

Алгоритм вычисления P -значения числа вхождений образцов в случайной последовательности.

Минимизация графа перекрытий

Фурлетова Е.И. *

Институт математических проблем биологии РАН – филиал Института прикладной математики им. М.В. Келдыша РАН, Пущино, Россия

Аннотация. Разработан алгоритм MINSUFREF для нахождения вероятности (P -значения) обнаружения S вхождений слов из набора $\mathcal{H}$ в случайной последовательности длины N . P -значение широко используется в компьютерных науках в качестве критерия для отбора слов-образцов определенного вида. В частности, в биоинформатике при поиске функциональных фрагментов в биологических последовательностях. Предложенный алгоритм основан на ранее разработанном алгоритме SUFPREF для вычисления точного P -значения. Алгоритм SUFPREF использует для вычислений граф перекрытий, вершины которого соответствуют словам из $\mathcal{H}$ и их перекрытиям. Ограничения SUFPREF состоят в том, что он допускает лишь наборы, состоящие из слов одинаковой длины, а также в ряде случаев не применим для наборов, состоящих из большого числа слов. В данной работе алгоритм расширен на наборы, состоящие из слов произвольных длин, с ограничением в том, что ни одно слово не должно быть фрагментом другого слова. Кроме того, на множестве префиксов слов из $\mathcal{H}$ введено отношение $\sim^R$ -эквивалентности, граф перекрытий минимизирован относительно этого отношения. Пространственная и временная сложности основной части работы алгоритма MINSUFREF линейны по числу вершин и ребер минимизированного графа перекрытий. Временная сложность алгоритма не зависит от размера алфавита и длин слов в наборе. Проведены компьютерные эксперименты, которые показали высокую эффективность MINSUFREF по сравнению с другими алгоритмами.

Ключевые слова: P -значение, вхождение образцов, перекрытие слов, автомат Ахо-Корасик, минимизация конечного автомата

ВВЕДЕНИЕ

Одной из важнейших задач компьютерных наук является поиск в последовательностях слов специального вида (образцов), которые играют определенную функциональную роль. Как правило, такие слова встречаются в последовательностях чаще или реже, чем ожидается. Этот подход получил особое применение в биоинформатике, в частности, при поиске факторов регуляции транскрипции [1, 2], а также при поиске оптимальных затравок для выравнивания биологических последовательностей [3]. Кроме того, такой подход применяется в метеорологии, психологии, теории надежности и других областях науки, см. [4, 5]. Для оценки статистической значимости числа вхождений образцов в последовательность зачастую используется P -значение числа вхождений, то есть вероятность встретить образцы из данного набора в последовательности

*evgfurletova@yandex.ru

заданной длины не менее заданного числа раз. Вычисление P -значения тесно связано с нахождением распределения времени ожидания появления образцов в последовательности испытаний [5, 6].

Начиная с 1980-х годов, методы нахождения распределения числа вхождений слов в случайных символьных последовательностях активно развивались. Подробный обзор существующих подходов изложен в работе [4]. Среди существующих методов встречаются как точные, так и использующие различные аппроксимации. Широкое распространение получило применение для вычислений конечных автоматов или специальных графов, см. [3, 4, 7, 8, 9, 10, 11, 12]. При этом временная и емкостная сложности алгоритма линейны по размеру (числу состояний и переходов) используемой структуры. В большинстве работ используется автомат Ахо-Корасик [13], распознающий язык $\Sigma^* \cdot \mathcal{H}$ (Σ — алфавит, $\mathcal{H}$ — набор образцов), или производные от него структуры. При решении практических задач построение или применение автомата Ахо-Корасик может быть затруднительным из-за огромного размера набора образцов. Поэтому в работах последних лет активно используются минимальные (приведенные) или редуцированные конечные автоматы, см. [3, 4, 8, 12]. Обратим внимание, мы будем различать понятия автомата и графа Ахо-Корасик; графом Ахо-Корасик будем называть бор с суффиксными ссылками, описанный в [13]. Автомат Ахо-Корасик — полный автомат, в котором из каждого состояния есть переход по каждому символу алфавита, переход соответствует определенному пути в графе Ахо-Корасик.

В работах [10, 11] мы разработали алгоритм *SufPref* для вычисления точного P -значения, основанный на рекуррентных соотношениях между специальными множествами символьных последовательностей. Алгоритм *SufPref* использует для вычислений граф перекрытий (*OvGraph*), вершины которого соответствуют словам из набора и их перекрытиям. *OvGraph* является подграфом графа Ахо-Корасик. Алгоритм был разработан для вероятностных моделей Бернулли, марковских моделей и скрытых марковских моделей. Временная и пространственная сложности алгоритма линейны по числу вершин графа *OvGraph*. Обычно перекрытия составляют относительно небольшую часть от числа всех префиксов слов из заданного набора. В среднем, их число меньше или равно числу слов в наборе, см. [14]. Причем нет явной зависимости числа перекрытий от размера алфавита. Благодаря этому *SufPref* во многих случаях оказался эффективнее других алгоритмов. Особенно при алфавитах больших размеров. Отметим, что временная сложность алгоритмов, использующих автомат Ахо-Корасик в изначальной или минимальной форме, пропорциональна размеру алфавита. Алгоритм *SufPref* имеет значительные ограничения, одно из которых — алгоритм допускает лишь наборы, состоящие из образцов одинаковой длины. Второе ограничение — сложность построения и использования графа перекрытий для наборов больших размеров, например, для наборов, заданных с помощью консенсусов из базы данных PROSITE [15]. Целью данной работы является преодоление этих ограничений.

В данной работе алгоритм *SufPref* расширен на случай, когда образцы в наборе имеют различную длину. Единственным ограничением является то, что в наборе не должно быть двух слов, одно из которых является фрагментом другого. Такое расширение значительно увеличивает круг допустимых наборов, преодоление ограничения — цель наших дальнейших исследований. Кроме того, предложен способ минимизации графа перекрытий. На множестве префиксов слов из заданного набора введено отношение $\sim^R$ -эквивалентности и предложена конструкция $\sim^R$ -минимального графа перекрытий (*MinOvGraph*), вершины которого соответствуют классам $\sim^R$ -эквивалентности вершин графа *OvGraph*. Отношение $\sim^R$ -эквивалентности позволяет производить вычисления вероятностей множеств символьных последовательностей по соотношениям

аналогичным соотношениям из [10], но для классов эквивалентности вместо одиночных вершин графа перекрытий. *MinOvGraph* также является подграфом графа $\sim^R$ -минимального автомата Ахо-Корасик, вершины которого — классы $\sim^R$ -эквивалентности вершин графа Ахо-Корасик. Итогом данной работы является алгоритм *MINSUFREF*, сочетающий в себе описанные модификации. Алгоритм *MINSUFREF* реализован в виде компьютерной программы, доступной по адресу <https://sourceforge.net/projects/minsufpref/>.

Отношение $\sim^R$ -эквивалентности было ранее предложено нами в работах [12, 16] и был разработан линейный алгоритм построения $\sim^R$ -минимального автомата Ахо-Корасик. В данной работе в определение добавлено дополнительное условие равенства длин эквивалентных состояний. Такое условие не усложняет разработанный алгоритм минимизации. Подобные подходы к минимизации автомата Ахо-Корасик были также предложены в работах [4, 17]. Следует отметить, что в классическом смысле при минимизации конечного автомата его состояния также разбиваются на классы эквивалентности, где отношением эквивалентности является неотличимость состояний автомата. И автомат называют минимальным (приведенным), если все его состояния неотличимы. Во многих случаях $\sim^R$ -минимальный автомат Ахо-Корасик является минимальным автоматом. Например, в [16] показано, что $\sim^R$ -эквивалентность тождественна неотличимости состояний автомата для случая, когда набор состоит из слов одинаковой длины. А также для случая, когда “длинные” фрагменты встречаются в словах из набора не более одного раза. И показано, что $\sim^R$ -эквивалентность влечет неотличимость состояний, однако обратное в общем случае неверно.

В данной работе описаны множества символьных последовательностей, приписанные к вершинам $\sim^R$ -минимального графа перекрытий, и выведены рекуррентные соотношения для этих множеств. С помощью этих соотношений могут быть получены формулы для вычисления вероятностей для любого распределения вероятностей символов в последовательностях. Для упрощения изложения в статье используется лишь модель Бернулли. Алгоритмы для вычисления вероятностей для других моделей будут подобными, с учетом особенностей моделей. В [10] описан способ перевода соотношений для символьных последовательностей в формулы для вычисления вероятностей при марковских и скрытых марковских моделях.

Проведены компьютерные эксперименты, которые показали высокую эффективность *MINSUFREF* по сравнению с другими алгоритмами, см. раздел 6. Продемонстрируем эффективность на примере набора образцов, заданного словом-консенсусом *TGTTTCCN{18}TGTTTCT*, здесь $N\{18\}$ обозначает 18 произвольных нуклеотидов. Набор является частью мотива, соответствующего факторам регуляции транскрипции *Stel2p* из базы данных *YEASTRACT* [18]. Такой набор содержит порядка $6,9 \cdot 10^{10}$ слов. Автомат Ахо-Корасик имеет более чем 10^{11} состояний. Его непосредственное построение затруднительно, в связи с чем затруднительно построение графа перекрытий. Минимальный автомат совпадает с $\sim^R$ -минимальным автоматом Ахо-Корасик и содержит 754 состояния и 3016 переходов, а $\sim^R$ -минимальный граф перекрытий содержит всего лишь 57 вершин и 241 ребро. Приведенные значения получены в ходе компьютерных экспериментов.

В работе [12] мы также использовали структуру, производную от $\sim^R$ -минимального автомата Ахо-Корасик, для нахождения приближенного P -значения с заданной точностью для вероятностных моделей Бернулли. Рассматривались лишь наборы образцов, заданные словами-консенсусами. Алгоритм обрабатывает последовательно перекрывающиеся

вхождения образцов (клампы). Временная и пространственная сложности алгоритма не зависят от длины случайной последовательности, однако в худшем случае они пропорциональны $m \times ||\mathcal{H}||^2$.

Данная статья организована следующим образом. В разделе 2 определены основные понятия, используемые в статье. В разделе 3 введено отношение $\sim^R$ -эквивалентности и даны определения $\sim^R$ -минимального автомата Ахо-Корасик и $\sim^R$ -минимального графа перекрытий. В четвертом разделе выведены основные рекуррентные соотношения на множествах символьных последовательностей, необходимые для вычисления P -значения. В пятом разделе описан алгоритм SufPref-D для нахождения P -значения, расширенный на случай набора, содержащего образцы произвольной длины. В разделе 6 описан алгоритм MinSufPref, являющийся оптимизацией SufPref-D, алгоритм использует минимизированный граф перекрытий. И в седьмом разделе описаны результаты сравнения MinSufPref с другими алгоритмами.

1. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Пусть задан конечный алфавит Σ , слово $x \in \Sigma^*$ и множество слов $X \subseteq \Sigma^*$. Длину слова x обозначим через $|x|$. Если все слова в множестве X имеют одинаковую длину, то будем обозначать эту длину через $|X|$. Мощность X (число слов в X) обозначим через $||X||$. Записи $x \cdot y$ или xy обозначают конкатенацию слов x и y . Для обозначения того, что слово v является префиксом (собственным префиксом) слова x будем использовать запись $v \preceq x$ ($v \prec x$). Аналогично запись $v \sqsubseteq x$ ($v \subset x$) обозначает, что v является суффиксом (собственным суффиксом) x . Отношения $\prec$, $\preceq$, $\subset$ и $\sqsubseteq$ являются отношениями частичного порядка.

Для префикса v слова x обозначим через $back(v, x)$ такой его суффикс, что $x = v \cdot back(v, x)$. Если v не является префиксом x , то значение $back(v, x)$ не определено.

Дано множество $\mathcal{H} = \{h_1, \dots, h_k\}$, состоящее из k слов в алфавите Σ , где $m_0 \leq |h_i| \leq m$. Самое короткое слово в $\mathcal{H}$ имеет длину m_0 , а самое длинное — m . Будем предполагать, что в $\mathcal{H}$ нет двух слов, одно из которых является фрагментом другого. Обозначим множества всех префиксов (собственных префиксов) слов из $\mathcal{H}$ через $Pref(\mathcal{H})$ ($PPref(\mathcal{H})$). Отметим, что $||Pref(\mathcal{H})|| \leq m \times ||\mathcal{H}|| + 1$.

