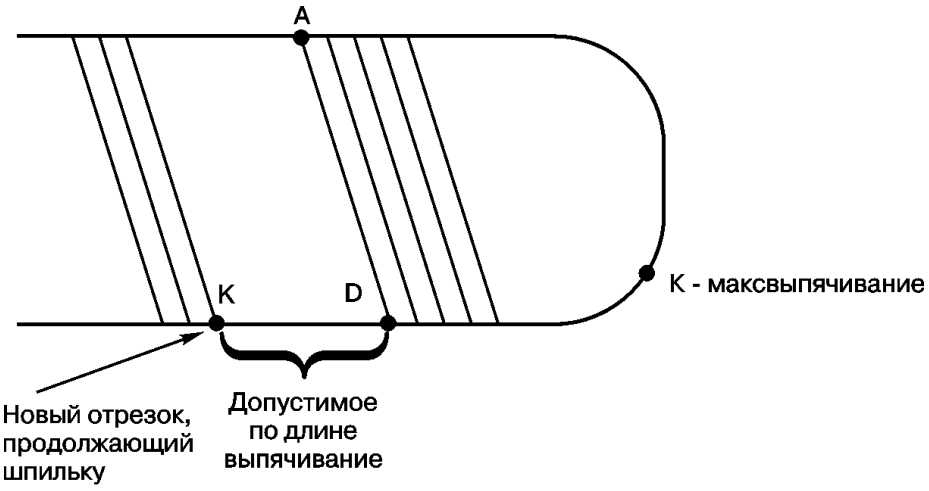

Рис. 6

Случай 1: A не комплементарно $K+1$. Тогда, в массиве продолж$[A]$ удаляется первый элемент (т.к. шпильки с концом в $K-$максвыпячивание не могут быть продолжены) и последний элемент равен пустому множеству (т.к. шпилек с концом в $K+1$ не существует). Массив максшпилька2$[A]$ не изменяется, т.к. новых шпилек не образовалось. Массив предшпилька$[A]$ заполняется пустыми множествами, т.к. предшпилек с началом в A и концом в $K+1$ не существует.

Случай 2. A комплементарно $K+1$. Тогда, в массиве продолж$[A]$ удаляется первый элемент, а последний заполняется следующим образом: сливаются последний элемент массива продолж$[A+1]$ и последний элемент предшпилька$[A+1]$ и в результате слияния мощности увеличиваются на единицу (т.е. учитываем пару $(A, K+1)$). Получившееся множество назовем set. Если A комплементарно $K+1$, то у нас "появляется шанс" получить новый отрезок. Как он может образоваться? Предшпилька$[A+1]$ (еще с концом в K) может "дорасти" до отрезка, значит, берем предшпильки наибольшей кратности МИНПОДРЯД-1; они хранятся в последнем элементе массива предшпилька$[A+1]$, рис. 7.

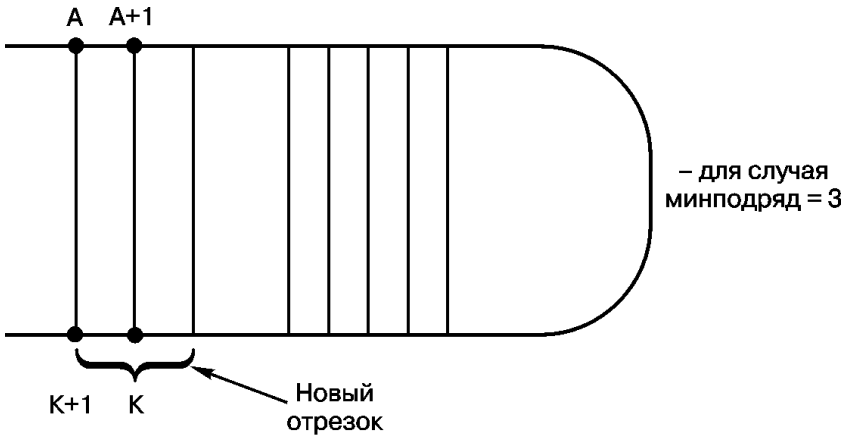

Рис. 7

Либо может продолжится отрезок с началом в A и концом в $K+1$. Качества шпилек с такими отрезками хранятся в последнем элементе массива продолж$[A+1]$, рис. 8.

