МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ им. М. В. ЛОМОНОСОВА

ФАКУЛЬТЕТ БИОИНЖЕНЕРИИ И БИОИНФОРМАТИКИ

Программа для поиска гомологов нуклеотидных последовательностей

Курсовая работа
студента IV курса
Ю. А. Пекова

Научный руководитель:
к.ф.-м.н., с.н.с. С. А. Спирин

Москва 2011

Оглавление

Глава 1

Введение

Задача поиска гомологов нуклеотидных последовательностей в банке данных является одной из важнейших задач биоинформатики. В случае кодирующих последовательностей для проведения такого поиска успешно используется программа TBLASTN. Но для поиска гомологов некодирующих последовательностей вполне удовлетворяющего инструмента нет. Для этого используется ряд программ, самые распространенные из них — FASTA [2,9], BLASTN [3,4,10] и discontiguous MEGABLAST [11], однако каждая из этих программ обладает существенным недостатком. FASTA при сравнении двух последовательностей выдает только одно выравнивание — то, которое имеет наивысший счет среди всех найденных выравниваний. Однако очень часто в последовательности из базы данных есть и другие значимые участки, гомологичные входной последовательности. Из-за того, что выравнивание с ними обладает чуть меньшим счетом, чем лучшее выравнивание, они не обнаруживаются программой. BLASTN использует ускоренный алгоритм поиска, записывая положение в базе данных всех слов длиной N нуклеотидов (N = 7, 11 или 15). Но такой подход приводит к уменьшению чувствительности: если в гомологичной последовательности не встретится участок длиной N букв, полностью совпадающий с участком входной последовательности, то выравнивание этих двух последовательностей не обнаружится. Discontiguous MEGABLAST индексирует в банке данных не слова, а паттерны длиной t. Паттерн соответствует слову, если в них совпадает хотя бы W букв в определенных местах. Это призвано увеличить чувствительность, но на практике в большинстве случаев чувствительность в сравнении с BLASTN уменьшается. Целью настоящей работы было создание компьютерной программы для поиска гомологов нуклеотидных последовательностей, превосходящей по чувствительности как программу FASTA, так и программу BLASTN.

Глава 2

Литературный обзор

Существует два основных подхода к построению выравнивания двух последовательностей — глобальное и локальное выравнивание. Оба этих подхода предназначены для построения выравнивания с максимальным весом. Различие состоит в том, что в первом случае выравнивание включает в себя обе последовательности целиком, а во втором для достижения максимального веса разрешается ограничиться лишь отрезками последовательностей.

Общий вес для каждого выравнивания вычисляется как сумма трех слагаемых. Первое слагаемое — количество позиций выравнивания, в которых содержатся одинаковые буквы (нуклеотиды), умноженное на некоторое положительное число — цену совпадения (match). Второе слагаемое — количество позиций выравнивания, в которых содержатся разные буквы, умноженное на некоторое отрицательное число — цену несовпадения (mismatch). Третье слагаемое — штраф за гэпы, неположительное число, которое зависит от наличия и количества гэпов в выравнивании.

Диагональю при сравнении двух последовательностей называется диагональ матрицы — таблицы локального сходства. Ее также можно описать как сдвиг одной последовательности относительно другой, или как полное выравнивание с гэпами только по краям. Все эти определения эквивалентны.

Для построения локального выравнивания обычно используется алгоритм Смита — Ватермана [7] или его модификации. Основное же различие среди программ поиска гомологов нуклеотидных последовательностей заключается в методах предварительного поиска участков возможного локального сходства.

Основные подходы и алгоритмы, используемые в поиске сходных последовательностей, описаны в следующих работах:

1. Karlin, S. and Altschul, S. F. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.

   При построении выравнивания важной задачей является оценка значимости найденного выравнивания. Для случая локального выравнивания без разрывов (без гэпов) в статье предлагается использовать статистический подход, основанный на распределении максимума весов выравнивания по независимым случайным последовательностям.

Чем меньше вероятность, что этот максимум будет больше, чем вес найденного выравнивания, тем более значима эта находка.

Ожидаемое число выравниваний по случайным последовательностям с весом, превышающим S, может быть рассчитано по формуле:

$$E = Kmne^{-\lambda S}, \tag{2.1}$$

где m и n — длины сравниваемых последовательностей, а K и λ — некоторые постоянные. Полученное значение E в литературе часто обозначается как "ожидаемая величина" или "expect value" (далее по тексту - $Evalue$). λ вычисляется как положительный корень уравнения

$$\sum_{i,j} p_i p_j \exp(\lambda s(i,j)) = 1,$$

где p_i — частота встречаемости буквы i, p_j — частота встречаемости буквы j, а $s(i,j)$ — соответствующий элемент матрицы замен. Постоянная K также зависит только от p_i, p_j и $s(i,j)$, но вычисляется по более сложной формуле.

Необходимо отметить, что формула (2.1) применима лишь при отрицательном математическом ожидании цены сопоставления двух случайно выбранных букв. Кроме того, алгоритм Смита — Ватермана выдает осмысленный результат только при выполнении того же условия (хотя формально может быть применён и без его выполнения).

Вероятность того, что существует вес выравнивания больший, чем S, рассчитывается по формуле:

$$P(x > S) = 1 - e^{-E}$$

При значениях $E < 0{,}01$ вероятность P практически совпадает с E.

