МОЛЕКУЛЯРНАЯ БИОЛОГИЯ, 2001, том 35, № 5, с. 1–9

ГЕНОМИКА. ПРОТЕАМИКА. БИОИНФОРМАТИКА

УДК 577.1

АЛГОРИТМ ВЫДЕЛЕНИЯ РЕГУЛЯТОРНЫХ СИГНАЛОВ В ПОСЛЕДОВАТЕЛЬНОСТЯХ ДНК

© 2001 г. Л. В. Данилова, К. Ю. Горбунов, М. С. Гельфанд*, В. А. Любецкий

Институт проблем передачи информации Российской академии наук, Москва, 101447

[1]Государственный научный центр ГосНИИГенетика, Москва, 113545

Поступила в редакцию 17.11.2000 г.

Предложен алгоритм для выделения регуляторных сигналов в последовательностях ДНК. Сложность алгоритма близка к квадратичной. Приведены результаты тестирования алгоритма на искусственных и природных последовательностях.

Ключевые слова: молекулярная биология, регуляторный сигнал, компьютерная логика, алгоритмы на графах.

Задача выделения регуляторных сигналов является одной из классических задач вычислительной биологии. Она приобрела особую популярность в последние несколько лет после того, как началось проведение массовых экспериментов по анализу экспрессии генов (например, [1, 2]; обзоры в [3, 4]), с одной стороны, и были опубликованы полные геномы многих бактерий и некоторых эукариот, что сделало возможным сравнительный анализ процессов регуляции (в частности, [5, 6]; обзор в [7]). Несмотря на всю важность, эта задача еще далека от решения. Точные (а тем самым, и переборные) методы занимают столь большое время, что практически не осуществимы и не пригодны для решения реальных задач.

Существующие подходы и алгоритмы выделения регуляторных сигналов из набора потенциальных регуляторных областей описаны в обзорах [8–11].

Эти алгоритмы можно разделить на две группы: оптимизационные и комбинаторные. Оптимизационные алгоритмы организованы следующим образом: задается некоторая характеристика качества набора потенциальных сайтов (например, информационное содержание). Далее, производится построение таких наборов, причем выбор представителей в каждой последовательности постепенно уточняется с целью максимизации качества. Таким образом, вся процедура сводится к поиску экстремума функции качества на пространстве наборов. Для этого применяются жадные алгоритмы [12, 13], алгоритмы максимизации ожидания [14–17], DMS [18], MEME [19–21]; а также стохастические алгоритмы: имитация теплового отжига [22]; Gibbs sampler [23, 24].

* Эл. почта: dlv@mail.ru

Комбинаторные алгоритмы также работают с наборами слов, однако, в этом случае целью является построить слово, присутствующее в каждой последовательности из выборки с наименьшими отклонениями (т.е. функцией качества является какая-то мера компактности полученного набора, например, его диаметр). К числу таких алгоритмов относятся ConsInd и MatInd [25, 26]; ITB [27]; WORDUP [28, 29]; CONSENSUS [30]; WINNOWER [31]; суффиксные деревья [32–34]; а также другие алгоритмы [35–43].

В настоящей работе предложен и тестирован *новый алгоритм для выделения сигнала из выборки невыравненных нуклеотидных последовательностей* (см. также [44]). Он является промежуточным с точки зрения приведенной выше классификации – производится оптимизация функции качества, однако, она определяется через попарное сходство слов, а не как информационное содержание набора.

АЛГОРИТМ

Постановка задачи. Дан набор из k нуклеотидных последовательностей, длины которых не фиксированы или примерно одинаковы (и в этом случае равны каждая какому-то числу n).

Системой назовем набор слов фиксированной длины l, по одному слову из одной последовательности (или по несколько слов из одной последовательности – здесь для краткости будем говорить о первом случае); в систему включаются слова из какой-то *заранее не фиксированной части исходных последовательностей;* система должна состоять из как можно более попарно похожих друг на друга слов (из по возможности большего числа последовательностей). Похожесть понимается, например, в смысле суммы попарных расстояний

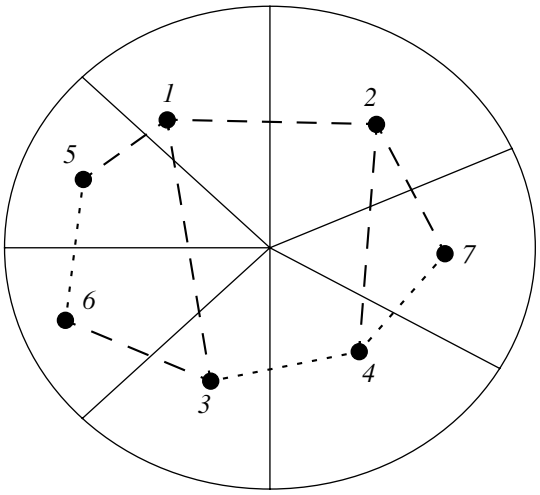

Рисунок 1. Процедура разбиения графа G.

