Measuring Over-generalization in the Minimal Multiple Generalizations of Biosequences

Yen Kaow Ng[1], Hirotaka Ono[2], and Takeshi Shinohara[3]

[1] Kyushu Institute of Technology
Graduate School of Computer Science and Systems, Iizuka, 820, Japan
kalngyk@daisy.ai.kyutech.ac.jp
[2] Kyushu University, Department of Computer Science and Communication Engineering, 6-10-1, Hakozaki, Fukuoka, 812-8581, Japan
ono@csce.kyushu-u.ac.jp
[3] Kyushu Institute of Technology
Department of Artificial Intelligence, Iizuka, 820, Japan
shino@ai.kyutech.ac.jp

Abstract. We consider the problem of finding a set of patterns that best characterizes a set of strings. To this end, Arimura *et. al.* [3] considered the use of minimal multiple generalizations (mmg) for such characterizations. Given any sample set, the mmgs are, roughly speaking, the most (syntactically) specific set of languages containing the sample within a given class of languages. Takae *et. al.* [17] found the mmgs of the class of pattern languages [1] which includes so-called sort symbols to be fairly accurate as predictors for signal peptides. We first reproduce their results using updated data. Then, by using a measure for estimating the level of over-generalizations made by the mmgs, we show results that explain the high level of accuracies resulting from the use of sort symbols, and discuss how better results can be obtained. The measure that we suggests here can also be applied to other types of patterns, e.g. the PROSITE patterns [4].

1 Introduction

Finding patterns that characterize a set of samples is a common task in molecular biology [6]. The samples are genes that are known to share certain traits, while the discovered patterns are to be used for the study of the genes, or for predicting if an unknown gene will demonstrate similar traits.

Computational means to find such patterns vary from statistical approaches to methods used in language learning [6, 7]. In language learning, the class of patterns most commonly used is the one which defines the *pattern languages* [1], in which case, a pattern is taken to be a string over a finite alphabet Σ and an infinite set $\{x_1, x_2, \ldots\}$ of variables, while the language generated by such a pattern is taken to be the strings obtainable by replacing all the variables in the pattern with strings over Σ. As an example, let $A, C \in \Sigma$, then the language of the pattern "Ax_1C" are the strings over Σ that begins with 'A' and ends with 'C'.

For such pattern languages, the problem of finding a characterization for a set of samples is frequently restated as the problem of looking for a language (or the unions of a set of languages) within a class that contains all the strings in the sample. Such approaches have been very actively studied within the learning theory community in the last decade, using either the pattern languages or its subclasses [14, 8]. We consider the problem first introduced by Arimura *et. al.* [3], where we are to find a collection of up to k languages within a class $\mathcal{L}$ of pattern languages which (1) together contains all the elements in the given sample S, and (2) is the most "*specific*" set of languages (we formally define this notion of "most specific languages" in Section 2) among every union of up to k languages in the class that contains S. Hence each number k, class $\mathcal{L}$ of languages, and set S of samples completely specify an instance of such a problem. A set of patterns that fulfills these conditions is called a k-mmg. Note that more than one k-mmg may exist for any given problem instance. Arimura *et. al.* gave a generic polynomial time algorithm (**MMG**) [3] for finding a k-mmg, for the problem instances where certain conditions are fulfilled [18].

There are a number of studies that followed up on the **MMG** algorithm [2, 19, 16], using the class of *regular patterns* [15] as search space. The regular patterns are patterns where each variable may appear at most once in it. For example, "Ax_1Cx_2" has the variables x_1 and x_2 each appearing only once in it. (As a counter-example, x_1 appears twice in "Ax_1Cx_1".) Takae *et. al.* [17] added an element called *sort symbol* to the regular patterns, resulting in a new pattern class called *sort regular patterns*.

A sort symbol is a letter associated with a non-empty set $S \subseteq \Sigma$, and is written $[A_1A_2\ldots]$ where A_1, A_2, $\ldots$ is an enumeration of the elements in S. A sort symbol associated with a set S in a pattern p can be replaced with any element in S in generating elements of the pattern language of p. Such sort symbols have been in common use within the molecular biology community for some time [4], usually under the name of *ambiguous letter* or *ambiguous character*. Takae *et. al.* found that the k-mmgs of sort regular pattern languages achieve higher accuracies than those of regular pattern languages of the same k in the prediction of signal peptide functions in unknown biosequences.

The present work aims to clarify Takae *et. al.*'s observations.