Слово $w \in PPref(\mathcal{H})$ будем называть **перекрытием** для $\mathcal{H}$, если в $\mathcal{H}$ найдутся два слова h_1 и h_2 такие, что w является собственным суффиксом h_1 и собственным префиксом h_2 . Множество всех перекрытий для $\mathcal{H}$ обозначим через $OV(\mathcal{H})$, где $OV(\mathcal{H}) \subset Pref(\mathcal{H})$. Отметим, что $\varepsilon \in OV(\mathcal{H})$. Также отметим, что $OV(\mathcal{H}) \cap \mathcal{H} = \emptyset$, так как предполагается, что никакое слово из $\mathcal{H}$ не является собственным суффиксом или префиксом другого слова из $\mathcal{H}$.

Далее в тексте будем использовать запись $\max\{x|G(x)\}$, которая обозначает слово x наибольшей длины, удовлетворяющее условию $G(x)$.

Суффиксной ссылкой (suffix link, border, failure function) слова $x \in Pref(\mathcal{H})$ будем называть собственный суффикс x наибольшей длины, который является перекрытием, то есть

$$sl(x) = \max\{v | v \in OV(\mathcal{H}) \text{ и } v \subset x\}.$$

Понятие суффиксной ссылки используется в различных алгоритмах анализа строк. В частности, в алгоритме Ахо-Корасик [13].

Для $w \in OV(\mathcal{H})$ также определим:

$$lp(w) = \max\{v | v \in OV(\mathcal{H}) \text{ и } v \prec w\}.$$

Другими словами, $lp(w)$ (левый предшественник) — это собственный префикс w наибольшей длины, который является перекрытием из $OV(\mathcal{H})$.

Положим $back(w) = back(lp(w), w)$.

Множество перекрытий и отношения порядка между ними будем рассматривать в виде специального графа.

Определение 1. Графом перекрытий $OvGraph(\mathcal{H}) = \langle OV(\mathcal{H}) \cup \mathcal{H}, E(\mathcal{H}) \rangle$ будем называть ориентированный граф с множеством вершин $OV(\mathcal{H}) \cup \mathcal{H}$ и множеством ребер $E(\mathcal{H})$, состоящих из прямых и обратных ребер, где

- прямое ребро ведет из вершины v в вершину w , если $v = lp(w)$;
- обратное ребро ведет из вершины w ($w \neq \varepsilon$) в вершину v , если $v = sl(w)$.

Прямое ребро $\langle v, w \rangle$ имеет метку $back(w)$.

Вершину, соответствующую пустому слову ε , будем называть **корнем**; вершины, соответствующие словам из $\mathcal{H}$, — **листьями**.

Отметим, что в каждую вершину графа перекрытий, кроме корня, входит ровно одно прямое ребро и из нее исходит ровно одно обратное ребро. Корень не имеет обратного ребра. Такое условие добавлено в определение графа перекрытий, так как предложенный в данной работе алгоритм вычисления P -значения, не требует наличие обратного ребра для корня.

Граф перекрытий имеет не более $m \times ||\mathcal{H}|| + 1$ вершин и $2 \times m \times ||\mathcal{H}||$ ребер.

Пример графа перекрытий $OvGraph(\mathcal{H})$ изображен на рисунке 1. Если в $OvGraph(\mathcal{H})$ убрать обратные ребра, то получится дерево, листья которого соответствуют словам из $\mathcal{H}$; внутренние вершины — перекрытиям, а корень — пустому слову ε . Также если убрать прямые ребра и развернуть обратные, то получится дерево.

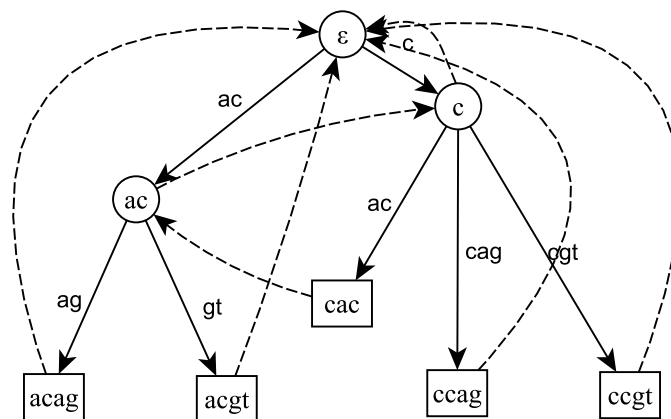


Рис. 1. Граф перекрытий $OvGraph(\mathcal{H})$ для $\mathcal{H} = \{acag, acgt, cac, ccag, ccgt\}$. Вершины, соответствующие словам из $\mathcal{H}$, обозначены прямоугольниками, а перекрытия — окружностями. Прямые ребра обозначены сплошными прямыми линиями; обратные ребра — пунктирными дугами.

Ниже иногда мы будем опускать запись “ $\mathcal{H}$ ” в названиях графов и автоматов.

Граф перекрытий может быть построен с использованием графа Ахо-Корасик [13]. Граф Ахо-Корасик является бором с суффиксными ссылками, который имеет следующую

структуру. $Pref(\mathcal{H})$ — множество вершин, граф имеет два типа ребер: прямые и обратные ребра. Прямое ребро ведет из вершины v в вершину w , если $w = va$, $a \in \Sigma$; обратные ребра соответствуют суффиксным ссылкам. Пример графа Ахо-Корасик представлен на рисунке 2.

Мы будем различать понятия автомата и графа Ахо-Корасик. Автомат Ахо-Корасик распознает язык $\Sigma^* \cdot \mathcal{H}$. Он также имеет множество состояний $Pref(\mathcal{H})$ и является полным, то есть существует переход из каждого состояния по каждому символу алфавита. Переход $\langle v, a, w \rangle$ в автомате соответствует пути в графе Ахо-Корасик, ведущему из v в w по суффиксным ссылкам и прямому ребру с меткой a . Число ребер графа Ахо-Корасик равно $2 \times ||Pref(\mathcal{H})|| - 1$; число переходов автомата Ахо-Корасик равно $||\Sigma|| \times ||Pref(\mathcal{H})||$.

Отметим, что если вершина графа Ахо-Корасик соответствует перекрытию, то к ней есть путь, ведущий по суффиксным ссылкам из терминальной вершины (листа), соответствующей некоторому слову из $\mathcal{H}$. Тогда для нахождения $OV(\mathcal{H})$ необходимо следовать по суффиксным ссылкам от листьев к корню ε ; $OV(\mathcal{H})$ состоит из всех вершин, кроме листьев, лежащих на этих путях. Далее путем обхода графа Ахо-Корасик в глубину, следуя по прямым ребрам от корня к листьям, удаляются все вершины, которые не соответствуют словам из $OV(\mathcal{H}) \cup \mathcal{H}$. Соседние из оставшихся вершин соединяются новыми ребрами. В итоге будет построен граф перекрытий. Данный алгоритм подробно описан в приложении к работе [10].

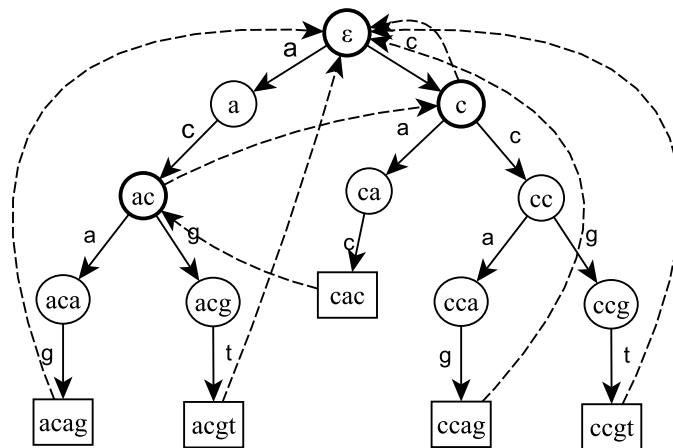


Рис. 2. Граф Ахо-Корасик для $\mathcal{H} = \{acag, acgt, sac, ccag, ccgt\}$. Вершины, соответствующие словам из $\mathcal{H}$, обозначены прямоугольниками, а остальные вершины — окружностями. Перекрытия выделены жирными окружностями. Прямые ребра обозначены сплошными прямыми линиями; суффиксные ссылки — пунктирными дугами. Для наглядности изображения суффиксные ссылки показаны только для перекрытий, кроме ε , и слов из $\mathcal{H}$.

2. Минимизация графа перекрытий

В данном разделе на множестве префиксов $Pref(\mathcal{H})$ введено отношение $\sim^R$ -эквивалентности. И определен $\sim^R$ -минимальный граф перекрытий (*MinOvGraph*), вершинами которого являются классы $\sim^R$ -эквивалентности вершин графа *OvGraph*. Отношение $\sim^R$ -эквивалентности позволяет минимизировать граф перекрытий и проводить вычисление вероятностей для классов эквивалентности его вершин.

Для описания конструкции и построения $\sim^R$ -минимального графа перекрытий также

определен $\sim^R$ -минимальный граф Ахо-Корасик (*MinACGraph*), который в данной работе играет роль вспомогательной структуры. *MinOvGraph* является подграфом *MinACGraph*.

2.1. $\sim^R$ -эквивалентность

Определим на множестве $Pref(\mathcal{H})$ отношение $\sim^R$ -эквивалентности.

Определение 2 $\sim^R$ -эквивалентность. Два слова $x, y \in Pref(\mathcal{H})$ будем называть $\sim^R$ -эквивалентными ($x \sim^R y$), если $x = y$ или одновременно выполняются следующие условия:

$$\begin{aligned} |x| &= |y| ; \\ xt \in \mathcal{H} &\Leftrightarrow yt \in \mathcal{H} ; \\ sl(x) &\sim^R sl(y) . \end{aligned}$$

Через $\hat{x}$ будем обозначать класс $\sim^R$ -эквивалентности для x , где $\hat{x} = \{y \mid y \in Pref(\mathcal{H}) \text{ и } x \sim^R y\}$. Также обозначим через $Pref_R(\mathcal{H})$ и $\mathcal{H}_R$ множества классов $\sim^R$ -эквивалентности на словах из $Pref(\mathcal{H})$ и $\mathcal{H}$ соответственно.

Отношение $\sim^R$ -эквивалентности было ранее введено нами в работах [12, 16], а также разработан алгоритм построения $\sim^R$ -минимального автомата Ахо-Корасик. В данной работе в определении $\sim^R$ -эквивалентности добавлено дополнительное условие равенства длин эквивалентных состояний. Данное условие не усложняет алгоритм минимизации.

Из определения 2 следует, что если $h \in \mathcal{H}$ и $h \sim^R h'$, то $h' \in \mathcal{H}$. Тогда $\hat{h} \subseteq \mathcal{H}$.

Из первого условия $\sim^R$ -эквивалентности следует, что в классе $\hat{x}$ все слова имеют одинаковую длину. Обозначим эту длину через $|\hat{x}|$. И из третьего условия $\sim^R$ -эквивалентности следует, что суффиксные ссылки эквивалентных состояний тоже эквивалентны. Тогда расширим определение суффиксной ссылки на класс $\hat{x}$, обозначив $sl(\hat{x}) = \widehat{sl(x)}$, где $x \in \hat{x}$.

Отметим, что если $\mathcal{H}$ состоит из слов одинаковой длины, то первое условие определения 2 следует из двух других условий.

Кроме того, следует отметить, что отношение $\sim^R$ -эквивалентности задано на множестве $Pref(\mathcal{H})$, которое совпадает с множеством вершин графа Ахо-Корасик, оно разбивает это множество на классы $\sim^R$ -эквивалентности. По второму условию $\sim^R$ -эквивалентности, если $x \sim^R y$, то $\{t \mid xt \in \mathcal{H}\} = \{t \mid yt \in \mathcal{H}\}$. Это тождественно тому, что поддеревья в графе Ахо-Корасик, состоящие из вершин и прямых ребер, с корнями в вершинах x и y являются подобными.