2. Pearson, W. R. and Lipman, D. J. 1988. Improved tools for biological sequence comparison.

   В данной статье предложена программа FASTA, предназначенная для эвристического поиска последовательностей (в том числе нуклеотидных). Для нахождения локального выравнивания с высоким весом она использует процедуру, состоящую из четырех этапов. На первом этапе создается таблица, в которую заносится расположение всех одинаковых букв длины $ktup$ в двух последовательностях. Для нуклеотидных последовательностей $ktup$ может принимать значения от 3 до 6. Затем отбираются 10 лучших (с наибольшим количеством совпадений слов) диагоналей.

   На втором этапе в отобранных диагоналях ищутся безразрывные участки с максимальным весом, так называемые "затравки". На третьем этапе проверяется, можно ли соединить эти затравки, используя разрывы и учитывая штрафы за них. Наконец, для последнего этапа отбираются диагонали, лежащие вокруг одного участка с самым высоким счетом, на которых вновь проводят локальное выравнивание с помощью модифицированных алгоритмов Нидельмана — Вунша и Смита — Ватермана.

Программа FASTA является самой ранней программой поиска гомологов среди распространенных ныне аналогов, и для нуклеотидных последовательностей считается самой чувствительной.

3. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990. Basic local alignment search tool.

   Предложенная в статье программа BLAST перед началом поиска индексирует последовательности из банка данных. При этом создается список, содержащий все слова фиксированной длины, которые локально выравниваются с этими последовательностями с весом выше порогового значения. Для каждого слова указывается расположение всех участков, с которыми оно выровнялось. Длина слова для нуклеотидных последовательностей — 7, 11 (по умолчанию) или 15. Затем происходит собственно поиск, в ходе которого все слова той же длины из последовательности запроса сравниваются со словами, найденными в базе данных при индексации. Если найдена пара одинаковых слов, то начинается процесс расширения совпадающего участка. Расширение происходит без разрывов, в обоих направлениях и до достижения максимального веса. Получающиеся в ходе этого выравнивания выдаются в качестве результата.

4. Autschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs.

   В данной статье описывается новая версия программы BLAST, которая расширяет совпадающие участки и строит выравнивания с учетом возможных разрывов.

5. Discontiguous MEGABLAST: http://www.ncbi.nlm.nih.gov/blast/discontiguous.shtml

   Discontiguous MEGABLAST по сравнению с BLASTN предназначен для поиска более дальних гомологов. При индексации банка данных записываются не целые слова, а паттерны длиной t (t = 16, 18 или 21). Каждому типу паттернов также соответствует число W: считается, что паттерн подобен слову, если в них совпадает хотя бы W букв в определенных местах (W = 11 или 12). После индексации банка данных каждое слово из последовательности запроса сравнивается с записанными паттернами, и при нахождении подобных паттернов соответствующий участок расширяется подобно тому, как это происходит в программе BLASTN.

   Для поиска используется несколько различных типов паттернов, предназначенных для поиска гомологов различной степени сходства для кодирующих или некодирующих последовательностей.

Глава 3

Описание программы

Ниже изложены основные принципы работы созданной программы Nhunt, а также входные и выходные данные.

На вход программе подается файл с банком последовательностей и файл с одной последовательностью (запрос), для которой мы хотим найти гомологи из базы данных. Все последовательности должны быть представлены в FASTA–формате. Работа программы состоит из трех основных этапов: поиск наилучших диагоналей, локальное выравнивание для найденных наилучших диагоналей и слияние лучших локальных выравниваний.

3.1 Поиск наилучших диагоналей

Диагональ численно определяется сдвигом одной последовательности относительно другой.

Вначале программа индексирует последовательность запроса. При этом для каждого возможного слова длиной $w + 1$ запоминаются места, в которых в этой последовательности встретились оно само или его "соседи". Слово A является соседом слова B, если выполняются следующие условия:

— длины слов A и B совпадают

— первые буквы слов A и B совпадают

— хотя бы w букв в одинаковых позициях в словах A и B совпадают

Например, для слова atg всеми возможными соседями являются следующие слова:
ata, atc, att, agg, acg, aag

Значение w является параметром программы и задается пользователем.

Точно так же индексируется последовательность, комплементарная к последовательности запроса. После этого начинается проход по последовательностям базы данных. При этом программа поочередно просматривает все позиции каждой последовательности базы, сдвигаясь на один нуклеотид. Для каждой позиции находится слово длиной $w + 1$, которое начинается в данной позиции. Например, для последовательности aaattcggcatta при $w + 1 = 3$ будут найдены следующие слова:
aaa, aat, att, ttc, tcg, cgg, ggc, gca, cat, att, tta

Все буквы, отличные от A, T, G и C, заменялись буквой A (это упрощает и ускоряет про-

цедуру поиска диагоналей, при этом практически не влияя на результат). Затем для каждого слова, найденного в позиции у последовательности базы, искалось идентичное ему слово из результата индексации запроса. Пусть оно (в качестве себя самого или соседа другого слова) встречалось k раз в запросе в позициях b_i. Тогда для всех i от 1 до k счет для каждой диагонали, численно задаваемой выражением

$$y - b_i + lenquery - w,$$

увеличивается на единицу. Здесь и далее $lenquery$ — длина последовательности запроса.

