ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И ЭКСПЕРТНЫЕ СИСТЕМЫ
(Вычислительные системы)
1996 год Выпуск 157

УДК 519.688:519.713.1

БЫСТРЫЙ ПОИСК В ТЕКСТОВЫХ БАЗАХ ДАННЫХ ПО ГРУППОВОМУ ЧАСТИЧНО СПЕЦИФИЦИРОВАННОМУ ЗАПРОСУ [1]

В.Д.Гусев, Л.А.Немытикова

Введение

В последние годы возникли и активно пополняются текстовые базы данных, содержащие слитные, предварительно неструктурированные тексты. Примерами таких баз являются:

- базы данных первичных структур ДНК- и РНК-молекул (суммарная длина текстов — порядка 10^6 символов, размер алфавита — 4, характерные длины текстов — от 10^3 до 10^5 символов);
- базы данных аминокислотных последовательностей (суммарная длина текстов — порядка 10^7 символов, размер алфавита — 20, характерные длины текстов — от 10^2 до 10^3 символов);
- базы данных древнерусских песнопений в знаменной форме записи (начальный этап создания). Суммарная длина хранящихся текстов — порядка 10^5 символов,

[1]Работа выполнена в рамках проекта, поддержанного грантом Госкомитета по высшему образованию РФ (Фундаментальные исследования в области информатики).

размер алфавита — порядка 200 символов, характерные длины текстов песнопений — 10^2 — 10^3 символов.

Как правило, алфавиты в перечисленных выше базах данных допускают различные варианты осмысленного агрегирования (разбиения на непересекающиеся подмножества близких по какому-то признаку элементов). *Частично специфицированный образец* — это цепочка символов, элементы которой в общем случае заданы с точностью до принадлежности к определенным подмножествам исходного алфавита. Такие образцы естественным образом возникают в задачах классификации при построении консенсусной (обобщающей) последовательности для группы близких текстов. Ситуация, когда поиск осуществляется по множеству частично специфицированных образцов, называется поиском по групповому частично специфицированному запросу.

Заметим, что указанная выше трактовка частично специфицированного образца не является типичной. В литературе в основном рассматривается случай частично специфицированного образца, отдельные элементы которого заданы точно, а все другие несущественны ("элементы типа X"). В некоторых отношениях отсутствие в образцах промежуточных градаций между точно заданными элементами и элементами типа X является облегчающим фактором.

Целью данной работы является создание быстрых алгоритмов поиска, ориентированных на реализацию на персональных компьютерах среднего класса. При этом предполагается, что *число элементов* в групповом запросе может быть очень велико (10^3 — 10^4 образцов) и все они предъявляются одновременно. Возможности ускорения поиска связаны с предварительным изучением структуры образцов (этап предобработки) и с запараллеливанием поиска для групп образцов с определенными структурными особенностями.

1. Постановка задачи

Пусть Σ — конечный алфавит, $|\Sigma|$ — размер алфавита, Z — текст, составленный из элементов Σ, $N = |Z|$ — длина текста Z, $Z[i]$ — i-й элемент текста Z, $Z[i:i+k]$ — фрагмент текста с i-го по $i+k$-й элемент включительно. Для обозначения различных (в общем случае пересекающихся) подмножеств алфавита Σ, содержащих не менее двух элементов, будем использовать алфавит D, непересекающийся с Σ. *Частично специфицированный образец* — это цепочка $p = b_1 b_2 \ldots b_m$, элементы которой принадлежат алфавиту $\Sigma \cup D$.

Задача поиска по частично специфицированному образцу p заключается в отыскании в тексте Z всех фрагментов, *согласованных* с p. Фрагмент $a_1 a_2 \ldots a_m$ $(a_i \in \Sigma,\ 1 \le i \le m)$ считается согласованным с образцом $p = b_1 b_2 \ldots b_m$ $(b_i \in \Sigma \cup D)$, если для всех $1 \le i \le m$ имеет место: $a_i = b_i$, если $b_i \in \Sigma$, либо $a_i \in b_i$, если $b_i \in D$. В случае группового частично специфицированного запроса $P = \{p_1, p_2, \ldots, p_n\}$, где n — число образцов, требуется определить в тексте Z все фрагменты, согласованные хотя бы с одним образцом из P. Это первая из рассматриваемых нами задач.

Для дальнейшего нам понадобится ввести некоторые характеристики "размытости" конкретного образца и группового запроса в целом. Пусть $p = b_1 b_2 \ldots b_m$ — частично специфицированный образец, $b_i \in \Sigma \cup D$, $1 \le i \le m$. Обозначим через $|b_i|$ число элементов в подмножестве Σ, обозначаемом символом b_i. Очевидно, что $|b_i| = 1$, если $b_i \in \Sigma$ и $|b_i| \ge 2$, если $b_i \in D$. Величину $|b_i|$ будем называть степенью неопределенности, характеризующей i-ю позицию образца. Для группового запроса в целом представляет интерес средняя степень неопределенности, приходящаяся на одну позицию:

$$r = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(\frac{1}{|p_i|} \cdot \sum_{k=1}^{|p_i|} |b_k| \right), \quad 1 \le r < |\Sigma|,$$

где n — число образцов в групповом запросе, $|p_i|$ — длина i-го образца. Разумно предполагать в дальнейшем, что

по крайней мере для коротких образцов выполняется соотношение $r \ll |\Sigma|$, иначе запрос будет слишком неопределенным. Для длинных образцов будем предполагать наличие локальной зоны (ядра) с невысокой средней степенью неопределенности.

Целочисленную величину

$$Q(p) = \prod_{k=1}^{m} |b_k|$$

назовем *степенью неопределенности* образца p.

Если $Q = 1$, образец задан точно (все $b_i \in \Sigma$). Значения Q, большие 1, характеризуют количество точно заданных образцов, согласующихся с p. Полную совокупность таких образцов назовем множеством, эквивалентным заданному частично специфицированному образцу (или просто *эквивалентным множеством*). Процесс получения из p эквивалентного ему множества $E(p)$ назовем *разверткой* частично специфицированного образца p.

Пример 1. Алфавит НК-последовательностей состоит из 4 символов: $\Sigma = \{A, G, C, T\}$. По типу азотистых оснований нуклеотиды можно разделить на пурины $R = \{A, G\}$ и пиримидины $Y = \{C, T\}$. По числу водородных связей, участвующих в комплементарном спаривании, тот же алфавит может быть разделен на группы "сильных" нуклеотидов $S = \{G, C\}$ и "слабых" $W = \{A, T\}$. Тогда для частично специфицированного образца $p = GARCWT$ эквивалентное ему множество точно заданных (или "константных") образцов имеет вид $E(p) = \{GAACAT, GAGCAT, GAACTT, GAGCTT\}$, т.е. $Q(p) = 4$.

Если число образцов в групповом запросе велико, увеличивается вероятность обнаружения взаимозависимых образцов. К простейшим типам зависимости можно отнести совпадение двух образцов, вложение одного образца в другой, пересечение двух или более образцов. Эти типы зависимости естественным образом интерпретируются на языке эквивалентных множеств, как совпадение, вложимость или пересечение соответствующих множеств. При

этом предполагается, что сравниваются образцы одинаковой длины. Если образцы имеют разную длину ($|p_1| = m_1, |p_2| = m_2$ и $m_1 < m_2$), речь может идти о вложении короткого образца в более длинный, т.е. о поиске внутри образца p_2 позиции $1 \leq i \leq m_2 - m_1 + 1$, такой что p_1 вложен в $p_2[i : i + m_1 - 1]$ (или наоборот). В аналогичной трактовке можно рассматривать задачу о пересечении короткого образца с более длинным.

Пример 2. Пусть вновь $\Sigma = \{A, G, C, T\}, D = \{R, W, Y, S\}$, где $R = \{A, G\}, W = \{A, T\}, Y = \{C, T\}, S = \{C, G\}$. Тогда для элементов группового запроса $P = \{p_1, p_2, p_3\}$, где $p_1 = GARC\overline{W}T, p_2 = ACRRRCWT, p_3 = SAAYWT$, имеют место следующие соотношения: образец p_1 вложен в суффиксный фрагмент образца p_2, начинающийся в третьей позиции; образцы p_1 и p_3 пересекаются по общим элементам из $E(p_1)$ и $E(p_3)$: $GAACAT$ и $GAACTT$; образец p_3 пересекается с фрагментом $RRRCWT$ из p_2, но не вложен в него.

Вторая из рассматриваемых нами задач состоит в выявлении взаимосвязей типа совпадений, вложений и пересечений различных образцов между элементами группового запроса. Мы будем сразу ориентироваться на наиболее общий случай — поиск пересечений, хотя частные задачи могут допускать более эффективное по сравнению с общим случаем решение.

Выявление взаимосвязей среди элементов группового запроса позволяет сократить число образцов в запросе и оптимизировать процедуру поиска образцов в тексте. Более того, возникает возможность сравнения групповых запросов, формируемых на основе различных методик. Если трактовать образцы в запросе как некоторую информативную систему признаков, характеризующих тот или иной класс символьных последовательностей, то выявление взаимосвязи на уровне образцов может означать в некоторых случаях наличие связи между соответствующими классами последовательностей.

2. Возможные подходы к решению

Простейшей задачей информационного поиска является обнаружение всех вхождений точно заданного образца $p = b_1 b_2 \ldots b_m$ $(b_i \in \Sigma,\ 1 \leq i \leq m)$ в текст $Z = a_1 a_2 \ldots a_N$ $(a_i \in \Sigma,\ 1 \leq i \leq N, m \ll N)$. Прямой алгоритм решения этой задачи в наихудшем случае имеет трудоемкость $O(N \cdot m)$. Алгоритмы, предложенные в [1,2], решают эту задачу с сублинейной в среднем трудоемкостью. Сублинейность означает, что число сравнений символов друг с другом, определяющее трудоемкость алгоритма, меньше длины текста. К примеру, при использовании алгоритма [2] для поиска образца длины 5 в английском тексте в среднем требуется просматривать лишь четвертую часть всех символов.

Для случая группового запроса $P = \{p_1, p_2, \ldots, p_n\}$ с *точно* заданными образцами p_i $(1 \leq i \leq n)$ Ахо и Корасик предложили эффективный алгоритм поиска [3], основанный на построении детерминированного конечного автомата по исходному множеству образцов (этап предобработки) и однократном "прогоне" текста через этот автомат (этап поиска). Автомат строится по принципу склеивания общих префиксных частей у образцов p_i. В каждом такте работы автомата считывается очередной символ текста, который сравнивается с очередным символом отслеживаемой в данный момент группы образцов, и в зависимости от результата сравнения осуществляется переход автомата в новое состояние.

Работа автомата описывается тремя функциями: переходов — $g(s, a)$, отказов — $f(s)$ и выхода $o(s)$, где s — текущее состояние, a — входной символ. Функция переходов указывает, в какое состояние должен перейти автомат из текущего состояния s при прочтении очередного символа текста a. Формально $g(s, a) = s'$, если существует выходящее из s ребро, помеченное символом a и связывающее состояния s и s'; в противном случае $g(s, a)$ полагается равной $f(s)$ (ситуация, обозначаемая термином "отказ", или переход по несовпадению).

Функция отказов определяет, в какое состояние переходит автомат из состояния s, если прерывается отслеживание текущего образца (ввиду его окончания или обнаружения несовпадения). Формально $f(s) = s''$, где цепочка символов, связывающая состояния 0 и s'', есть максимальный (по всем образцам p_i) префикс, являющийся одновременно суффиксом цепочки, связывающей состояния 0 и s. Использование функции отказов позволяет избежать возвратов назад по тексту, что обеспечивает линейность процедуры поиска по N.

Функция выхода $o(s)$ отличается от нуля лишь для состояний, являющихся конечными для тех или иных образцов. Значение $o(s)$ задается списком — перечислением соответствующих образцов.

Суммарная трудоемкость алгоритма по оценкам авторов составляет $O(N+\sum_i |p_i|)$ (вместо $O(N \cdot n)$ при n-кратном использовании алгоритмов типа [1,2]). Первый член суммы — трудоемкость поиска по тексту с помощью построенного автомата, второй — трудоемкость предобработки образцов (формирование функций $g(s,a), f(s), o(s)$). Последняя оценка не совсем корректна, поскольку в ней опущен мультипликативный член, соответствующий мощности алфавита, которая может быть очень большой. Более того, при большом числе образцов и большом размере алфавита очень существенным лимитирующим фактором становятся затраты оперативной памяти, определяемые в наихудшем случае произведением $|\Sigma| \cdot \sum_i |p_i|$.