Пример 1. Пусть задано множество слов $\mathcal{H} = \{acag, acgt, cac, ccag, ccgt\}$. На рисунке 2 представлен соответствующий граф Ахо-Корасик. Слова ac и cc $\sim^R$ -эквивалентны, так как поддеревья в графе Ахо-Корасик с вершинами, соответствующими этим словам, подобны и $sl(ac) = sl(cc) = c$. Отметим, что ac является перекрытием для $\mathcal{H}$, а cc — нет.

Нижеуказанные теорема и следствие из нее представлены в работе [16], добавленное условие в определении 2 равенства длин эквивалентных слов значительно не влияет на ход доказательства.

Теорема 1. [16] Пусть $x, y \in Pref(\mathcal{H})$ и $x \sim^R y$. Тогда $\forall a \in \Sigma$:

1. $sl(xa) \stackrel{R}{\sim} sl(ya)$;
2. если $xa, ya \in Pref(\mathcal{H})$, то $xa \stackrel{R}{\sim} ya$.

Следствие 1. Пусть $x \stackrel{R}{\sim} y$ и $xt \in Pref(\mathcal{H})$. Тогда $yt \in Pref(\mathcal{H})$ и $xt \stackrel{R}{\sim} yt$.

Из указанной теоремы следует, что если две вершины x и y являются $\stackrel{R}{\sim}$ -эквивалентными, то вершины, к которым ведут пути из x и y по слову $t \in \Sigma^*$ также $\stackrel{R}{\sim}$ -эквивалентны.

Отношение $\stackrel{R}{\sim}$ -эквивалентности позволяет минимизировать граф перекрытий и граф Ахо-Корасик. Подобно построению графа перекрытий для построения $\stackrel{R}{\sim}$ -минимального графа перекрытий потребуется конструкция $\stackrel{R}{\sim}$ -минимального графа Ахо-Корасик.

Обратим внимание на то, что отношение $\stackrel{R}{\sim}$ -эквивалентности можно было бы определить напрямую на множестве $OV(\mathcal{H}) \cup \mathcal{H}$. Однако такой подход некорректен. Возможен случай, когда $v, w \in OV(\mathcal{H})$ и $v \stackrel{R}{\sim} w$ и можно подобрать такое $t \in \Sigma$, что $vt \in OV(\mathcal{H})$, а $wt \notin OV(\mathcal{H})$, хотя по следствию 1 имеем $vt \stackrel{R}{\sim} wt$.

2.2. $\stackrel{R}{\sim}$ -минимальный граф Ахо-Корасик

Определение 3. $\stackrel{R}{\sim}$ -минимальный граф Ахо-Корасик $MinACGraph(\mathcal{H})$ — ориентированный граф с множеством вершин $Pref_R(\mathcal{H})$, а также множеством ребер $E_R^A(\mathcal{H})$, состоящих из прямых и обратных ребер, где

- прямое ребро ведет из вершины $\hat{v}$ в вершину $\hat{w}$, если $\hat{w} = \hat{v}a$, где $a \in \Sigma$;
- обратное ребро ведет из вершины $\hat{w}$ в вершину $\hat{v}$, если $\hat{v} = sl(\hat{w})$.

Прямое ребро $\langle \hat{v}, \hat{v}a \rangle$ имеет метку a .

Корректность определения $\stackrel{R}{\sim}$ -минимального графа Ахо-Корасик следует из определения $\stackrel{R}{\sim}$ -эквивалентности и теоремы 1. Ниже на рисунке 3 представлен пример $\stackrel{R}{\sim}$ -минимального графа Ахо-Корасик. Соответствующий этому графу конечный автомат является минимальным (приведенным).

Удалим в $\stackrel{R}{\sim}$ -минимальном графе Ахо-Корасик все вершины, кроме тех, которые $\stackrel{R}{\sim}$ -эквивалентны словам из $OV(\mathcal{H}) \cup \mathcal{H}$, и соединим прямыми ребрами соседние среди оставшихся вершин. Полученный граф будем называть $\stackrel{R}{\sim}$ -минимальным графом перекрытий. Способ построения $\stackrel{R}{\sim}$ -минимального графа перекрытий аналогичен способу построения графа перекрытий путем обхода графа Ахо-Корасик. Пример $\stackrel{R}{\sim}$ -минимального графа перекрытий представлен на рисунке 4.

2.3. $\stackrel{R}{\sim}$ -минимальный граф перекрытий

Опишем формально конструкцию $\stackrel{R}{\sim}$ -минимального графа перекрытий.

Обозначим через $OV_R(\mathcal{H})$ подмножество $Pref_R(\mathcal{H})$ классов $\stackrel{R}{\sim}$ -эквивалентности, содержащих перекрытия из $OV(\mathcal{H})$, то есть

$$OV_R(\mathcal{H}) = \{\hat{w} \in Pref_R(\mathcal{H}) \mid \hat{w} \cap OV(\mathcal{H}) \neq \emptyset\}.$$

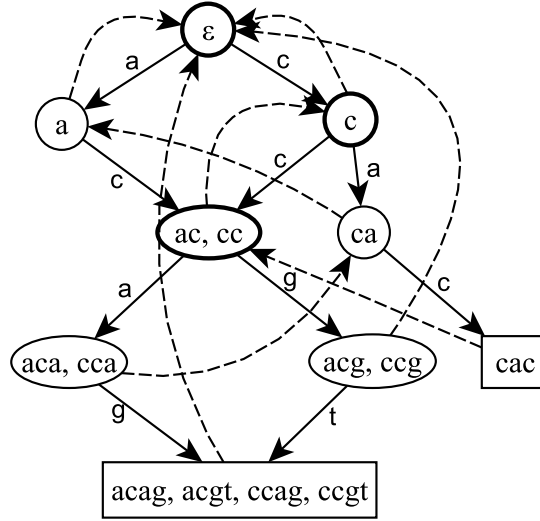


Рис. 3. $\sim^R$ -минимальный граф Ахо-Корасик для $\mathcal{H} = \{acag, acgt, cac, ccag, ccgt\}$. Обозначения вершин и ребер аналогичны обозначениям на рисунке 2. Вершины, содержащие перекрытия, выделены жирными окружностями.

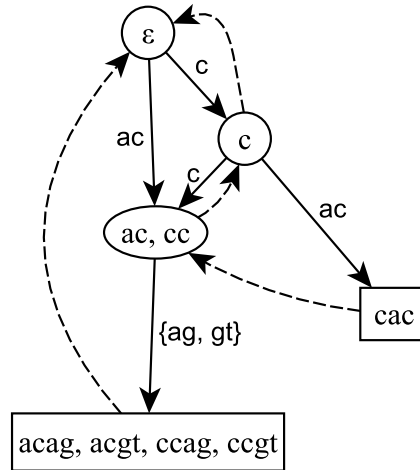


Рис. 4. $\sim^R$ -минимальный граф перекрытий для $\mathcal{H} = \{acag, acgt, cac, ccag, ccgt\}$. Обозначения вершин и ребер аналогичны обозначениям на рисунке 1.

Важно отметить, что классы из $OV_R(\mathcal{H})$ могут содержать слова, которые не являются перекрытиями. Например, рассмотрим $\mathcal{H}$ из примера 1, см. рис. 3 и 4. Имеем $\widehat{ac} \in OV_R(\mathcal{H})$, где $\widehat{ac} = \widehat{cc} = \{ac, cc\}$, причем ac является перекрытием, а cc — нет. Однако $\widehat{cc} \in OV_R(\mathcal{H})$.

Далее нас будут интересовать лишь те слова из $Pref(\mathcal{H})$, классы $\sim^R$ -эквивалентности которых принадлежат $OV_R(\mathcal{H}) \cup \mathcal{H}_R$.

Пусть $w \in Pref(\mathcal{H})$ и $\widehat{w} \in OV_R(\mathcal{H})$. Обозначим через $lp_R(w)$ собственный префикс слова w наибольшей длины, который $\sim^R$ -эквивалентен некоторому перекрытию из $OV(\mathcal{H})$, то есть

$$lp_R(w) = \max\{v \mid v \prec w \text{ и } \widehat{v} \in OV_R(\mathcal{H})\}.$$

В вышеуказанном примере $lp_R(ccgt) = cc$, где $cc \notin OV(\mathcal{H})$, но $\widehat{cc} \in OV_R(\mathcal{H})$.

Понятия $lp_R(w)$ и $lp(w)$ похожи. Однако $lp_R(w)$, в отличие от $lp(w)$, может не являться перекрытием, но должен быть эквивалентным некоторому перекрытию.

Для различных слов w, w' из класса $\hat{w} \in OV_R(\mathcal{H})$ префиксы $lp_R(w)$ и $lp_R(w')$ могут быть не $\sim^R$ -эквивалентными. Тогда для $\hat{w}$ определим множество:

$$LPREDS_R(\hat{w}) = \{\widehat{lp_R(w)} \mid w \in \hat{w}\}.$$

Отметим, что $LPREDS_R(\hat{w}) \subseteq OV_R(\mathcal{H})$.

Пусть $\hat{v} \in LPREDS_R(\hat{w})$. Обозначим:

$$BACK_R(\hat{v}, \hat{w}) = \bigcup_{\substack{w \in \hat{w} \\ lp_R(w) \in \hat{v}}} back(v, w).$$

Другими словами, $BACK_R(\hat{v}, \hat{w})$ — объединение меток путей в $\sim^R$ -минимальном графе Ахо-Корасик, ведущих от $\hat{v}$ к $\hat{w}$ и не содержащих других вершин из $OV_R(\mathcal{H})$. Понятие $BACK_R(\hat{v}, \hat{w})$ схоже с $back(v, w)$. Однако на пути из v в w с меткой $back(v, w)$ могут быть перекрытия, отличные от v и w .

Если $\hat{v} \notin LPREDS_R(\hat{w})$, то $BACK_R(\hat{v}, \hat{w}) = \emptyset$.

Утверждение 1. Пусть $\hat{w} \in OV(\mathcal{H})$ и $\hat{v} \in LPREDS_R(\hat{w})$. Также пусть $v, v' \in \hat{v}$. Тогда $\bigcup_{w \in \hat{w}; v=lp_R(w)} back(v, w) = \bigcup_{w \in \hat{w}; v'=lp_R(w)} back(v', w)$.

Доказательство. Обозначим $A = \bigcup_{w \in \hat{w}; v=lp_R(w)} back(v, w)$ и $B = \bigcup_{w \in \hat{w}; v'=lp_R(w)} back(v', w)$. Покажем, что $A \subseteq B$. Пусть $w \in \hat{w}$ и $v = lp_R(w)$. Тогда слово w можно представить в виде $w = v \cdot back(v, w)$, где $back(v, w) \in A$. Из $\sim^R$ -эквивалентности слов v и v' следует, что $v \cdot back(v, w) \sim^R v' \cdot back(v, w)$. Тогда $v' \cdot back(v, w) \in \hat{w}$.

Пусть $w' = v' \cdot back(v, w)$. Покажем, что $v' = lp_R(w')$. Допустим противное. Пусть $lp_R(w') = r'$, где $r' = v' \cdot t$, t — непустое слово из Σ^* . Тогда $r' \sim^R$ -эквивалентно префиксу $r = v \cdot t$ слова w . Причем $\hat{r}' = \hat{r} \in OV_R(\mathcal{H})$. Так как $|r| > |v|$, то $v \neq lp_R(w)$. Получаем противоречие. Тогда $v' = lp_R(w')$ и $back(v, w) \in B$.

Симметрично, $B \subseteq A$. Отсюда следует, что $A = B$.

□

Следствие 2. Пусть $\hat{w} \in OV(\mathcal{H})$ и $\hat{v} \in LPREDS_R(\hat{w})$. Также пусть $v \in \hat{v}$. Тогда

$$BACK_R(\hat{v}, \hat{w}) = \bigcup_{\substack{w \in \hat{w} \\ v=lp_R(w)}} back(v, w). \quad (2.1)$$

Определение 4. $\sim^R$ -минимальный граф перекрытий $MinOvGraph(\mathcal{H}) = \langle OV_R(\mathcal{H}) \cup \mathcal{H}_R; E_R(\mathcal{H}) \rangle$ — это ориентированный граф с множеством вершин $OV_R(\mathcal{H}) \cup \mathcal{H}_R$, а также множеством ребер $E_R(\mathcal{H})$, состоящих из прямых и обратных ребер, где

- прямое ребро ведет из вершины $\hat{v}$ в вершину $\hat{w}$, если $\hat{v} \in LPREDS_R(\hat{w})$;
- обратное ребро ведет из вершины $\hat{w}$ в вершину $\hat{v}$, если $\hat{v} = sl(\hat{w})$.