We first note that any sort regular pattern can be expressed as the union of a few regular patterns. For example, "$x_1[AC]x_2$" is the union of "x_1Ax_2" and "x_1Cx_2". Hence allowing the use of sort symbols achieves similar effects as allowing more languages to be in a union. From this observation, we can perhaps compare the k-mmgs of sort regular patterns to the k'-mmgs of regular patterns for some k' that is "suitably" larger than k. However, such k' and k cannot be straight-forwardly decided from the alphabet size, or the number of occurrences of sort symbols that appeared in the k-mmg. This is because allowing for more languages to be union-ed provides more flexibility than simply allowing the use of sort symbols, since patterns in a union can bear no similarity at all to each other. In a more delicate example, we see that the union of the languages of "Ax_1C" and "Cx_1A" cannot be expressed as the language of a single sort regular pattern.

To illustrate this difficulty in comparisons more clearly, we found, through experiments, that a 4-mmg of sort regular pattern languages where 3 occurrences of sort symbols are allowed in each pattern to be somewhat comparable to an 8-mmg of regular pattern languages, in terms of accuracies in signal peptide prediction. These numbers do not appear to be intuitively derivable to us.

Nevertheless, we did find a measure that has a very good correlation with the accuracy values we obtained. This measure is the total number of strings in the language of a k-mmg of up to approximately the length of the longest sample used. We expected that such a count would give us a good indication of the amount of over-generalization that a k-mmg makes. (The computational complexity involved in such countings has been discussed in [11, 9, 13], and learning-related results of it can be found in [12]. We note also that such a counting can be done for many kinds of patterns studied in molecular biology [6, 7], for instance the PROSITE patterns [4].)

What does the correlation between this measure and the accuracies of the k-mmgs tell us? We answer this at the end of the paper.

This paper is structured as follows. We first introduce the terminology required in our subsequent discussion. This is followed by a detailed description of our experimental setup, where we also make some remarks that have been left out in Takae *et. al.*'s earlier work. We then present results from the experiments and finally, discuss their significance.

2 Preliminaries

The symbol N denotes the set of natural numbers. For a set $\mathcal{A}$ we denote by $card(\mathcal{A})$ the cardinality of $\mathcal{A}$. A *word* over a non-empty alphabet $\mathcal{A}$ is a string of symbols taken from $\mathcal{A}$. The empty word is a null string and is denoted ε. For a set $\mathcal{A}$, the symbols $\mathcal{A}^*$, $\mathcal{A}^+$, and $\mathcal{A}^{\leq n}$ denote the sets of all the words, non-empty words, and words of length n or less over $\mathcal{A}$, respectively. For any sets $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} - \mathcal{B}$ denotes the set $\{x \mid x \in \mathcal{A} \text{ and } x \notin \mathcal{B}\}$.

Let Σ be a finite set of alphabets. A *sort symbol* is a letter associated with a non-empty set $S \subseteq \Sigma$, and is written $[S]$, or $[A_1 A_2 \ldots]$ where $A_1, A_2, \ldots$ is an enumeration of the elements of S. Π denotes the set $\{[S] | S \neq \emptyset \wedge S \subseteq \Sigma\}$. For any $A \in \Sigma$, $[A]$ is treated as equivalent to A, and hence Σ is treated as a proper subset of Π.

A *regular pattern* is a string over $\Sigma \cup \{*\}$, and a *sort regular pattern* is a string over $\Pi \cup \{*\}$, where the '$*$' symbol is called a *variable*. A *substitution* θ for a pattern p is a set of replacements for variables in p with patterns, and sort symbols with sort symbols. The image of a pattern p under a substitution θ is written $p\theta$. Every occurrence of a variable or sort symbol may be replaced independently. A sort symbol $[S]$ can only be replaced with another sort symbol $[S']$ where $S' \subseteq S$. We write $p \preceq q$ if $p = q\theta$ for some substitution θ. We write $p \prec q$ if $p \preceq q$ but not $q \preceq p$. Given two sets of patterns P and Q, we write $P \sqsubseteq Q$ if for each $p \in P$, $p \preceq q$ for some $q \in Q$, and $P \sqsubset Q$ just in case $P \sqsubseteq Q$ but not $Q \sqsubseteq P$. In the latter case, P is also said to be more *specific* than Q, and Q more *general* than P.

The *language* $L(p)$ *for a pattern* p is the set of all $w \in \Sigma^*$ where $w = p\theta$ for some substitution θ, that is, $L(p) = \{w \in \Sigma^* \mid w \preceq p\}$. For a set of patterns P, $L(P)$ is the union of $L(p)$ of each $p \in P$. Note that for any sets of patterns P, Q, if $P \sqsubseteq Q$ then $L(P) \subseteq L(Q)$ [15]. Given $k \in N$, a finite set $S \subseteq \Sigma^*$ and a class of patterns $\wp \subseteq (\Pi \cup \{*\})^*$, a set of patterns $P \subseteq \wp$ is a *k-minimal multiple generalization* (or k-mmg) for S within $\wp$ if $card(P) \leq k$, $S \subseteq L(P)$, and there are no other set of up to k patterns Q in $\wp$ where $Q \sqsubset P$ and $S \subseteq L(Q)$. Note that for each k, $\wp$, and S there may be more than one such k-mmgs.

2.1 The MMG algorithm

We introduce the **MMG** algorithm and our implementation of it in this section. Given a positive integer k, a set of strings S and a class of patterns $\wp$, the following is the listing for the **MMG** algorithm given in [3] for finding a k-mmg. All patterns in the following are implicitly elements of $\wp$.

MMG(k, S)
(1) Let $P \leftarrow \{\text{“}*\text{”}\}$.
(2) Let $\Delta k \leftarrow k$.
(3) While $\Delta k \geq 2$ and there exists $p \in P$ and
$Q \sqsubseteq \{p\}$ of more than one pattern where
(i) $S - L(P - \{p\}) \subseteq L(Q)$, but
(ii) no $Q' \subset Q$ has $S - L(P - \{p\}) \subseteq L(Q')$ (i.e. no pattern in Q is redundant),
(3.1) Replace each $q \in Q$ with a more specific $q' \prec q$ until any further such replacement will result in $S \not\subseteq L(P) - L(\{p\}) \cup L(Q)$.
(3.2) Let $P \leftarrow P - \{p\} \cup Q$.
(3.3) Let $\Delta k \leftarrow \Delta k - |Q| + 1$.
(4) Output P.

A detailed description of the algorithm can be found in [3], here we only explain the parts which is needed in our discussion. Intuitively, the **MMG** algorithm starts with a most general pattern, and continues making the patterns more specific until no further refinement can be performed.

To make the search for Q at step (3) efficient, the **MMG** algorithm looks for candidates for Q from only a subset $\rho(p)$ of refinements of p of cardinality polynomial in the length of p. Such a ρ, called a *refinement operator*, is said to be *complete with respect to the class* $\wp$ just in case for all $p, p' \in \wp$, $p' \preceq p \Leftrightarrow p' \in \rho^+(p)$, where ρ^+ is the transitive closure of ρ. It has been shown that for a complete refinement operator ρ, for any p, such Q that fulfill the requirement at step (3) exists if and only if some $Q'' \subseteq \rho(p)$ also fulfills the requirement — hence limiting the search to $\rho(p)$ does not make the search less complete.

Refinement operator In this paper we use the pattern class where only up to a number of occurrences of the letters in $\Pi - \Sigma$ and $\{*\}$ are allowed in each pattern. (The regular patterns are simply patterns where 0 occurrence of the letters in $\Pi - \Sigma$ are allowed.) Let $\wp(m, s)$ denote the class of patterns with at most m occurrences of variables and at most s occurrences of the letters in $\Pi - \Sigma$ in each pattern. It is clear that for each m, $s \in N$, $\wp(m, s) \subset \wp(m+1, s)$ and $\wp(m, s) \subset \wp(m, s+1)$. For each class $\wp(m, s)$, we use the following as refinement operator.

$\rho(p)$
(1) Let $P \leftarrow \emptyset$.
(2) For each variable occurrence '$*$' in p and
for each q in { "$*[\Sigma]*$", "$*[\Sigma]$", "$[\Sigma]*$", ε },
(2.1) Let p' be the pattern obtained from p by replacing the '$*$' with q,
(2.2) if $p' \in \wp(m, s)$, add p' to P.
(3) For each sort symbol occurrence $[S]$ in p, for each $S' \in \{S - \{X\} \mid X \in S\}$,
(3.1) Add the pattern obtained by replacing $[S]$ with $[S']$ to P.
(4) Output P.

Hence in this case, $card(\rho(p))$ is linear in the length of p, $|p|$ say, and furthermore, $\rho(p)$ can be computed in time $O(|p|)$.[4] We let $m = 4$ throughout this paper, that is, we consider only the classes $\wp(4, s)$ for some $s \in N$. For this reason we also write $\wp(4, s)$ simply $\wp(s)$.

Implementation Our implementation of the algorithm is in C, and compiles on GCC 3.3 and 3.4 (on both `gcc` and `g++`). The program supports most of the modifications studied in the earlier publications [2, 17, 16], and provides other useful features such as the automatic compilation of alphabet from the input sample (that is, the alphabet for the amino acids is not coded into the program). The program can be downloaded from http://www.daisy.ai.kyutech.ac.jp/~kalngyk/.

[4] Note that this operator is not complete with respect to any class $\wp(m, s)$ where $1 \leq m \leq 2$ and $s \geq 1$. As an example, let A, $C \in \Sigma$, $p =$"$*A[AC]A*$" and $q =$"$*$". Clearly, p, $q \in \wp(2, 1)$ and $p \preceq q$, but it can be shown that $p \notin \rho^+(q)$.

3 Experimental setup and reproduced earlier results

Our samples are biosequences selected from the flat files for bacterial (release 137.0), plant (release 138.0) and rodent (release 138.0) sequences from NCBI GenBank [5]. All the signal peptide entries (that is, entries with the primary tag `sig_peptide`) were extracted, and from among these, we discarded the entries where the corresponding coding sequence (CDS)

1. has more than one translation table specified, or
2. translates into a sequence different from that given in the file.

We also removed the signal peptide entries which

1. translates into a sequence containing the '$*$' symbol, or
2. does not begin with a Methionine, or where
3. the location specified is fuzzy, or references an external sequence.

Finally, we remove the prefixing Methionine from each sequence obtained. We denote this set of sequences by $\texttt{POS}_T$, where T is a sequence type of either bacterial, plant, or rodent. The following table lists the number of signal peptide sequences obtained using this method. Entries in brackets are numbers for the sequences used in [17] (corrected to count only distinct sequences) and the increase comes from the growth in the available sequences from GenBank.

Sequence type T	$card(\texttt{POS}_T)$	$\texttt{POS}_T$ min/max/ave length	$card(\texttt{NEG}_T)$
bacterial	1869 (422)	10 / 31 / 23	18690
plant	1694 (324)	10 / 30 / 22	16940
rodent	1803 (745)	10 / 30 / 21	18030

Table 1. sequences used in finding k-mmgs ($\texttt{POS}_T$) and in evaluation ($\texttt{POS}_T$, $\texttt{NEG}_T$)

The negative examples $\texttt{NEG}_T$ for each type T are (1) randomly chosen substrings of (randomly selected) type T CDSs, of a randomly decided length between 21 to 35, which are (2) not a substring of any sequence in $\texttt{POS}_T$. While it is not guaranteed that sequences obtained this way will not turn out to be signal peptides, we trust that this odds is small, and by using a large number of sequences we can reduce its effects.

3.1 Measures for evaluating a k-mmg

accuracy. For a sequence type T, let $S \subset \texttt{POS}_T$ be the samples used in deriving a k-mmg P. For each such P and S we define the following measures of accuracy,

1. The *positive accuracy*, $p(P,S) = 100 \times \frac{card((\texttt{POS}_T - S) \cap L(P))}{card(\texttt{POS}_T - S)}$
2. The *negative accuracy*, $n(P) = 100 \times \frac{card(\texttt{NEG}_T - L(P))}{card(\texttt{NEG}_T)}$
3. The *overall accuracy*, $acc(P,S) = \sqrt{p(P,S) \cdot n(P)}$

coverage. For a k-mmg P and a length n, the quantity $card(L(P)^{\leq n})$ gives us an indication of the amount of over-generalization P has made within $\Sigma^{\leq n}$. In this paper we are interested in the case where $n = 30$, since that is roughly the longest length of our positive examples. Hence for a k-mmg P we define the *coverage of* P,

$$cov(P) = card(L(P)^{\leq 30})$$

In our experiment, to count $cov(P)$, we first construct a DFA [10] that accepts the words in $L(P)$, and then count the number of distinct paths of up to length 30 which reaches an accepting state [13].

3.2 Reproduced earlier results using updated samples

We first reproduce the results as obtained in [17]. For each T and for each n in $\{50, 100, 150, \ldots, 800\}$, we randomly choose 100 sets of samples $S_1^n, S_2^n, \ldots, S_{100}^n$ each of size n from $\texttt{POS}_T$. From each such S_i^n we obtain a 5-mmg P_i^n from the class $\wp(9)$. We then compute the average accuracy $\sum_i acc(P_i^n, S_i^n)/100$, of the 100 5-mmgs obtained from samples of the size n, and we plot these accuracy values against n to see how this accuracy depends on the input size. These dependencies are quite similar to those observed in [17]. In our plots (Fig. 1) we also show the averages of $n(P_i^n)$ and $p(P_i^n, S_i^n)$ for a better picture of what leads to the trend observed in $acc(P_i^n, S_i^n)$. We see that the positive accuracy increases but negative accuracy decreases as we increase the sample size. This tells us that the k-mmgs produced becomes more general as we add more samples to **MMG**.

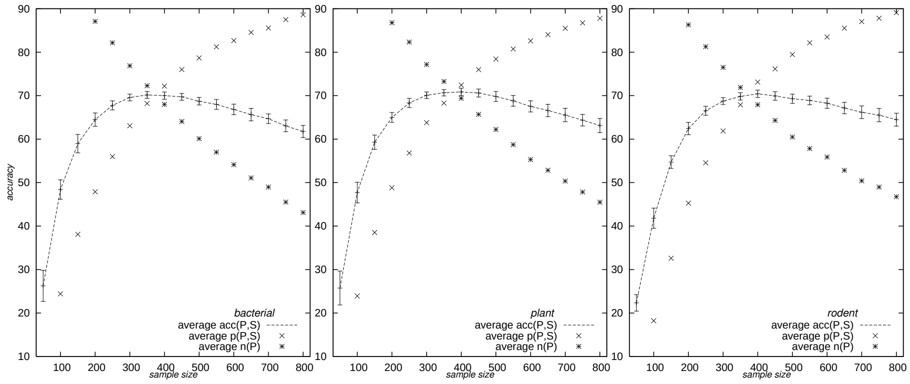

Fig. 1. accuracies ($acc(P, S)$) vs input sample sizes ($card(S)$) of k-mmgs

4 Results: correlation of accuracies with coverages

In this section we study the correlation between the accuracies of k-mmgs with their coverages, to examine the effects of over-generalization on the k-mmgs, under various conditions. We only reproduce the plots for bacterial samples here, but all the results in this section are also observed on the plant and rodent samples.

4.1 Coverages of k-mmgs from the same k, $\wp(s)$ and sample S

We first compare among the k-mmgs obtainable from the same sample. We run a modified form of the **MMG** algorithm to make it produce more than one k-mmg on a given sample set. This is done in a straight-forward manner, by letting **MMG** continue on a few different branches at step (3.1), each branch with a different order on the patterns in Q to refine.

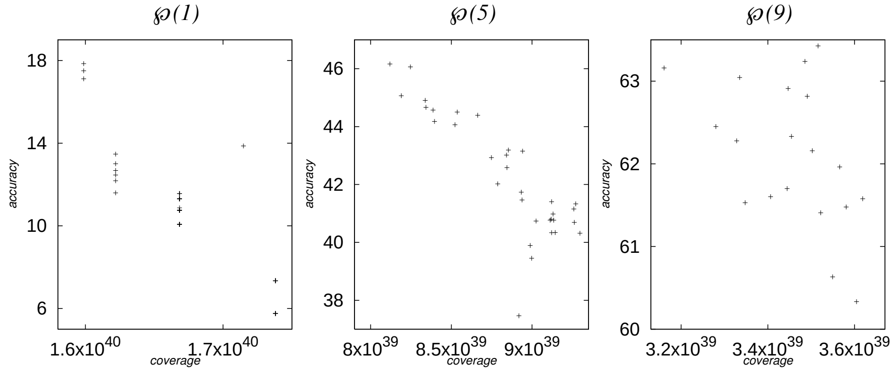

Fig. 2. $acc(P, S)$ vs $cov(P)$ of 5-mmgs obtained from the same sample.

Fig. 2 shows accuracies ($acc(P, S)$) against coverage ($cov(P)$) for the 5-mmgs obtained from respectively $\wp(1)$, $\wp(5)$ and $\wp(9)$, all from the same sample (of size 800). We see a very close correlation for the case of $\wp(1)$ and the correlation becomes less obvious for the case of $\wp(9)$, since the accuracies then conglomerate around similar values.

4.2 Coverages of k-mmgs for various k, $\wp(s)$ at different sample sizes

The correlation of accuracies with coverages can be observed across k-mmgs obtained from different samples as well as classes. Fig. 3 shows these values for k-mmgs obtained from the classes

1. 3-mmgs from $\wp(0)$ (legend $\boxdot$)
2. 3-mmgs from $\wp(3)$ (legend $\times$)
3. 5-mmgs from $\wp(0)$ (legend $\triangle$)
4. 5-mmgs from $\wp(5)$ (legend $\circ$)
5. 5-mmgs from $\wp(9)$ (legend $+$)

In each graph in Fig. 3 we fix a sample size n, and for each of the classes we obtain 50 k-mmgs, each from a different sample of size n. Comparing among the

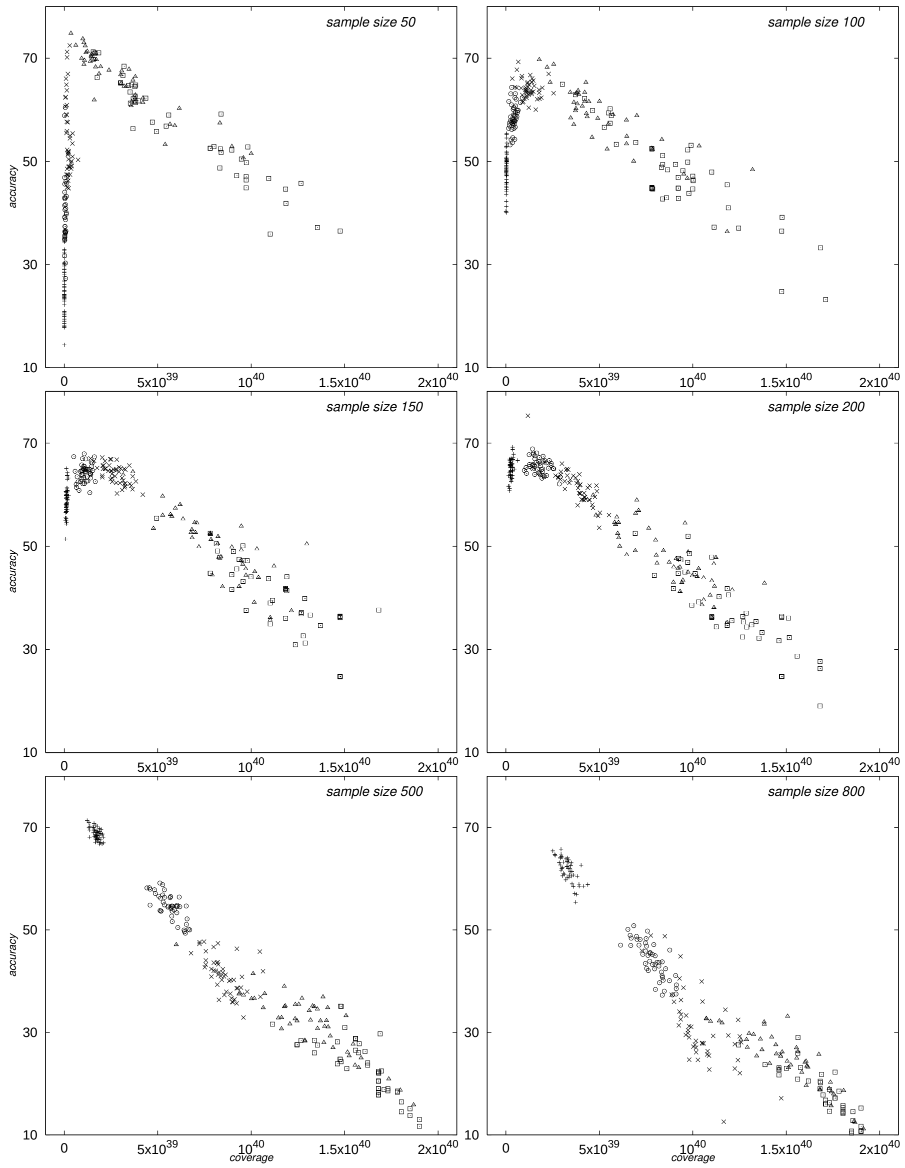

Fig. 3. $acc(P,S)$ vs $cov(P)$ of k-mmgs obtained from different classes and samples.
$\boxdot$ = 3-mmgs from $\wp(0)$ $\times$ = 3-mmgs from $\wp(3)$ $\triangle$ = 5-mmgs from $\wp(0)$
$\bigcirc$= 5-mmgs from $\wp(5)$ $+$ = 5-mmgs from $\wp(9)$.

graphs, we see how increasing the sample size makes the k-mmgs more general, and we also see how this affects their accuracies. The values obtained are in consistency with those in Section 4.1. Interestingly, k-mmgs of similar coverages almost always score about the same accuracies, regardless of k, or the class where the patterns are drawn. We also notice, from the plots for samples of the sizes 50∼150, that k-mmgs with coverages below a certain threshold score lower accuracies as their coverages decrease. We look at this region of the plots more closely in Section 4.4.

4.3 Similarly accurate k-mmgs of different k, $\wp(s)$ at different sample sizes

Allowing letters in $\Pi - \Sigma$ to occur in patterns achieves similar effects as increasing the number k of patterns allowed in a k-mmg. However, given any k-mmg from $\wp(n)$ for some k and n, it is difficult to decide if some k'-mmgs from $\wp(n')$ for some $k' \neq k$ and $n' \neq n$ will be similarly accurate for protein prediction. Nevertheless, we found a specific case where such accuracies are comparable for all the sample sizes used in our tests: the 4-mmgs obtained from $\wp(3)$ (legend $\boxdot$) and 8-mmgs from $\wp(0)$ (legend $\times$). This is shown in Fig. 4.

Interestingly, while the two different classes produce k-mmgs of similar accuracies, it is very much faster for the **MMG** algorithm to find 4-mmgs in $\wp(3)$ than 8-mmgs in $\wp(0)$, since the runtime of the algorithm increases more in k than in the size of the refinement operator.

4.4 Overly specific k-mmgs

We now show the details on the positive correlation between accuracies and coverage in the low coverage region in Fig. 3, by showing the x-axis in log scale.

In Fig. 5 we show respectively, the accuracies, the positive accuracies, and the negative accuracies of

(1) 5-mmgs from $\wp(4)$ (legend $\boxdot$)
(2) 5-mmgs from $\wp(5)$ (legend $\times$)
(3) 5-mmgs from $\wp(9)$ (legend $\circ$)
(4) 7-mmgs from $\wp(20)$ (legend $\blacksquare$)

obtained from samples of size 50.

We see a rapid decline of the positive accuracies around the coverage range of 10^{30}∼10^{38}, and for this region the trend of the accuracies follows closely that of the positive accuracies.

5 Conclusions

We now revisit the proposition made by Takae *et. al.* in [17], where they observed that the k-mmgs of sort regular patterns achieve higher accuracies than k-mmgs of regular patterns. We have shown that these higher accuracies are brought forth by a reduction of over-generalization in the k-mmgs of the sort regular patterns. We also argued that similar reduction in over-generalization can be achieved by increasing the number k of patterns allowed in a k-mmg, though we lack a theoretical basis to make any meaningful comparison between the two. Nevertheless, in Section 4.3 we gave an empirical case where k-mmgs of a higher k of only regular patterns scored similar accuracies as the k-mmgs of sort regular

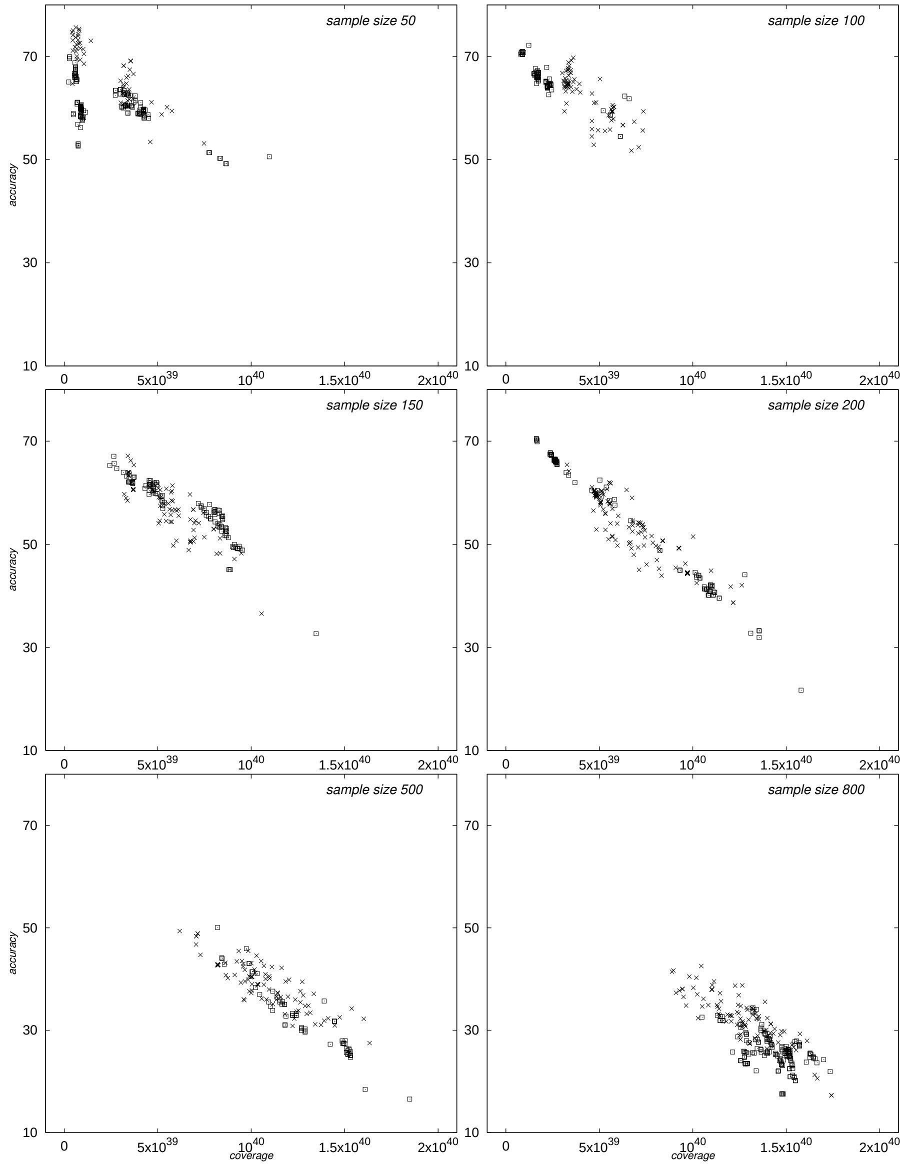

Fig. 4. $acc(P, S)$ vs $cov(P)$ of 4-mmgs from $\wp(3)$ and 8-mmgs from $\wp(0)$.
$\boxdot$ = 4-mmgs from $\wp(3)$ $\times$ = 8-mmgs from $\wp(0)$

patterns. It is noteworthy, however, that the **MMG** algorithm finds k-mmgs of sort regular patterns very much faster than it finds k-mmgs of regular patterns of similar accuracies. This is because the runtime of the algorithm grows faster on k than the usage of letters in $\Pi - \Sigma$ (which results in a larger refinement operator).

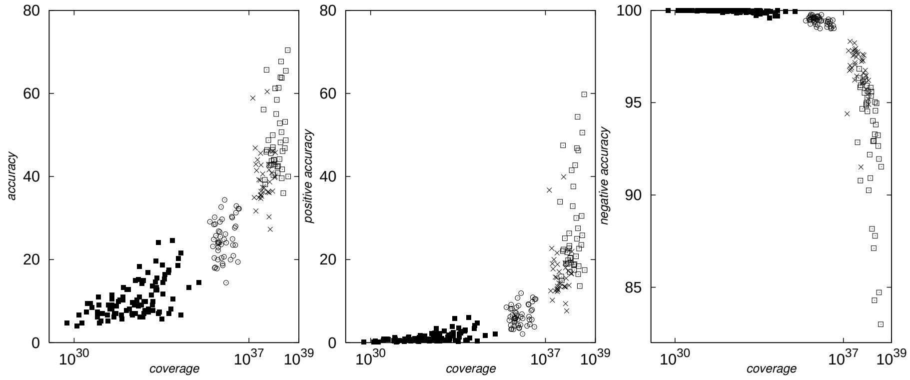

Fig. 5. details of $acc(P, S)$, $p(P, S)$ and $n(P)$, against $cov(P)$ for samples of size 50

By counting the coverages of the k-mmgs, that is, the number of strings of up to a certain length in the languages of the k-mmgs, we also found that k-mmgs with similar coverages always achieve similar accuracies — and this is regardless of the number k, or the kind of patterns used in the k-mmg. For the samples we used, we found that in general the accuracy values correlate negatively with the coverages, except for very small values of coverages, where the opposite trend is observed. This implies that in general, the k-mmgs obtained in our tests are overgeneralized, and hence as a rule it is preferable to find more specific k-mmgs, although we should also be on the guard so that they do not become overly specific, for which we have seen that the accuracies will go into a rapid decline very suddenly. In practice, with respect to any samples of a specific kind, it may be possible for us to locate the range of coverage values where the k-mmgs' accuracies would be highest, and then use these coverage values as a guide in whether or not to accept an **MMG** output.

We note that such coverage may also be useful for making comparisons between pattern sets that are otherwise incomparable through syntax or set inclusion, a possibility which may be interesting in both theory and practice.

Finally, we hope that the idea of coverage can be adopted in the analysis of other types of patterns used to represent biological sequences, such as the PROSITE patterns [4].

Acknowledgement

We would like to thank the reviewers for helpful comments. Yen Kaow Ng is supported by the Japanese Government Scholarship of the Ministry of Education, Science, Sports, Culture and Technology of Japan. Hirotaka Ono is supported by the Scientific Grant in Aid of the Ministry of Education, Science, Sports, Culture and Technology of Japan.

References

1. D. Angluin. Finding patterns common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.
2. H. Arimura, R. Fujino, T. Shinohara, and S. Arikawa. Protein motif discovery from positive examples by Minimal Multiple Generalization over regular patterns. In *Proceedings of the Genome Informatics Workshop*, pages 39–48, 1994.
3. H. Arimura, T. Shinohara, and S. Otsuki. Finding minimal generalizations for unions of pattern languages and its application to inductive inference from positive data. In *Proc. of the 11th Ann. Symp. on Theoretical Aspects of Comp. Sci. (STACS'94)*, volume 775 of *Lecture Notes in Computer Science*, pages 649–660. Springer-Verlag, 1994.
4. A. Bairoch. PROSITE: A dictionary of sites and patterns in proteins. *Nucl. Acids Res.*, 25(19):2241–2245, 1991.
5. D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler. Genbank: update. *Nucl. Acids Res.*, 32(Database-Issue):23–26, 2004.
6. A. Brāzma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *J. Comp. Biol.*, 5(2):277–304, 1998.
7. B. Brejova, T. Vinar, and M. Li. *Pattern Discovery: Methods and Software*, chapter 29, pages 491–522. Humana Press, 2003.
8. J. Case, S. