University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1995-60

Finding a Good Collection of Patterns Covering a Set of Sequences *

Alvis Brāzma, Esko Ukkonen, Jaak Vilo

Helsinki, December 1995

Report C-1995-60

University of Helsinki
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

* The research reported here has been supported by Academy of Finland, Finnish Center for International Mobility (CIMO) and a Research Grant 93.593 of Latvian Council of Sciences.

Finding a Good Collection of Patterns Covering a Set of Sequences †

Alvis Brāzma, Esko Ukkonen, Jaak Vilo

Report C-1995-60
22 pages
Helsinki, December 1995

Abstract. We consider a problem of learning of unions of pattern languages from positive examples. We consider three different classes of patterns - regular patterns, substring patterns and the so called PROSITE patterns. By regular patterns we understand patterns where each variable symbol can appear only once. By substring patterns we understand a subclass of regular patterns of the type $x\alpha y$, where x and y are variables and α is a string of constant symbols. The PROSITE patterns is a class of patterns used for classification of bio-sequences in PROSITE database. We present an algorithm which, given a set of sequences, finds a 'good' collection of patterns 'covering' this set. The notion of a 'good covering' is defined as the most probable collection of patterns likely to produce the examples in some simple and natural probabilistic model. We show that this criterion is equivalent to the so called Minimum Description Length (MDL) principle. We present a polynomial-time algorithm for approximating the optimal cover within a logarithmic factor and prove its performance guarantees. In the case of substring patterns the running time of the algorithm is almost linear.

Keywords: Pattern discovery, strings, MDL, PROSITE patterns, union of patterns, weighted set cover

CR Classification: I.2.6 F.2.2 J.3

†The research reported here has been supported by Academy of Finland, Finnish Center for International Mobility (CIMO) and a Research Grant 93.593 of Latvian Council of Sciences.

Contents

1. Introduction

The general problem we are considering in this paper is - given a set of sequences believed to share some common property, find some simple 'characterization' of this set. For instance, the set may consist of sequences representing the primary structure of some protein molecules. The set may contain noise, i.e., some of the sequences may actually not possess the property. The 'reasons' why the sequences may possess the property may be different for different sequences, therefore a simple single 'characterization' covering all sequences may not be possible.

For a rigorous treatment of the problem we should precisely define what we mean by 'characterization'. As already suggested by the example above, we are primarily interested in *biosequences*, the sequences corresponding to biological macromolecules. Depending on what properties of biosequences we want to consider, we should choose an appropriate formalism. Choosing the formalism is always a compromise between what can be realistically efficiently computed from examples, and what is expressive enough to capture the properties we are interested in. The pattern languages of Angluin [Ang80] are natural formalism for describing common patterns in strings. In many applications different regular subclasses of the pattern languages (such a pattern language belongs also to the set of regular languages) are sufficient at least for reasonable approximations. In molecular bioinformatics even such a simple subclass of regular patterns as substring patterns, i.e., a presence of a certain subword in strings, has been used for prediction and classification [WAG84, Sta89]. A number of more powerful subclasses of regular patterns have also been introduced by biologists [SS90, SAC90, OUSK92, NG94], and even databases of such patterns have been created (e.g., PROSITE database [Bai92]). In the current paper we will consider similar subclasses of regular patterns for characterization of sequences. We will consider three cases, first a class of substring patterns, then the class of regular patterns in a sense of [Shi83], and then a class of patterns of the type that is used in PROSITE database [Bai92], which include most of the subclasses of regular language patterns used by biologists. We call them simply patterns.

Quite a large number of algorithms have been developed for automatic discovery of patterns in a given set of biosequences, for instance [SAC90, AKM+92, NG94, JCH95, SVS95] (for a survey see [BJEG95]). To have applications to biology, such algorithms have to be able to cope with noise, i.e., with the fact, that some of the given sequences can actually be misclassified or contain errors. A common way, how this problem is being dealt with in most of these algorithms, is to look for patterns common to at least a certain fixed percentage of examples. This actually requires to assume a certain upper bound on the noise level, say 30%, meaning that at least 70% of the examples should be correct. Then the algorithm simply looks for patterns common to at least 70% of sequences. Such an algorithm will usually not work, if in reality the level of noise is, say, 31%. One heuristic approach to deal with the problem in case of very simple patterns is developed in [WB95], but there the examples are assumed to be correctly prealigned and have equal length.

In this paper we propose a more flexible approach for automatic discovery of patterns where we do not have to make any assumptions about the noise level. Moreover, our algorithm does not necessarily look for a single pattern common to a fixed percentage of the sequences, but instead, it tries to find *a set of patterns* "covering" the sequences. More formally it means that our target class is not just single patterns, but unions of patterns.

A nontrivial problem in this case is – how to define what is a 'good' set of patterns for the given set of sequences. Obviously, we want to define it so that the trivial answers – an empty pattern, or a simple union of all examples – would automatically be excluded. Therefore, a trivial application of Occam's Razor is not appropriate here.

A similar problem for the case where the number of patterns is bounded by some given constant k has been considered in [ASO94]. In this case it is possible to define the best set of patterns in a relatively simple way - as 'a minimally general collection of at most k regular patterns that explains all the positive examples'. However, like in the case of noisy examples, this implies assuming some constant k (the upper bound on the number of patterns). In our approach, we do not want to make such assumptions. Their approach, however, does not work if we omit the phrase 'at most k'.

At first glance our aim to find the 'best' set of unbounded number of patterns from only positive examples may seem impossible, since the language defined by an unbounded number of unions of patterns (even subword patterns) does not have finite *elasticity* [Wri89] and it is not immediately clear how to avoid simultaneously the two trivial hypothesis – that consisting of union of all examples, and the universal expression. Moreover, by using arguments of [Gol67] it can be actually shown, that the union of unbounded number of patterns is not inferable in the limit from only positive examples. But this contradicts our intuition in the sense that in many cases for a human being it is quite easy to find the nontrivial 'best' set of patterns for a set of positive examples without any *a priori* bounds on the number of patterns.

In [KMU95] a different approach to learning of the unions of patterns, based on the so called Rissanen's Minimum Description Length (MDL) principle [Ris78, LV95a], has been introduced. According to MDL principle, the best pattern set is the one which minimizes the sum of

- the length (in bits) of the patterns; and
- the length (in bits) of data when encoded with the help of the patterns.

The class of patterns considered in [KMU95] is the so called simple patterns where variables correspond to unit-length substitutions (therefore all words in the language have equal length and the language is finite) and proves, that the best pattern according to MDL principle, while using some natural coding scheme, can be approximated within a logarithmic factor in polynomial time (in some sense).

In this paper we use a similar approach for different classes of patterns, and additionally, we substantiate the particular coding scheme from the point of view of Bayesian learning. We assume certain simple and natural properties about the probability distribution of different patterns and examples, and then look for the patterns having the highest conditional probability, assuming, that they have 'generated' the given examples. The pattern sets with the highest such probability are defined as the 'best'. We show that under these assumptions, the 'best' pattern set is the same as the best according to MDL principle in a simple and natural encoding scheme. Unfortunately the problem of finding the pattern set ensuring the minimum description length is NP-hard even in the case of substring patterns. Therefore, instead of trying to solve the problem precisely, we try to approximate the optimal solution. We develop a greedy polynomial-time algorithm for approximating MDL and prove that the algorithm approximates the optimal solution within a logarithmic factor. We show that for substring patterns the algorithm can be implemented in the way that it approximates the optimal set of patterns in time $O(nN)$, where n is the number of examples, and N is total length of examples. In case of more complicated patterns (PROSITE type), the algorithm is still polynomial in the total length of examples plus the number of candidate patterns it has to check. The candidate patterns can be provided to the algorithm by methods developed in earlier bio-pattern discovery algorithms (e.g., [JCH95]). Thus, in the case of more complicated patterns, we effectively provide a method how to generalize the existing pattern discovery algorithms so that they are applicable also for the case of unions of patterns of unbounded number, and for dealing with noise without assuming any bounds on its level. (MDL principle automatically provides method how to deal with noise without pre-assuming any fixed bounds on it.)