Для ускорения работы программы слишком длинные последовательности базы данных нарезаются на более короткие куски, каждый из которых далее обрабатывается независимо. Чтобы предотвратить возможную потерю хороших диагоналей, к началу каждого следующего куска добавляется окончание предыдущего, равное по длине последовательности запроса. После прохождения таким образом всей базы данных из всех возможных диагоналей отбираются те, для которых выполняется условие:

$$D_k \geq E_q + \sigma \cdot p, \tag{3.1}$$

где D_k — счет k-ой диагонали, p — задаваемый пользователем параметр, E_q — ожидаемый счет и σ — среднеквадратичное отклонение. Эта оценка основана на предположении, что в каждой позиции каждой диагонали P_n — вероятность случайного совпадения слова из последовательности запроса и слова из банка данных — одинакова и зависит только от длины слова. Такая вероятность может быть вычислена по формуле $P_n = \frac{1+3 \cdot w}{4^{w+1}}$, где $w + 1$ — длина слова. Тогда ожидаемый счет для каждой диагонали может быть рассчитан как произведение длины запроса (то есть длины диагонали) на вероятность совпадения слов в одной позиции:

$$E_q = P_n \cdot lenquery,$$

В рамках предложенной модели количество соседей для каждой диагонали имеет биномиальное распределение, для которого среднеквадратичное отклонение задается следующей формулой:

$$\sigma = \sqrt{lenquery \cdot P_n \cdot (1 - P_n)}$$

Таким образом, чем больше счет диагонали отличается от ожидаемого, тем более вероятно, что она не является случайной и на ней следует провести локальное выравнивание.

3.2 Локальное выравнивание для лучших диагоналей

Для отобранных вышеописанным способом лучших диагоналей в несколько итераций проводится локальное выравнивание с помощью упрощённого алгоритма Смита — Ватермана. Посмотрим, как это происходит, на примере одной короткой диагонали.

Сначала обе последовательности в соответствии с номером диагонали записываются друг под другом:

					C	A	C	C	G	C	G	A	A		
G	C	A	A	T	C	A	G	C	C	T	A	T	A	C	G	G

Затем удаляются краевые участки, в которых представлена только одна последовательность и в которых выравнивание невозможно:

C	A	C	C	G	C	G	A	A
A	T	C	A	G	C	C	T	A

Далее с правого края добавляется “лишняя“ пара букв (F), а все позиции (пары букв) нумеруются, начиная с левого края:

1	2	3	4	5	6	7	8	9	10
C	A	C	C	G	C	G	A	A	F
A	T	C	A	G	C	C	T	A	F

Затем каждой позиции такого выравнивания присваивается число в соответствии со следующими правилами (параметры $match$ и $mismatch$ задаются пользователем):

— последней по счету позиции (с парой букв F) присваивается число 0.

— если в позиции под номером n буквы совпадают, то ей присваивается число $S_{n+1}+match$, где S_{n+1} — число, присвоенное позиции $n+1$, $match$ — натуральное число, которое по смыслу является добавленным весом за совпадение букв.

— если в позиции под номером n буквы не совпадают, то ей присваивается число $S_{n+1} + mismatch$, где S_{n+1} — число, присвоенное позиции $n+1$, $mismatch$ — отрицательное целое число, которое по смыслу является штрафом за несовпадение букв. Однако если $S_{n+1} + mismatch < 0$, то позиции присваивается число 0.

— для пары букв, одна или обе из которых являются“кодами неопределённости” (“ambiguity codes”), то есть обозначают подмножества множества A, T, G, C, вместо значений match и mismatch используется среднее значение цены сопоставления по всем выборам букв A, T, G, C из соответствующих множеств.

Для примера в приведенном ниже выравнивании использованы следующие значения: $match = 5, mismatch = -4$.

1	2	3	4	5	6	7	8	9	10
C	A	C	C	G	C	G	A	A	F
A	T	C	A	G	C	C	T	A	F
3	7	11	6	10	5	0	1	5	0

Затем выбирается позиция, которой присвоено наибольшее число (в данном случае это позиция под номером 3). От этой позиции вправо начинается расширение выравнивания, которое заканчивается первой встреченной позицией, которой присвоено число 0 (в данном случае это позиция 7). Таким образом, в первое найденное локальное выравнивание попадаются позиции от 3 до 6 включительно.

После этого участок найденного выравнивания ”вырезается”, а участки, оставшиеся по краям, проверяются на длину. Если их длина больше заданного пользователем параметра

L (по умолчанию $L = 10$), то с ними по отдельности происходит так же процедура, что и с исходным выравниванием, и так далее, пока не останется ни одного кусочка, большего или равного L. Каждому вырезанному участку локального выравнивания сопоставляется счет S, равный числу, присвоенному первой позиции этого участка (в нашем примере это число 11).

3.3 Слияние участков локального сходства

Среди найденных участков локального сходства отбираются те, для которых вес

$$S \geq z \cdot match, \tag{3.2}$$

где z — натуральное число, которое задается пользователем. Отобранные участки со всех диагоналей сортируются по номеру позиции в банке данных, с которого они начинаются. В том случае, если пара соседних участков расположена достаточно близко, из запроса и из банка данных вырезаются последовательности максимальной длины, лежащие между началом первого участка и концом второго. Эти две последовательности выравниваются алгоритмом Смита — Ватермана для локального выравнивания двух последовательностей с афинными штрафами за гэпы. Афинные штрафы означают, что штраф за участок идущих подряд n гэпов рассчитывается как сумма $-GapOpen - n \cdot GapExtention$, где $GapOpen$ - штраф за открытие участка гэпов, $GapExtention$ - штраф за каждый гэп в отдельности. Оба этих параметра являются неотрицательными целыми числами и задаются пользователем.