Очевидно, что описанная конструкция может быть успешно использована для поиска по единичному частично специфицированному образцу или ограниченному их количеству, если предварительно представить их в виде эквивалентных множеств константных образцов. Трудоемкость алгоритма составит в наихудшем случае $O(N+|\Sigma| \cdot \sum_i |p_i| \cdot Q(p_i))$, затраты памяти — $O(|\Sigma| \cdot \sum_i |p_i| Q(p_i))$. Критичным для приведенных оценок является параметр

$Q(p_i)$ — степень неопределенности образца, которая *экспоненциальным* образом растет в зависимости от числа недоопределенных позиций в образце. "Плохо определенными" в этом смысле являются образцы с большим числом нерегулярно расположенных элементов типа X. Даже единичный "плохо определенный" образец может потребовать при развертке нерационально больших затрат памяти для построения автомата [4]. Эти проблемы многократно усугубляются в случае группового частично специфицированного запроса.

Прямой алгоритм поиска образца с точно заданными элементами и элементами типа X состоит в сканировании текста окном ширины $|p|$ и анализе в пределах окна лишь тех позиций текста, которые соответствуют точно заданным позициям образца (поиск по "маске"). В случае совпадения содержимого позиций, выделяемых маской в образце и в соответствующем фрагменте текста, образец считается найденным (нет нужды проверять позиции с элементами типа X). Трудоемкость алгоритма в наихудшем случае $O(N \cdot l)$, где $l < |p|$ — число позиций, выделяемых маской. В случае частично специфицированного образца общего вида (с элементами агрегированного алфавита) совпадение символов в пределах маски не избавляет от проверки оставшихся позиций, что увеличивает трудоемкость в наихудшем случае до $O(N \cdot |p|)$. При этом предполагается, что проверка принадлежности элементов текста определенным подмножеством из Σ осуществляется с константной трудоемкостью (например, эта информация может быть затабулирована).

Описанный подход находит свое дальнейшее развитие в [5] и в принципе напоминает метод "квантования" точно заданного образца, изложенный в [6], однако выбор маски не оптимизируется (как в [6]), а диктуется расположением точно заданных позиций. Применительно к групповому частично специфицированному запросу этот метод имеет ограниченное применение, поскольку удается запараллелить лишь поиск таких подгрупп образцов, которые характеризуются одинаковым расположением точ-

но заданных символов. Аналогичные соображения применимы и к методам, основанным на хешировании образцов по точно заданным позициям [7].

Изложенные выше возможные подходы к решению задачи поиска по групповому частично специфицированному запросу показывают, что для запросов большой мощности удовлетворительных решений в общем случае не существует. Это вынуждает проводить разработку специальных аппаратно-программных комплексов с параллельной архитектурой [8].

Предлагаемое нами решение ориентировано на персональные компьютеры среднего класса. Суть его состоит в том, чтобы воспользоваться наиболее привлекательной из известных конструкций для группового поиска — автоматом Ахо — Корасик, обойдя трудности, связанные с экспоненциальным нарастанием затрат по памяти, путем упаковывания в автомат не самих образцов, а некоторых их фрагментов ("ядер"). Длина ядра l, как правило, значительно меньше длины образца. За счет этого достигается двойной эффект: 1) существенно уменьшается число состояний автомата, т.е. экономится память; 2) не успевает себя проявить эффект экспоненциального нарастания числа элементов эквивалентного множества, поскольку в рассматриваемом варианте осуществляется развертка не образцов, а ядер, выбор которых предварительно оптимизируется. Идея использования ядер восходит к алгоритмам отыскания несовершенных повторов в тексте [9].

Процедура поиска становится двухэтапной: найденное в тексте ядро требуется расширить до размеров образца (в наихудшем случае), чтобы убедиться, что соответствие между текстом и образцом сохраняется по всей длине последнего. Заметим однако, что процедура расширения не требует больших затрат, поскольку чаще всего заканчивается уже на первом шаге (обнаруживается несоответствие).

3. Алгоритм 1 поиска по групповому частично специфицированному запросу (построение автомата по образцам)

1. Построение автомата.

ШАГ 1. Выбор размера ядра. Выбираем размер ядра l, исходя из мощности алфавита Σ, степени неопределенности образцов в запросе и имеющихся ресурсов оперативной памяти (значение $l = 3$-4 наиболее приемлемо для алфавитов средней мощности ($|\Sigma| = 20$-50)).

ШАГ 2. Выбор ядра. В каждом образце $p_i = b_1 b_2 \dots b_m$, $1 \le i \le n$, выбираем ядро z_i длины l в виде фрагмента $b_k b_{k+1} \dots b_{k+l-1}$ с минимальным значением $Q(z_i)$. Положение оптимального ядра в образце p_i (номер позиции k) фиксируем в i-й ячейке массива I размерности n.

ШАГ 3. Развертка ядер. Осуществляем развертку каждого из ядер z_i, $1 \le i \le n$, совмещая ее с построением автомата по множествам $E(z_i)$. На выходе получаем функции $g(s, a), f(s)$ и $o(s)$ для множества ядер.

2. Поиск по тексту.

ШАГ 4. Обнаружение ядер. Пусть Z — анализируемый текст. Пропускаем его через автомат, рассматривая каждый символ в качестве очередного управляющего воздействия. Фиксируем символы текста, переводящие автомат в одно из конечных состояний. Каждый из них является последним элементом ядра, характеризующего совокупность образцов, связанных с соответствующим конечным состоянием.

ШАГ 5. Расширение ядер. Рассматриваем последовательно ядра, выделенные в тексте Z на шаге 4. Пусть очередному ядру α длины l соответствует конечное состояние s^*. Проверяем, согласуются ли лево- и правосторонние расширения ядра α в тексте Z с образцами, представленными в списке $o(s^*)$. Для этого выравниваем каждый образец с соответствующим фрагментом текста по общему ядру α, используя информацию, содержащуюся в массиве I (см. ниже), и проверяем вложимость элементов текста из окрестностей ядра α в подмножества алфавита

Σ, фигурирующие в образце. Проверка ведется до первого несогласования либо до исчерпания всех элементов образца ($|p_i| - l$ проверок в наихудшем случае). Если все проверки прошли успешно, фиксируем обнаружение образца p в тексте Z.

				ядро α		
Z:	...	a_{k-2}	a_{k-1}	$a_k\ a_{k+1}...a_{k+l-1}$	a_{k+l}	...
p:		b_1	b_2	$b_3\ b_4...b_{3+l-1}$	b_{3+l}	
				z_i		
		$\vdots$	$\vdots$		$\vdots$	
$I(i)=3$		?	$a_{k-1} \in b_2$?	$\alpha \in E(z_i)$	$a_{k+l} \in b_{3+l}$?	

3. Оценка суммарной трудоемкости. Шаг 2 алгоритма реализуется сканированием каждого образца окном ширины l и вычислением значения Q для каждого окна. Учитывая зацепленность соседних окон, Q можно не вычислять каждый раз заново, а лишь корректировать по простой рекуррентной схеме. Трудоемкость шага — $O(\sum_i |p_i|)$.

На шаге 3 каждое из частично специфицированных ядер z_i, $1 \le i \le n$, заменяется эквивалентным множеством из $Q(z_i)$ точно заданных ядер длины l. Полное количество образцов, по которым строится автомат, есть $\sum_i Q(z_i) = n \cdot \bar{Q}_l$, где $\bar{Q}_l$ — средняя степень неопределенности, приходящаяся на одно ядро длины l. Трудоемкость построения автомата по ядрам пропорциональна их суммарной длине $n \cdot l \cdot \bar{Q}_l$.

Средняя суммарная трудоемкость шагов 1 и 2 (трудоемкость предобработки) составляет $O(\sum_i |p_i| + n \cdot l \cdot \bar{Q}_l)$. Вес второго слагаемого определяется значением параметра $\bar{Q}_l$. Мы предполагаем и имеем тому многочисленные экспериментальные подтверждения, что для реальных групповых запросов с большим числом образцов значение $\bar{Q}_l$ невелико для относительно небольших, но уже гарантирующих эффективность поиска значений l ($l = 3, 4$). Неявно это означает, что образцы в групповом запросе

уже построены по *ядерному* принципу в том смысле, что большая их часть содержит относительно "консервативный" (т.е. с малой степенью неопределенности) участок хотя бы небольшой длины.

Шаг 4 соответствует стандартному поиску по тексту с использованием конечного автомата. Трудоемкость его составляет $O(N)$, где N — длина текста.

Трудоемкость процедуры расширения (шаг 5) определяется числом выявляемых в тексте ядер, длиной списка образцов, идентифицируемых каждым ядром, и числом сопоставляемых символов при каждом расширении. Естественно считать, что для группового запроса большой мощности (а именно такой случай нас интересует) в ядерном автомате будут представлены почти все потенциально возможные l-символьные цепочки, где l — размер ядра. Это означает, что почти каждая из $(N-l+1)$ l-грамм текста Z будет идентифицирована как ядро. Следовательно, число выявляемых в тексте ядер по порядку сопоставимо с длиной текста N.

В ядерном автомате $n\bar{Q}_l$ ядер распределены по $|\Sigma|^l$ конечным состояниям, причем почти все состояния непусты. Это означает, что каждое ядро, выделяемое в тексте, в среднем будет сравниваться при расширении с $\frac{n \cdot \bar{Q}_l}{|\Sigma|^l}$ образцами.

Сравнение цепочки текста с образцом p в наихудшем случае требует $(|p| - l)$ операций. Трудоемкость в среднем близка к константной, если число элементов типа X в образце не слишком велико. Процедуру сравнения можно рассматривать как последовательность испытаний Бернулли, где успеху соответствует несогласованность символов в идентичных позициях, что эквивалентно прерыванию процесса сопоставления (образец не найден), а неудаче — согласованность (сопоставление продолжается). Нас интересует математическое ожидание числа неудач Y, предшествующих первому успеху. Известно [10], что случайная величина Y имеет геометрическое распределение $P(Y = k) = q^k \cdot p, \quad k = 0, 1, 2, \ldots$, где p и q соответственно

вероятности успеха и неудачи в последовательности испытаний Бернулли. Тогда

$$E(Y) = \sum_k k \cdot P(Y = k) = qp(1 + 2q + 3q + \ldots) = qp(1-q)^{-2} = \frac{q}{p},$$

т.е. для обнаружения первого несогласования требуется в среднем $1 + \frac{q}{p}$ сравнений. Если сравниваются элементы алфавита Σ, то вероятность неудачи $q = \sum_{a \in \Sigma} p_a^2$ (вероятность совпадения двух случайно выбираемых символов), соответственно $p = 1 - q$. Если, к примеру, $p_a = \frac{1}{|\Sigma|}$ для всех a, то $q = \frac{1}{|\Sigma|}$, $p = \frac{|\Sigma| - 1}{|\Sigma|}$ и число сравнений в среднем составит $1 + \frac{1}{|\Sigma| - 1}$, что близко к 1 уже для небольших алфавитов.

Если сравнивается элемент алфавита Σ с элементом алфавита D, то вероятность неудачи увеличивается пропорционально числу элементов в агрегированном множестве из D (соответственно p уменьшается). Выше мы предположили, что среднее число элементов r, приходящееся на одну позицию в разумно организованном групповом частично специфицированном запросе, существенно меньше размера алфавита Σ. В этом случае вероятность успеха p все еще будет превалировать над q, т.е. среднее число сравнений не превысит 2.

Таким образом, трудоемкость поиска (шаги 4 и 5) составляет в среднем $O\left(N + \frac{N \cdot n\bar{Q}_l}{|\Sigma|^l}\right)$, а суммарная трудоемкость поиска и предобработки — $O\left[N\left(1 + \frac{n\bar{Q}_l}{|\Sigma|^l}\right) + \sum_i |p_i| + n\bar{Q}_l \cdot l\right]$. Последнее выражение можно представить в виде $O\left(\left[N + \sum_i |p_i|\right] + n\bar{Q}_l \cdot \left(\frac{N}{|\Sigma|^l} + l\right)\right)$, где член в квадратных скобках — трудоемкость стандартного алгоритма Ахо — Ко-

расик, а второй член — добавка, обусловленная частичной специфицированностью запроса.

4. Алгоритм 2 поиска по групповому частично специфицированному запросу (построение автомата по тексту)

Анализ трудоемкости алгоритма 1 показывает, что при относительно небольших длинах текста N поиск осуществляется быстро, а основное время уходит на построение автомата по большому числу образцов. В оценке трудоемкости превалирует член $n \cdot \bar{Q}_l \cdot l$. Можно поменять местами текст и образцы: построить автомат по всевозможным фрагментам длины l исходного текста, а образцы пропускать через автомат, рассматривая их в качестве управляющих последовательностей. При тех же условиях (N — относительно невелико, n — велико) будем иметь малые затраты на построение автомата и существенно большие — на прогон образцов через автомат. Последние однако оказываются меньшими по величине, чем $n \cdot \bar{Q}_l \cdot l$ в определенном диапазоне значений N. Аналогичный вывод справедлив и в отношении затрат памяти. Это обусловило разработку описанного ниже алгоритма. В его основе также лежит "ядерный" принцип.

1. Построение автомата.

ШАГ 1. Выбор размера ядра. Выбираем значение l из тех же соображений, что и в предыдущем алгоритме.