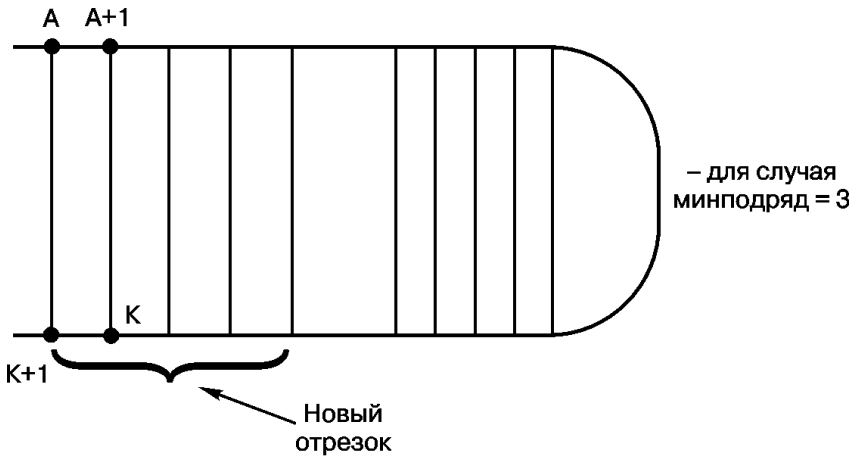

Рис. 8

Операция слияния позволяет выбрать лучшие качества из обоих вышеописанных случаев. В массив максшпилька2$[A]$ вливается множество set. Заметим, что на каждом шаге мы вливаем в максшпильку2$[A]$ только качества шпилек с концом в K. Но так как операция слияния позволяет оставить "лучшие" качества, то очевидно, что качества лучших шпилек останутся в окончательном массиве максшпилька2$[A]$. Массив предшпилька$[A]$ перезаполняется следующим образом: в предшпильки кратности $r > 1$ кладутся соответствующие столбцы ($r-1$-ые) массива предшпилька$[A+1]$ (еще с концом в K). Заполним, например, второй столбец массива предшпилька$[A]$, то есть максимальные качества предшпилек кратности 2. Очевидно, такие предшпильки получаются из предшпилек с началом в $A+1$, концом в K и кратности 1 (рис. 9).

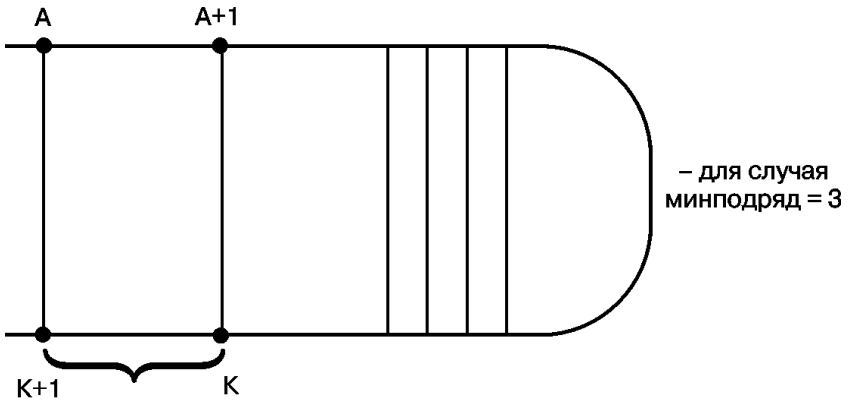

Рис. 9

Значит, берем первый столбец массива предшпилька$[A+1]$, увеличиваем в нем мощности на 1 и кладем его во второй столбец массива предшпилька$[A]$.

Первый столбец массива предшпилька$[A]$, то есть предшпильки кратности 1, заполняется следующим образом: необходимо к паре $(A, K+1)$ подобрать наилучшую по мощности шпильку. Для этого сливаем элементы массивов продолж$[A+1]$, ..., продолж$[A+$МАКСВЫПЯЧИВАНИЕ$+1]$ и увеличиваем мощности полученного множества на 1, чтобы учесть пару $(A, K+1)$, рис. 10.

Очевидно, что качества шпилек, которые может продолжить пара $(A, K+1)$ имеют начало в отрезке $[A+1, \ldots, A+$МАКСВЫПЯЧИВАНИЕ$+1]$ и конец в отрезке $[K-$МАКСВЫПЯЧИВАНИЕ$, \ldots, K]$.