Если счет получившегося выравнивания больше, чем счет каждого из первоначальных участков, первый участок из пары удаляется, а второй заменяется на новое выравнивание. Далее проверяется следующая пара, в которой второй участок из предыдущей пары становится первым, и так далее до проверки последней пары.

Все оставшиеся участки, которые не были слиты с соседями, также выравниваются алгоритмом Смита — Ватермана.

Далее по формуле, предложенной в [1], для каждого выравнивания вычисляется $Evalue$:

$$Evalue = Kmne^{-\lambda S},$$

где m и n — длины последовательностей запроса и базы данных, а K и λ — некоторые постоянные, получение которых описано в главе 5.

В выходной файл записываются все участки локального выравнивания с $Evalue$, меньшим или равным пороговому значению $Evalue$, которое также задается пользовательским параметром.

Программа реализована на языке программирования C.

Глава 4

Руководство пользователя

Исполняемые файлы для архитектур Linux x86/amd64 и исходный код программы для самостоятельной компиляции доступны по адресу: http://mouse.belozersky.msu.ru/~bennigsen/nhunt

Программа запускается из командной строки, ниже приведен минимальный формат запуска:
nhunt -i input.fasta -d database.fasta,
где input.fasta — название файла с последовательностью запроса, database.fasta — название файла с последовательностями банка данных.

Полный список ключей программы:

1. Ключ –i. Query File. Название файла с последовательностью запроса, для которой ведется поиск гомологов. Обязательный параметр.

2. Ключ –d. Database. Название файла с последовательностями банка данных, в которых ведется поиск гомологов. Обязательный параметр.

3. Ключ –o. nhunt report Output File. Название выходного файла, в который записываются результаты работы программы. Значение по умолчанию: “nhunt.out”.

4. Ключ –e. Expectation value. Положительное число. Порог на значение E-value. Выравнивания с E-value, превышающим заданный этим параметром порог, не выдаются в результатах. Значение по умолчанию: 1.0

5. Ключ –s. Minimal score for output alignments. Неотрицательное целое число. Порог на значение счета выравнивания. Выравнивания с счетом меньшим, чем заданный этим параметром порог, не выдаются в результатах. Значение по умолчанию: 0

6. Ключ –p. Diagonals selection threshold. Положительное число. Порог для отбора лучших диагоналей (см. формулу (3.1)). Значение по умолчанию: 10.

7. Ключ –w. Number of coincident letters. Натуральное число. Минимальное количество букв, которые должны совпадать в двух “соседних“ словах. Значение по умолчанию: 5.

8. Ключ –z. Score threshold. Натуральное число. Определяет минимальный счет для участка локального сходства (см. формулу (3.2). Значение по умолчанию: 5.

9. Ключ –r. Reward for a nucleotide match. Неотрицательное целое число. Цена совпадения двух букв в выравнивании. Значение по умолчанию: 5.

10. Ключ –q. Penalty for a nucleotide mismatch. Неположительное целое число. Цена за несовпадение двух букв в выравнивании. Значение по умолчанию: -4.

11. Ключ –G. Cost to open a gap. Неотрицательное число. Штраф за появление группы идущих подряд гэпов в выравнивании. Значение по умолчанию: 10.

12. Ключ –E. Cost to extend a gap. Неотрицательное число. Штраф за появление каждого нового гэпа в выравнивании. Значение по умолчанию: 5.

13. Ключ –m. Alignment view option. Данный параметр определяет вид выходного файла. Возможные значения:

    ”0”. Выходной файл содержит выровненные последовательности и информацию об этих выравниваниях.

    ”8”. Выходной файл содержит таблицу с информацией о выравниваниях, но не сами выравнивания.

    “9“. То же, что в предыдущем пункте, но в файле также содержатся комментарии к таблице.

    Значение по умолчанию: 0.

Актуальный список параметров можно получить, вызвав программу без параметров.

Глава 5

Выбор параметров K и λ

Строго говоря, формула (2.1) применима только для подсчета вероятностей появления выравниваний без гэпов (без разрывов). При том же условии выводятся и формулы для расчета параметров K и λ. Общей формулы для подсчета вероятностей появления выравниваний с гэпами в настоящий момент не существует. В связи с этим для расчета $Evalue$ в программе была использована формула (2.1), однако параметры K и λ были выведены на основе статистического моделирования.

Процедура моделирования состояла из нескольких этапов:

1. Программа nhunt запускалась для поиска гомологов случайных последовательностей запроса на случайных базах данных. Случайные последовательности генерировались на основе бернуллиевской модели, то есть вероятности появления каждой из четырех букв в каждой позиции были одинаковы и независимы. Длина последовательности банка данных — $5 \cdot 10^6$, длины последовательностей запроса — 50, 75, 100, 150 и 200. Программа запускалась по 10 раз для каждой из длин запроса, причем для каждого нового поиска заново генерировалась как последовательность запроса, так и последовательность банка данных.

2. Все веса найденных в результате этих запусков выравниваний были отсортированы по возрастанию. Затем для каждого S_i (значения i-того веса) была вычислена величина $E_i = \frac{N_{align}}{n \cdot m}$, где m, n — длины последовательностей запроса и базы данных, а N_{align} — число найденных выравниваний, вес которых оказался большим, чем S_i.

3. Введем величину $P_i = Ke^{-\lambda S_i}$, где K и λ — искомые параметры. Из формулы (2.1) видно, что в том случае, когда эти параметры подобраны верно, дожно выполняться равенство

$$N_{align} = nmKe^{-\lambda S_i} \Leftrightarrow P_i = E_i.$$

Следовательно, наша цель — подобрать значение P_i таким образом, чтобы оно было как можно ближе к экспериментальному значению E_i. Введем также обозначения $f_i = \ln P_{i+1} - \ln P_i$ и $F_i = \ln E_{i+1} - \ln E_i$. Такие замены были произведены для того, чтобы подбирать параметры по очереди: вначале путем минимизации разницы между f_i и F_i подбирается параметр λ, а затем уже параметр K.

Легко видеть, что единсвенным аргументом функции f_i является λ:

$$f_i = \ln P_{i+1} - \ln P_i = \ln Ke^{-\lambda S_{i+1}} - \ln Ke^{-\lambda S_i} = \ln K - \lambda S_{i+1} - \ln K + \lambda S_i = \lambda(S_i - S_{i+1})$$

Оптимальное значение λ_w параметра λ было подобрано методом наименьших квадратов с использованием математического пакета GNU Octave:

$$\lambda_w = argmin(\sum_i (F_i - f_i)^2)$$

4. Для нахождения оптимального значения K_w параметра K приравняем $E_i = P_i$ и решим получившиеся уравнения $E_i = Ke^{-\lambda_w \cdot S_i}$ относительно K. Среднеарифметическое этих решений и определим как оптимальное значение K_w.

Необходимо отметить, что получаемые значения K_w и λ_w достаточно сильно зависили от того, используем ли мы для оптимизации весь отсортированный список весов S_i или лишь его часть, начиная с некоторого S_k. На наш взгляд, это может свидетельствовать о том, что истинный вид зависимости *Evalue* от счета выравнивания с гэпами отличается от зависимости, предложенной в формуле (2.1). Так как нас в первую очередь интересует *Evalue* для высоких значений счета (то есть для редко встречаемых выравниваний), то в некоторых случаях для оптимизации использовался список весов лишь начиная с некоторого S_k. Такой подход позволял точнее оценить K_w и λ_w для высоких значений счета.

Таблица 5.1: Значения K_w и λ_w при различных параметрах

match	*mismatch*	*GapOpen*	*GapExtention*	K_w	λ_w
5	-2	10	5	0.005	0.065
5	-2	15	5	0.192	0.074
5	-2	20	5	0.01	0.098
5	-3	10	5	0.071	0.169
5	-3	15	5	0.202	0.185
5	-3	20	5	0.15	0.182
5	-4	10	5	0.107	0.204
5	-4	15	5	0.172	0.214
5	-4	20	5	0.085	0.204
5	-5	10	5	0.065	0.206
5	-5	15	5	0.332	0.233
5	-5	20	5	0.383	0.235
5	-10	10	5	0.246	0.26
5	-10	15	5	0.56	0.276
5	-10	20	5	0.604	0.277
5	-15	10	5	0.365	0.275
5	-15	15	5	0.407	0.278
5	-15	20	5	0.288	0.271

Процедура оптимизации проводилась для каждого набора параметров $\{match, mismatch, GapOpen, GapExtention\}$ (см. главу 4), где $match = 5$, $mismatch$ принимает одно из значений набора $\{-2, -3, -4, -5, -10, -15\}$, $GapOpen$ — одно из значений набора $\{10, 15, 20\}$, $GapExtention = 5$. Полученные значения K_w и λ_w представлены в таблице 5.1. Другие параметры программы подбирались таким образом, чтобы получить максимальное число найденных выравниваний за разумное время.

Пользователь также может задать параметры $match = R$ и $mismatch = Q$, которые отличаются от представленных в таблице. В том случае, если значения параметров R и Q удовлетворяют требованию отрицательного математического ожидания цены сопоставления двух случайно выбранных букв, то оба значения умножаются на величину $5/R$, вследствие чего значение параметра $match$ становится равным 5 и таким образом как бы нормируется по таблице 5.1. В ином случае пользователю предлагается выбрать другие параметры. Если $Q \cdot 5/R < -15$, пользователю также предлагается ввести другие значения параметров.

Параметры $GapOpen$ и $GapExtention$ нормируются на значение $match = 5$ по вышеописанной процедуре, аналогично параметру $mismatch$. Если получившиеся новые значения отличаются от представленных в таблице, то для дальнейших вычислений выбираются наборы параметров, который ближе всего к заданным пользователем значениям $GapOpen$ и $GapExtention$; кроме того, программа выдает предупреждение, что расчет $Evalue$ может быть некорректным, и предлагает использовать варианты значений параметров из таблицы 5.1.

Далее параметры K_w и λ_w подбираются методом линейной интерполяции, основываясь на уже вычисленных значениях K_w и λ_w для различных наборов параметров. Для компенсации изменения значений параметров в $5/R$ раз окончательное значение λ'_w параметра λ вычисляется следующим образом:

$$\lambda'_w = \lambda_w \cdot 5/R$$

Глава 6

Результаты тестирования программы

Для сравнения программы Nhunt с программами FASTA и BLASTN они были попарно запущены для поиска гомологов различных РНК в геномах различных прокариот. Показателем качества программы служит количество находок, имеющих вес не меньше заданного. Для того, чтобы такое сравнение было корректно, вес каждой находки программ FASTA и BLASTN пересчитывался в соответсвии с параметрами, заданными для программы Nhunt по умолчанию.