Прямое ребро $\langle \hat{v}, \hat{w} \rangle$ имеет метку $BACK_R(\hat{v}, \hat{w})$.

Пример $\sim^R$ -минимального графа перекрытий представлен на рисунке 4. Из каждой вершины $\sim^R$ -минимального графа перекрытий, кроме $\hat{\varepsilon}$, исходит ровно одно обратное ребро, но, в отличие от изначального графа перекрытий, в вершину может входить несколько прямых ребер.

3. МНОЖЕСТВА СИМВОЛЬНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

В данном разделе определим множества символьных последовательностей и выведем соотношения между этими множествами, которые позволяют вычислить вероятность (P -значение) обнаружить S вхождений слов из $\mathcal{H}$ в случайной последовательности длины N . Нижеуказанные соотношения были представлены нами в работе [10] для случая, когда $\mathcal{H}$ состоит из слов одинаковой длины. В данной работе рассматривается произвольное конечное множество $\mathcal{H}$, никакое слово в котором не является фактором другого слова. При данном обобщении соотношения из [10] также верны, поэтому здесь будут лишь выписаны требуемые формулы с пояснениями без формального доказательства.

3.1. Основные соотношения

Пусть даны натуральные числа n, s .

Определение 5. *Определим множество:*

$$B(n, s) = \{t \in \Sigma^n \mid t \text{ содержит не менее } s \text{ вхождений } \mathcal{H}\}.$$

Тогда вероятность $Prob(B(N, S))$ — искомое P -значение.

Исходя из определения, имеем $B(n, 0) = \Sigma^n$.

Для нахождения $B(n, s)$ определим дополнительные множества.

Определение 6. *Пусть $h \in \mathcal{H}$. Определим:*

$$\begin{aligned} R(n, s, h) &= \{t \in \Sigma^n \mid t \text{ содержит ровно } s \text{ вхождений } \mathcal{H} \text{ \& } \\ &\quad t \text{ оканчивается на } h\}; \\ R(n, \geq s, h) &= \{t \in \Sigma^n \mid t \text{ содержит не менее } s \text{ вхождений } \mathcal{H} \text{ \& } \\ &\quad t \text{ оканчивается на } h\}. \end{aligned}$$

Также для $w \in OV(\mathcal{H})$ доопределим $R(n, s, w) = \bigcup_{h \in \mathcal{H}; w \subset h} R(n, s, h)$ и $R(n, \geq s, w) = \bigcup_{h \in \mathcal{H}; w \subset h} R(n, \geq s, h)$. Если $x \in Pref(\mathcal{H}) \setminus (OV(\mathcal{H}) \cup \mathcal{H})$, то определим $R(n, s, x) = \emptyset$.

Для $w \in OV(\mathcal{H})$ справедливо соотношение:

$$R(n, s, w) = \bigcup_{v \in OV(\mathcal{H}) \cup \mathcal{H}; w=sl(v)} R(n, s, v). \quad (3.2)$$

Отметим, что $R(n, s, \varepsilon) = \bigcup_{h \in \mathcal{H}} R(n, s, h)$.

Для любых слов $h_1, h_2 \in \mathcal{H}$ выполняется $R(n, s, h_1) \cap R(n, s, h_2) = \emptyset$, так как ни одно слово из $\mathcal{H}$ не является фактором другого слова.

Справедливо соотношение:

$$R(n, s, h) = R(n, \geq s, h) \setminus R(n, \geq s+1, h). \quad (3.3)$$

Пусть $t \in B(n, s)$. Тогда либо префикс строки t длины $n-1$ содержит не менее s вхождений $\mathcal{H}$, либо t оканчивается s -ым вхождением $\mathcal{H}$. Отсюда

$$B(n, s) = B(n - 1, s) \cdot \Sigma \cup R(n, s, \varepsilon) . \quad (3.4)$$

Определение 7. Дополнительно для $w \in Pref(\mathcal{H}) \setminus (\mathcal{H} \cup \varepsilon)$ определим:

$$D(n, s, w) = \{t \in B(n, s) \mid w \text{ является суффиксом } t \text{ \& } s\text{-ое вхождение } \mathcal{H} \text{ пересекает } w \text{ на конце } t\} .$$

Отметим, что если $n < m_0 + s - 1$, то $B(n, s) = R(n, s, h) = R(n, \geq s, h) = D(n, s, w) = \emptyset$.

Пусть $n \geq m_0 + s - 1$. Выведем соотношения для нахождения $R(n, \geq s, h)$ и $D(n, s, w)$.

Пусть $t \in R(n, \geq s, h)$. Тогда t содержит не менее s вхождений $\mathcal{H}$ и оканчивается на $h \in \mathcal{H}$.

Если $s = 1$, то префикс t длины $n - |h|$ может быть произвольной строкой в алфавите Σ . Отсюда

$$R(n, \geq 1, h) = \Sigma^{n-|h|} \cdot h . \quad (3.5)$$

Теперь пусть $s > 1$. Возможны два случая расположения $(s - 1)$ -го вхождения $\mathcal{H}$ в t относительно слова h на конце t . В первом случае $(s - 1)$ -е вхождение $\mathcal{H}$ лежит в префиксе t' длины $n - |h|$ строки t и не пересекает h . Тогда $t = t' \cdot h$ и $t \in B(n - |h|, s - 1) \cdot h$.

Во втором случае $(s - 1)$ -е вхождение $\mathcal{H}$ пересекает h на конце строки t . Причем оно также пересекает префикс $lp(h)$ слова h ; пересечение является перекрытием из $OV(\mathcal{H})$. Пусть $t = t' \cdot lp(h) \cdot back(h)$, где $h = lp(h) \cdot back(h)$. Тогда $t' \cdot lp(h) \in D(n - |h| + |lp(h)|, s - 1, lp(h))$. Отсюда получаем:

$$R(n, \geq s, h) = B(n - |h|, s - 1) \cdot h \cup D(n - |h| + |lp(h)|, s - 1, lp(h)) \cdot back(h) . \quad (3.6)$$

Пусть $w \in Pref(\mathcal{H}) \setminus (\mathcal{H} \cup \varepsilon)$, $n \geq m_0$, $s \geq 1$. Аналогично рассуждая, получим соотношение:

$$D(n, s, w) = D(n - |w| + |lp(w)|, s, lp(w)) \cdot back(w) \cup R(n, s, w) . \quad (3.7)$$

Если в соотношениях (3.6) и (3.7) заменить $lp(x)$ на $lp_R(x)$ ($x = w$ или h), а $back(x)$ на $back(lp_R(x), x)$, то полученные соотношения будут также верны.

3.2. Вероятности множеств символьных последовательностей

Представленные выше соотношения для множеств символьных последовательностей легко преобразуются в соответствующие формулы для вычисления вероятностей этих множеств при заданной модели распределения вероятностей букв в словах.

При модели Бернулли вероятность встретить символ из алфавита в некоторой позиции случайной последовательности не зависит ни от этой позиции, ни от предшествующих позиций последовательности. Относительно модели Бернулли вероятность слова равна произведению вероятностей символов этого слова. Вероятность конкатенации слов равна произведению вероятностей этих слов. Вероятность множества слов равна сумме вероятностей слов из этого множества.

При марковских и скрытых марковских моделях вероятность встретить символ из алфавита в случайной последовательности зависит от того, какие стоят символы в предыдущих позициях. Подробный вывод формул и сами формулы для вычисления

вероятностей вышеуказанных множеств символьных последовательностей при марковских и скрытых марковских моделях представлены в работе [10].

4. АЛГОРИТМ ВЫЧИСЛЕНИЯ P -ЗНАЧЕНИЯ

В работе [10] мы разработали алгоритм SUF^{PREF} для вычисления вероятности обнаружения по крайней мере S вхождений слов из множества $\mathcal{H}$ в случайной последовательности длины N . В основе алгоритма лежит граф перекрытий. Для каждого значения $n \leq N$ алгоритм в два этапа обходит граф перекрытий. На первом этапе алгоритм обходит граф перекрытий в глубину по прямым ребрам (от корня к листьям), вычисляя вероятности D -множеств для вершин из $OV(\mathcal{H})$ и вероятности R -множеств для вершин из $\mathcal{H}$. На втором этапе алгоритм обходит граф по обратным ребрам (от листьев к корню), находя вероятности R -множеств для вершин из $OV(\mathcal{H})$, суммируя соответствующие вероятности, вычисленные для вершин, из которых ведут в рассматриваемую вершину обратные ребра. Найденные вероятности R -множеств для ряда последних значений n сохраняются в специальных структурах. Способ вычисления вероятностей D и R -множеств предполагает, что $\mathcal{H}$ состоит из слов одинаковой длины. В [10] также была предложена оптимизация SUF^{PREF} путем удаления листьев, а также входящих в них и исходящих из них ребер, и замены этих ребер глубокими ребрами. Это идея также реализована в $\text{MinSUF}^{\text{PREF}}$, см. подраздел 5.4.

В данной работе предложен другой порядок вычисления вероятностей R и D -множеств, а также изменены используемые структуры данных. В данной версии алгоритма (алгоритм $\text{SUF}^{\text{PREF-D}}$) для каждой вершины из $OV(\mathcal{H})$ сохраняются вероятности D -множеств, вместо R -множеств, вычисленные на предыдущих шагах алгоритма, благодаря чему для вычисления вероятностей вида $R(n, s, h)$ на шаге n не требуется предварительно вычислять все вероятности вида $D(n - |h| + w, s, w)$, где $w \prec h$ и $w \in OV(\mathcal{H})$, как на первом этапе алгоритма SUF^{PREF} из [10]. Это не ухудшает сложность алгоритма, тем временем позволяя рассматривать множества $\mathcal{H}$, состоящие из слов произвольной длины.

Здесь будет рассмотрена лишь модель Бернулли. Схема алгоритма $\text{SUF}^{\text{PREF-D}}$ для других вероятностных моделей аналогична с учетом свойств этих моделей.

4.1. Описание алгоритма $\text{SUF}^{\text{PREF-D}}$

Алгоритм $\text{SUF}^{\text{PREF-D}}$ для заданных множества слов $\mathcal{H}$ и чисел S и N вычисляет вероятность $\text{Prob}(B(S, N))$, см. алгоритм 1. Структура данных алгоритма представлена в подразделе 4.2.

На предварительном шаге алгоритм строит граф перекрытий для $\mathcal{H}$.

Далее алгоритм в цикле по $n = m_0, \dots, N$ вычисляет вероятности вышеописанных множеств символьных последовательностей. Шаг n имеет два этапа.

На первом этапе для каждого слова $h \in \mathcal{H}$ и числа $s = 1, \dots, S+1$ алгоритм вычисляет вероятности $\text{Prob}(R(n, \geq s, h))$ и $\text{Prob}(R(n, s, h))$ (для $s \leq S$). Найденные вероятности $\text{Prob}(R(n, s, h))$ добавляются к значениям ячеек массива $Rsets(sl(h))[s]$, приписанного к вершине $sl(h)$; описание массива представлено в подразделе 4.2. Если $n < |h| + s - 1$, то $\text{Prob}(R(n, \geq s, h)) = \text{Prob}(R(n, s, h)) = 0$. Иначе для вычислений используются соотношения (3.3), (3.5) и (3.6). Для вычислений требуются вероятности B и D -множеств, которые заранее были сохранены в массивах $Bsets$ и $Dsets$, см. подраздел 4.2.