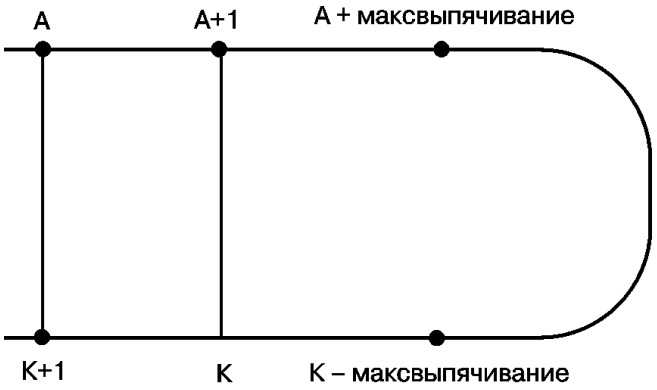

Рис. 10

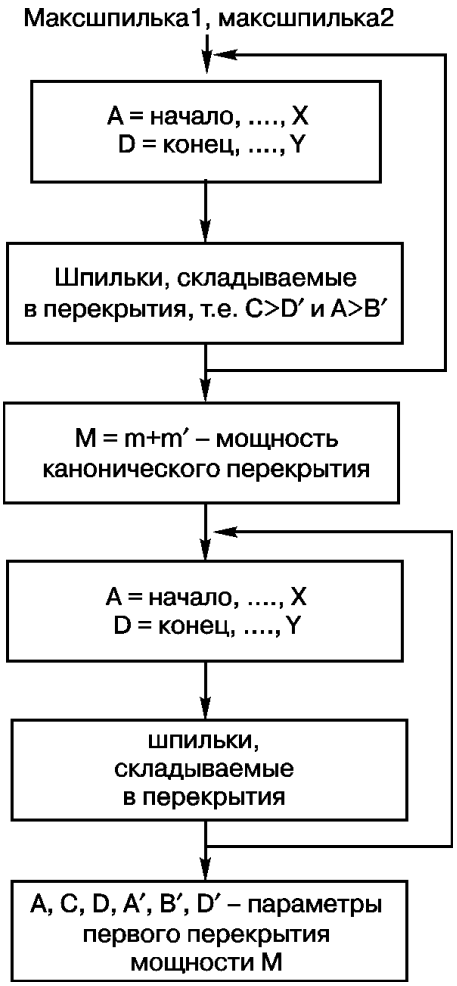

Рис. 11

Качества таких шпилек лежат в массивах продолж[A+1], ..., продолж[A+МАКСВЫПЯЧИВАНИЕ+1]. Заполнение массива максшпилька1. Очевидно, что массив максшпилька1 заполняется симметричным образом к массиву максшпилька2. Чтобы его заполнить, меняем направление последовательности, то есть бывший последний элемент становится первым и так далее. Далее, используя несложные функции отображения симметричности, находим на этой “перевернутой” последовательности мас-

сив максшпилька2 и инвертируем его элементы. Алгоритм выбора канонического перекрытия при фиксированном X описан на рис. 11:

Общая схема этапа 3 приведена на рис. 12.

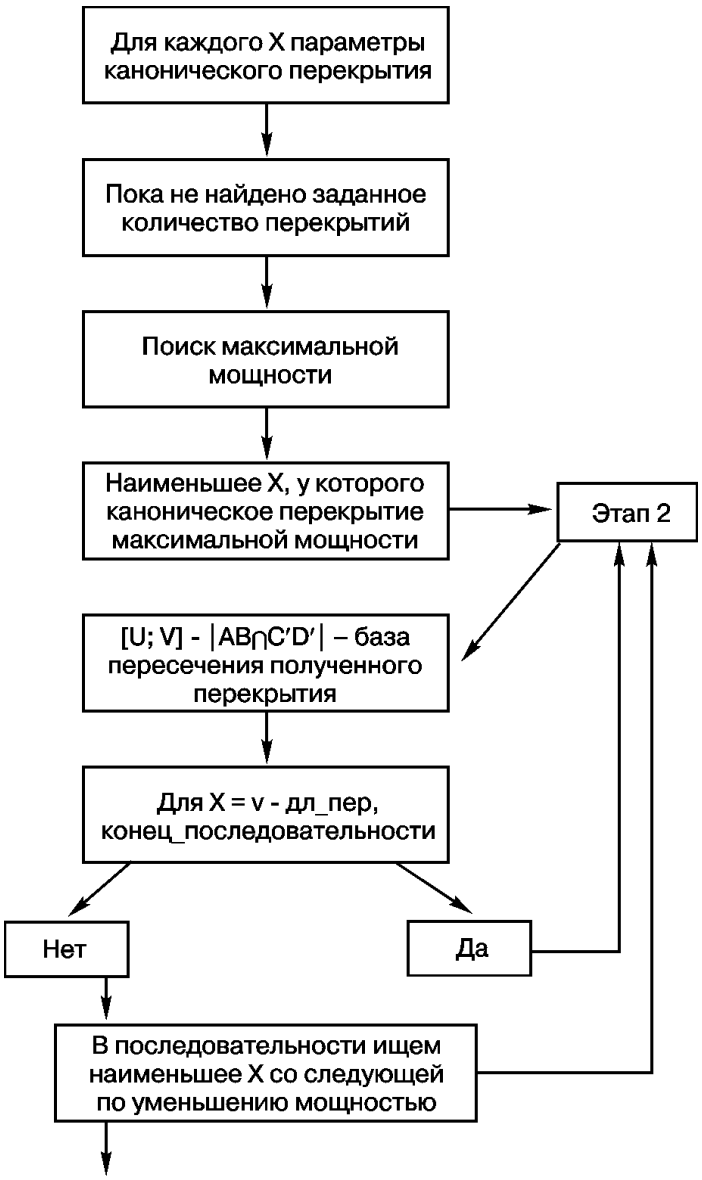

Рис. 12

Отметим, что при разных X алгоритм может выдавать параметры одинаковых перекрытий. Рассмотрим максимальное перекрытие в некоторой последовательности. Пусть его базой пересечения является отрезок $[A = 30 \ldots C' = 40]$. Тогда для $X = 30$ это перекрытие будет каноническим; но и для $X = 31, \ldots, 36$ оно также будет каноническим, так как мы рассматриваем максимальное перекрытие, и во всех случаях отрезок $[X \ldots Y]$ лежит в базе пересечения. Этап 3 отсеивает одинаковые перекрытия (рис. 12) Общая схема этапа 2 приведена на рис. 13.

Здесь двумерный массив полн индексируется переменными B и C; полн$[C, B]$ хранит максимальную мощность шпильки с концами A, B, C, D. Двумерный массив неполн также индексируется переменными B, C; неполн$[C, B]$ является антиочередью длины МИНПОДРЯД-1, i-ый элемент

Рис. 13

которой хранит максимальную мощность внутренней предшпильки кратности i. Внутренней предшпилькой кратности i ($i = 1, \ldots$,минподряд-1) называется шпилька, у которой отрезок с концами B, C имеет длину i и $C - B + 1$ может быть больше константы макспетля. Рассмотрим нахождение второй шпильки. Сначала находим такое минимальное B, что мощность шпильки с концами A, B, C, D равна m. B ищется индукцией; начальное значение индукции — $B = A$, полн и неполн заполняются очевидно. Опишем индуктивный переход $B \to B + 1$. Пусть c не комплементарно $B + 1$. Тогда полн$[c, B+1]$ равен минус бесконечности (так как шпилек с такими концами не существует), и аналогично неполн равен антиочереди из минус бесконечностей. Если c комплементарно $B + 1$, то массив полн$[c, B + 1]$ заполняется максимумом из poln$[c + 1, B]$ и последнего элемента неполн$[c + 1, B]$. Массив неполн перезаполняется следующим образом: в i-ый столбец, где $i = 2, \ldots$,минподряд-1, нужно положить элементы $i - 1$ столбца массива неполн$[c + 1, B]$, увеличенные на единицу. Осталось перезаполнить первый элемент массива неполн$[c, B + 1]$. В него кладем максимум элементов полн$[u, v]$, где $u = c, \ldots, c+$максвыпячивание и $v = B, \ldots, B-$максвыпячивание, увеличенный на единицу.

Теперь по A, B, C, D и массивам полн и неполн восстановим шпильку (например, вторую; первая будет заполняться аналогично).

Встанем в точки B, C — начальное положение и проверим, что полн$[B, C] = M$. Тогда запомним пару (B, C) — она входит в нашу шпильку. Увеличим C на 1, и B уменьшим на 1, т.е. встанем в позицию $(B - 1, C + 1)$ (рис.14), и будем искать полн$[B - 1, C + 1] = M - 1$. Пара $(B - 1, C + 1)$ также входит в нашу шпильку. Таким образом восстановим первый отрезок (сначала по массиву полн, а последние нуклеотиды отрезка по массиву неполн). Затем находим начало нового отрезка. Пусть восстановили m нуклеотидов шпильки и остановились в позиции (B', C'). Тогда определим цикл, который от B', уменьшаясь, и от C', увеличиваясь, ищем такие B'', C'', что полн$[B'', C''] = M - m'$. Эти B'', C'' будут началом нового отрезка. Проведен счет, который дал биологически содержательные, а в ряде случаев новые результаты, для следующих случаев: B. subtilis pyr operon, Attenuators of Escherichia coli, Salmonella tyrhi, Yersinia pestis, Vibrio cholerae, Haemophilus influenzae (trpEGDC), Haemophilus influenzae (trpBA); Attenuators of pheA operon тех же организмов и Actinobacillus actinomycetemcomitans, Xanihomonas campestris и ряда других.