6.1 Сравнение программ Nhunt и FASTA

Программы Nhunt и FASTA тестировались на поиске гомологов валиновой тРНК *E. coli* в геноме *E. coli* и в геномах пяти различных архейных микроорганизмов. Для каждой из программ было измерено время работы и количество находок, имеющих вес больше, чем заданные значения. Программа Nhunt была запущена с параметрами по умолчанию, программа FASTA - с параметрами $GapOpen = 10$, $GapExtention = 5$ и $ktup = 3$ (описание последнего параметра см. в главе 2); значение остальных параметров также были заданы по умолчанию.

В таблицах с результатами тестирования для каждой из сравниваемых программ представлены время ее работы в секундах и количество находок, вес которых не меньше, чем различные заданные значения весов. Кроме того, для каждого из заданного значения весов в скобках представлено значение $Evalue$, вычисленное по формуле (2.1); значения параметров k и λ взяты из таблицы 5.1.

6.1.1 Поиск гомологов тРНК *E. coli* в геноме *E. coli*

- Входная последовательность: валиновая тРНК *E. coli* (длина последовательности - 77 нуклеотидов)
- Последовательность базы данных: геном *E. coli K-12* (длина генома — 4639675 нуклеотидов)

Результаты тестирования представлены в таблице 6.1:

Таблица 6.1: Количество находок гомологов тРНК *E. coli* в геноме *E. coli*

Программа	Время	300 ($1 \cdot 10^{-19}$)	200 ($7 \cdot 10^{-11}$)	150 ($2 \cdot 10^{-6}$)	100 ($5 \cdot 10^{-2}$)	75 (8.6)
FASTA	4.7	2	6	21	38	58
Nhunt	0.6	2	6	21	46	83

6.1.2 Поиск гомологов тРНК *E. coli* в геномах архей

- Входная последовательность: валиновая тРНК *E. coli* (длина последовательности - 77 нуклеотидов)
- Последовательности базы данных: геномы архей *Archaeoglobus fulgidus DSM 4304*, *Halobacterium sp. NRC-1*, *Sulfolobus solfataricus P2*, *Pyrobaculum aerophilum str. IM2*, *Picrophilus torridus DSM 9790* (суммарная длина геномов — 10953209 нуклеотидов)

Результаты тестирования представлены в таблице 6.2:

Таблица 6.2: Количество находок гомологов тРНК *E. coli* в геномах архей

Программа	Время	140 ($4 \cdot 10^{-19}$)	120 ($2 \cdot 10^{-3}$)	100 (0.12)	85 (2.7)	75 (20.5)
FASTA	5.5	1	6	14	27	49
Nhunt	1.4	2	8	17	42	129

6.2 Сравнение программ Nhunt и BLASTN

Программы Nhunt и BLASTN тестировались на поиске гомологов нескольких тРНК *E. coli* в геномах пяти различных архейных микроорганизмов, а также на поиске гомологов группы miscRNA *E. coli* в геномах трех бактерий. Для каждой из программ было измерено количество находок, имеющих вес больше, чем заданные значения. Программа Nhunt была запущена с параметрами по умолчанию, программа BLASTN — с подобранными нами параметрами, при которых находится наибольшее число выравниваний. Ниже представлена строка запуска программы BLASTN:

```
blastall -p blastn -F F -W 7 -r 5 -q -4 -G 10 -i input.fasta -d database.fasta
```

Здесь ключ $-r$ задает цену за совпадение двух букв, $-q$ — цену за несовпадение, $-G$ — штраф за появление группы гэпов, $-E$ — штраф за отдельный гэп, $-W$ - длину слова при индексации базы данных (описание последнего параметра см. в главе 2). Остальные параметры заданы по умолчанию.

В таблицах с результатами тестирования для каждой из сравниваемых программ представлено количество находок, вес которых не меньше, чем различные заданные значения

весов. Кроме того, для каждого из заданного значения весов в скобках представлено значение *Evalue*, вычисленное по формуле (2.1); значения параметров K и λ взяты из таблицы 5.1.

6.2.1 Поиск гомологов пяти тРНК *E. coli* в геномах архей

- Входные последовательности: пять тРНК *E. coli* — валиновая, селеноцистеиновая, аспарагиновая, лейциновая и аргининовая (средняя длина последовательностей - 82 нуклеотида)
- Последовательности базы данных: геномы архей *Archaeoglobus fulgidus DSM 4304*, *Halobacterium sp. NRC-1*, *Sulfolobus solfataricus P2*, *Pyrobaculum aerophilum str. IM2*, *Picrophilus torridus DSM 9790* (суммарная длина геномов — 10953209 нуклеотидов)

Результаты тестирования представлены в таблице 6.3:

Таблица 6.3: Количество находок гомологов пяти тРНК *E. coli* в геномах архей

Программа	200 ($2 \cdot 10^{-10}$)	150 ($5 \cdot 10^{-6}$)	125 ($8 \cdot 10^{-4}$)	100 (0.13)	75 (21.8)
BLASTN	1	18	38	75	139
Nhunt	2	21	54	121	357