Приведен пример разбиения графа при $k = 7$. *1 шаг*: разбиваем на два части P_1 и P_2, таким образом, что $P_1 = \{1, 3, 5, 6\}$ и $P_2 = \{2, 4, 7\}$, (1, 2) – основное ребро этого разбиения, а (3, 4) – вспомогательное; *2 шаг:* разбиваем часть P_1 на $P_{11} = \{1, 5\}$ и $P_{12} = \{3, 6\}$ с основным ребром (1, 3) и вспомогательным (5, 6), а часть P_2 на $P_{21} = \{2, 7\}$ и $P_{22} = \{4\}$ с основным ребром (2, 4) и вспомогательным (4, 7); *3 шаг*: разбиваем часть P_{11} на $P_{111} = \{1\}$ и $P_{112} = \{5\}$ с основным ребром (1, 5), часть P_{12} на $P_{121} = \{3\}$ и $P_{122} = \{6\}$ с основным ребром (3, 6), часть P_{21} на $P_{211} = \{2\}$ и $P_{212} = \{7\}$ с основным ребром (2,7).

Хэмминга между словами, входящими в систему, или в смысле какой-то другой фиксированной метрики между этими словами, или, лучше сказать, в смысле максимизации какого-то фиксированного "качества системы". Качество системы определяется, например, как сумма попарных "расстояний" между ее словами, вычисляемых с помощью функции $F(x, y)$, которая для двух слов x и y длины l отражает степень их похожести между собой (например, количество совпадающих букв в них), а также отражает степень наличия других желательных характеристик у такой пары слов. Интуитивно такая система понимается как *сигнал*; а слово, являющееся представителем сигнала в какой-то из исходных последовательностей, – как *регуляторный участок (сайт)* в этой последовательности.

Серьезная алгоритмическая и теоретическая проблема состоит в том, что некоторые последовательности из исходной выборки могут в принципе не содержать регуляторных участков, относящихся к рассматриваемому сигналу. Поэтому важно, что описанный алгоритм находит сигнал даже в том случае, когда он содержится в относительно небольшой части всех исходных последовательностей.

При необходимости можно учитывать структурные особенности сигналов. Так, например, если есть основания полагать, что сигнал является (комплементарным) палиндромом (в частности, если известно, что регуляторный белок связывается с ДНК как димер), то можно использовать, например, функцию $F(x, y) = S(x, y) + 0.5\ [\max(S(x, pal(x)), S(x', pal(x'))) + \max(S(y, pal(y)), S(y', pal(y')))]$, где $S(x, y)$ – количество совпадающих букв в словах x и y, а $pal(x)$ – слово, полученное из x обращением с заменой каждой буквы на комплементарную и x' – слово x без последней буквы. Эта весовая функция выбрана в значительной мере произвольно, и алгоритм не зависит от вида функции F, кроме очевидного соображения, что особо большая сложность ее вычисления приводит к увеличению времени работы алгоритма. Впрочем, нет основания ожидать появления биологически содержательных и при этом настолько сложно вычислимых функций F, что это приведет к заметному изменению времени работы алгоритма.

Предлагаемый алгоритм решает исходную задачу за время **квадратичное от числа k исходных последовательностей и кубичное от длины n каждой из них.** Трудно представить себе, что возможен алгоритм для решения этой задачи с лучшей оценкой времени его работы.

Качественное описание алгоритма. Сначала образуем вспомогательный граф G, который остается фиксированным в процессе работы алгоритма. Граф G состоит из k вершин и всех ребер, которые возникают в процессе выполнения следующей процедуры, см. пример на рис. 1, где $k = 7$. На первом шаге все вершины графа G разбиваются на две равные (с точностью до единицы, если k нечетное) части и между этими частями проводятся два ребра (A, B) и (C, D), не выходящие из одной вершины (пусть, скажем, A и C находятся в одной части, а B и D в другой). Любое из этих ребер, скажем, (A, B), назовем *основным относительно этого разбиения,* а другое – *вспомогательным.* На рис. 1 $(A, B) = (1, 2)$ и $(C, D) = (3, 4)$. Также проводятся два диагональных ребра (A, D) и (C, B), которые не показаны на рисунке. Далее итеративно повторяем такое разбиение "вглубь" графа G. А именно, каждую из двух полученных частей графа G снова разбиваем на две (в том же смысле) равные части так, что A и C, как и B и D, находятся в разных частях этих разбиений. Относительно уже этих разбиений основные ребра определены однозначно: это (A, C) и (B, D), а вспомогательные ребра (по возможности, не выходящие из той же вершины, что и основные) выбираются произвольно. На рис. 1 это – ребра $(5, 6)$ и $(4, 7)$. И так далее, каждую появившуюся в этой процедуре не одновершинную часть P разбиваем на две равные части P_1 и P_2 так, чтобы основные ребра новых частей соединяли концы основного и вспомогательного ребер предыдущей части P. Конечно, при этом каждое P равно объединению его частей P_1 и P_2 .

Процедура разбиений прекращается, когда все части P станут одновершинными; на самом деле, можно остановиться, когда эти части станут просто мелкими (из 1–3 вершин).

Содержание основного цикла алгоритма в общих словах можно описать так: для мелких частей любой вопрос решается в лоб, перебором, а для других частей выполняется индуктивный переход от некоторой уже вычисленной (уже известной) информации для P_1 и P_2 к однотипной информации о P. Иными словами, по информации f для P_1 и g для P_2 легко вычисляется однотипная (с f и g) информация h для P. Конечно, это только наводящее соображение, которое мы сейчас уточним.