§2. Алгоритм из §1 не всегда годится, потому что вторичная структура далеко не всегда характеризуется максимальной мощностью. Поэтому полезен алгоритм, который находит консервативную альтернативную вторичную структуру на основе ее похожести в семействе фрагментов РНК родственных организмов. Здесь нами применялся следующий алгоритм (в этой публикации опишем его без деталей).

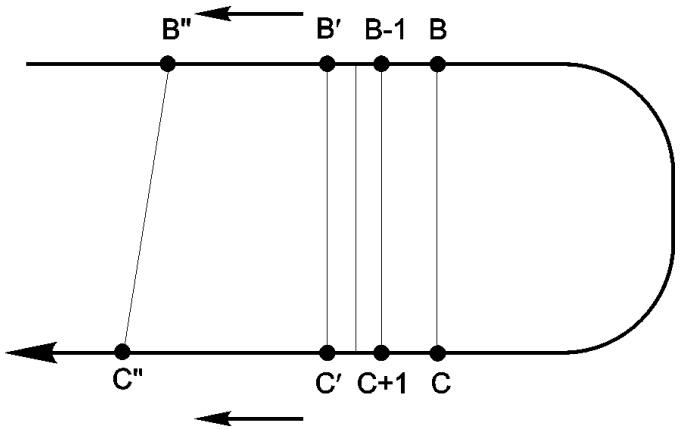

Рис. 14

Как уже упоминалось, каждая вторичная структура состоит из нескольких так называемых шпилек, т.е. двух упорядоченных наборов отрезков (двух оснований: левого и правого), где каждый i-ый отрезок левого основания комплементарен i-му отрезку правого основания. При этом направления обхода от первого к последнему отрезку в этих наборах противоположны: в первом наборе — слева направо, во втором — справа налево, и соседние отрезки расположены недалеко друг от друга. Шпильку, имеющую по одному отрезку в каждом основании, будем называть спиралью. Таким образом, элементарными частями вторичной структуры можно считать спирали. Отметим, что искомая структура практически всегда содержит достаточно длинные спирали.

Предлагаемый алгоритм вначале строит в каждом из данных фрагментов РНК большой запас спиралей, начиная с самых длинных. Это делается методом динамического программирования, где для каждых двух позиций A и $K > A$ ищется самая длинная спираль с началом в A и концом в K (найти ее легко, если известна аналогичная спираль с началом в $A - 1$ и концом в $K + 1$). Отметим, что сходная программа применялась в §1 для поиска в данной РНК наиболее мощного перекрытия.

Далее, объединяем в шпильки те спирали, которые друг относительно друга расположены соответствующим образом. С большей затратой времени такой запас шпилек можно получить непосредственным применением программы из §1 к участкам $[A, K]$ данного фрагмента РНК. Итак, получаем n списков шпилек: $L_1, \ldots, L_n$. В каждом из них упорядочиваем шпильки по возрастанию номера позиции начала левого основания. Теперь проредим эти списки. Это делается в два этапа. На первом прореживание происходит на основе условия о сходности порядка расположения шпилек в данном наборе фрагментов РНК, а на втором — на основе условия о похожести оставшихся шпилек как слов.

На *первом этапе* прореживания для каждой пары списков L_i, L_j делаем следующее. Каждый список рассматриваем как слово, состоящее из шпилек как отдельных букв. Исходим из того, что паре похожих вторичных структур соответствует пара таких подслов (из первого и второго списков слов), что после выбрасывания из них некоторого числа лишних шпилек получаются два слова одинаковой длины, в которых соответствующие шпильки имеют большую степень сходства. Поэтому, естественно применить к паре исходных слов L_i и L_j алгоритм Смита-Уотермена, находящий в двух данных последовательностях пару подслов, наиболее близких в смысле сходства двух слов, полученных из соответствующих подслов некоторым выравниванием. Напомним, что выравниванием двух слов называется процесс вставок в них пробелов (делеций), приводящий к двум словам одинаковой длины, которые уже сравниваются побуквенно.