На втором этапе шага n алгоритм обходит граф перекрытий, следуя по обратным ребрам от листьев, соответствующих словам из $\mathcal{H}$, к корню ε . При обходе алгоритм для каждой вершины $w \in OV(\mathcal{H})$ в цикле по $s = 1, \dots, S$ находит вероятности

$Prob(R(n, s, w))$ и $Prob(D(n, s, w))$ (для $w \neq \varepsilon$). Для нахождения $Prob(R(n, s, w))$ используется массив $Rsets$, который инициализируется в начале шага n . Во время рассмотрения вершины v , из которой в w ведет обратное ребро, алгоритм добавляет к $Rsets(w)[s]$ вероятность $Prob(R(n, s, v))$. Обход организован таким образом, чтобы на момент рассмотрения вершины w все вершины, из которых обратные ребра ведут в w , уже были рассмотрены. Тогда к началу рассмотрения w имеем $Rsets(w)[s] = Prob(R(n, s, w))$. Для $w \neq \varepsilon$ алгоритм также находит $Prob(D(n, s, w))$, используя вычисленную вероятность $Prob(R(n, s, w))$. Если $n < m_0 + s - 1$, то $Prob(D(n, s, w)) = 0$, иначе $Prob(D(n, s, w))$ вычисляется по соотношению (3.7). Найденные вероятности D -множеств сохраняются в массиве $Dsets(w)$.

В конце шага n алгоритм для всех $s = 1, \dots, S$ вычисляет вероятности $Prob(B(n, s))$ по формуле (3.4), которые сохраняются в массиве $Bsets$. Вероятности $Prob(R(n, s, \varepsilon))$ были найдены на втором этапе шага n .

В итоге, после выполнения шага N вычислена искомая вероятность $Prob(B(N, S))$.

4.2. Структура данных алгоритма SufPref-D

Данные, используемые при работе алгоритма SufPref-D, имеют следующую структуру.

Перманентные данные

Граф перекрытий $OvGraph(\mathcal{H}) = \langle OV(\mathcal{H}) \cup \mathcal{H}, E(\mathcal{H}) \rangle$. Дополнительно для каждой вершины $w \in (OV(\mathcal{H}) \cup \mathcal{H}) \setminus \varepsilon$ хранятся вероятности $Prob(back(w))$.

Изменяемые данные

Во время выполнения алгоритма хранятся следующие массивы данных:

1. Двумерный массив $Bsets$ размера $m \times S$. В момент начала выполнения шага n :

$$Bsets[k][s] = Prob(B(n - m + i, s)),$$

где $k = (n - m + i) \bmod m$; $i = 0, \dots, m - 1$ и $s = 1, \dots, S$. Если $n - m + i < m_0 + s - 1$, то $Bsets[k][s] = 0$.

2. Для каждой вершины $w \in OV(\mathcal{H})$ двумерный массив $Dsets(w)$ размера $m_w \times S$ ($m_w = \max\{|x| - |w| \mid x \in (\mathcal{H} \cup OV(\mathcal{H})) \& w = lp(x)\}$). В момент начала выполнения шага n :

$$Dsets(w)[k][s] = Prob(D(n - m_w + i, s, w)),$$

где $k = (n - m_w + i) \bmod m_w$; $i = 0, \dots, m_w - 1$ и $s = 1, \dots, S$. Если $n - m_w + i < m_0 + s - 1$, то $Dsets(w)[k][s] = 0$.

3. Для каждой вершины $w \in OV(\mathcal{H})$ локальный массив $Rsets(w)$ размера S , где $Rsets(w)[s] = Prob(R(n, s, w))$; $s = 1, \dots, S$. Данный массив используется только во время выполнения шага n алгоритма при вычислении вероятностей R -множеств.

Algorithm 1: SUFFPREF-D

Input: $\Sigma, N, S, \mathcal{H}$
Output: $Prob(B(N, S))$

// 1. Предварительный шаг
1 Построение графа перекрытий, инициализация массивов $Bsets$ и $Dsets$;

// 2. Основной алгоритм
2 **for** $n = m_0, \dots, N$ **do**
3 Инициализация массивов $Rsets$;
4 // 2.1. Находим $Prob(R(n, s, h))$
5 **for** $h \in \mathcal{H}; s = 1, \dots, S + 1$ **do**
6 **if** $n \geq |h| + s - 1$ **then**
7 Находим $Prob(R(n, \geq s, h))$ по формулам (3.5), (3.6);
8 **if** $s > 1$ **then**
9 $Prob(R(n, s - 1, h)) = Prob(R(n, \geq s - 1, h)) - Prob(R(n, \geq s, h))$;
10 $Rsets(sl(h))[s - 1] += Prob(R(n, s - 1, h))$;
11 **end**
12 **end**
13 // 2.2. Обходим $OVGraph$ по обратным ребрам. Находим
14 $Prob(R(n, s, w))$, $Prob(D(n, s, w))$ и $Prob(B(n, s))$
15 **for** $s = 1, \dots, S$ **do**
16 **if** $n \geq m_0 + s - 1$ **then**
17 **for** $w \in OV(\mathcal{H}) \setminus \varepsilon$ **do**
18 // К моменту рассмотрения w вероятность $Prob(R(n, s, w))$
19 уже вычислена и хранится в $Rsets(w)[s]$
20 Находим $Prob(D(n, s, w))$ по формуле (3.7) и сохраняем в $Dsets(w)$;
21 $Rsets(sl(w))[s] += Prob(R(n, s, w))$;
22 **end**
23 Находим $Prob(B(n, s))$ по формуле (3.4) и сохраняем в $Bsets$;
24 **end**
25 **end**
26 **end**

Теорема 2. Алгоритм SUFFPREF-D корректно вычисляет вероятность $Prob(B(N, S))$.

Доказательство. Алгоритм на шаге n ($n = m_0, \dots, N$) в цикле по s ($s = 1, \dots, S$) вычисляет вероятности $Prob(B(n, s))$ и дополнительные вероятности $Prob(R(n, \geq s, w))$, $Prob(R(n, s, w))$ ($w \in OV(\mathcal{H}) \cup \mathcal{H}$) и $Prob(D(n, s, w))$ ($w \in OV(\mathcal{H})$). Вычисление производится по формулам (3.2), (3.4), (3.5), (3.6) и (3.7). Вычисление по указанным формулам производится корректно, так как величины, которые фигурируют в формулах, вычислены заранее либо на текущем шаге n , либо на предыдущих шагах работы алгоритма и хранятся в структурах данных, описанных в подразделе 4.2. В итоге, на шаге N алгоритм находит требуемую вероятность $Prob(B(N, S))$.

□

Теорема 3. Пусть задана модель Бернулли распределения вероятностей символов в алфавите. Алгоритм SUFFPREF-D имеет временную сложность

$O(N \times S \times (||OV(\mathcal{H})|| + ||\mathcal{H}||)) + F(\mathcal{H})$ и емкостную сложность $O(S \times m \times ||OV(\mathcal{H})||) + F(\mathcal{H})$, где $F(\mathcal{H}) = O(m \times \mathcal{H})$ — оценка временной и емкостной сложности построения и хранения графа перекрытий.

Доказательство.

Временная сложность $F(\mathcal{H})$ построения графа перекрытий зависит от способа построения. Если граф перекрытий строится с помощью графа Ахо-Корасик, то $F(\mathcal{H}) = O(m \times ||\mathcal{H}||)$. Временная сложность основной части алгоритма определяется сложностью выполнения основного цикла **for** (см. строки 2–22 схемы алгоритма SufPref-D); цикл выполняется N раз. Предполагается, что $N \geq m_0$, иначе искомая вероятность $Prob(B(N, S))$ равняется нулю. В свою очередь, временная сложность одного прохода основного цикла определяется сложностью выполнения двух внутренних циклов **for**. Один проход каждого из этих циклов имеет временную сложность $O(1)$. При фиксированном значении n первый внутренний цикл **for** (см. строки 4–12) выполняется $(S+1) \times ||\mathcal{H}||$ раз, и второй внутренний цикл (см. строки 13–21) выполняется $S \times ||OV(\mathcal{H})||$ раз. Отсюда получаем указанную оценку временной сложности алгоритма.

Емкостная сложность алгоритма определяется объемом памяти требуемым для хранения на предварительном шаге графа Ахо-Корасик ($F(\mathcal{H}) = O(m \times ||\mathcal{H}||)$) и графа перекрытий ($O(||OV(\mathcal{H})|| + ||\mathcal{H}||)$), а также массивов $Bsets$ ($O(S \times m)$), $Dsets$ ($O(S \times m \times ||OV(\mathcal{H})||)$) и $Rsets$ ($O(S \times ||OV(\mathcal{H})||)$). Отсюда получаем указанную оценку емкостной сложности.

□

Исходя из указанных оценок, алгоритм SufPref-D имеет такие же временную и емкостную сложности, как и изначальный алгоритм SufPref, но в отличие от SufPref позволяет рассматривать наборы, содержащие образцы различной длины. Для марковских и скрытых марковских моделей оценки сложностей алгоритма SufPref-D также совпадают с оценками соответствующих сложностей SufPref, см. [10].

5. ОПТИМИЗАЦИЯ АЛГОРИТМА

В разделе 4 был представлен алгоритм SufPref-D для нахождения $Prob(B(N, S))$, который является модификацией SufPref. Однако этот алгоритм не является оптимальным, так как граф перекрытий, основная структура данных, используемая при вычислениях, может содержать $\sim^R$ -эквивалентные вершины. В данном разделе предложена оптимизация (алгоритм MinSufPref), использующая для вычислений $\sim^R$ -минимальный граф перекрытий, что позволяет значительно улучшить сложность алгоритма.

В подразделе 5.1 представлено обобщение множеств и формул из раздела 3 на классы $\sim^R$ -эквивалентности. И в подразделе 5.2 дано описание алгоритма MinSufPref и его структур данных.

5.1. Множества символьных последовательностей для классов $\sim^R$ -эквивалентности

Обобщим понятия R и D -множеств на классы эквивалентности $\hat{x} \in OV_R(\mathcal{H}) \cup \mathcal{H}_R$ следующим образом: $R(n, s, \hat{x}) = \cup_{x \in \hat{x}} R(n, s, x)$ и $D(n, s, \hat{x}) = \cup_{x \in \hat{x}} D(n, s, x)$ (для $\hat{x} \in OV_R(\mathcal{H})$). Отметим, что $R(n, s, \varepsilon) = R(n, s, \hat{\varepsilon}) = \cup_{\hat{x} \in OV_R(\mathcal{H}) \cup \mathcal{H}_R} R(n, s, \hat{x}) = \cup_{h \in \mathcal{H}} R(n, s, h)$.

Ниже представлены соотношения для D и R -множеств.

Утверждение 2. Пусть $\hat{w} \in OV_R(\mathcal{H}) \setminus \hat{\varepsilon}$, $s \geq 1$ и $n \geq m_0 + s - 1$. Тогда

$$D(n, s, \hat{w}) = \left(\bigcup_{\hat{v} \in LPRED_S_R(\hat{w})} D(n - |\hat{w}| + |\hat{v}|, s, \hat{v}) \cdot BACK_R(\hat{v}, \hat{w}) \right) \bigcup R(n, s, \hat{w}). \quad (5.8)$$

Доказательство. Используя соотношение (3.7), представим $D(n, s, \hat{w})$ в следующем виде:

$$D(n, s, \hat{w}) = \bigcup_{w \in \hat{w}} D(n, s, w) = \left(\bigcup_{w \in \hat{w}} D(n - |w| + |lp_R(w)|, s, lp_R(w)) \cdot back(lp_R(w), w) \right) \bigcup R(n, s, \hat{w}).$$

Сгруппируем $w \in \hat{w}$ таким образом, чтобы слова в одной группе имели $\sim^R$ -эквивалентные префиксы $lp_R(w)$. И заменив объединение $\bigcup_{w \in \hat{w}}$ в вышеуказанной формуле тождественным ему двойным объединением $\bigcup_{\hat{v} \in LPRED_S_R(\hat{w})} \bigcup_{w \in \hat{w}; lp_R(w) \in \hat{v}}$, получим соотношение:

$$D(n, s, \hat{w}) = \left(\bigcup_{\hat{v} \in LPRED_S_R(\hat{w})} \bigcup_{w \in \hat{w}; lp_R(w) \in \hat{v}} D(n - |\hat{w}| + |\hat{v}|, s, lp_R(w)) \cdot back(lp_R(w), w) \right) \bigcup R(n, s, \hat{w}). \quad (5.9)$$

При выводе (5.9) также использовалось свойство, что все слова из одного класса $\hat{x}$ имеют одинаковую длину $|\hat{x}|$.

Рассмотрим отдельно внутреннее объединение:

$$A = \bigcup_{w \in \hat{w}; lp_R(w) \in \hat{v}} D(n - |\hat{w}| + |\hat{v}|, s, lp_R(w)) \cdot back(lp_R(w), w). \quad (5.10)$$

Теперь сгруппируем $w \in \hat{w}$ таким образом, чтобы слова в одной группе имели одинаковые префиксы $lp_R(w)$, принадлежащие классу $\hat{v}$. И также заменив объединение в формуле (5.10) соответствующим двойным объединением, получим:

$$A = \bigcup_{v \in \hat{v}} \bigcup_{w \in \hat{w}; lp_R(w) = v} D(n - |\hat{w}| + |\hat{v}|, s, v) \cdot back(v, w).$$

Так как множитель $D(n - |\hat{w}| + |\hat{v}|, s, v)$ не зависит от w , то его можно вынести за пределы внутреннего объединения в выведенной формуле для A . Вместе с тем, используя соотношение (2.1), получаем:

$$\begin{aligned} A &= \bigcup_{v \in \hat{v}} \left(D(n - |\hat{w}| + |\hat{v}|, s, v) \cdot \bigcup_{w \in \hat{w}; lp_R(w) = v} back(v, w) \right) = \\ &= \left(\bigcup_{v \in \hat{v}} D(n - |\hat{w}| + |\hat{v}|, s, v) \right) \cdot BACK_R(\hat{v}, \hat{w}) = \\ &= D(n - |\hat{w}| + |\hat{v}|, s, \hat{v}) \cdot BACK_R(\hat{v}, \hat{w}). \end{aligned}$$

В итоге, заменив в двойном объединении в (5.9) внутреннее объединение на $D(n -$

$|\widehat{w}| + |\widehat{v}|, s, \widehat{v}) \cdot \text{BACK}_R(\widehat{v}, \widehat{w})$, приходим к соотношению, указанному в условии данного утверждения.

□

Рассуждая аналогично, получим подобную формулу для R -множеств. Пусть $\widehat{h} \in \mathcal{H}_R$, $s \geq 1$ и $n \geq |\widehat{h}| + s - 1$. Тогда

$$R(n, \geq 1, \widehat{h}) = \Sigma^{n-|\widehat{h}|} \cdot \widehat{h}; \quad (5.11)$$

$$R(n, \geq s, \widehat{h}) = B(n - |\widehat{h}|, s - 1) \cdot \widehat{h} + \bigcup_{\widehat{v} \in \text{LPREDS}_R(\widehat{h})} D(n - |\widehat{h}| + |\widehat{v}|, s - 1, \widehat{v}) \cdot \text{BACK}_R(\widehat{v}, \widehat{h}) \quad (5.12)$$

Для получения формул для нахождения $R(n, s, \widehat{w})$, перепишем соотношение (3.2) следующим образом:

$$R(n, s, \widehat{w}) = \bigcup_{\widehat{v} \in \text{OV}_R(\mathcal{H}) \cup \mathcal{H}_R; \widehat{w} = sl(\widehat{v})} R(n, s, \widehat{v}). \quad (5.13)$$

5.2. Алгоритм MINSUF_PREF

5.2.1. Структура данных

Структура данных MINSUF_PREF аналогична структуре данных алгоритма SufPref-D.

Перманентные данные

$\sim^R$ -минимальный граф перекрытий $\text{MinOvGraph}(\mathcal{H}) = \langle \text{OV}_R(\mathcal{H}) \cup \mathcal{H}_R, E_R(\mathcal{H}) \rangle$. Также для каждого класса $\widehat{w} \in (\text{OV}_R(\mathcal{H}) \cup \mathcal{H}_R) \setminus \widehat{\varepsilon}$ хранится список вероятностей $\{\text{Prob}(\text{BACK}_R(\widehat{v}, \widehat{w})) \mid \widehat{v} \in \text{LPREDS}_R(\widehat{w})\}$. Так как вероятность $\text{Prob}(\text{BACK}_R(\widehat{v}, \widehat{w}))$ является меткой прямого ребра $\langle \widehat{v}, \widehat{w} \rangle$ графа MinOvGraph , то общее число сохраненных вероятностей такого вида не превышает $\|E_R\|$.

Изменяемые данные

Во время выполнения алгоритма хранятся следующие массивы данных:

1. Двумерный массив $Bsets$; такой же, что описан в разделе 4.2.
2. Аналогично для каждой вершины $\widehat{w} \in \text{OV}_R(\mathcal{H})$ хранится двумерный массив $Dsets(\widehat{w})$ размера $m_w \times S$ ($m_w = \max\{|\widehat{x}| - |\widehat{w}| \mid \widehat{x} \in (\text{OV}_R(\mathcal{H}) \cup \mathcal{H}_R) \& \widehat{w} \in \text{LPREDS}_R(\widehat{x})\}$). В момент начала выполнения шага n :

$$Dsets(\widehat{w})[k][s] = \text{Prob}(D(n - m_w + i, s, \widehat{w})),$$

где $k = (n - m_w + i) \bmod m_w$; $i = 0, \dots, m_w - 1$ и $s = 1, \dots, S$. Если $n - m_w + i < m_0 + s - 1$, то $Dsets(\widehat{w})[k][s] = 0$.

3. Для каждой вершины $\widehat{w} \in \text{OV}_R(\mathcal{H})$ локальный массив $Rsets(\widehat{w})$ размера S , где $Rsets(\widehat{w})[s] = \text{Prob}(R(n, s, \widehat{w}))$; $s = 1, \dots, S$.

5.2.2. Описание алгоритма

Схема алгоритма **MinSufPref** также аналогична схеме алгоритма **SufPref-D**, см. алгоритмы 1 и 2. Основное отличие состоит в том, что **MinSufPref** использует $\sim^R$ -минимальный граф перекрытий *MinOvGraph*, вместо исходного графа перекрытий *OvGraph*. Во всех строках алгоритма 2 вершины графа перекрытий заменены их классами $\sim^R$ -эквивалентности, то есть вершинами *MinOvGraph*. Другое отличие состоит в том, что вероятности $Prob(R(n, s, \hat{h}))$ и $Prob(D(n, s, \hat{w}))$ вычисляются в циклах по $LPREDS_R(\hat{x})$ ($\hat{x} = \hat{h}$ или $\hat{w}$), см. строки 6 и 16 алгоритма 2. Номера формул для вычисления вероятностей в алгоритме 1 заменены номерами формул для вычисления соответствующих вероятностей для классов $\sim^R$ -эквивалентности из раздела 5.1.

Algorithm 2: MINSUF_PREF

Input: $\Sigma, N, S, \mathcal{H}$
Output: $Prob(B(N, S))$

// 1. Предварительный шаг
1 Построение $MinOvGraph(\mathcal{H}) = \langle OV_R(\mathcal{H}) \cup \mathcal{H}_R, E_R(\mathcal{H}) \rangle$; инициализация массивов $Bsets$ и $Dsets$;

// 2. Основной алгоритм
2 **for** $n = m_0, \dots, N$ **do**
3 Инициализация массивов $Rsets$;
 // 2.1. Находим $Prob(R(n, s, \hat{h}))$
4 **for** $\hat{h} \in \mathcal{H}_R: s = 1, \dots, S + 1$ **do**
5 **if** $n \geq |\hat{h}| + s - 1$ **then**
6 Находим $Prob(R(n, \geq s, \hat{h}))$ по формулам (5.11), (5.12); если $s > 1$, то
 вычисления производятся в цикле по $\hat{v} \in LPREDS_R(\hat{h})$;
7 **if** $s > 1$ **then**
8 $Prob(R(n, s - 1, \hat{h})) = Prob(R(n, \geq s - 1, \hat{h})) - Prob(R(n, \geq s, \hat{h}))$;
9 $Rsets(sl(\hat{h}))[s - 1] += Prob(R(n, s - 1, \hat{h}))$;
10 **end**
11 **end**
12 **end**
 // 2.2. Обходим $MinOvGraph$ по обратным ребрам. Находим
 $Prob(R(n, s, \hat{w}))$, $Prob(D(n, s, \hat{w}))$ и $Prob(B(n, s))$
13 **for** $s = 1, \dots, S$ **do**
14 **if** $n \geq m_0 + s - 1$ **then**
15 **for** $\hat{w} \in OV_R(\mathcal{H}) \setminus \hat{\varepsilon}$ **do**
16 // К моменту рассмотрения $\hat{w}$ вероятность $Prob(R(n, s, \hat{w}))$
 уже вычислена и хранится в $Rsets(\hat{w})[s]$
17 Находим $Prob(D(n, s, \hat{w}))$ по формуле (5.8) в цикле по
 $\hat{v} \in LPREDS_R(\hat{w})$ и сохраняем в $Dsets(\hat{w})$;
18 $Rsets(sl(\hat{w}))[s] += Prob(R(n, s, \hat{w}))$;
19 **end**
20 Находим $Prob(B(n, s))$ по формуле (3.4) и сохраняем в $Bsets$;
21 **end**
22 **end**

5.3. Корректность вычислений. Оценка сложности алгоритма

Теорема 4. Алгоритм $MINSUF_PREF$ корректно вычисляет вероятность $Prob(B(N, S))$.

Доказательство. Аналогично доказательству теоремы 2 с учетом того, что вычисления производятся не для одиночных вершин, а для соответствующих им классов R -эквивалентности.

□

В следующей теореме представлены оценки временной и емкостной сложностей

алгоритма **MinSufPref** для вероятностных моделей Бернулли.

Теорема 5. Пусть задана модель Бернулли распределения вероятностей символов в алфавите. Алгоритм **MinSufPref** имеет временную сложность $O(N \times S \times ||E_R(\mathcal{H})||) + F(\mathcal{H})$ и емкостную сложность $O(S \times m \times ||OV_R(\mathcal{H})|| + ||E_R(\mathcal{H})|| + P(\mathcal{H}))$, где $F(\mathcal{H})$ и $P(\mathcal{H})$ — временная и емкостная сложности предварительной части алгоритма.

Доказательство.

Временная сложность $F(\mathcal{H})$ предварительной части алгоритма **MinSufPref** определяется сложностью построения *MinOvGraph* и вспомогательной структуры *MinACGraph*. Временная сложность построения *MinOvGraph* линейна по числу ребер *MinACGraph*. Оценка $F(\mathcal{H})$ зависит от алгоритма построения *MinACGraph*, в худшем случае оценка равняется $O(m \times ||\mathcal{H}||)$, а в лучшем — $O(||E_R^A(\mathcal{H})||)$, где $||E_R^A(\mathcal{H})||$ — число ребер *MinACGraph*.

Временная сложность основной части **MinSufPref** (см. алгоритм 2) определяется сложностью выполнения основного цикла **for**, которая, в свою очередь, определяется сложностью выполнения двух внутренних циклов **for**. Совместное число операций, совершаемых при выполнении внутренних циклов **for** в строках 4–12 и 13–21 при фиксированных n и s пропорционально числу ребер *MinOvGraph*. Тогда временная сложность общего выполнения этих циклов для всех значений $n = m_0, \dots, N$ и $s = 1, \dots, S + 1$ имеет оценку $O(N \times S \times ||E_R(\mathcal{H})||)$. Отсюда получаем указанную в теореме оценку временной сложности.

Емкостная сложность. Доказательство оценок для емкостной сложности напрямую следует из описания структуры данных, см. подраздел 5.2.1.

□

Отметим, что основываясь на алгоритме из [16], граф *MinACGraph* может быть построен за линейное время по числу вершин графа Ахо-Корасик. В этом случае $F(\mathcal{H})$ и $P(\mathcal{H})$ имеют оценку $O(m \times ||\mathcal{H}||)$. Алгоритм из [16] требует построения изначального графа Ахо-Корасик, что может быть невозможным, когда $\mathcal{H}$ имеет огромный размер. В [12] мы разработали алгоритм построения *MinACGraph* для набора образцов, заданного словом-шаблоном, напрямую, т. е. без построения графа Ахо-Корасик; обе сложности этого алгоритма линейны по числу ребер *MinACGraph*. Также возможность прямого построения минимального автомата Ахо-Корасик рассматривалась в работе [3], однако предложенный там алгоритм не является линейным и не имеет подробного описания с четким доказательством. В дальнейшем планируется разработка алгоритма прямого построения графа *MinACGraph*, линейного по числу его ребер, для произвольного набора слов.