Внешний цикл алгоритма состоит во взаимно однозначном приписывании каждой вершине графа G одной из исходных последовательностей; одну из таких (текущих, но фиксированных на каждой отдельной итерации этого цикла) расстановок последовательностей по вершинам графа G обозначим r; тогда $r(A)$ – это последовательность, приписанная вершине A, в которой, как и во всех других приписанных последовательностях, ищется сигнал. Граф G помогает организовать этот поиск. Тело этого цикла состоит в следующем.

Последовательность $r(A)$ будем называть последовательностью *над* A.

Для фиксированного r выполняется новый цикл (называемый *сборкой*), который состоит в *переходе* от информации о двух "соседних" (и более мелких) частях P_1 и P_2 графа G к информации об объединении этих частей в (более крупную) часть P графа G. Точнее, от информации о совокупности последовательностей над A, где A пробегает часть P_1 , и информации о совокупности последовательностях над B, где B пробегает часть P_2, к информации о совокупности последовательностей над C, где C пробегает часть P.

Условимся далее говорить о совокупности последовательностей $r(A)$, где A пробегает часть P, как о *"последовательностях над P"*. Систему слов, выбранных (не более, чем) по одному из каждой последовательности над P, будем называть сигналом (системой слов) над P. Конечно, мы ищем сигнал над $P = G$, который и называем сигналом; но ищем этот сигнал как специальный частный случай сигнала над произвольным P.

Наконец, наилучшую возможную систему слов, выбранных (не более, чем) по одному из каждой последовательности над P, у которой значения сигнала над основным ребром части P равны двум наперед заданным (вообще говоря, произвольным) словам x и y из соответствующих последовательностей, назовем *продолжением слов x и y до сигнала над P*. Понятно, что продолжение каких-то слов x и y до сигнала над P может не быть (осмысленным) сигналом над P по той тривиальной причине, что все портят эти "закрепленные" значения x и y.

Итак, упомянутый выше переход (индуктивный шаг) состоит в следующем.

Пусть для двух частей P_1 и P_2 с основными ребрами соответственно (A, C) и (B, D), полученных разбиением подграфа P (в исходном графе G) с основным ребром (A, B), уже определены два набора каждый из t лучших сигналов (систем), которые в первом наборе все над над P_1, а во втором наборе все над P_2. Разумеется, просто объединение лучшего сигнала над P_1 с лучшим сигналом над P_2 не является даже (осмысленным) сигналом над P (в общем случае).

Поэтому поступим тоньше. Упомянутый набор, скажем для P_1, состоит не из лучших сигналов над P_1, а из "наилучших (возможных) сигналов с закрепленными концами" над P_1, т.е. первый набор состоит из продолжений любых двух слов x и y из соответственно последовательностей над A и C (где x и y имеют взаимное качество большее некоторого фиксированного *порога* u) на множество P_1. И аналогично, второй набор состоит из t наилучших продолжений на все множество P_2 слов x и y, произвольно взятых над вершинами B и D.

Например, на рис. 1 пусть $P_1 = \{1, 5, 6, 3\}$ и $P_2 = \{2, 7, 4\}$; $(A, C) = (1, 3)$ и $(B, D) = (2, 4)$; P равно объединению множеств P_1 и P_2 (то, что в данном случае P совпало с G, конечно, случайно; так всегда будет в конце этой индукции, но не на предшествующих ему шагах). Пусть $t = 1$ (к слову сказать, так и было в тех биологических случаях, для которых этот алгоритм реально применялся). Тогда на предыдущих итерациях обсуждаемого цикла были вычислены две функции $f(x, y)$ и $g(x, y)$; функция $f(x, y)$ по любой паре слов: x из последовательности над A и y из последовательности над C, которые близки друг к другу более, чем на некоторый фиксированный порог u, – выдает наилучший сигнал в выборке $\{r(1), r(3), r(5), r(6)\}$, соответствующей множеству P_1 (при фиксированном r); аналогично функция $g(x, y)$ по любой паре слов: x из последовательности над B и y из последовательности над D, которые в том же смысле достаточно близки друг к другу, – выдает наилучший сигнал в выборке $\{r(2), r(4), r(7)\}$, соответствующей множеству P_2 (при том же фиксированном r). На этой итерации строится функция $h(x, y)$, которая выдает аналогичную информацию по отношению к более крупному множеству P.

А именно, в новом цикле *перебирая все слова* x_1 и y_1 из соответственно последовательностей над C и D, для которых попарное качество слов x с x_1 и y с y_1 выше этого порога (точнее, для которых определены продолжения), выберем по паре x, x_1 и отдельно по паре y, y_1 (с помощью соответ-

ственно f и g) продолжения на P_1 и P_2, которые при объединении на все P дают t лучших сигналов над всем P.

Если условие пороговости не может быть обеспечено, то соответствующее значение сигнала над P считается по определению равным нулю; иными словами, из-за пороговости возникают частично определенные сигналы над G, что, впрочем, естественно и с точки зрения постановки исходной задачи.

Кроме того, ведется список (реестр) S полученных сигналов. Как только возникает очередной сигнал s, он проверяется на похожесть с уже включенными в S сигналами (с той же областью определения). Это делается так: если на достаточно большой доле последовательностей сигнал s не имеет новых по сравнению с сигналами из S слов, то предполагаем, что среди этих не новых слов из s значительную часть составляют реальные сайты, и тогда смотрим, насколько близко к этой *сигнальной* совокупности каждое из оставшихся в s слов. Если среди них не обнаружено слов, близких к ней, то не включаем этот сигнал s в S.