Вспомним алгоритм Смита-Уотермена. Обозначим через S меру близости двух (невыровненных) слов как максимальное сходство всех выравниваний этих слов, а через s некоторую меру сходства на парах букв, включая пробел. Пусть $H_{ij} \geq 0$ максимально возможное $S(u, v)$ по всем таким подсловам u и v, что u оканчивается в первом слове в позиции i, а v оканчивается во втором слове в позиции

j (если этот максимум меньше нуля, полагаем $H_{ij} = 0$). Если для всех i и j положить $H_{i0} = H_{j0} = 0$, то H_{ij} подсчитывается индуктивно по формуле $H_{ij} = \max(0, H_{i-1,j-1} + s(x,y), H_{i-1,j} + s(x, probel), H_{i,j-1} + s(probel, y))$, где x — i-ая буква первого слова, а y — j-ая буква второго слова. Таким образом, функция H может быть легко вычислена. Отсюда найдем максимальное значение функции H_{ij} по всем i, j и пару подслов u, v, на которых этот максимум достигается. Известна и модификация этого алгоритма, позволяющая находить не одну, а несколько лучших пар подслов.

Так как в качестве букв у нас рассматриваются шпильки, то в нашем алгоритме вместо функции s находится процедура, вычисляющая степень сходства двух шпилек. Чаще всего сходство шпилек проявляется на спиралях, так как выпячивания обычно неконсервативны. Поэтому соединим составляющие шпильку спирали в одно слово и будем измерять сходство двух шпилек как максимальное сходство (в смысле выравнивания соответствующих слов), что также делается алгоритмом Смита-Уотермена. Для этого следует разумно задать степень сходства между буквами: мы не хотим, чтобы пара длинных шпилек имела слишком большое преимущество над парой коротких шпилек за счет лишь своей длины. Поэтому сходство между различными буквами измеряем отрицательным числом (например, мы брали $s(a,a) = 1$, $s(a,b) = -0.9$, $s(a, probel) = -1$).

Важен выбор меры сходства между шпилькой и пробелом на первом шаге первого этапа, т.е. выбор штрафа за делецию. Он должен зависеть от того, насколько плотная структура ищется. Если ищется структура, начало и конец которой находятся в данных фрагментах далеко друг от друга, то предполагается, что делеций много, и штраф за них не должен быть большим. Если же ищется плотная структура, занимающая небольшое место, то штраф за делецию делается большим и тогда делеций может быть мало.

Далее, в каждом списке L_i оставляется небольшое число шпилек: лишь те, которые вошли в полученное в результате работы первого этапа подслово (состоящее из шпилек) хотя бы при одном j из пары (L_i, L_j); затем из них оставляются шпильки, которые сопоставлены не пробелу, а шпильке со степенью сходства более заданного порога.

На этих сокращенных списках запускается *второй этап* прореживания. Это некоторый алгоритм поиска системы попарно похожих слов, например, алгоритм из [2] поиска общей системы слов (сигнала) в наборе последовательностей (в нем все слова рассматриваются как независимые друг от друга элементы, поэтому вместо них могут стоять объекты любой природы). При этом используется тот алгоритм из [2], который выдает не одну лучшую систему, а некоторое заранее заданное их количество — не меньше предполагаемого количества шпилек в искомой вторичной структуре.

В нашей реализации описываемого алгоритма хорошо показал себя простейший вариант: каждой шпильке ставилось в соответствие качество, равное сумме ее сходств со всеми шпильками, имеющими с ней максимальные сходства в своих списках, и на выходе появлялось заданное число шпилек лучшего качества.

Программа была успешно тестирована на последовательностях tRNK кишечной палочки и некоторых аминокислотных оперонах. Планируется продолжать такое тестирование.

СПИСОК ЛИТЕРАТУРЫ

1. Верещагин Н.К., Любецкий В.А. Алгоритм определения вторичной структуры РНК. В сб.: *Труды научно-исследовательского семинара логического центра ИФ РАН*. М.: Изд-во РАН, 2000, вып. 14, стр. 99–109.
2. Горбунов К.Ю., Любецкий В.А. Алгоритм выявления регуляторного сигнала в наборе последовательностей. В сб.: *Логические исследования*, М.: Наука, 2000, вып. 7, стр. 159–164.

Статью представил к публикации член редколлегии В.А. Любецкий