5.4. Дальнейшая оптимизация. Удаление терминальных вершин

Алгоритм **MinSufPref** может быть модифицирован следующим образом. Удалим в *MinOvGraph* терминальные вершины (листья) $\hat{h} \in \mathcal{H}_R$, а также входящие в них прямые ребра $\langle \hat{v}, \hat{h} \rangle$ и выходящие обратные ребра $\langle \hat{h}, \hat{r} \rangle$, где $\hat{v} \in LPREDS_R(\hat{h})$ и $\hat{r} = sl(\hat{h})$, заменив их на ребра вида $\langle \hat{v}, \hat{r} \rangle$. Такие ребра будем называть глубокими. Метка глубокого ребра совпадает с меткой соответствующего прямого ребра и равняется $BACK_R(\hat{v}, \hat{h})$. Пример модифицированного графа *MinOvGraph* представлен на рисунке 5.

Идея описанной процедуры состоит в следующем. В цикле по $\hat{h} \in \mathcal{H}_R$ алгоритм вычисляет вероятности R -множеств по формуле (5.12) (см. строки 4–8 Алгоритма 2). При вычислениях по второй части формулы, алгоритм обходит прямые ребра $\langle \hat{v}, \hat{h} \rangle$,

вычисляя сумму величин вида $Prob(D(\dots)) \cdot Prob(BACK_R(\hat{v}, \hat{h}))$. Затем в строке 9, следуя по обратному ребру, алгоритм добавляет найденные вероятности к ячейкам массива $Rsets(\hat{r})$. Однако можно явно не вычислять вероятности R -множеств, а напрямую добавлять величины $Prob(D(\dots)) \cdot Prob(BACK_R(\hat{v}, \hat{h}))$ к ячейкам $Rsets(\hat{r})$, следуя по глубоким ребрам $\langle \hat{v}, \hat{r} \rangle$. В итоге, $Rsets(\hat{r})$ будет содержать такие же величины.

Благодаря такой модификации, число вершин и ребер $\sim^R$ -минимального графа перекрытий уменьшится на $||\mathcal{H}_R||$, в результате чего уменьшится объем требуемой памяти и алгоритм будет выполнять на $||\mathcal{H}_R||$ меньше операций.

Подобная оптимизация графа перекрытий *OvGraph* также использовалась в алгоритме SufPref. В *OvGraph* удалены терминальные вершины, между вершинами v и r из $OV(\mathcal{H})$ построено глубокое ребро, если существует слово $h \in \mathcal{H}$, такое что $v = lp(h)$ и $r = sl(h)$. Метка глубокого ребра равняется $\cup_{h \in \mathcal{H}; v=lp(h); r=sl(h)} back(h)$.

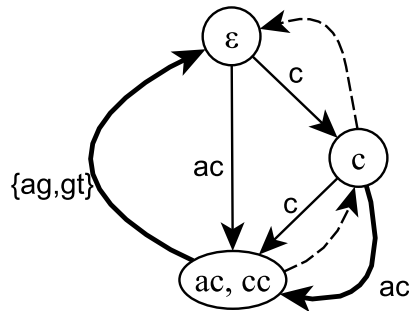


Рис. 5. $\sim^R$ -минимальный граф перекрытий для $\mathcal{H} = \{acag, acgt, sac, ccag, ccgt\}$ с удаленными терминальными вершинами. Удалены терминальные вершины, прямые и обратные ребра заменены глубокими. Глубокие ребра обозначены сплошными жирными дугами.

6. СРАВНЕНИЕ С ДРУГИМИ ПОДХОДАМИ

Для сравнения алгоритма MinSufPref с другими алгоритмами были проведены компьютерные эксперименты. Рассмотрены два множества наборов образцов (мотивов): 1) мотивы, заданные матрицами позиционных весов (МПВ) из базы данных HOCOMOCO [19]; 2) мотивы, заданные консенсусами из базы данных PROSITE [15]. Для каждого из мотивов построены следующие структуры: автомат Ахо-Корасик *ACA*ut, R -минимальный автомат Ахо-Корасик *MinACA*ut, граф перекрытий *OvGraph* и R -минимальный граф перекрытий *MinOvGraph*. В графах *OvGraph* и *MinOvGraph* удалены терминальные вершины, прямые и обратные ребра заменены глубокими ребрами, см. подраздел 5.4. Среди алгоритмов для вычисления p -значения: *ACA*ut используется в алгоритме АноPro [9]; *MinACA*ut используется в ряде алгоритмов, в частности, в SPatt [8]; *OvGraph* — в SufPref. Отметим, что, исходя из конструкции, *MinOvGraph* имеет наименьшее число вершин среди указанных структур. Временная сложность каждого из алгоритмов пропорциональна числу переходов или ребер используемой структуры. Обозначим через T_{AC} и T_{MAC} число переходов между состояниями автоматов *ACA*ut и *MinACA*ut соответственно. Так как автоматы *ACA*ut и *MinACA*ut являются полными, то $||T_{AC}|| = ||\Sigma|| \times ||Pref(\mathcal{H})||$ и $||T_{MAC}|| = ||\Sigma|| \times ||Pref_R(\mathcal{H})||$, где $||Pref(\mathcal{H})||$ ($||Pref_R(\mathcal{H})||$) — число состояний *ACA*ut (*MinACA*ut). Для рассмотренных мотивов было проведено сравнение чисел переходов и ребер построенных структур. Также с помощью алгоритмов MinSufPref и SufPref были вычислены и сравнены p -значения обнаружения отобранных

мотивов в случайных последовательностях. Для вычислений использовался процессор Intel Core i5-3230M с тактовой частотой 2.6 GHz. Подробная информация о мотивах, построенных структурах и вычисленных p -значениях опубликована в виде таблиц в [21].

6.1. Мотивы, заданные матрицами позиционных весов

База данных HOCOMOCO содержит МПВ, описывающие участки связывания транскрипционных факторов (ТФ) в геноме мыши и человека. Матрица МПВ и порог задают мотив, состоящий из слов одинаковой длины в алфавите нуклеотидов, характерных для соответствующего участка связывания. В ходе данной работы были рассмотрены матрицы МПВ для участков связывания ТФ в геноме человека и пороги, соответствующие уровню значимости 0.0001. Были отобраны мотивы длины 12–18, состоящие из 1000–4000000 слов, всего отобрано 152 мотива. Также с помощью алгоритмов SufPref и MinSufPref были вычислены p -значения при $S = 3$ и $N = 1000$ и равномерном распределении нуклеотидов; для каждого из мотивов вычисленные вероятности обоими алгоритмами совпали (с точностью до погрешности в вычислениях в 15-ом и больших десятичных знаках), см.[21], что еще раз подтверждает корректность предложенного подхода. Приведем результаты вычислений.

- *ACAut /MinOvGraph*

Среди рассмотренных мотивов количество переходов $||T_{AC}||$ автомата *ACAut* в 8–320 раз (в среднем в 34 раза) превышает количество ребер $||E_R(\mathcal{H})||$ графа *MinOvGraph*.

- *OvGraph /MinOvGraph*

В 90% случаев $||E(\mathcal{H})||$ незначительно превышает $||E_R(\mathcal{H})||$, не более чем в 2 раза; в 9% случаев $||E_R(\mathcal{H})||$ больше $||E(\mathcal{H})||$, но менее чем на 20%.

- *MinACAut /MinOvGraph*

В 87% случаев $||T_{MAC}||$ превышает $||E_R(\mathcal{H})||$, в среднем в 3.4 раза (максимальное превышение в 34 раза); в 13% случаев $||E_R(\mathcal{H})||$ превышает $||T_{MAC}||$, но не более чем в 3 раза. В таблице 1 представлены результаты анализа отношения $||T_{MAC}||/||E_R(\mathcal{H})||$, вычислены 10, 30, 50, 70 и 90-перцентили, где α -перцентиль — значение, ниже которого находится $\alpha\%$ данных. Процент случаев, для которых это отношение больше единицы снижается с ростом количества слов в мотивах, однако среднее значение отношения и среди мотивов больших размеров остается большим единицы.

Следует отметить, что в ходе проведения экспериментов обнаружены мотивы, заданные с помощью МПВ из базы данных HOCOMOCO, но не входящие в число отобранных, для которых $||T_{MAC}||$ значительно превышает $||E_R(\mathcal{H})||$. Например, для мотива, заданного матрицей SOX8.H12INVIVO.0.PSM.A и порогом 9.3728 (мотив имеет длину 17 и состоит из 171691 слов), $||T_{MAC}|| = 234188$ и $||E_R(\mathcal{H})|| = 312$, $||T_{MAC}||/||E_R(\mathcal{H})|| = 750.6$.

6.2. Мотивы, заданные консенсусами из базы данных PROSITE

База данных PROSITE содержит консенсусы и МПВ, описывающие различные семейства белков, а также их домены и сайты. Консенсус представляет собой слово, буквы которого являются подмножествами алфавита Σ , в данном случае Σ — алфавит

аминокислот. Слово в алфавите Σ принадлежит мотиву, если символ, стоящий в каждой его позиции, принадлежит соответствующей букве консенсуса. Было отобрано 776 мотивов, заданных консенсусами из PROSITE, из них 771 состоят из слов одинаковой длины, 5 — из слов различной длины. Отметим, что *SufPref* допускает лишь наборы, состоящие из слов одинаковой длины. Рассмотренные мотивы содержат от 1 до $10^{50.3}$ слов. В 262-х мотивах слова не имеют перекрытий, поэтому число вершин и ребер *OvGraph* и *MinOvGraph* равно 1. Отметим, что для мотива, заданного консенсусом, нами был разработан прямой алгоритм построения минимального автомата, без построения изначального автомата Ахо-Красик, что значительно повышает эффективность алгоритма *MinSufPref*. Для каждого мотива были также построены *MinACAut* и *MinOvGraph*. Число состояний автомата *ACAut* имеет оценку $O(m \times ||\mathcal{H}||)$, поэтому для большей части мотивов его построение невозможно, также невозможно построение графа *OvGraph*, так как для этого требуется наличие предварительно построенного автомата *ACAut*. Автомат *ACAut* и граф *OvGraph* построены лишь для мотивов, содержащих не более 10^7 слов. Для отобранных мотивов также были вычислены p -значения при $S = 1$ и $N = 1000$ и равномерном распределении аминокислот. Общее время построения структур и вычисления вероятностей составило 697 секунд. Приведем результаты вычислений.

- *ACAut /MinOvGraph*

ACAut и *OvGraph* построены лишь для 324 мотивов, из которых 189 не имеют перекрытий между словами. Среди этих мотивов $||T_{AC}||$ превышает $||E_R(\mathcal{H})||$ в $80 - 10^{8.7}$ раз.

- *OvGraph /MinOvGraph*

$||E(\mathcal{H})||$ превышает $||E_R(\mathcal{H})||$ для всех мотивов, для которых был построен *OvGraph*, превышение в 1–1050 раз.

- *MinACAut /MinOvGraph*

Во всех, кроме трёх случаев, $||T_{MAC}||$ превышает $||E_R(\mathcal{H})||$, в среднем в 265.55 раз, медианное значение равно 129.71, см. таблицу 1. Максимальное превышение в 21360 раз, в соответствующем мотиве слова не имеют перекрытий. Среди этих трёх случаев максимальное превышение $||E_R(\mathcal{H})||$ над $||T_{MAC}||$ — в 7.8 раз.

В работе [8] предложен алгоритм SPAT, использующий для вычисления точного p -значения минимальный автомат *MinACAut*. В [8] также проводились вычисления для мотивов, заданных консенсусами из базы данных PROSITE. Авторы работы рассмотрели трансмембранные белки из базы данных SWISS-PROT [20], общей длиной 84192 аминокислоты. Были отобраны те консенсусы из PROSITE, для которых мотивы были обнаружены по крайней 1 раз в последовательностях из SWISS-PROT. Затем с помощью программы SPAT вычислили точное p -значение при $S = S_{oc}$ и $N = 84192$, где S_{oc} — число обнаруженных вхождений, вычисление производилось процессором Intel 2.6 GHz P4. Самое большое время заняло вычисление p -значения при $S = 15$ для консенсуса с ID = PS00029, оно составило 5173.3 секунды. Мы также вычислили p -значение для этого консенсуса с теми же параметрами с помощью программы *MinSufPref*, время вычислений составило всего лишь 319 секунд (в 16.2 раз быстрее), что говорит о высокой эффективности разработанных в ходе данной работы алгоритмов. Для мотива, заданного консенсусом с ID = PS00029, *MinACAut* имеет 20480 состояний и 409600 переходов; *MinOvGraph* имеет 4890 вершин и 47345 ребер.

Таким образом, *MinSufPref* показал высокую эффективность по сравнению с другими алгоритмами. Для мотивов из *PROSITE* алгоритм *MinSufPref* значительно превосходит по эффективности рассмотренные алгоритмы. Для мотивов из базы данных *HOCOMOCO* размеры *OvGraph* и *MinOvGraph* мало отличаются, т. е. *MinSufPref* и *SufPref* сравнимы по эффективности. Однако для большинства мотивов из *PROSITE* алгоритм *SufPref* имеет большую сложность или не применим.

Таблица 1. Отношение числа переходов *MinACAut* к числу ребер *MinOvGraph*. В таблице для мотивов из баз данных *HOCOMOCO* и *PROSITE* указаны минимальное, максимальное и 10, 30, 50, 70 и 90-перцентили значения отношения $||T_{MAC}||/||E_R||$.

База данных	процентиль	$ T_{MAC} / E_R $
HOCOMOCO	min	0.32
	10%	0.86
	30%	1.55
	50%	2.27
	70%	3.52
	90%	7.3
	max	34.03
PROSITE	min	0.13
	10%	23.93
	30%	62
	50%	129.71
	70%	240
	90%	520
	max	21360

7. ВЫВОДЫ

В работе предложен алгоритм вычисления *P*-значения обнаружения слов из заданного набора в случайной последовательности. За основу был взят алгоритм *SufPref* для нахождения *P*-значения, производящий вычисления путем обхода графа перекрытий. Алгоритм *SufPref* был модифицирован таким образом, чтобы появилась возможность работать с наборами, состоящими из слов различной длины. Для этого был изменен порядок вычисления вероятностей *D* и *R*-множеств, а также структура данных, хранящихся во время работы алгоритма. Данная модификация не изменяет оценки временной и емкостной сложности работы алгоритма *SufPref*.

На множестве префиксов слов из заданного набора введено отношение $\sim^R$ -эквивалентности. Граф перекрытий заменен $\sim^R$ -минимальным графом перекрытий, вершины которого соответствуют классам $\sim^R$ -эквивалентности вершин изначального графа перекрытий. Предложен алгоритм построения $\sim^R$ -минимального графа перекрытий

с использованием вспомогательной структуры — $\sim^R$ -минимального графа Ахо-Корасик. $\sim^R$ -минимальный граф перекрытий является подграфом $\sim^R$ -минимального графа Ахо-Корасик. Также преобразованы соотношения для вычисления вероятностей множеств символьных последовательностей таким образом, чтобы была возможность производить вычисления вероятностей для классов $\sim^R$ -эквивалентности, вместо одиночных вершин графа перекрытий.

Разработанный алгоритм MINSUF_PREF, совмещает в себе обе указанные модификации. Временная и пространственная сложности основной части алгоритма MINSUF_PREF линейный по числу вершин и ребер $\sim^R$ -минимального графа перекрытий. Минимизация графа перекрытий позволяет значительно повысить эффективность алгоритма.

Проведенные компьютерные эксперименты показали высокую эффективность MINSUF_PREF по сравнению с другими алгоритмами. Особенно для наборов образцов, заданных с помощью консенсусов. Превосходство над другими алгоритмами повышается с увеличением размера алфавита.

Данная работа является продолжением проекта, начатого под руководством М.А. Ройтберга в сотрудничестве с М. Regnier.

СПИСОК ЛИТЕРАТУРЫ

1. Qian Z., Lu L., Qi L., Li Y. An efficient method for statistical significance calculation of transcription factor binding sites. *Bioinformatics*. 2007. V. 2. No. 5. P. 169–174. doi: [10.6026/97320630002169](https://doi.org/10.6026/97320630002169)
2. Berman B.P., Pfeiffer B.D., Lavery T.R., Salzberg S.L., Rubin G.M., Eisen M.B., Celniker S.E. Computational identification of developmental enhancers: conservation and function of transcription factor binding-site clusters in *Drosophila melanogaster* and *Drosophila pseudoobscura*. *Genome Biol.* 2004. V. 5. P. R61. doi: [10.1186/gb-2004-5-9-r61](https://doi.org/10.1186/gb-2004-5-9-r61)
3. Martin D.E.K., Noe L. Faster exact distributions of pattern statistics through sequential elimination of states. *Annals of the Institute of Statistical Mathematics*. 2017. V. 69. P. 231–248. doi: [10.1007/s10463-015-0540-y](https://doi.org/10.1007/s10463-015-0540-y)
4. Martin D.E.K. Computation of exact probabilities associated with overlapping pattern occurrences. *Wiley Interdisciplinary Reviews: Computational Statistics*. 2019. V. 11. No. 5. doi: [10.1002/wics.1477](https://doi.org/10.1002/wics.1477)
5. Moshkovitz I., Barron Y. The waiting time distribution of competing patterns in Markov-dependent Bernoulli trials. *Axioms*. 2025. V. 14. No. 3. P. 221. doi: [10.3390/axioms14030221](https://doi.org/10.3390/axioms14030221)
6. Martin D.E.K., Aston J.A.D. Waiting time distribution of generalized later patterns. *Computational Statistics and Data Analysis*. 2008. V. 52. No. 11. P. 4879–4890. doi: [10.1016/j.csda.2008.04.019](https://doi.org/10.1016/j.csda.2008.04.019)
7. Nicodeme P., Salvy B., Flajolet P. Motif statistics. *Theoretical Computer Science*. 2002. V. 287. No. 2. P. 593–617. doi: [10.1016/S0304-3975\(01\)00264-X](https://doi.org/10.1016/S0304-3975(01)00264-X)
8. Nuel G. Pattern Markov chains: optimal Markov chain embedding through deterministic finite automata. *Journal of Applied Probability*. 2008. V. 45. No. 1. P. 226–243. URL: <http://www.jstor.org/stable/27595936> (accessed 31.07.2025).
9. Boeva V., Clement J., Regnier M., Roytberg M.A., Makeev V.J. Exact p-value calculation for heterotypic clusters of regulatory motifs and its application in computational annotation of cis-regulatory modules. *Algorithms for molecular biology*. 2007. V. 2. No. 13. doi: [10.1186/1748-7188-2-13](https://doi.org/10.1186/1748-7188-2-13)
10. Regnier M., Furlotova E., Yakovlev V., Roytberg M. Analysis of pattern overlaps and exact computation of *P*-values of pattern occurrences numbers: case of Hidden Markov Models.

- Algorithms for molecular biology*. 2014. V. 9. No. 25. doi: [10.1186/s13015-014-0025-1](https://doi.org/10.1186/s13015-014-0025-1)
11. Regnier M., Kirakossian Z., Furletova E.I., Roytberg M.A. A word counting graph. In *London Algorithmics 2008: Theory and Practice (Texts in Algorithmics)*. 2008. 31 p.
 12. Furletova E., Holub J., Regnier M. Minimized compact automaton for clumps over degenerate patterns. 2019. URL: <https://hal.inria.fr/hal-01940837> (accessed 31.07.2025).
 13. Aho A.V., Corasick M.J. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*. 1975. V. 18. No. 6. P. 333–340. doi: [10.1145/360825.360855](https://doi.org/10.1145/360825.360855)
 14. Фурлетова Е.И., Ройтберг М.А. О числе перекрытий слов в паттернах. *Математическая биология и биоинформатика*. 2016. Т. 11. № 1. С. 14–23. doi: [10.17537/2016.11.14](https://doi.org/10.17537/2016.11.14)
 15. Sigrist C.J.A., Castro E., Cerutti L., Cuche B.A., Hulo N., Bridge A., Bougueleret L., Xenarios I. New and continuing developments at PROSITE. *Nucleic Acids Res.* 2013. V. 41. No. D1. P. D344–D347. doi: [10.1093/nar/gks1067](https://doi.org/10.1093/nar/gks1067)
 16. Фурлетова Е.И. Минимизация автомата Ахо-Корасик за линейное время. *Дискретная математика*. 2023. Т. 35. № 2. С. 125–142. doi: [10.4213/dm1730](https://doi.org/10.4213/dm1730)
 17. AitMous O., Bassino F., Nicaud C. Building the minimal automaton of $A * X$ in linear time, when X is of bounded cardinality. *Combinatorial Pattern Matching. Lecture Notes in Computer Science*. 2010. V. 6129. doi: [10.1007/978-3-642-13509-5_25](https://doi.org/10.1007/978-3-642-13509-5_25)
 18. Teixeira M.C., Monteiro P.T., Palma M., Costa C., Godinho C.P., Pais P., Cavaleiro M., Antunes M., Lemos A., Pedreira T., Sá-Correia I. YEASTRACT: an upgraded database for the analysis of transcription regulatory networks in *Saccharomyces cerevisiae*. *Nucleic Acids Res.* 2018. V. 46. No. D1. P. D348–D353. doi: [10.1093/nar/gkx842](https://doi.org/10.1093/nar/gkx842)
 19. Kulakovskiy I.V., Vorontsov I.E., Yevshin I.S., Sharipov R.N., Fedorova A.D., Rumynskiy E.I., Medvedeva Y.A., Magana-Mora A., Bajic V.B., Papatsenko D.A. HOCOMOCO: towards a complete collection of transcription factor binding models for human and mouse via large-scale ChIP-Seq analysis. *Nucleic Acids Res.* 2018. V. 46. No. D1. P. D252–D259. doi: [10.1093/nar/gkx1106](https://doi.org/10.1093/nar/gkx1106)
 20. Gasteiger E., Jung E., Bairoch A. SWISS-PROT: Connecting biological knowledge via a protein database. *Curr. Issues Mol. Biol.* 2001. V. 3. No. 3. P. 47–55.
 21. Фурлетова Е.И. Сравнение минимального графа перекрытий с другими структурами. *Mendeley Data*. 2025. V. 1. doi: [10.17632/27y9yy5bnz.1](https://doi.org/10.17632/27y9yy5bnz.1)

Рукопись поступила в редакцию 21.06.2025.

Переработанный вариант поступил 27.07.2025.

Дата опубликования 12.08.2025.

An Algorithm for Computation of P -Value of Patterns Occurrences in a Random Sequence. Overlap Graph Minimization

Furletova E.I.

Institute of Mathematical Problems of Biology RAS, Keldysh Institute of Applied Mathematics of RAS, Pushchino, Russia

Abstract. We have developed algorithm MINSUFREF for computing of the probability (P -value) of S occurrences of words from a set $\mathcal{H}$ in a random sequence of length N . It is widely used in computer sciences as a criterion for selecting of patterns of a special kind. In particular, in bioinformatics, when searching for functional fragments in biological sequences. The proposed algorithm is based on the previously developed algorithm SUFREF for exact P -value computation. SUFREF uses an overlap graph for calculations, the nodes of the graph correspond to the words from $\mathcal{H}$ and their overlaps. The restrictions of SUFREF are that it admits only sets consisting of words of a same length, and also the algorithm may not be applicable for sets consisting of a large number of words. In this paper, the algorithm is expanded to sets consisting of words of arbitrary lengths, but with the constraint that a set can not contain words occurring in other its words. Also the $\sim^R$ -equivalence relation on the prefixes of pattern words is introduced, the overlap graph is minimized in accordance to this relation. The space and time complexities of the main part of MINSUFREF are linear on the number of nodes and edges of the minimized overlap graph. The time complexity is independent of the alphabet size and the lengths of words in $\mathcal{H}$. We have performed computer experiments that showed high efficiency of MINSUFREF compared to similar algorithms.

Key words: P -value, patterns occurrences, an overlap, Aho-Corasick automaton, automaton minimization