Внешний цикл, состоящий в расстановке последовательностей по вершинам графа G, обеспечивает, что любая пара последовательностей хотя бы раз будет приписана вершинам графа G так, что окажется соединенной в нем ребром; при этом очередная расстановка выбирается из условия о том, чтобы больше пар последовательностей, не соединенных на предыдущих итерациях, сейчас соединились. Это обеспечивает разумное количество итераций внешнего цикла при достаточном разнообразии обсчитанных расстановок r.

Последнее важно для получения достаточно представительной статистики на следующем, последнем цикле работы алгоритма.

А именно, каждой позиции в каждой последовательности (где находится, скажем, буква i) ставится в соответствие число, которое отражает меру того, что буква i входит в искомый сайт. Второй вариант состоит в том, чтобы сопоставлять такое число каждому слову, отражая тем самым меру того, что это слово входит в искомый сайт (далее рассмотрим этот вариант). Это число равно сумме качеств по всем полученным сигналам, которые включают данное слово. Здесь под качеством понимается качество не всего сигнала, а качество слова из сигнала (который его содержит) по отношению ко всему этому сигналу, т.е. сумма значений $F(x, y)$, где x – упомянутое слово, а y пробегает все остальные слова этого сигнала. Таким образом, слова, входящие в "реальный" сигнал, будут помечены в исходных последовательностях числами, которые заметно больше чисел, которые помечают другие слова.

Идея алгоритма в следующем. Понятие наилучшего (т.е. претендующего на реальность) сигнала требует учитывать все пары последовательностей (в которых имеется сигнал). Для каждой расстановки r выбираются сигналы, которые соответствуют определению наилучшего сигнала только для пар последовательностей, связанных ребром в графе G (при данной расстановке), а таких ребер неизмеримо меньше, чем пар всевозможных последовательностей. Но за счет выбора подходящей цепочки расстановок удается получить представительную статистику для последующего выделения сигнала. Этот абзац интуитивно поясняет идею алгоритма, вопрос о доказательствах здесь не обсуждается.

РЕАЛИЗАЦИЯ АЛГОРИТМА И ВЫЧИСЛИТЕЛЬНЫЙ ЭКСПЕРИМЕНТ

Алгоритм был реализован в виде компьютерной программы, которую использовали в основном для поиска сигнала в наборе генетических последовательностей с характерной длиной от 100 до 200 нуклеотидов; в этом частном случае сигнал представлял собой систему отдельных слов с характерной длиной от 15 до 30. В этом варианте программа была написана на языке Object Pascal с помощью среды программирования Delphi. На ее вход подавался текстовый файл, содержащий набор генетических последовательностей, а на ее выходе создавался файл, содержащий функцию, которая каждому слову сопоставляет степень достоверности того, что оно входит в сигнал. С помощью этой программы при пороге u, равном половине длины сигнала, на компьютере Pentium-Celeron 300 MHz с оперативной памятью 64 MB обработка 19 последовательностей длины 200 заняла 12 мин, а 9 последовательностей той же длины – 1.2 мин.

Счет, проведенный на многих примерах, показал, что в подавляющем большинстве случаев алгоритм предлагал хотя бы один сайт в каждой из содержащих сигнал последовательностей.

Чтобы проверить, не содержит ли некоторая последовательность других сигнальных слов, применялся следующий прием. Вместо каждой буквы найденного в данной последовательности слова ставился знак "звездочка", т.е. буква, которую программа считала отличной от всех букв и даже от самой себя. После этого алгоритм находил в последовательности другое сигнальное слово (если оно там было) и т.д.

В редких случаях алгоритм не находил в какой-то последовательности ни одного сигнального слова (хотя оно там было). Причиной этого был "мусор", т.е. совокупность очень похожих друг на друга, но не сигнальных слов. Этот мусор выделялся сравнительно большими числами и его легко было увидеть (особенно, с учетом биологических соображений). Тогда "мусорные" слова забива-

Таблица 1. Результаты тестирования на искусственных последовательностях*

	0, 1, 2	3	4	5	6
0	XXXXXXXXXX	XXXXXXXXXX	XXXXXXXXXX	1061011555	1220021000
1	XXXXXXXXXX	XXXXXXXXXX	XXXXX9XXXX	0063	–
2	XXXXXXXXXX	XXXXXXXXXX	XXXX97XX7X	0053	–
3	XXXXXXXXXX	XXXXXXXXXX	X9X478XX9X	0065	–
4	XXXXXXXXXX	XXXXXXXXXX	799286X96X	1000	–
5	XXXXXXXXXX	XXXXXXXXXX	3498168858	0020	–
6	XXXXXXXXXX	XXXXXXXXXX	05X0477052	0000	–
7	XXXXXXXXXX	XXXXXXXXXX	0330053404	0000	–
8	XXXXXXXXXX	XXXXXXXXXX	2240004025	0000	–
9	XXXXXXXXXX	XXXXXXXX9X	2650427021	0000	–
10	XXXXXXXXXX	XXXXXXXXXX	8423552057	–	–

	0, 1	2	3	4
0	5	5555555555	5555555555	554530255
1	5	5555555555	5555555555	255413144
2	5	5555555555	5555555555	514330023
3	5	5555555555	5555453541	005000030
4	5	5555555555	5535255554	003020011
5	5	4555555555	3403535443	000300013

* Приведено количество правильно найденных сайтов, каждый символ соответствует одному независимому тесту (X обозначает 10). Верхняя таблица: выборка из 10 последовательностей. Нижняя таблица: выборка из 5 последовательностей. Строки: количество добавленных последовательностей, не содержащих сайтов, к исходным, содержащим сайт, (от 0 до 10 и от 0 до 5 соответственно). Столбцы: количество измененных букв (от 0 до 6 и от 0 до 4 соответственно).

лись звездочками, программа запускалась снова и находила ранее не найденные сигнальные слова.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Искусственная выборка. В контролируемых условиях для определения характеристик алгоритма проведено его тестирование на искусственной выборке. Сгенерированы последовательности длины 200 в четырехбуквенном алфавите, в каждую из них было подставлено одно и тоже слово длины 16. Затем в каждом из этих слов случайным образом "портили" по несколько букв (имитация ослабления сигнала), а также добавляли последовательности, не содержащие сигнала (имитация загрязнения выборки).

Результаты тестирования приведены в табл. 1.

Сайты устойчиво находятся при внесении в сигнал до 2 ошибок при выборке из 5 последовательностей, до 3 ошибок – из 10 последовательностей, а также, когда количество лишних последовательностей (не содержащих сайты) составляло до 50% выборки. Случай 10 последовательностей исследован более подробно. При ошибках в 4 позициях сигнала результат зависит от чистоты выборки: приемлемые результаты получаются при количестве последовательностей (не содержащие сайтов) до 3–4 (в большинстве тестов сайты правильно определяются практически во всех последовательностях). При дальнейшем загрязнении выборки сигнал может не быть обнаружен, доля таких тестов повышается с увеличением лишних последовательностей. При более слабом сигнале сайты обнаруживаются только в отдельных тестах.

Природные выборки. По аналогичной схеме рассмотрены три выборки последовательностей, содержащих регуляторные сайты *Escherichia coli*. Процедуру загрязнения выборки лишними последовательностями имитировали следующим образом: маскировали лучший из сайтов, полученных на очередном этапе тестирования (нуклеотиды, составляющие этот сайт заменяли знаком *). Тем самым, не только появлялись последовательности, не содержащие сайтов, но и постепенно ослаблялся сам сигнал.

Последовательности регуляторных сайтов взяты из базы данных dpinteract [45], а фрагменты последовательностей, содержащие эти сайты, извлечены из полного генома *E. coli* при помощи программы GenomeExplorer [46].

Таблица 2. Выделение сайтов связывания PurR[1]

	0	0	0	1	1	2	2	3	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	11	13
Оперон	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M
purR-1	88	8	*	8	*	8	*	14	*	7	*	6	*	6	*	6	*	13	*	7	*	**12**	*	**12**	*	7	*	7
purR-2	0		72		63		41		27		24		24		23		23		15		7		7		7		7	
purEK	88	0	88	0	*	12	*	7	*	18	*	12	*	12	*	19	*	12	*	7	*	7	*	7	*	7	*	7
CvpApurF	88	0	88	0	76	0	*	22	*	22	*	21	*	14	*	14	*	13	*	7	*	7	*	7	*	7	*	7
purC	87	0	80	0	70	0	59	0	29	0	18	6	18	6	22	6	24	7	15	7	15	7	8	7	8	7	7	7
purMN	88	0	88	0	70	0	60	0	30	0	28	0	*	22	*	42	*	13	*	13	*	13	*	6	*	7	*	12
purL	88	0	88	0	75	0	60	6	*	24	*	24	*	28	*	20	*	13	*	7	*	12	*	12	*	12	*	12
purB	80	0	80	0	59	7	45	6	27	6	17	6	17	6	16	6	16	7	13	7	13	7	8	7	8	7	**6**	**11**
guaBA	79	0	80	0	67	0	54	0	27	0	17	7	17	6	23	14	24	7	*	14	*	14	*	14	*	14	*	21
purHD	88	0	88	0	70	0	59	0	30	0	27	7	27	7	24	7	24	7	7	**13**	8	7	8	6	7	7	**0**	**14**
glyA	80	0	80	0	70	0	59	0	29	0	26	6	26	8	25	7	17	7	8	7	8	7	8	7	**8**	**12**	**0**	**8**
pyrD	88	0	88	0	70	0	49	0	29	0	26	0	26	0	*	12	*	12	*	7	*	7	*	7	*	7	*	11
prsA	88	0	88	0	70	0	60	0	30	0	*	28	*	21	*	19	*	7	*	12	*	7	*	7	*	7	*	12
glnB	80	0	80	0	63	0	53	0	27	0	16	8	15	8	14	7	7	6	13	6	13	6	13	6	*	6	*	7
purA-1	80	0	80	0	63	0	53	0	27	0	24	6	24	6	31	5	22	**6**	7	7	7	7	11	7	11	7	*	7
purA-2	0		0		0		0		0		0		0		7		**6**		**6**		**6**		**0**		**0**		**0**	
codBA	88	0	88	0	75	0	60	0	30	0	27	0	26	0	32	6	17	8	15	8	*	8	*	10	*	8	*	8
pyrC	80	0	80	0	69	0	59	0	29	0	18	5	18	5	9	7	24	7	15	6	7	7	7	6	7	7	7	7
purT	88	0	88	0	76	0	61	0	30	0	27	0	27	0	26	5	*	7	*	12	*	12	*	12	*	7	*	12
GcvTHP	72	0	72	0	63	0	45	7	26	7	15	0	14	7	14	6	15	7	15	6	15	6	7	6	7	6	7	7
speAB	70	8	70	8	54	5	45	6	18	5	17	8	17	8	23	5	16	7	15	7	15	7	*	7	*	11	*	13