ШАГ 2. Построение автомата по тексту. Скользящим окном размера l выделяем в тексте фрагменты длины l ("образцы"), сдвигаясь каждый раз на один символ. Начальная позиция фрагмента интерпретируется как номер образца. Всего таким образом выделяем $(N - l + 1)$ образцов, среди которых могут быть повторяющиеся, но имеющие при этом разные номера. Строим по этим образцам автомат, игнорируя функцию отказов (она не нужна) и формируя функцию выхода в виде списков вхождений соответствующих l-грамм в текст.

2. Поиск с помощью автомата.

ШАГ 3. Формирование поисковых запросов. В каждом из частично специфицированных образцов p_i, $1 \le i \le n$, выделяем ядро z_i длины l с минимальной степенью неопределенности $Q(z_i)$. Формируем множество $E(z_i)$ константных ядер, эквивалентно представляющих (в смысле полноты поиска) частично специфицированное ядро z_i.

ШАГ 4. Обнаружение ядер в тексте. Цепочки константных символов, фигурирующие в $E(z_i)$, $1 \le i \le n$, последовательно подаем на вход автомата. Если соответствующая цепочка представлена в автомате списком своих вхождений в текст, осуществляем расширение всех вхождений цепочки с целью проверки соответствия i-му образцу (см. шаг 5). Если цепочка не обнаруживается в автомате (отказ), переходим к анализу следующей цепочки.

ШАГ 5. Расширение ядер по тексту. Реализуется аналогично шагу 5 алгоритма 1 с единственной разницей: в данном случае различные фрагменты текста (с одинаковым ядром) проверяются на согласованность с единственным образцом, тогда как в алгоритме 1 единственный фрагмент текста проверяется на согласованность с разными образцами, пересекающимися по данному ядру.

3. Оценка трудоемкости алгоритма 2. Рассмотрим два крайних случая: 1) $N < |\Sigma|^l$ и 2) $N \gg |\Sigma|^l$. Первый случай, как правило, соответствует поиску в одиночном тексте, второй — поиску по базе данных. Порядок трудоемкости в первом случае составляет $N \cdot l + \sum_i |p_i| + n \cdot \bar{Q}_l$. Здесь первый член — трудоемкость построения автомата по тексту (сумма длин образцов), второй — трудоемкость выбора оптимальных ядер в образцах, а третий — трудоемкость поиска, пропорциональная числу обращений к автомату (поиск по автомату считаем константным, так как при выполнении условия 1 он, в большинстве случаев, будет заканчиваться отказом). При большом n доминирующим в оценке трудоемкости является член $n \cdot \bar{Q}_l$ (вместо

$n \cdot \bar{Q}_l \cdot l$ в алгоритме 1), что и дает алгоритму 2 выигрыш во времени при не слишком больших длинах текста.

Порядок трудоемкости во втором случае составляет $N \cdot l + \sum_i |p_i| + n \cdot \bar{Q}_l \cdot l + \frac{n \cdot \bar{Q}_l \cdot N}{|\Sigma|^l}$. Здесь первые два члена имеют тот же смысл, что и выше, третий член — время прохождения по автомату (при выполнении условия 2 почти все из $|\Sigma|^l$ возможных путей будут реализованы в автомате и отказов почти не будет), четвертый член — оценка средней трудоемкости расширений. Различие с алгоритмом 1 будет лишь в первом члене (не в пользу алгоритма 2). Главный же ограничивающий фактор алгоритма 2 — непрерывный рост памяти, затрачиваемой на построение автомата при увеличении N. Эта память расходуется в основном для представления функции выхода. Заметим, что в алгоритме 1 затраты памяти под автомат лимитировались не длиной текста, а числом образцов, что, как правило, менее ограничительно.

5. Алгоритм выявления совпадающих, вложенных и пересекающихся образцов

В принципе, данная задача похожа на предыдущую, но усложняется этап предобработки и редуцируется этап поиска (отсутствует входной текст). Как следствие последнего отпадает необходимость в вычислении функции отказов $f(s)$.

Два обстоятельства усугубляют дефицит оперативной памяти.

1. Поскольку речь идет не только о сравнении образцов одинаковой длины, но и о выявлении вложений более коротких образцов в более длинные, осуществляется формальное приведение всех образцов к одной длине. При этом количество образцов в групповом запросе возрастает.

2. Переход к ядрам теперь производится не с целью минимизации информации об исходном образце, заносимой в автомат, а с целью выявления общих фрагментов

у разных образцов. Такие ядра должны располагаться во всех образцах в одинаковых позициях, чтобы гарантировать обнаружение всех взаимосвязанных образцов. Это увеличивает в среднем степень их неопределенности $\bar{Q}_l$.

Указанные обстоятельства заставляют сделать дополнительный шаг с целью минимизации затрат памяти. А именно, осуществляется разбиение исходного множества образцов на пересекающиеся в общем случае подмножества, такие что образцы из разных подмножеств не подлежат дальнейшему сопоставлению друг с другом, т.е. совпадения, вложения и пересечения образцов могут иметь место только внутри каждого подмножества, но не между ними. Для каждого из выделенных подмножеств строится автомат ядерного типа, конечные состояния которого содержат уже относительно короткие списки образцов, подлежащих дальнейшему сопоставлению друг с другом (этап расширения ядер).

1. Описание алгоритма.

ШАГ 1. Приведение всех образцов к одинаковой длине. Вычисляем гистограмму длин образцов в групповом запросе и определяем значение длины L, которому соответствует главный пик гистограммы. Приводим все образцы к указанной длине следующим образом. Если $|p_i| > L$, заменяем образец p_i совокупностью из $|p_i| - L + 1$ образцов, выделяемых из p_i скользящим окном длины L. Если $|p_i| < L$, заменяем образец p_i совокупностью из $L - |p_i| + 1$ образцов длины L, содержащих p_i соответственно в позициях $1, 2, \ldots, L - |p_i| + 1$. Позиции, дополняющие p_i до образца длины L, заполняем фиктивными символами Y, имеющими статус элементов типа X. Можно показать, что приведение всех образцов к длине L минимизирует возрастание суммарной длины группового запроса и гарантирует в то же время обнаружение всех вхождений коротких образцов в более длинные (аналогично для пересечений).

ШАГ 2. Лексикографическое разбиение множества образцов на подмножества. После выполнения шага 1 имеем множество образцов $P' = \{p_1, p_2, \ldots, p_{n'}\}$, где $p_i =$

$= b_1^i b_2^i \ldots b_L^i$, $|p_i| = L$, $1 \le i \le n'$ и $n' > n$ (n — число образцов в исходном запросе). Разбиение множества P' на подмножества будем производить в соответствии с содержимым одной и той же позиции в каждом образце. Номер позиции k^* выбираем из компромиссных соображений: с одной стороны, желательно иметь в позиции k^* минимальное число элементов типа Y (в общем случае желательно минимизировать по k значение $R_k = \sum_{i=1}^{n'} |b_k^i|$, $1 \le k \le L$); с другой стороны, распределение элементов алфавита Σ и D в позиции k^* должно быть максимально равномерным, чтобы обеспечить наилучшее приближение к равномощному разбиению. Разбиваем множество P' на $|\Sigma|$ пересекающихся подмножеств $P'_j = \{p_i : b_{k^*}^i$ содержит в себе j-й элемент алфавита $\Sigma\}$. Таким образом, множества P'_j, $1 \le j \le |\Sigma|$, пересекаются лишь на тех образцах, у которых $|b_{k^*}^i| > 1$. К примеру, если i-й образец представлен в k^*-позиции двухэлементным подмножеством $\{\alpha, \beta\}$ из Σ, он попадает одновременно в P'_α и P'_β. При этом может существовать другой образец с номером i', который также попадет в P'_α и P'_β и пересечется с i-м образцом. Этот факт будет выявлен в дальнейшем как при анализе множества P'_α, так и при анализе P'_β. Подобное дублирование можно рассматривать как плату за возможность избежать перекрестного сравнения образцов из P'_α и P'_β.

ШАГ 3. Построение автоматов на ядрах. Разбиение множества P' на подмножества P'_j сводит исходную задачу к серии последовательно решаемых задач меньшей размерности. Шаги 3 и 4 реализуются для каждого из подмножеств P'_j, $1 \le j \le |\Sigma|$. Задаем размер ядра l, руководствуясь теми же соображениями, что и в предыдущих алгоритмах, и в зависимости от значения k^* фиксируем положение ядра слева или справа от позиции k^* единым для всех образцов способом. Пусть, для определенности, ядро расположено в позициях $k^*+1, k^*+2, \ldots, k^*+l$ каждого образца. Осуществляем развертку каждого из частично специфицированных ядер и по полученным эквивалент-

ным множествам строим автомат, игнорируя вычисление функции отказов. Если анализируется подмножество P'_j, соответствующее j-му элементу алфавита, то каждое конечное состояние автомата представлено списком образцов, содержащих (среди прочих) j-й элемент в позиции k^* и обладающих общей константной цепочкой, расположенной в позициях $k^*+1,\ldots,k^*+l$.

ШАГ 4. Расширение образцов. Рассматриваем поочередно конечные состояния автомата, содержащие не менее двух образцов. Пусть текущее конечное состояние представлено списком из t образцов. Чтобы избежать сопоставления каждого с каждым, организуем массив размерности $|\Sigma|$ и осуществляем анализ t образцов по первой из позиций, не использовавшихся при сортировке и последующем построении автомата (это позиции с номерами $k^* > k > k^*+l$). Элемент массива с номером j, $1 \le j \le |\Sigma|$, содержит список тех из t образцов, которые допускают появление j-го символа алфавита Σ в анализируемой k-й позиции. Все списки формируются путем однократного просмотра t подмножеств, указанных в k-й позиции. Дальнейшему анализу с привлечением очередной позиции подвергаются лишь списки, содержащие не менее 2 образцов. Процесс расширения заканчивается, как только на очередном шаге все списки оказываются пустыми или одноэлементными, либо по исчерпании всех $L-l-1$ проверяемых позиций.

ШАГ 5. Устранение дублирований. При описании шага 2 отмечалось, что вследствие наличия общих образцов во множествах P'_α и $P'_\beta (\alpha, \beta \in \Sigma, \alpha \ne \beta)$ могут возникать ситуации, когда такие образцы пересекаются, и эти случаи фиксируются независимо как при анализе множества P'_α, так и при анализе P'_β. При составлении сводного списка пересечений для всего группового запроса такого рода дублирования должны быть устранены. Это эквивалентно обнаружению одинаковых пар индексов (i,k) во множестве номеров попарно пересекающихся образцов, являющихся общими хотя бы для двух множеств P'_j. Задача решается целочисленной сортировкой по первому индек-

су i с последующей сортировкой по второму индексу k групп объектов с одинаковым значением i.

2. Оценка трудоемкости. Пусть длины образцов в исходном групповом запросе $P = \{p_1, p_2, \ldots, p_n\}$ меняются в диапазоне от l_{min} до l_{max}. Обозначим через n_s число образцов с длиной s. Очевидно, что $n = \sum_{s=l_{min}}^{l_{max}} n_s$. В результате выполнения шага 1 число образцов в запросе возрастает до

$$n' = n_L + \sum_{k=1}^{l_{max}-L} (k+1) \cdot n_{L+k} + \sum_{k=1}^{L-l_{min}} (k+1) \cdot n_{L-k},$$

где L — значение s, на котором достигается максимум величины n_s. Вовсе необязательно формировать образцы в явном виде, что потребует значительных затрат памяти. В сжатой форме эти образцы представлены в исходном запросе и при реализации шагов 2 и 3 необходимая информация о новых образцах извлекается алгоритмическим путем (см. описание шага 1) из исходных образцов. Таким образом шаг 1 фактически реализуется параллельно с шагами 2 и 3, хотя для лучшего понимания в описании алгоритма он фигурирует отдельно. Вычисление гистограммы длин потребует затрат $O(\sum_i |p_i|)$.

Перебора по всем позициям на шаге 2 в принципе можно избежать. Зная l_{min}, L, l_{max}, параметры n_s ($l_{min} \le s \le l_{max}$) и алгоритм конструирования новых образцов, можно выбрать значение k^* из соображений минимизации числа элементов типа Y в искомой позиции. Если для данной позиции разбиение по элементам алфавита не будет слишком неравномерным, можно этим и ограничиться. Трудоемкость шага 2 тогда будет определяться суммарным числом элементов в подмножествах, представленных в позиции k^*. По порядку это сопоставимо с произведением $n' \cdot r$, где r — средняя степень неопределенности, приходящаяся на одну позицию в новом групповом запросе ($1 < r < |\Sigma|$).

Шаг 3 (построение автомата по частично специфицированным ядрам в каждом из подмножеств P'_j) имеет трудоемкость пропорциональную суммарной длине упаковываемых в автомат константных ядер. Вследствие дублирования некоторых образцов в разных подмножествах P'_j количество развертываемых частично специфицированных ядер превосходит суммарное число образцов n' примерно в r раз. Число генерируемых при развертке константных ядер, соответственно, возрастает еще в $\bar{Q}_l$ раз. Таким образом суммарная трудоемкость шага 3 составляет $O(n' \cdot r \cdot \bar{Q}_l \cdot l)$.