Кроме того, в данном тестировании координаты найденных гомологов сравнивались с координатами аннотированных в геномах архей тРНК. Для каждого найденного выравнивания была подсчитана длина “правильного” фрагмента (то есть число нуклеотидов гомолога, лежащих в пределах координат аннотированной тРНК) и длина “неправильного” фрагмента (то есть число нуклеотидов гомолога, лежащих вне предела координат аннотированной тРНК). Результаты подсчета представлены на рис. 6.1. На данном графике каждой программе соответствует график, а каждому найденному выравниванию соответствует точка. Значение ординаты этой точки выражает суммарную длину “правильных” фрагментов для всех выравниваний, имеющих счет больший, чем данное. Значение абсциссы выражает суммарную длину “неправильных” фрагментов для всех выравниваний, имеющих счет больший, чем данное. Точки, обозначающие самые лучшие выравнивания, находятся в начале координатной сетки. Следовательно, при заданном значении абсциссы качество программы можно охарактеризовать площадью под графиком: действительно, площадь тем больше, чем больше суммарная длина найденных “правильных” фрагментов.

В этом тесте для сравнения была также запущена программа BLASTN с параметрами по умолчанию (обозначена как “BLASTN (default)”).

6.2.2 Поиск гомологов группы miscРНК *E. coli* в геномах бактерий

Под miscРНК бактерии будем понимать все РНК, которые не являются транспортными или рибосомальными. Очевидно, что в данную группу входит также большое число т. н. “ан-

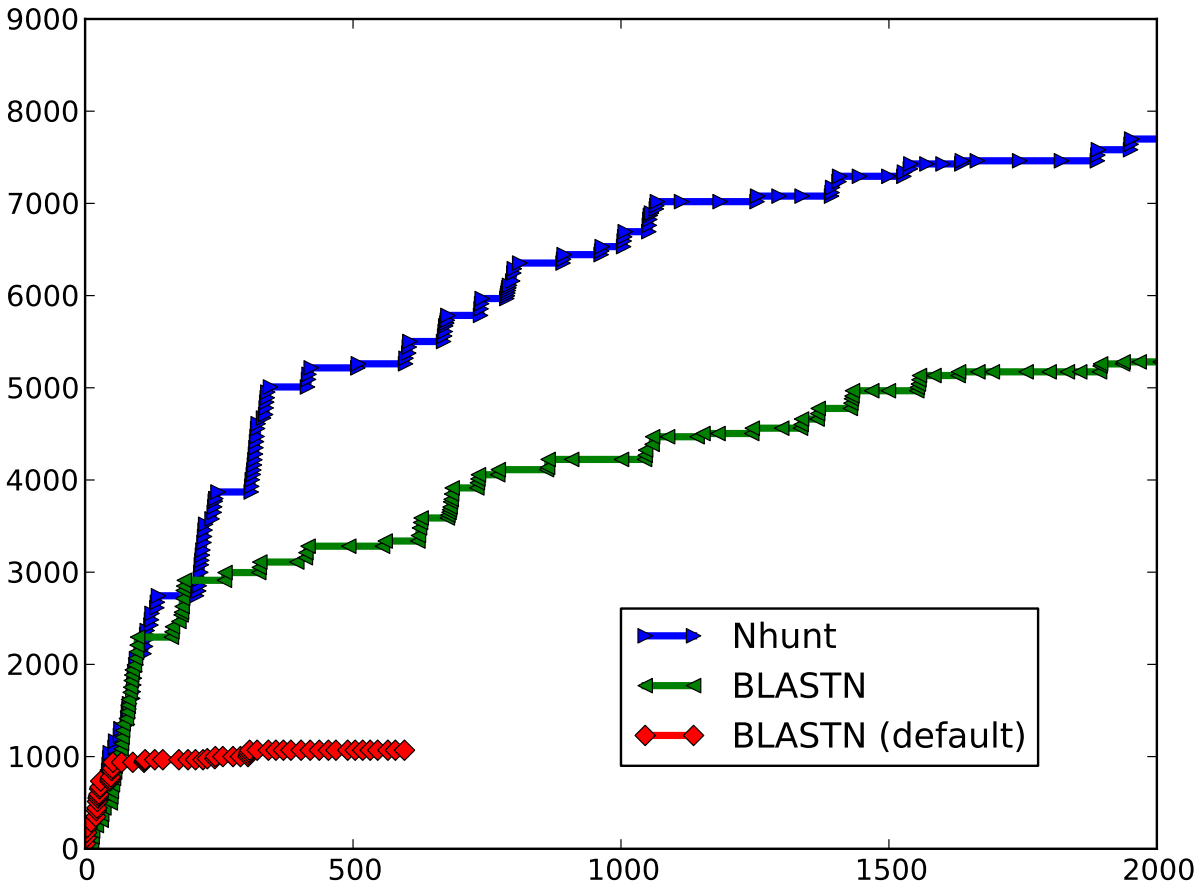

Рис. 6.1: Суммарное число “правильных” и “неправильных” фрагментов

тисмысловых” РНК, которые полностью или частично кодируют последовательность, комплементарную гену бактерии; это свойство обеспечивает клетке дополнительную ступень регуляции трансляции. Поэтому для того, чтобы не сравнивать последовательности генов различных бактерий, из всей совокупности miscРНК *E. coli* были выбраны те, которые не имеют гомологов среди генов *E. coli*. В штамме *Escherichia coli 55989*, где проаннотированно наибольшее число miscRNA (110), таковых оказалось 64. В качестве последовательностей банка данных были выбраны три бактерии, отличные друг от друга по степени родства с *E. coli*.

- Входные последовательности: группа из 64 miscРНК *E. coli 55989* (средняя длина последовательностей - 142 нуклеотида)
- Последовательности базы данных: геномы бактерий *Bacillus cereus B4264*, *Pseudomonas aeruginosa PAO1* и *Yersinia pestis KIM* (средняя длина геномов — 5428065 нуклеотидов)

Результаты тестирования представлены в таблице 6.4:

Таблица 6.4: Количество находок гомологов группы miscRNA *E. coli* в геномах бактерий

Программа	200 ($1.6 \cdot 10^{-10}$)	150 ($4 \cdot 10^{-6}$)	125 ($7 \cdot 10^{-4}$)	100 (0.11)	75 (18.7)
B. cereus					
BLASTN	2	5	14	146	859
Nhunt	2	6	15	238	4135
P. aeruginosa					
BLASTN	4	6	11	97	369
Nhunt	4	8	13	187	2060
Y. pestis					
BLASTN	23	29	34	112	684
Nhunt	25	30	41	167	2796

Была также предпринята попытка оценить качество полученных выравниваний способом, аналогичным описанному в разделе 6.2.1. Под “правильными” фрагментами в данном случае понимаются фрагменты гомолога, принадлежащие межгенным промежуткам. Показано, что в находках обеих программ практически независимо от выбранного веса суммарная длина “правильных” фрагментов выравниваний с бóльшим весом примерно в два раза превосходит суммарную длину “неправильных” фрагментов. Значение данного соотношения пока остается неясным; ожидалось, что с падением веса и, следовательно, качества находок данное соотношение будет резко падать, так как по суммарной длине доля межгенных участков не превышает 15 % по отношению к длине всего генома.

Глава 7

Обсуждение

Результаты тестирований показали, что по чувствительности (то есть по количеству находок с весом, больше заданного) программа Nhunt превосходит как программу FASTA, так и программу BLASTN. Это различие менее заметно в области больши́х весов и становится все более выражено при уменьшении порогового веса. Это говорит о том, что хорошие выравнивания с больши́м весом одинакого хорошо обнаруживаются всеми программами, однако значительное число менее качественных находок обраруживается лишь программой Nhunt.

Другая оценка, независимая от веса выравнивания и направленная на подсчет суммы "правильных" фрагментов, также показала превосходство программы Nhunt. Более высокая площадь под графиком для этой программы (см. рис. 6.1) по сравнению с программой BLASTN означает, что выравнивания, найденные программой Nhunt, но не найденные программой BLASTN, являются "правильными" и, следовательно, обладают биологическим смыслом.

По скорости работы программа Nhunt заметно превосходит программу FASTA, хотя и работает гораздо медленнее, чем программа BLASTN. Дальнейшие планы по улучшению программы Nhunt включают в себя также ускорение программы.

Кроме того, программа Nhunt имеет множество параметров, позволяющих регулировать соотношение "скорость работы/чувствительность". Параметры, заданные по умолчанию, на наш взгляд, позволяют достичь разумного компромисса между этими значениями. Однако за счет увеличения времени работы чувствительность можно заметно повысить.

Дополнительно показано, что программа BLASTN с подобранными нами параметрами заметно провосходит программу BLASTN, запущенную с параметрами по умолчанию. Это может означать, что параметры по умолчанию в программе BLASTN подобраны неоптимально.

Глава 8

Выводы

1. Нами была создана компьютерная программа Nhunt для поиска гомологов нуклеотидных последовательностей.

2. Эта программа была успешно протестирована на ряде примеров, определены значения параметров, необходимых для правильной оценки значимости найденных выравниваний.

3. При сравнении программы Nhunt с аналогичными программами показано, что она превосходит программу FASTA как в скорости, так и в чувствительности, и превосходит в чувствительности программу BLASTN.

Литература

1. Karlin, S. and Altschul, S. F. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proceedings of the National Academy of Sciences of the USA 87:2264-2268.

2. Pearson, W. R. and Lipman, D. J. 1988. Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences of the USA 4:2444-2448.

3. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J. 1990. Basic local alignment search tool. Journal of Molecular Biology 215:403-410.

4. Autschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 25:3389-3402.

5. Dembo, A. and Karlin, S. 1991. Strong limit theorems of empirical functionals for large exceedances of partial sums of i.i.d. variables. Annals of Probability 19:1737-1755.

6. Дурбин Р., Эдди Ш., Крог А., Митчисон Г. Анализ биологических последовательностей. М.-Ижевск, НИЦ "Регулярная и хаотическая динамика", Институт компьютерных исследований, 2006.

7. Smith, T. F. and Waterman, M. S. 1981. Identification of common molecular subsequences. Journal of Molecular Biology 147:195-197.

8. Gotea V., Veeramachaneni V. and Makalowski W. Mastering seeds for genomic size nucleotide BLAST searches. Nucleic Acids Research, 2003, 31:6935-6941.

9. http://faculty.virginia.edu/wrpearson/fasta/

10. ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/user_manual.pdf

11. http://www.ncbi.nlm.nih.gov/blast/discontiguous.shtml