[1]Первая строка: количество последовательностей, не содержащих сайтов, и общее количество замаскированных сайтов (обозначены звездочками). Вторая строка: C – вес истинного сайта, M – максимальный вес ложного сайта. Жирный шрифт – пропущенный истинный сайт; лучший ложный сайт той же или большей силы, что и лучший истинный (кроме нулевого веса).

Пуриновый регулон. Выборка регуляторных областей генов, регулируемых пуриновым репрессором PurR, состояла из 19 последовательностей длиной 200 нуклеотидов, содержащих 21 сайт длиной 16 нуклеотидов. Две последовательности содержали по два сайта, остальные – по одному. Результаты тестирования приведены в табл. 2. Первые ошибки появляются при маскировке 8–9 сайтов, однако даже если выборка более чем наполовину состояла из последовательностей, не содержащих сайты, большинство сайтов опознавалось правильно. Когда в одной последовательности содержится два сайта, то второй находится при маскировке уже найденного. В пуриновой выборке таких последовательностей две.

Аргининовый регулон. Выборка регуляторных областей генов, регулируемых аргининовым репрессором ArgR, состояла из 9 последовательностей длиной 200 нуклеотидов, содержащих 19 сайтов длиной 18 нуклеотидов. Одна последовательность содержала три сайта, остальные – по два. Результаты тестирования приведены в табл. 3. Аргининовый бокс – слабый сигнал, и специфичность регуляции осуществляется за счет кооперативного узнавания мультимерными комплексами молекул репрессора пар сайтов, расположенных на фиксированном расстоянии друг от друга [47]. Тем не менее, сайты связывания аргининового репрессора находятся достаточно уверенно даже при маскировке значительного числа лучших сайтов: первые потери обнаруживаются при маскировке 5 сайтов, причем в одной из последовательностей оказываются замаскированными оба сайта. Второй сайт находится также, как и в пуриновой выборке, при маскировке одного, уже найденного. В случае трех сайтов – процедура та же, т.е. маскируем два уже найденных и находим третий.

Таблица 3. Выделение сайтов связывания ArgR[1]

	0	0	0	1	0	2	0	3	0	4	1	5	1	6	1	7	1	8	0	9
Оперон	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M	C	M
argR-1	33	0	36	0	*	0	*	8	*	8	*	0	*	**13**	*	**18**	*	**9**	*	**18**
argR-2	0		0		24		34		18		8		**0**		**8**		**8**		**8**	
argA-1	33	0	32	0	19	0	19	0	17	8	9	7	9	8	**0**	**16**	**0**	**16**	*	**16**
argA-2	0		0		9		26		17		9		9		**9**		**0**		**9**	
argCBH-1	0	0	**2**	**6**	18	0	37	7	27	7	16	8	**8**	**8**	**8**	**8**	**0**	**9**	**10**	**18**
argCBH-2	39		*		*		*		*		*		*		*		*		*	
argD-1	0	0	0	0	0	0	0	0	8	7	7	7	**7**	**8**	**8**	**8**	**8**	**7**	**8**	**16**
argD-2	35		35		30		51		*		*		*		*		*		*	
argE-1	0	0	0	0	9	0	10	0	10	0	8	0	**0**	8	**0**	**9**	**0**	**9**	**10**	**18**
argE-2	39		27		22		43		38		8		9		**9**		**8**		*	
argF-1	36	0	36	0	31	0	21	0	19	0	**6**	**14**	10	8	*	**17**	*	**8**	*	**18**
argE-2	0		0		0		30		27		**6**		**0**		**7**		**7**		**10**	
argG-1	11	0	0	0	10	0	10	0	19	0	**8**	8	8	7	**9**	**16**	**0**	**9**	**10**	**16**
argG-2	0		0		0		0		0		**0**		8		**0**		**0**		**0**	
argG-3	24		36		22		42		29		10		*		*		*		*	
argI-1	36	0	35	0	31	0	*	0	*	0	*	15	*	8	*	16	*	8	*	**16**
argI-2	0		0		0		47		44		*		*		*		*		**10**	
carAB-1	30	0	16	0	24	0	18	0	19	0	9	8	9	8	17	8	*	**8**	*	**16**
carAB-2	0		7		0		9		8		**8**		**8**		**0**		**0**		**8**	

Обозначения как в табл. 2. Последняя пара столбцов: в каждой последовательности замаскирован лучший сайт.

Регулон катаболитной репрессии. Выборка регуляторных областей генов, регулируемых белком CRP, состояла из 31 последовательности длиной 200 нуклеотидов, содержащих 48 сайтов длиной 22 нуклеотидов. В 16 последовательностях содержался один сайт, в остальных – от двух до четырех. Выборка сайтов связывания CRP содержит множество слабых сайтов, многие из которых не были найдены даже в первоначальной постановке (табл. 4), поэтому в этом случае тесты с маскировкой сайтов не проводились. Следует отметить, что взаимодействия CRP с регуляторными участками сложны и включают динамические переключения с одних сайтов на другие [48]. Поэтому нельзя исключать, что некоторые из ошибочно опознанных сайтов на самом деле функциональны.

Тестирование показало практическую применимость предложенного алгоритма. В настоящее время он используется в практике анализа регуляторных взаимодействий; в частности, с его помощью анализируются неохарактеризованные регулоны сахарного метаболизма гамма-протеобактерий, а также – начат систематический анализ регуляции в пирококках.

Дальнейшее развитие алгоритма будет проводиться в нескольких направлениях. Актуальной является проблема повышения устойчивости к шуму, в особенности, к загрязнению выборки. Предполагается модифицировать алгоритм таким образом, чтобы он явно принимал во внимание возможность загрязнения, а ожидаемое количество лишних последовательностей являлось параметром, выставляемым исходя из априорных соображений или настраиваемым автоматически.

Необходимо также принять во внимание статистические особенности исследуемых геномов, в частности, неравномерность нуклеотидного состава и особенности частот олигонуклеотидов (слово, не являющееся случайным по отношению ко всему геному вряд ли является специфическим регуляторным сигналом).

Следует аккуратно исследовать возможности использования различных типов симметрий (палиндромы, прямые повторы и т.п.). Наконец, алгоритм должен более специфически учитывать возможность появления в последовательности нескольких сайтов, а в выборке – нескольких различных сигналов.

Мы благодарны А.А. Миронову за помощь в работе с GenomeExplorer и ценное обсуждение. Эта работа была частично поддержана грантами Merck Genome Research Institute (переведите,пожа-

Таблица 4. Выделение сайтов связывания CRP. Обозначения как в табл. 2

	C	M
aldB	**8**	**15**
ansB	**0**	**18**
araB-1	17	8
araB-2	**7**	
cdd-1	**0**	7
cdd-2	28	
crp-1	**0**	14
crp-2	16	
cya	27	9
cytR-1	17	16
cytR-2	7	
dadAX-1	43	8
dadAX-2	**8**	
deoP-1	**9**	**17**
deop-2	**8**	
fur	26	13
gal	**8**	**21**
glpACB-1	26	8
glpACB-2	**8**	
glpACB-3	**8**	
glpD	17	8
glpFK-1	**0**	15
glpFK-2	32	
gut	34	14
ilvB	**10**	**17**
lac-1	9	7
lac-2	24	
malEpKp-1	8	6
malEpKp-2	9	
malEpKp-3	9	
malEpKp-4	**0**	
malT	18	16
melR	**16**	**30**
mtl	**17**	**21**
nupG-1	20	9
nupG-2	19	
ompA	**16**	**16**
ompR	24	16
ptsH-1	**9**	**26**
ptsH-2	**16**	
rhaS	19	8
rot-1	**8**	8
rot-2	27	
tdcA	28	8
tnaL	28	8
tsx-1	**10**	**26**
tsx-2	7	
uxuAB	23	8

луйста) (244), INTAS (99-1476), Российского фонда фундаментальных исследований (99-04-48247 и 00-15-99362), the Howard Hughes Medical Institute (55000309) и программой “Геном человека”.

СПИСОК ЛИТЕРАТУРЫ

1. *Spellman P.T. et al.* // Mol. Biol. Cell. 1998. V. 9. P. 3273–3297.
2. *Roth F.R. et al.* // Nature Biotechnol. 1998. V. 16. P. 939–945.
3. *Bassett D.E. Jr. et al.* // Nature Genet. 1999. V. 21: P. 51–55.
4. *Bucher P.* // Curr. Opin. Struct. Biol. 1999. V. 9. P. 400–407.
5. *Gelfand M.S., Koonin E.V., Mironov A.A.* // Nucleic Acids Res. 2000. V. 28. P. 695–705.
6. *McGuire A.M., Hughes J.D., Church G.M.* // Genome Res. 2000. V. 10. P. 744–757.
7. *Gelfand M.S.* // Res. Microbiol. 1999. V. 150. P. 755–771.
8. *Gelfand M.S.* // J. Comput. Biol. 1995. V. 2. P. 87–115.
9. *Frech K., Quandt K., Werner T.* // Comput. Appl. Biosci. 1997. V. 13. P. 89–97.
10. *Duret L., Bucher P.* //. Curr. Opin. Struct. Biol. 1997. V. 7. P. 399–406.
11. *Fickett J.W., Wasserman W.W.* // Curr. Opin. Biotechnol. 2000. V. 11. P. 19–24.
12. *Stormo G.D., Hartzell G.W.* III. // Proc.Natl. Acad. Sci. USA. 1989 V. 86. P. 1183–1187.
13. *Hertz G.Z., Hartzell G.W. III, Stormo G.D.* // Comput. Appl. Biosci. 1990 V. 6. P. 81–92.
14. *Lawrence C.E., Reilly A.A.* // PROTEINS: Structure, Function, Genetics. 1990. V. 7. P. 41–51.
15. *Cardon L.R., Stormo G.D.* // J. Mol. Biol. 1992 V. 223. P. 159–170.
16. *Frishman D., Mironov A., Gelfand M.* // Gene. 1999 V. 234. P. 257–265.
17. *Gelfand M.S., Koonin E.V., Mironov A.A.* // Nucleic Acids Res. 2000. V. 28. P. 695–705.
18. *Hu Y.-J., Sandmeyer S., McLaughlin C., Kibler D.* // Bioinformatics. 2000. V. 16. P. 222–232.
19. *Bailey T.L., Elkan C.P.* // Proc. 2nd Int. Conf. on Intelligent Systems for Molecular Biology ISMB'1994. 1994 P. 28–36.
20. *Bailey T.L., Elkan C.P.* // Proc. 3rd Int. Conf. on Intelligent Sysems for Molecular Biology ISMB'1995. 1995. P. 21–29.
21. *Grundy W.N., Bailey T.L., Elkan C.P.* // Comput. Appl. Biosci. 1996. V. 12. P. 303–310.
22. *Lukashin A.V., Engelbrecht J., Brunak S.* // Nucleic Acids Res. 1992. V. 20. P. 2511–2516.
23. *Lawrence C.E., Altschul S.F., Boguski M.S., Liu J.S., Neuwald A.F., Wootton J.C.* // Science. 1993. V. 262. P. 208–214.
24. *Roth F.P., Hughes D., Estep P.W., Church G.M.* // Nature Biotech. 1998. V. 16. P. 939–945.
25. *Frech K., Herrmann G., Werner T.* // Nuceic. Acids Res. 1993. V. 21. P. 1655–1664.

26. *Quandt K., Frech K., Karas H., Wingender E., Werner T.* // Nucleic Acids Res. 1995. V. 23. P. 4878–4884.
27. *Kielbasa Sz.M., Korbel J.O., Beule D., Schuchhardt J., Herzel H.* // Proc. German Conf. on Bioinformatics GCB'2000. 2000. P. 55–62.
28. *Pesole G., Prunella N., Liuni S., Attimonelli M., Saccone C.* // Nucleic Acids Res. 1992. V. 20. P. 287102875.
29. *Liuni S., Prunella N., Pesole G., Dorazio T., Stella E., Distante A.* // Comput. Appl. Biosci. 1993 V. 9. P. 701–707.
30. *Hertz G.Z., Stormo G.D.* // Bioinformatics. 1999. V. 15. P. 563–577.
31. *Pevzner P.A., Sze S.-H.* // Proc. 8th Int. Conf. on Intelligent Systems for Molecular Biology ISMB'2000. 2000. P. 269–278.
32. *Jonassen I.* // Comput. Appl. Biosci. 1997. V. 13. P. 509–522.
33. *Brazma A., Jonassen I., Vilo J., Ukkonen E.* // Genome Res. 1998. V. 8. P. 1202–1215.
34. *Marsan L., Sagot M.-F.* // Proc. 4th Annu. Int. Conf. on Computational Molecular Biology RECOMB'2000. 2000. P. 210–219.
35. *Ulyanov A.V., Stormo G.D.* // Nucleic Acids Res. 1995. V. 23. P. 1434–1440.
36. *Fraenkel Y.M., Mandel Y., Friedberg D., Margalit H.* // Comput. Appl. Biosci. 1995. V. 11. P. 379–387.
37. *Rocke E., Tompa M.* // Proc. 2nd Annu. Int. Conf. on Computational Molecular Biology RECOMB'98. 1998. P. 228–233.
38. *Tompa M.* // Proc. 7th Int. Conf. on Intelligent Systems for Molecular Biology ISMB'1999. 1999. P. 262–271.
39. *Rigoutsos I., Floratos A.* //. Bioinformatics. 1998. V. 14. P. 55–67.
40. *van Helden J., Andre B., Collado-Vides J.* // J. Mol. Biol. 1998. V. 281. P. 827–842.
41. *Jensen L.J., Knudsen S.* // Bioinformatics. 2000. V. 16. P. 326–333.
42. *Cho R.J., Campbell M.J., Winzeler E.A., Steinmetz L., Conway A., Wodicka L., Wolfsberg T.G., Gabrielian A.E., Landsman D., Lockhart D.J., Davis R.W.* // Molecular Cell. 1998. V. 2. P. 65–73.
43. *Wolfsberg T.G., Gabrielian A.E., Campbell M.J., Cho R.J., Spouge J.L., Landsman D.* // Genome Res. 1999. V. 9. P. 775–792.
44. *Вьюгин В.В., Горбунов К.Ю., Любецкий В.А.* // Труды конференции “Проблемы управления и моделирования в сложных системах” / Под ред. Мясникова В.П. М.: РАН, 2000. С. 130–137.
45. *Robison K., McGuire A.M., Church G.M.* // J. Mol. Biol. 1998. V. 284. P. 241–254.
46. *Миронов А.А., Винокурова Н.П., Гельфанд М.С.* // Молекуляр. биология. 2000. Т. 34. С. 253–264.
47. *Maas W.K.* // Microbiol. Rev. 1994. V. 58. P. 631–640.
48. *Busby S., Kolb A.* // Regulation of Gene Expression in Escherichia coli. / Eds. Lin E.C.C., Lynch A.S. Место издания? E.G.Landes Co., 1995. Ch. 12. P. 255–279.22.