При оценке трудоемкости шага 4 будем предполагать, что каждое из подмножеств P'_j, $1 \leq j \leq |\Sigma|$, в среднем содержит порядка $\frac{n' \cdot r}{|\Sigma|}$ частично специфицированных образцов, из которых при развертке выделяется порядка $\frac{n' \cdot r\bar{Q}_l}{|\Sigma|}$ константных ядер длины l. При равномерном их распределении по конечным состояниям автомата в среднем с каждым конечным состоянием будет связано порядка $\bar{t} = \frac{n' \cdot r \cdot \bar{Q}_l}{|\Sigma|^{l+1}}$ образцов. Расширение этих образцов на один символ потребует порядка $r\bar{t}$ условных операций (просмотр всех элементов каждого из $\bar{t}$ агрегированных подмножеств). При выполнении условия $r \ll |\Sigma|$ несогласованность образцов в подавляющем большинстве случаев выявляется за 1-2 шага. В наихудшем случае (обнаружение пересечения) потребуется проверка всех $L-l-1$ позиций. Суммарные затраты по всем автоматам составят в среднем порядка $\frac{r^2 \cdot n' \cdot \bar{Q}_l}{|\Sigma|^l}$ условных операций.

Трудоемкость шага 5 пропорциональна числу пар (групп) пересекающихся образцов, выявляемых на шагах 3, 4, и не может превышать их по порядку сложности.

Основной вклад в суммарную трудоемкость вносит шаг 3 алгоритма, явно превалирующий над шагами 1,2 и 4 (к примеру, отношение трудоемкости шага 3 к шагу 4

составляет $\frac{l \cdot |\Sigma|^l}{r}$, что много больше 1, поскольку $r < |\Sigma|$). Таким образом, порядок трудоемкости поиска пересекающихся образцов в среднем составляет $n'r \cdot \bar{Q}_l \cdot l$. Наихудший случай соответствует ситуации, когда любая пара образцов пересекается (например, элемент $\alpha \in \Sigma$ присутствует в каждой позиции каждого образца). При этом не выполняется предположение о равномощности разбиения и в конечном состоянии автомата, соответствующем ядру α^l, будет фигурировать полный список образцов, который не сократится и при последующем расширении, так что трудоемкость шага 4 составит $O(n' \cdot r^2 \cdot (L-l-1))$ и он будет сопоставим с шагом 3, а иногда и превалировать над ним. Подобные патологические случаи, как правило, не представляют практического интереса.

6. Замечания о практической реализации

1. Хотя наши алгоритмы ориентированы преимущественно на образцы с элементами агрегированного алфавита, допускается и появление элементов типа X, в количестве, не приводящем к нарушению условия $r \ll |\Sigma|$. При оптимальном выборе ядра, как правило, удается избежать наличия в нем элементов типа X. Отдельные исключения не приводят к существенному увеличению трудоемкости. Для общего случая нами разработана схема автомата, допускающего использование элементов типа X наряду с символами исходного алфавита (речь идет уже о недетерминированном автомате). Этот автомат требует специального рассмотрения, выходящего за рамки данной работы.

2. В автомате, построенном по ядрам относительно небольшой длины, реализуются почти все из $|\Sigma|^l$ конечных состояний (такова специфика группового запроса большой мощности). В подобной ситуации конечные состояния составляют большую часть всех состояний автомата. Существенной экономии памяти можно добиться, задавая функцию переходов (но не функцию отказов) лишь для внутренних состояний. При этом необходимо, чтобы

номера внутренних состояний предшествовали номерам конечных. Соответствующая идея была изложена в [11] и она дает наиболее ощутимый эффект именно в нашем случае.

3. Учет разнообразных факторов, связанных с изменением размера ядра, достаточно сложен. К примеру, изменение размера ядра приводит к изменению трудоемкости построения автомата и затрат памяти под функции переходов и выходов. Изменяется также трудоемкость поиска по тексту (обнаруживается больше или меньше ядер). К этому следует добавить, что вследствие существующей иерархии "памятей" в компьютере (статическая, динамическая, расширенная, дисковая) переход на следующую ступень иерархии сопровождается замедлением вычислений, иногда весьма существенным. Поэтому оптимизацию размера ядра целесообразно проводить экспериментальным путем.

4. Изложенные выше соображения о целесообразности применения алгоритма 2 в режиме обработки одиночных текстов относительно небольшой длины остаются в силе даже тогда, когда автомат по групповому частично специфицированному запросу уже построен и хранится в виде файлов на диске. Это объясняется тем, что перенос автомата в оперативную память требует времени, сопоставимого с временем его построения. Поэтому любая возможность компактизации автомата и самих образцов заслуживает внимания.

Способ построения автомата по тексту представляет интерес еще с одной точки зрения. А именно, в случае когда поисковая ситуация является динамической, т.е. групповые запросы непрерывно сменяют друг друга, а текст, пусть даже довольно длинный, не меняется, выгоднее может оказаться ориентация на текст.

7. Апробация алгоритмов

Для апробации использовался реальный групповой частично специфицированный запрос большой мощности

(порядка 23000 образцов), сформированный на основе анализа большого количества протеиновых семейств [12]. Каждый образец представлял собой консенсусное описание какой-либо информативной зоны в соответствующем семействе. Длины образцов варьировались от 6 до 10 элементов. Алфавит Σ состоял из 20 символов (аминокислот), алфавит D содержал порядка 2300 элементов (различных подмножеств Σ). Для примера ниже приведен один из образцов: $N-[DE]-[FGHY]-[CN]-[DKLN]-[CRT]-B-C$. Здесь черточки разделяют позиции образца, все символы — элементы алфавита Σ, агрегированные подмножества заданы перечислением своих элементов в квадратных скобках. Средняя степень неопределенности, приходящаяся на одну позицию образца $r = \frac{18}{8} = 2,25$, степнь неопределенности всего образца $Q = 2 \cdot 4 \cdot 2 \cdot 4 \cdot 3 = 192$.

В целом для рассматриваемого группового запроса средняя степень неопределенности на один образец составила $\bar{Q}(p) \approx 27$. Оптимизация размера ядра дала близкие результаты по суммарному времени предобработки и поиска для $l = 3$ и 4 (чуть лучше для $l = 3$). При оптимизированном выборе положения ядра средняя степень неопределенности, приходящаяся на одно ядро, составила $\bar{Q}_l \approx 3,3$, т.е. при трехкратном (примерно) уменьшении размера ядра по сравнению со средней длиной образцов степень неопределенности снижается более чем в восемь раз, что является существенным доводом в пользу ядерного алгоритма.