The structure of this paper is the following. In the next section we show that under some simple assumptions regarding the probability distribution, the Bayesian and MDL approaches are equivalent. In Section 3 we prove the MDL approximation results and describe the respective algorithm. In conclusions we discuss some of the implications of our results and possible future research directions.

2. Relations between Bayesian and MDL approaches

Let Σ be a finite alphabet, and let $* \notin \Sigma$. We define a *regular pattern* π as a nonempty string over the alphabet $\Sigma \cup \{*\}$. We define the language $\mathcal{L}(\pi)$ generated by the pattern π as the set of all words obtained from π by replacing $*$ characters in π by arbitrary strings over Σ^*. A *substring pattern* is a regular pattern of the type $*\varepsilon*$, where $\varepsilon \in \Sigma^+$. We say that a sequence α *contains* or *matches* the pattern π, if $\alpha \in \mathcal{L}(\pi)$.

Our aim is, given a set $\alpha_1 \ldots \alpha_n$ of strings over Σ, to find a 'good' collection (i.e., a set) of patterns $\pi_1, \ldots, \pi_k$ such that

$$\{\alpha_1, \ldots, \alpha_n\} \subset \mathcal{L}(\pi_1) \cup \ldots \cup \mathcal{L}(\pi_k).$$

We want to define the notion of a 'good' collection in some natural way corresponding to our intuition. For instance, we want to exclude such trivial solutions as $k = n$ and $\pi_i = \alpha_i$, for $i = 1, \ldots, n$, or $k = 1$ and $\pi_1 = *$.

For the purpose of defining what is a 'good' pattern we will assume that strings in the language $L(\pi)$ are distributed according to some simple probability distribution. Moreover, we will also assume that patterns themselves have some probabilities of occurring - longer patterns having smaller probabilities in accordance with Occam's Razor principle (which in this context means that smaller hypothesis are more likely). We will define the best pattern set $\{\pi_1, \ldots, \pi_k\}$, given the data $\{\alpha_1, \ldots, \alpha_n\}$, as the most probable one assuming the given probability distribution.

Let $B_1, \ldots, B_k$ be a partition of A (i.e., $B_1, \ldots, B_k$ are disjoint and their union is A) and let $\Pi = \{\pi_1, \ldots, \pi_k\}$ be a set of patterns such that the pattern π_j matches all the sequences of the set $B_j = \{\beta_1^j, \ldots, \beta_{l_j}^j\}$. We call $\Omega = \{(\pi_1, B_1), \ldots, (\pi_k, B_k)\}$ a *cover* of A. We call Π the *pattern set* of Ω. Thus, the notion of the cover includes both - the partition and the patterns, and our problem can be formulated as finding the best cover.

Let us denote the length of a string α by $|\alpha|$. For a set $A = \{\alpha_1, \ldots, \alpha_n\}$ we define $\|A\| = \sum_{j=1}^{n} |\alpha_j|$, and $|A| = n$.

Let \$ be one more character such that $\$ \notin \Sigma$. Let us assume some probability distribution P_1 over the set $\Sigma \cup \{*\} \cup \{\$\}$, i.e., $P_1(a_1) + \cdots + P_1(a_m) + P_1(*) + P_1(\$) = 1$. Let $\pi = \pi'\$$, $\pi' \in (\Sigma \cup \{*\})^*$. Let $P_1(\$) = q$. Let us assume that patterns π are generated in a random Bernoulli process consecutively outputting letters from $\Sigma \cup \{*\} \cup \{\$\}$ with probabilities P_1 and \$ signaling the end of the pattern.

Let $\pi = \pi'\$$ be such a string, where $\pi' = e_1 \ldots e_h \in (\Sigma \cup \{*\})^*$. We define the *a priori* probability of the pattern π as

$$P_1(\pi) = P_1(e_1) \cdots P_1(e_h) P_1(\$). \tag{1}$$

Thus, the probability q_l that a pattern has the length l is

$$q_l = (1-q)^{(l-1)} q.$$

Note that a similar length distribution has been used in [QR89] for MDL learning of decision trees. Note also that $\sum_{\pi \in (\Sigma \cup \{*\})^*} P_1(\pi) = 1$.

Let $\Pi = \{\pi_1, \ldots, \pi_k\}$. We define the *a priori* probability of the pattern set Π as

$$P_1(\Pi) = P_1(\pi_1) \cdots P_1(\pi_k) \cdot (1-r)^k r \tag{2}$$

for some $0 < r < 1$. Parameters q and r determine the *a priori* average length of patterns and how many patterns we want on average. The distribution is consistent with Occam's Razor: smaller number of shorter patterns are considered as *a priori* more probable. Note that the sum of probabilities of all pattern sets is normalized to 1, i.e., $\sum_{\Pi} P_1(\Pi) = 1$.

Next, let us assume that sequences α are obtained by filling in the characters $*$ in π_i' by some random substrings generated in a similar Bernoulli process. More precisely, let $P(a_i)$ be some probability distribution over $a_i \in \Sigma$. In practice we take $P(a_i)$ proportional to the relative frequency f_i of a_i in a sequence database (or in A). Let $\$_2$ be one more character disjoint with Σ with a probability $P_2(\$_2) = p$, and let us normalize the probabilities $P(a_i)$ to $P_2(a_i) = (1-p)P(a_i)$. Let some sequences γ_j be generated by outputting characters from $\Sigma \cup \{\$_2\}$ with probabilities P_2 and $\$_2$ signaling the end.

Let $\gamma_j = \gamma_j'\$_2$, where $\gamma_j' = c_1 \dots c_l \in \Sigma^*$. We define $P_2(\gamma_j) = P_2(c_1) \cdot \ldots \cdot P_2(c_l) \cdot P_2(\$_2)$. We assume that biosequences are obtained from patterns π by filling in the characters $*$ with sequences γ_j' in π'.

Let π be a pattern with h characters $*$. Let $\gamma_i = \gamma_i'\$$ and $\gamma_i' \in \Sigma^*$ $(i = 1, \dots, h)$. We define the conditional probability of a sequence α given π as

$$Pr(\alpha|\pi) = \begin{cases} P_2(\gamma_1) \cdots P_2(\gamma_h), \text{where substituting } \gamma_1', \dots, \gamma_h' \text{ for } * \text{ in } \pi \text{ gives } \alpha, & \text{if } \alpha \in \mathcal{L}(\pi'); \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Given a set $A = \{\alpha_1, \dots, \alpha_n\}$, we define

$$Pr(A|\pi) = Pr(\alpha_1|\pi) \cdot \ldots \cdot Pr(\alpha_n|\pi). \tag{4}$$

Similarly, if we are given a cover $\Omega = \{(\pi_1, B_1), \dots, (\pi_k, B_k)\}$ of the set A, then we define

$$Pr(A|\Omega) = Pr(B_1|\pi_1) \cdot \ldots \cdot Pr(B_k|\pi_k), \tag{5}$$

and

$$Pr(A|\Pi) = \max \{Pr(A|\Omega) \mid \Pi \text{ is the pattern set of } \Omega\}. \tag{6}$$

The definition of $Pr(A|\Pi)$ through maximum means that we are defining the best pattern set not as just the most probable, but as the one providing the most probable way how, given the patterns, the strings A can be obtained. This is consistent with an assumption that a presence of a pattern in a sequence determines some bio-physical property of the respective molecule (corresponding to that sequence). If only one pattern $\pi \in \Pi$ is present in a sequence then the respective molecule should always have this property. If more than one pattern is present in a sequence, then the 'strongest' pattern wins, where the property of being the 'strongest' is determined not only by this sequence, but to some extent also by the overall composition of the set.

The probability $p = P_2(\$_2)$ can be estimated from the average length of biosequences in the respective family. The ratio $P_1(\$)$ to $P_2(\$_2)$ determines the most probable relative length of strings to patterns. By varying this ratio we will slant the fitness of patterns towards longer patterns common to fewer strings or vice versa. Normally it is very small, and sometimes can be assumed to be 0. Note that this model in essence is similar to that of [NLL95].

If we assume the above defined probability distributions for the pattern set and the data set, we will say that we are assuming the *Bernoulli model for regular patterns.*

We define $Pr(A) = \sum_\Pi Pr(A|\Pi)P_1(\Pi)$, and apply Bayesian theorem obtaining

$$Pr(\Pi|A) = \frac{Pr(A|\Pi)P_1(\Pi)}{Pr(A)}. \tag{7}$$

We define Π maximizing $Pr(\Pi|A)$, i.e.,

$$\max_{\Pi} Pr(\Pi|A) = \max_{\Pi} \frac{Pr(A|\Pi)P_1(\Pi)}{Pr(A)} = \max_{\Omega} \frac{Pr(A|\Omega)P_1(\Pi)}{Pr(A)} \tag{8}$$

as the *most probable pattern set* for the given A in Bernoulli model. To maximize (8) we can just as well maximize $\log P(\Pi|A)$, or minimize

$$-\log Pr(A|\Omega) - \log P_1(\Pi) + \log Pr(A).$$

Since $P(A)$ is constant under varying hypothesis, this equals minimizing

$$-\log Pr(A|\Omega) - \log P_1(\Pi).$$

Let us find the expressions for $Pr(A|\Omega)$ and $P_1(\Pi)$. Let $\Omega = \{(\pi_1, B_1), \ldots, (\pi_k, B_k)\}$, $l_j = |B_j|$, $(j = 1, \ldots, k)$.

$$\log Pr(A|\Omega) = \sum_{j=1}^{k} \log Pr(B_j|\pi_j)$$

Further

$$\log Pr(B_j|\pi_j) = \sum_{i=1}^{l_j} \log Pr(\alpha_i^j|\pi_j),$$

$$\log Pr(\alpha_i^j|\pi_j) = \sum_{v=1}^{h} \log P_2(\gamma_{i,}^j)_v.$$

If we denote by $N_i(\Gamma)$ the number of a character $a_i \in \Sigma'$ in a set of strings Γ (where Σ' is some alphabet), then by summing over the characters of the alphabet $\Sigma \cup \{\$_2\}$, we obtain

$$\log Pr(\alpha_i^j|\pi_j) = \sum_{y=1}^{m+1} N_y(\Gamma_i^j) \log P_2(a_y),$$

where $\Gamma_i^j = \{(\gamma_i^j)_1, \ldots, (\gamma_i^j)_h\}$, $m = |\Sigma|$, and $N_{m+1}(\Gamma)$ is the number of characters $\$_2$ in Γ. The last expression equals

$$\sum_{y=1}^{m} N_y(\alpha_i^j) \log P_2(a_y) - \sum_{y=1}^{m} N_y(\pi_j) \log P_2(a_y) - N_*(\pi_j) \log P_2(\$_2),$$

where $N_*(\pi_j)$ is the number of characters $*$ in π_j. Consequently

$$\log Pr(\Pi|A) = \sum_{h=1}^{m} (N_i(A) \log P_2(a_h) - \sum_{j=1}^{k} l_j N_i(\pi_j) \log P_2(a_h)) - H \log P_2(\$_2),$$

where H is the total number of sequences γ generated to obtain A from Π.

Let us introduce the following denotation, which will play an important role in our MDL expressions. Let P_x be a probability distribution over some alphabet Σ'. For $\alpha = f_1 \ldots f_h \in \Sigma'^*$ denote

$$L_{P_x}(\alpha) = -\sum_{i=1}^{h} \log P_x(f_i).$$

$L_{P_x}(\alpha)$ can be interpreted as the length of α in bits in some optimal coding in respect to the probability distribution P_x. We assume that the string α may also include characters which are not included in the probability distribution P_x and we assume that these characters are ignored in the length L_{P_x} (i.e., more precisely we define $L_{P_x}(\alpha) = -\sum_{i=1}^{h} x(f_i) \log P_x(f_i)$, where $x(f_i) = 1$ if $P_x(f_i) > 0$ and $x(f_i) = 0$ otherwise).

For a set $A = \{\alpha_1, \ldots, \alpha_l\}$, we define

$$L_{P_x}(A) = \sum_{i=1}^{l} L_{P_x}(\alpha_i).$$

Note that $L_{P_x}(\alpha)$ or $L_{P_x}(A)$ can be obtained also by summing over the characters $a_i \in \Sigma'$, i.e., $L_{P_x}(\alpha) = -\sum_{i=1}^{m} N_i(\alpha) \log P_x(a_i)$ and $L_{P_x}(A) = \sum_{i=1}^{m} N_i(A) \log P_x(a_i)$, where $m = |\Sigma'|$. Therefore we obtain that

$$-\log Pr(\Pi|A) = L_{P_2}(A) - \sum_{j=1}^{k} l_j(L_{P_2}(\pi_j) + N_*(\pi_j) \log p), \tag{9}$$

where $N_*(\pi_j)$ is the number of characters $*$ in π_j (and $p = P_2(\$_2)$). Note that in $L_{P_2}(\pi_j)$ the characters $*$ and $\$$ are not counted (as they are not in the distribution P_2). In a similar way we obtain that

$$-\log P_1(\Pi) = L_{P_1}(\Pi) - k \log(1-r) - \log r. \tag{10}$$

After some regrouping of the terms we obtain that the most probable pattern set Π is the pattern set of some cover Ω minimizing the expression

$$M(\Omega) = M_1(\Omega) + M_2(\Omega), \tag{11}$$

where M_1 and M_2 are defined as

$$M_1(\Omega) = L_{P_1}(\Pi) - \log r,$$

and

$$M_2(\Omega) = L_{P_2}(A) - \sum_{j=1}^{k} l_j(L_{P_2}(\pi_j) + N_*(\pi_j) \log p) - k \log(1-r).$$

Next note that we can obtain a similar expression from completely different considerations. Let $A = \{\alpha_1, \ldots, \alpha_n\}$, where $\alpha_i \in \Sigma^*$, $(i = 1, \ldots, n)$. Let us assume, that we want to transmit the sequences A over a channel. The trivial way how to do it would be simply to transmit $\alpha_1, \ldots, \alpha_n$ one after another. If we abstract from the fact that some delimiters separating strings $\alpha_1, \ldots, \alpha_n$ also have to be transmitted, the message length (in characters) would be $\|A\|$.

Now, suppose we know that a pattern $\pi = \varepsilon_1 * \varepsilon_2 * \ldots * \varepsilon_h$ is present in all of these sequences. This means, that for $\alpha_i = \varepsilon_1 \gamma_i^1 \varepsilon_2 \ldots \gamma_i^{h-1} \varepsilon_h$, for some $\gamma_i^1, \ldots, \gamma_i^{h-1} \in \Sigma^*$. Then the message can be easily compressed by using this information. Namely, we could first transmit π, and then $\gamma_1^1, \gamma_1^2, \ldots, \gamma_1^{h-1}, \gamma_2^1, \ldots, \gamma_2^{h-1}, \ldots, \ldots, \gamma_n^{h-1}, \ldots, \gamma_n^{h-1}$.

Similarly, if $\Omega = \{(\pi_j, B_j) | j = 1, \ldots, k\}$ is a cover of A then A can be transmitted using Ω as

for $j = 1$ to k do
    send π_j
    for $i = 1$ to l_j do
        for $q = 1$ to h_{j-1} do send $\gamma_i^{q,j}$

where $l_j = |B_j|$, the size of set B_j. It can be easily seen, that the message length obtained by this coding scheme equals

$$\|A\| - \sum_{j=1}^{k} (l_j |\pi_j| - |\pi_j|),$$

where the second term can be considered as a 'compression' with respect to the trivial encoding. In practice, if we want to transmit A we also need to transmit some delimiter, say ';', separating the strings. Let the length in bits of this delimiter be a. If we want to transmit the message encoded by a cover Ω, then we need to transmit two kinds of delimiters: one, say the same ';', to be transmitted after each 'group' B_j, and another, say ',', to be transmitted between strings inside the same group. Let the length of the encoding of the other delimiter (',') in bits be b. Let the length in bits of characters from Σ be c. Then, it can be easily seen, that the compression of the message length (in bits) in comparison to trivial encoding (with delimiters) can be expressed as c times

$$\sum_{j=1}^{k} (u_j \cdot l_j - w_j),$$

where $w_j = |\pi_j| + \frac{a}{c}$, and $u_j = |\pi_j| + \frac{a}{c} - 2\frac{b}{c}$ (each pattern π_j requires to send also $2l_j$ delimiters ',', but saves $l_j - 1$ delimiter ';'). Note that by varying the relative length a, b, and c (and consequently w_j and u_j), we can 'slant' the optimum towards longer patterns common to fewer strings, or vice versa. Such flexibility is important in practical applications, since it allows to vary the optimum, and thus, in the end, to come up with different answers.

Now note that $M(\Omega)$ in fact corresponds to a message length in a similar but slightly more efficient coding scheme. First we send the message '$\pi_1'\$\pi_2'\$\ldots\pi_k'\$:$', i.e., the patterns separated by delimiters \$ and some final delimiter ':'. Characters of the pattern alphabet $\Sigma \cup \{*\}$

and the delimiters \$ and : are encoded by bit strings of different length depending on the probabilities P_1 of these characters in some prefix coding. If we choose the code lengths appropriately (more probable characters are encoded in shorter way) then the length of this part of message is approximately $M_1(\Omega)$. The real message length may be slightly larger than $M_1(\Omega)$ because of the discrete nature of coding.

After that we send substrings by which we have to replace $*$ characters in patterns to obtain strings from A. The substrings are separated by $\$_2$. Some additional delimiter is signaling the switching to the next pattern. The characters are again coded by bit-strings of different length according to the probability distribution P_2. The length of this part of the message is approximately $M_2(\Omega)$. We will call such encoding scheme *encoding by regular patterns*.

Thus we see that maximizing the conditional probability $P(\Pi|A)$ is equivalent to minimizing the message length (obtained by the above described coding scheme) – the so called Rissanen's Minimum Description Length (MDL) principle [Ris78].

The compression in the coding by regular patterns (in respect to the trivial encoding for the probability distribution P_1) is approximately equal to

$$C'(\Omega) = \sum_{j=1}^{k} l_j(L_{P_2}(\pi_j) + N_*(\pi_j)\log p) - L_{P_2}(\Pi) - k\log(1-r) - \log r.$$

Let Π be the pattern set of Ω. We define $C(\Omega)$ as the *abstract compression of A in the encoding by regular patterns* Π. Evidently minimization of $M(\Omega)$ equals to maximization of $C(\Omega)$. Thus we obtain

Theorem 1 *Given a set of strings $A = \{\alpha_1, \ldots, \alpha_n\}$, the most probable pattern set* Π *for A in the Bernoulli model is the pattern set of a cover Ω that gives the maximal abstract compression $C(\Omega)$ in the encoding of the set A by regular patterns.*

If we exclude from the compression expression the terms which are constant or depend only on A but not on Π, we obtain that we want to maximize

$$C(\Omega) = \sum_{j=1}^{k} (u_j \cdot l_j - w_j), \tag{12}$$

where

$$u_j = L_{P_2}(\pi_j) + N_*(\pi_j)\log p, \tag{13}$$

and

$$w_j = L_{P_1}(\pi_j) - \log(1-r). \tag{14}$$

Thus to find the most probable Π we may maximize (12). By varying relative values of $P_1(\$)$, $P_1(*)$, $P_2(\$_2)$, and r, we can slant the optimum towards longer patterns common to less strings or vice-versa; or towards more or fewer gaps (i.e., characters $*$) in patterns.

As the substring patterns are a subclass of regular patterns, the most probable pattern set Π for substring patterns can be obtained very similarly. The only difference is that the patterns are generated aas strings in the alphabet Σ and $*$ prefixed and appended to these strings. The probability distribution P_1 is obtained as $P_1(a_i) = (1 - 3P_1(\$))P(a_i)$.

Now let us turn to a class of patterns used in PROSITE database.

2.1 PROSITE patterns

PROSITE is a database of protein sequence families. For most of the families a pattern common to the sequences in this family is given. An informal descriptive definition of PROSITE patterns can be found at: `http://expasy.hcuge.ch/txt/prosuser.txt`. In bio-computing literature the syntax of PROSITE patterns is usually explained by examples. For instance, a presence of a pattern

`[YA]-x-D-T-D-S-[LIVMT]`

means that a string should contain a substring beginning with either `Y` or `A`, followed by an arbitrary character from the basic alphabet, followed by string `DTDS`, followed by `L`, or `I`, or `V`, or `M`, or `T`.

`W-x(2)-[LI]-[SAG]-x(4,5)-R`

means that a string should contain `W` followed by two arbitrary characters from the basic alphabet, followed by either `L` or `I`, followed by `S`, or `A`, or `G`, followed by from 4 to 5 arbitrary characters, followed by `R`.

For our purposes we need a more formal definition. Let $\Phi = \{b_1, \ldots, b_s\}$ be an alphabet, disjoint with Σ, by which in fact we denote some fixed subsets $K_1, \ldots, K_s$ of Σ. For each b_i we define

$$\mathcal{L}(b_i) = K_i,$$

$(i = 1, \ldots, s)$. Let us assume, that $s \geq m+1$, $K_1 = \{a_1\}, \ldots, K_m = \{a_m\}$, and $K_s = \Sigma$. We will frequently identify b_i with a_i, for $1 \leq i \leq m$, and we will denote b_s by x.

Let X be the set of all objects of the type $x(t, v)$, where t and v are nonnegative integers, such that $t \leq v$. We define $\mathcal{L}(x(t, v))$ as the set of all words over Σ^* of length between t and v, i.e.,

$$\mathcal{L}(x(t, v)) = \{\alpha \mid \alpha \in \Sigma^*,\ t \leq |\alpha| \leq v\}.$$

Thus $x(t, v)$ effectively means the wildcard of flexible length from t to v. If $\phi = d_1 \ldots d_h$ is a string over an alphabet $\Phi \cup X$ then define

$$\mathcal{L}(\phi) = \{\delta_1, \ldots, \delta_k \mid \delta_1 \in \mathcal{L}(d_1), \ldots, \delta_k \in \mathcal{L}(d_k)\}.$$

A *P-pattern* is a string of the type

$$\pi = *\phi * .$$

We define

$$\mathcal{L}(*\phi*) = \{\gamma\beta\delta | \beta \in \mathcal{L}(\phi), \gamma \in \Sigma^*, \delta \in \Sigma^*\}.$$

As x is equivalent to $x(1,1)$, we assume that the wildcards are only in the latter form. The definition of P-patterns covers in practice the most frequently used subclass of patterns that can be expressed in PROSITE pattern notation. Note that if we restrict characters of ϕ to Σ, then we obtain a subclass of *substring patterns*.

Let $\pi = *d_1 \dots d_h*$ be a P-pattern, and let us assume that the string $d_1 \dots d_h$ is generated in a similar probabilistic process as for substring patterns, except that the alphabet is $\Phi \cup \{\$\}$, the probability distribution P_1 is over $\Phi \cup \{\$\}$, and, together with each wildcard character x we generate two integers t and v to obtain $x(t,v)$. We assume that t and v are bounded by a certain constant g (this assumption is not essential in our model, however, it simplifies the situation). Let two integers t and v $(t < s)$ be drawn randomly with equal probabilities from the interval $[0..g]$.

The probability distribution P_1 should be regarded as a tool for giving preferences to different characters representing single amino-acids as well as their groups used in the patterns. It may be natural to assign probabilities proportional to the amino-acid frequencies to characters from Σ. For the group characters, theoretically substantiated probabilities can possibly be calculated from substitution matrices such as PAM [Day78] or Blosum [HH91]. Alternatively we can use some heuristics giving preferences to the groups that we want to use. Even simpler heuristics is to assign the probability 0 to the groups we do not use, and to distribute the probabilities uniformly among all other characters. The probability of the pattern generated in the described process can be defined as in (1).

The optimal message length for a P-pattern π is

$$M_1(\pi) = L_{P_1}(\pi) + 2\#X(\pi) \cdot \log g,$$

where $\#X(\pi)$ is the number of wildcard characters x in the pattern π.

Next assume that the sequences α_i are generated from patterns in a similar process as in the case of the substring patterns, but additionally the pattern characters b_j with $j > m$ are replaced by $a_i \in K_j$ (where $K_j = \mathcal{L}(b_i)$) and gaps $x(t,v)$ are filled. The probability $P(a_i|b_j)$ of a substitution of a_i for b_j is proportional to $P(a_i)$ and can be regarded as the conditional probability of a_i on condition b_j. More precisely, let $P(K_j) = \sum_{a_i \in K_j} P(a_i)$, then $P(a_i|b_j) = \frac{P(a_i)}{P(K_j)}$ if $a_i \in K_j$, and 0 otherwise. The gaps $x(t,v)$ are filled in by randomly choosing an integer y between t and v, and generating a string of the length y by outputting y characters according to the probability distribution P. The probabilities $Pr(\alpha|\pi)$, $Pr(A|\pi)$, $Pr(A|\Omega)$, and $Pr(A|\Pi)$ can be defined similarly as in (3)-(6). We call the described probability model *Bernoulli model for P-patterns.*

Next let us calculate the optimal message length corresponding to the maximal $Pr(B|\pi)$, for sequences $B = \{\alpha_1, \dots, \alpha_l\}$ containing a pattern π. To send the set $B = \{\alpha_1, \dots, \alpha_l\}$ we send the prefixes and the suffixes of each string α_i exactly like in the case of substring patterns, but additionally we send the information regarding the positions in π that are not characters from Σ. We will deal separately with characters form Φ and from X. If $\Omega_1 = \{(\pi, B)\}$, then the message length for B can be expressed as

$$M_2(\Omega_1) = L_{P_2}(\mathrm{Prefix}(B)) + L_{P_2}(\mathrm{Suffix}(B)) + M_2^1(\Omega_1) + M_2^2(\Omega_1),$$

where the first and the second terms mean the message length corresponding to the sets of prefixes and suffixes, $M_2^1(\Omega_1)$ stands for the additional information for characters b_i, $i > m$, and $M_2^2(\Omega_1)$ for the characters $x(t, v)$.

Let us begin with $M_2^1(\Omega_1)$. We fix a particular ordering of elements $(a_{h_1}, \ldots, a_{h_d})$ for each given set $K_h = \{a_{h_1}, \ldots, a_{h_d}\}$. For each position in π containing a character $b_i \in \Phi$ we have to send the information specifying the particular character a_{h_j} in this position in the string. To specify this information takes

$$M_2^1(\Omega_1) = -\sum_{j=1}^{s}\sum_{i=1}^{m} N_{i,j}(B) \log P(a_i|b_j)$$

bits, where $N_{i,j}(B)$ is the number of substitutions of a_i for b_j that have been applied to obtain strings of B from the pattern π. Let $\hat{\Phi}(\pi) = \langle i_1, \ldots, i_k\rangle$ be positions in π that are from Φ (i.e., for of all the characters except those of the type $x(t, v)$). Let us denote by $\pi(h)$ the character $b_j \in \Phi$ in the position h of π. Then for large $\|B\|$ the last expression tends to

$$-l \cdot \sum_{j=1}^{s}\sum_{i=1}^{m} N_j(\pi) P(a_i|b_j) \log P(a_i|b_j) = -l \cdot \sum_{h\in\hat{\Phi}(\pi)} \sum_{i=1}^{m} P(a_i|\pi(h)) \log P(a_i|\pi(h)),$$

where $l = |B|$.

For the characters of $x(t, v)$, we have to send y ($t \le y \le v$) characters from the basic alphabet Σ, and additionally $\log_2(v - t + 1)$ bits specifying the particular y (note, that in this way we do not have to send a delimiter). Thus

$$M_2^2(\Omega_1) = L_P(\mathrm{Gaps}(B)) + \sum_{x(t_i,v_i)} \log(v_i - t_i + 1),$$

where Gaps(B) is the set of sequences substituted for wild-cards in π to obtain B. The second term represents the information about the length of each substitution sent together with the substitution itself. Thus, the message length corresponding to the most probable pattern π is $M((\pi, B)) = M'((\pi, B)) - \log r = M_1(\pi) + M_2((\pi, B)) - \log r$. (The term $-\log r$ comes from the delimiter separating M_1 and M_2.)

The minimization of $M(\Omega_1)$ is equivalent to maximization of the compression $C(\Omega_1)$ in respect to $L_{P_2}(B)$, which equals

$$C(\Omega_1) = L_{P_2}(B) - (L_{P_2}(\mathrm{Prefix}(B)) + L_{P_2}(\mathrm{Suffix}(B)) + L_P(\mathrm{Gaps}(B)) + M_2^1(\Omega_1)).$$

For large l this tends to

$$F'(\pi, l) = l \cdot (I(\pi) + 2\log p + f(\pi)) - M_1(\pi),$$

where,

$$I(\pi) = I_1(\pi) + I_2(\pi),$$
$$I_1(\pi) = \sum_{j \in \hat{\Phi}(\pi)} I'(\pi(j)),$$

$$I'(b_h) = -\sum_{i=1}^{m}(P_2(a_i)\log P_2(a_i) - P_2(a_i|b_h)\log P_2(a_i|b_h)) =$$
$$= -\sum_{a\in\Sigma} P_2(a)\log P_2(a) + \sum_{a\in K_h} \frac{P_2(a)}{P_2(K_h)} \log \frac{P_2(a)}{P_2(K_h)},$$

and

$$I_2(\pi) = -\sum_{x(t_i,v_i)} \log(v_i - t_i + 1),$$

where the sum is over all characters from X in π, and

$$f(\pi) = \frac{1}{3}\, g\; |\pi|\; P_1(x)\; \log(1 - P_2(\$_2)).$$

$I'(b_h)$ can be interpreted as the decrease in uncertainty provided by the knowledge that the character in the given position has been chosen from a limited variety K_h. The term $2\log p$ comes from the delimiters $\$_2$. The term $f(\pi)$ comes from the observation that $L_{P_1}(\mathrm{Gaps}(B)) - L_P(\mathrm{Gaps}(B)) = \|\mathrm{Gaps}(B)\| \cdot (1 - \log P_2(\$_2))$. The value of $\|\mathrm{Gaps}\|$ tends to $l \cdot \frac{1}{3}g|\pi|P_1(x)$ for large l. As $P_2(\$_2)$ and moreover $\log(1 - P_2(\$))$ is very small, the term $f(\pi)$ can be ignored in practice.

Note that $I_1(\pi)$ is exactly the same as the first term in the pattern significance measure I of [JCH95]. The second term in [JCH95], which is a heuristic correction for flexibilities, in our case is replaced by a theoretically more substantiated correction I_2, suggesting logarithmic, rather than linear, dependence on the gap length.

For a cover $\Omega = \{(\pi_1, B_1), \ldots, (\pi_k, B_k)\}$ of A, the total message length is $M(\Omega) = \sum_{j=1}^{k} M'((\pi_i, B_i)) - k\log(1-r) - \log r$ and the total fitness is $F(\Omega) = \sum_{j=1}^{k} F'(\pi_j, l_j) - k\log(1-r)$, where $l_j = |B_j|$. Thus we obtain

Theorem 2 *Given a set of strings $A = \{\alpha_1, \ldots, \alpha_n\}$, the most probable pattern set Π in the Bernoulli model for P-patterns is the pattern set of a cover Ω that maximizes the fitness function*

$$F(\Omega) = \sum_{j=1}^{k}(u_j \cdot l_j - w_j),$$

where $u_j = I(\pi_j) + 2\log p + f(\pi_j)$, and $w_j = L_{P_1}(\pi_j) + 2\#X(\pi_j)\log g - \log(1-r)$.

The dominant term is u_i and it corresponds to the compression gained by using pattern π_j. The term w_j corresponds to the length needed to describe pattern π_j plus delimiters separating groups B_j.

Assuming that P_1 is uniform and ignoring $f(\pi)$, we obtain that $F'(\pi, l)$ can be expressed as $F''(\pi, l) = l \cdot (I(\pi) - c_0) - (c_1|\pi| + c_2\#X(\pi) + c_3)$ for $c_0 = -2\log p$, $c_1 = \log|\Phi|$, $c_2 = 2\log g$, and $c_3 = -\log(1-r) - \log(1-q)$. As c_3 depends only on q and r, which do not have good theoretical estimates, in practice it can be adjusted experimentally.

3. MDL approximation algorithm

MDL principle has been successfully used in many practical applications (e.g. [QR89]). It has also some theoretical justifications from Kolmogorov complexity point of view [LV95b]. Because Kolmogorov complexity is not computable, some other means for justifying MDL that lead to computable coding schemes are important.

In the previous section we have shown that under some simple assumptions regarding the probability distribution, the 'most probable' pattern set Π (together with the corresponding partition $B = (B_1, \ldots, B_k)$ of A) can be obtained by minimizing the description length of A under a certain simple coding scheme. In this section we will deal with the problem of minimizing the description length, i.e., with the implementation of MDL principle.

More precisely, the problem that we want to solve is – given the set A, find a cover $\Omega = \{(\pi_j, B_j) \mid j = 1, \ldots, k\}$, $B_j = \{\beta_1^j, \ldots, \beta_{l_j}^j\}$, such that expression $M(\Omega)$ reaches minimum or $C(\Omega)$ or $F(\Omega)$ reaches maximum. The most probable pattern set Π will be the pattern set of Ω. Let us denote this minimum value of $M(\Omega)$ by M_{opt}.

Because maximization of $C(\Omega)$ or $F(\Omega)$ contains a set covering problem, which is NP-hard, as a special case [KMU95], we obtain the following proposition.

Proposition 1 *The problem of finding M_{opt} for a given set A is NP-hard.*

Since the proposition in reality means that finding of M_{opt} is infeasible, we will try to approximate it instead of trying to compute M_{opt} exactly. We will prove two approximation theorems – the first for the case of substring patterns and the second for the case of regular patterns.

Theorem 3 *There exists an algorithm* `Greedy_substrings` *that, given a set of strings A, finds a substring pattern cover Ω_S of A, such that*

$$M(\Omega_S) \leq M_{opt} \times \log(|A|) + O(1)$$

in time $O(|A| \cdot \|A\|)$.

Unfortunately, in the case of more general patterns the efficiency has to suffer, but still, even in this case, we can have an algorithm which is in some sense polynomial. For exact analysis of the algorithm, let $\Delta = \{\pi_1, \ldots, \pi_K\}$ be a set of patterns. By $M_{opt(\Delta)}$ we will denote the minimal value of $M(\Omega)$ for the pattern to be chosen only from the set Δ. Let further Θ be a set of pairs $\{(\pi_i, F_i) \mid i = 1, \ldots, K\}$ where each π_i is a pattern and each $F_i = \{\xi_1^i, \ldots, \xi_{l_i}^i\}$ is the subset of A consisting of strings ξ_j^i that contain pattern π_i (i.e., $\Theta = \{(\pi_i, F_i) \mid F_i = \{\xi_1^i, \ldots, \xi_{l_i}^i\}, \xi_j^i \in L(\pi_i)\}$). We will call Θ the *set of hypotheses*. Let $size(\Theta) = |F_1| + \cdots + |F_K|$, and let us denote by $\Delta(\Theta)$ the pattern set used in Θ. By $M_{opt(\Theta)}$ we will denote the minimal value of $M(\Omega)$ for the patterns to be chosen from $\Delta(\Theta)$ with the corresponding sets B_i in Ω as subsets of F_i.

Theorem 4 *There exists an algorithm* `Greedy` *that, given a set of strings A and a set of hypotheses Θ, finds a cover Ω_Θ, such that*

$$M(\Omega_\Theta) \leq M_{opt(\Theta)} \times \log |A| + O(1)$$

in time $O(size(\Theta) \cdot \log |A|)$.

Note that a number of efficient heuristics are available for finding a hypotheses set Θ (e.g., [JCH95]).

Corollary 1 *There exists an algorithm* `Greedy2` *that, given a set of strings A and a set of patterns Δ, finds a cover Ω_Δ, such that*

$$M(\Omega_\Delta) \leq M_{opt(\Delta)} \times \log |A| + O(1)$$

in time $O(\|A\| \cdot \|\Delta\|^2 + |\Delta| \cdot |A| \cdot \log |A|)$.

The Corollary 1 follows from the fact that we can match all patterns of Δ against A with a non-deterministic automata of size at most $\|\Delta\|$ in time $O(\|A\| \cdot \|\Delta\|^2)$ and produce a hypotheses set Θ of size at most $|\Delta| \cdot |A|$.

Let $M_{opt(v)}$ be the minimal value of $M(\Omega)$ if the patterns in Ω can contain only up to v variables (characters $*$). Then we obtain the following corollary.

Corollary 2 *There exists an algorithm* `Greedy3` *that for a fixed v, given a set of strings A, finds a cover Ω_v such that*

$$M(\Omega_v) \leq M_{opt(v)} \times \log |A| + O(1)$$

in polynomial time in $\|A\|$.

The Corollary 2 follows from the fact that there exists only polynomial (in size of A) number of regular patterns with up to some fixed number v of variables that match at least one string from the set A. For a bounded v this set of patterns can be computed from the set A itself in polynomial time.

Next we will describe the algorithm `Greedy` and prove that it satisfies Theorem 4. Then we will show, how it can be implemented in a manner that it satisfies Theorem 3 in the case of substring patterns.

Let u, and w be functions such that $u(\omega_j) = u_j$, and $w(\omega_j) = w_j$ defined by (13) and (14) (or by appropriate expressions or PROSITE patterns). Then the algorithm that computes a cover Ω_Θ in a greedy fashion can be described as follows.

`Greedy`(A, Θ) return cover of A
    $U \leftarrow A$; $\Omega \leftarrow \emptyset$; $\Gamma \leftarrow \Theta$
    while $U \neq \emptyset$ do
        find $(\pi_i, F_i) \in \Gamma$ maximizing $\frac{u(\pi_i)\cdot|(F_i \cap U)| - w(\pi_i)}{|(F_i \cap U)|}$
        $\Omega \leftarrow \Omega \cup \{(\pi_i, F_i \cap U)\}$
        $U \leftarrow U - F_i$
        $\Gamma \leftarrow \Gamma - \{(\pi_i, F_i)\}$
    return Ω

The algorithm is a variation of the greedy algorithm for the weighted set cover problem [Chv79]. As suggested by (12), in our case the cost (in fact, the compression achieved) of selecting pattern π_i to the cover is $u(\pi_i) \cdot |T_i| - w(\pi_i)$ where $T \subseteq A$ consists of strings in A that are covered by π_i but not by any pattern that has already been selected by the algorithm to the greedy cover. Pattern π_i that gives the maximal relative compression is selected ("greedily") next.

To satisfy the complexity requirements of the theorem, the algorithm should be implemented in a slightly less straight-forward way. Before executing the main loop that implements the greedy-choice of the next pattern, we compute values

$$c_i = c(\pi_i, |F_i|) = \frac{u(\pi_i) \cdot |F_i| - w(\pi_i)}{|F_i|} = u(\pi_i) - \frac{w(\pi_i)}{|F_i|}$$

for every i and put the pairs (i, c_i) into a priority queue (implemented by heap, for example) so that the top element has a maximal cost c_i. This can be done in time $O(K) \leq O(size(\Theta))$.

After that the greedy-choice loop can be implemented as follows:

while $U \neq \emptyset$ do
    retrieve (i, c_i) from the top of the priority queue
    $\Omega \leftarrow \Omega \cup \{(\pi_i, F_i)\}$; $U \leftarrow U - F_i$; $\Gamma \leftarrow \Gamma - \{(\pi_i, F_i)\}$
    for every $\alpha_h \in F_i$ do
        for every F_j such that $\alpha_h \in F_j$ do
            $F_j \leftarrow F_j - \{\alpha_h\}$
            $|F_j| \leftarrow |F_j| - 1$
            $c_j \leftarrow c(\pi_j, |F_j|)$
            update the priority queue according to changed (j, c_j)

The innermost operations are performed totally $\sum_{i=1}^{K} |F_i| = size(\Theta)$ times. Each subtraction can be performed in constant time (e.g., by using linked list representation of sets F_i and α_h in all F_j). The most costly operation in the inner loop is updating the priority queue, which can be performed in time $O(\log |A|)$ if the queue is implemented by a heap, for example. Thus, the total time of the greedy choice loop is bounded by $O(size(\Theta) \cdot \log |A|)$ as required by the theorem.

Next, let us prove that the cover Ω_Θ returned by the algorithm satisfies the theorems. We need to introduce a couple of useful notations. Let

$$C_j(x) = u_j x - w_j,$$

for some reals u_j and w_j. Thus if, $\Omega = \{(\pi_j, B_j) | i = 1, \ldots, k \text{ and } B_j = \{\beta_1, \ldots, \beta_{l_j}\}\}$ is a cover of A, and C', u_j and w_j are defined by expressions (12), (13) and (14) on page 9, then

$$C'(\Omega) = \sum_{j=i}^{k} C_j(l_j).$$

Let $\Theta = \{(\pi_j, F_j) | i = 1, \ldots, K\}$ where $F_j \subset A$ and $\bigcup_{j=1}^{K} F_j = A$, be a set of hypotheses. Let $\Delta(\Theta) = \{\pi_1, \ldots, \pi_K\}$, and let $I = (i_1, \ldots, i_k)$ be a k-tuple of disjoint integers, such that each i_j satisfies $1 \leq i_j \leq K$. We define an *ordered set cover* (of A) as

$$\Omega_o(\Theta, I) = \{(\pi_{i_1}, T_1), \ldots, (\pi_{i_k}, T_k)\},$$

where sets T_j are defined as follows: $T_1 = F_{i_1}$, $T_2 = F_{i_2} - T_1$, $T_3 = F_{i_3} - (T_1 \cup T_2), \ldots, T_k = F_{i_k} - (T_1 \cup \ldots \cup T_{k-1})$. The set $\{T_1, \ldots, T_k\}$ will be called the *partition corresponding to I*. Thus

$$C'(\Omega_o(\Theta, I)) = \sum_{j=1}^{k} C_{i_j}(|T_j|).$$

From the linearity of function C' we can easily obtain

Lemma 1 *Let Ω_1 be some cover of a set A, and let $\Delta = \{\pi_1, \ldots, \pi_k\}$ be the set of patterns of Ω_1. Let Θ be a set of hypotheses for A with the pattern set $\Delta(\Theta) = \Delta$. Let us assume, that $|\pi_1| \geq \ldots \geq |\pi_k|$. Then*

$$C'(\Omega_o(\Theta, (1, \ldots, k))) \geq C'(\Omega_1).$$

As the corollary we obtain the fact, that we can assume that the optimal cover is expressed as an ordered cover. Note that the algorithm `Greedy` constructs an ordered cover.

For the next few paragraphs we will closely follow [KMU95]. We will compare some optimal ordered cover, say $I = (i_1, \ldots, i_k)$, with the ordered cover constructed by the algorithm, say $J = (j_1, \ldots, j_p)$. Let $\{T_1, \ldots, T_k\}$ and $\{P_1, \ldots, P_p\}$ be the partitions corresponding to I and

J. Let us distribute the total compression $C'(\Omega_o(\Theta, J))$ of $\Omega_o(\Theta, J)$ to elements $\alpha \in A$ as follows. If $\alpha \in P_r$, then α gets

$$c_\alpha = \frac{C_{j_r}(|P_r|)}{|P_r|}$$

Let $H(n)$ be the harmonic number of n, i.e., $H(n) = \sum_{i=1}^{n}(1/i)$. We will use the fact that $H(n) = \ln n + O(1)$.

Lemma 2 $\sum_{\alpha \in T_r} c_\alpha \geq u_{i_r} \cdot |T_r| - w_{i_r} \cdot H(|T_r|)$.

The proof of this lemma is similar to the proof of Lemma 3 in [KMU95], except the minus sign before w_{i_r} and $\leq$ signs changed to $\geq$.

Next, from Lemma 2 we obtain that

$$C'(\Omega_o(\Theta, J)) = \sum_{r=1}^{k} \sum_{\alpha \in T_r} c_\alpha \geq \sum_{r=1}^{k}[u_{i_r} \cdot |T_r| - w_{i_r} \cdot H(|T_r|)] \geq \sum_{r=1}^{k}[u_{i_r} \cdot |T_r| - w_{i_r} \cdot H(|A|)].$$

According to the definition $C'(\Omega_o(\Theta, I)) = \sum_{r=1}^{k}[u_{i_r} \cdot |T_r| - w_{i_r}]$. Therefore,

$$C'(\Omega_o(\Theta, I)) - C'(\Omega_o(\Theta, J)) \leq H(|A|) \sum_{r=1}^{k} w_{i_r}.$$

Now, using $M_{opt} = M(\Omega_o(\Theta, I)) \geq \sum_{r=1}^{k} w_{i_r}$ (follows from definition of M) we get

$$M(\Omega_o(\Theta, J)) - M(\Omega_o(\Theta, I)) = C'(\Omega_o(\Theta, I)) - C'(\Omega_o(\Theta, J)) \leq M(\Omega_o(\Theta, I)) \cdot (H(|A|) - 1),$$

hence

$$M(\Omega_o(\Theta, J)) \leq M(\Omega_o(\Theta, I)) \cdot \ln|A| + O(1) = M_{opt} \cdot \ln|A| + O(1),$$

i.e., Theorem 4 follows.

To prove Theorem 3 we have to show additionally, how to implement the algorithm in time $O(|A| \cdot \|A\|)$ in the case of substring patterns. This can be done by using generalized suffix trees (GST). GST generalizes the suffix trees (see for example [McC76]) for multiple strings. It is a trie-like data structure that represents all suffixes of multiple strings. Each leaf node contains a list of the names of the original strings that have as a suffix the string that leads from root to this leaf. All substrings of every string in GST are represented by some (may be partial) path from the root. GST can be constructed in linear time on the sum of the lengths of strings it contains [Hui92].

For the substring pattern case the input of our algorithm is only the set $U = A$, and the pairs (π_i, F_i) are found by the algorithm `Greedy_substring` from A. To implement this in the required time, we first build the GST in time $O(\|A\|)$. Next for $i = 1, \ldots, |A|$ we find the longest substring π_i present in at least i strings of A. For each i we also compute $c_i = c(\pi_i, i)$

and keep track of the best π_i (i.e., π_i having the highest c_i) so far. This can be done in time $O(\|A\|)$ by using the construction described on page 239 of [Hui92]. After this we pick the substring pattern π_i having the highest cost $c(\pi_i, F_i)$ among all possible subsets F_i of A, such that $F_i = L(\pi_i) \cap A$.

Next we find the subset F_i itself (note, that for computing $c(\pi_i, F_i)$ we did not need F_i itself, since the cost depends only on $|F_i|$, which equals i). Having π_i we can find F_i in time $O(\|A\|)$ from the leaves of the subtree under π_i or by scanning A by a fast substring search algorithm. At the same time we can compute $U \leftarrow U - F_i$.

After finding the new U we repeat the process again (starting by the building of a new GST for the updated U). Since in each such step U becomes at least one element smaller, this can be repeated no more than $|A|$ times giving the total time $O(|A| \cdot \|A\|)$ as required by the Theorem 3.

4. Conclusions

We have studied a new approach to automatic pattern discovery in a set of sequences, which is based on finding a 'good' collection of patterns covering all sequences. We have shown that we can efficiently approximate the 'best' cover within a logarithmic factor. As far as we know, this is the first result regarding the learning of unbounded number of unions of regular patterns which allow variable length substitutions.

The developed MDL expression can be regarded as a new and non-heuristic 'significance' measure for patterns. It covers also the case in which gaps of variable length are allowed. Finding a good 'significance' measure is an important problem in molecular bio-informatics, and it has always been difficult to deal with gaps of variable length.

For the case of substring patterns, we have also developed a very efficient pattern discovery algorithm based on generalized suffix-trees. In case of more general patterns, the algorithm looses much of its efficiency. One apparent future research directions is to try to find subclasses of interesting patterns where more efficient discovery algorithms are possible, and to carry out computing experiments. A more theoretical research direction is to try to relax the independence assumption and to find the respective MDL coding schemas (which would also mean finding more general 'significance' measures).

5. Acknowledgements

The authors wish to thank Inge Jonassen for valuable comments and suggestions.

References

[AKM+92] S. Arikawa, S. Kuhara, S. Miyano, A. Shinohara, and T. Shinohara. A learning algorithm for elementary formal systems and its experiments on identification of transmembrane domains. In *Proc. 25th Hawaii Int. Conf. on System Sci.*, pages 675–684, 1992.

[Ang80] D. Angluin. Finding patterns common to a set of strings. *J. of Comp. and Syst. Sci.*, 21:46–62, 1980.

[ASO94] Hiroki Arimura, Takeshi Shinohara, and Sesuko Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proc. of the 11th STACS, Lecture Notes in Comp. Sci., 755*, pages 649–660. Springer, 1994.

[Bai92] Amos Bairoch. PROSITE: a dictionary of sites and patterns in proteins. *Nucleic Acids Research*, 20:2013–2018, 1992.

[BJEG95] Alvis Brazma, Inge Jonassen, Ingvar Eidhammer, and David Gilbert. Approaches to automatic discovery of patterns in biosequences. Technical Report TR-113, Department of Informatics, University of Bergen, Bergen, Norway, December 1995.

[Chv79] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[Day78] M. O. Dayhoff. *Atlas of protein sequence and structure*, volume 5. National Biomedical Research Foundation, 1978.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[HH91] Steven Henikoff and Jorja G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19(23):6565–6572, 1991.

[Hui92] Lucas Chi Kwong Hui. Color set size problem with application to string matching. In *Proc. of Combinatorial Pattern Matching*, pages 230–243. Springer-Verlag, 1992.

[JCH95] Inge Jonassen, John F. Collins, and Desmond G. Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Science*, 4(8):1587–1595, 1995.

[KMU95] Pekka Kilpeläinen, Heikki Mannila, and Esko Ukkonen. MDL learning of unions of simple pattern languages from positive examples. In *Proceedings of the 2nd European conference EuroCOLT'95*, pages 252–260, 1995.

[LV95a] Ming Li and Paul Vitányi. Computational machine learning in theory and praxis. *Lecture Notes in Computer Science*, 1000:518–535, 1995.

[LV95b] Ming Li and Paul Vitanyi. Computational machine learning in theory and praxis. Technical Report NC-TR-95-052, Royal Holloway, University of London, Department of Computer Science, Egham, Surrey TW20 0EX, England, September 1995. (To appear in Lecture Notes in Computer Science 1000).

[McC76] E. M. McCreight. A space–economical suffix tree construction algorithm. *Journal of the ACM*, 23:262–272, 1976.

[NG94] Andrew F. Neuwald and Philip Green. Detecting patterns in protein sequences. *Journal of Molecular Biology*, 239:689–712, 1994.

[NLL95] Andrew F. Neuwald, Jun S. Liu, and Charles E. Lawrence. Gibbs motif sampling: Detection of bacterial outer membrane protein repeats. *Protein Science*, 4:1618–1632, 1995.

[OUSK92] A. Ogiwara, I. Uchiyama, Y. Seto, and M. Kanehisa. Construction of a dictionary of sequence motifs that characterize groups of related proteins. *Protein Engineering*, 5(6):479–488, 1992.

[QR89] J. R. Quinlan and R. L. Rivests. Inferring decision trees using the minimum decription length principle. *Information and Computation*, 80:227–248, 1989.

[Ris78] J. Rissanen. Modeling by the shortest data description. *Automatica-J.IFAC*, 14:465–471, 1978.

[SAC90] Hamilton O. Smith, Thomas M. Annau, and Srinivasan Chandrasegaran. Finding sequence motifs in groups of functionally related proteins. In *Proc. Natl. Acad. Sci. USA*, pages 826–830, Jan 1990.

[Shi83] T. Shinohara. Polynomial time inference of extended regular pattern languages. *Lect. Notes in Comp. Sci.*, 147:115–127, 1983.

[SS90] Randall F. Smith and Temple F. Smith. Automatic generation of primary sequence patterns from sets of related protein sequences. In *Proc. Natl. Acad. Sci. USA*, pages 118–122, Jan 1990.

[Sta89] Rodger Staden. Methods for discovering novel motifs in nucleic acid sequences. *CABIOS*, 5(4):293–298, 1989.

[SVS95] Marie-France Sagot, Alain Viari, and Henri Soldano. A distance-based block searching algorithm. In Christopher Rawlings et al, editor, *Proc. of Third International Conference on Intelligent Systems for Molecular Biology*, pages 322–331, Menlo Park, California, July 1995. AAAI Press.

[WAG84] M S Waterman, R Arratia, and D J Galas. Pattern recognition in several sequences: Consensus and alignment. *Bulletin of Mathematical Biology*, 46(4):515–527, 1984.

[WB95] Thomas D. Wu and Douglas L. Brutlag. Identificaion of protein motifs using conserved amino acid properties and partitioning techniques. In Christopher Rawlings et al, editor, *Proc. of Third International Conference on Intelligent Systems for Molecular Biology*, pages 402–410, Melno Park, California, July 1995. AAAI Press.

[Wri89] K. Wright. Identification of unions of languages drawn from an identifiable class. In *Proc. of the 2nd Annual Workshop on Computational Learnign Therory*, pages 328–333, 1989.