Процедура расширения при поиске по тексту работает очень быстро: среднее число проверок до получения результата (положительного или отрицательного) составило $\approx 1,2$. При построенном по образцам автомате (алгоритм 1) сам поиск на персональных компьютерах IBM PC с 486-м (и даже 386-м) процессором идет практически в реальном масштабе времени.

Тестирование второго алгоритма показало, что для длин текстов до 20-30 тысяч символов он обладает ощутимым выигрышем во времени по сравнению с первым алгоритмом. Такого диапазона длин вполне достаточно

для режима обработки одиночных текстов, поскольку самые длинные аминокислотные последовательности обычно не превышают 10 тысяч символов.

При тестировании третьего алгоритма было выявлено 543 различных образцов, обнаруживших взаимосвязи типа "совпадение", "вложение" или "пересечение". Общее число пар образцов, находящихся друг с другом в указанных отношениях, составило 443, т.е. некоторые образцы фигурировали в этих парах многократно.

Указанные 443 пары были разбиты на три группы. В *первую* вошли пары образцов, принадлежащих одному семейству белков. Они свидетельствуют о наличии повторяющихся фрагментов (возможно, зашумленных) в аминокислотных последовательностях данного семейства. Во *вторую* группу вошли образцы, связывающие заведомо родственные семейства. О силе связи двух семейств свидетельствует количество пар пересекающихся образцов. В *третью* группу (наиболее интересную) вошли образцы, связывающие семейства, формально не относимые к родственным. Выявленные взаимосвязи могут заставить пересмотреть вопрос о предполагаемой независимости семейств.

З а к л ю ч е н и е

Рассмотрена задача поиска в текстовых базах данных по групповому частично специфицированному запросу большой мощности. Предполагается, что элементы поискового образца могут быть заданы с точностью до принадлежности к произвольному подмножеству исходного алфавита. В традиционных постановках обычно фигурирует единичный частично специфицированный запрос, некоторые элементы которого заданы точно, а другие неопределены, в том смысле, что несущественно, какие символы могут стоять в данной позиции.

Прямое сведение данной задачи к известному алгоритму поиска по детерминированному групповому запросу не проходит из-за экспоненциально возрастающих (в

зависимости от числа частично специфицированных позиций) затрат памяти и времени. При разумных и не слишком ограничительных предположениях о степени неопределенности ("размытости") частично специфицированных образцов предложена двухэтапная схема решения задачи. Первый этап сводится к выявлению опорных ядер в каждом из образцов (коротких фрагментов с минимальной степенью неопределенности). Найденные ядра заменяются эквивалентным (с точки зрения полноты поиска) множеством константных ядер, по которым строится распознающий автомат. На этапе поиска производится обнаружение константных ядер в анализируемом тексте и их расширение до размеров образца с целью проверки согласованности символов текста с элементами образца, не вошедшими в состав ядра. Использование "ядерного" принципа способствует кардинальному снижению трудоемкости по сравнению с прямым методом поиска.

Показано, что для не слишком длинных текстов можно построить более эффективный алгоритм поиска, если текст и образцы поменять местами. Это означает, что автомат ядерного типа строится уже не по образцам, а по отдельным фрагментам текста, а образцы играют роль управляющих воздействий. Данный алгоритм предназначен для поиска в одиночных текстах.

В сходном ключе рассмотрена вспомогательная задача, актуальная для запросов большой мощности. Предложен алгоритм выявления совпадений, вложений и пересечений образцов в групповом запросе. Это начальный шаг к выявлению структуры запросов. Он представляет интерес не только для оптмизации процедуры поиска, но и для выявления взаимосвязей в механизме формирования группоого запроса.

Все алгоритмы апробированы на реальных групповых запросах, возникающих при анализе генетических текстов (ДНК- и аминокислотных последовательностей). На персональных компьютерах среднего класса обработка идет практически в реальном масштабе времени.

Литература

1. KNUTH D.E., MORRIS J.H., PRATT V.R. Fast pattern matching in strings //J.Computing. — 1977. — Vol.6. — P. 323-350.

2. BOYER R.S., MOORE J.S. A fast string searching algorithm //Communications of the ACM. — 1977. — Vol. 20. — P.767-772.

3. AHO A.V., CORASICK M.J. Efficient string matching: an aid to bibliographic search //Communications of the ACM. — 1975. — Vol. 18, № 6. — P. 333-340.

4. ABARBANEL R.M., WIENEKE P.R., MANSFIELD E., JAFFE D.A. and BRUTLAG D.I. Rapid searches for complex patterns in biological molecules //Nucleic Acids Research. — 1984. — Vol. 12, № 1. — P.263-280.

5. PINTER Ron Y. Efficient string matching with don't-care patterns //Combinatorial algorithms on words (ed. by A.Apostolico and Z.Galil). — Springer Verlag, 1985. — NATO ASI Series, Vol. F12. — P. 11-29.

6. VISHKIN U. Deterministic sampling — a new technique for fast pattern matching //SIAM J. Comput. — 1991. — Vol. 20, № 1. — P.22-40.

7. RIVEST R.L. Partial-match retrieval algorithms //SIAM J.Comput. — 1976. — Vol. 5, № 1. — P. 19-50.

8. ISENMAN M.E., SHASHA D.E. Performance and architectural issues for string matching //IEEE Trans. on Computers. — 1990. — Vol. 39, № 2. — P. 238-250.

9. ГУСЕВ В.Д., КУЛИЧКОВ В.А., НИКУЛИН А.Е. Алгоритм поиска несовершенных повторов в генетических текстах //Анализ символьных последовательностей. — Новосибирск, 1985. — Вып. 113: Вычислительные системы. — С.107-122.

10. ФЕЛЛЕР В. Введение в теорию вероятностей и ее приложения. — М.: Мир, 1964. — 498 с.

11. AOL Junichi, YAMAMOTO Yoneo, SHIMADA Ryosaku. A method for improving string pattern matching machines //IEEE Trans. on Software Engineering. — 1984. — Vol. SE-10, № 1. — P.116-120.

12. Банк образов протеиновых семейств PROF-IMAGE для быстрого оценивания вероятных функций аминокислотных последовательностей /А.Г.Бачинский, А.А.Ярыгин, В.А.Куличков, Е.Г.Гусева //Молекулярная биология. — 1995. — Т. 29, N 4. — С. 907-917.

Поступила в редакцию
6 сентября 1996 года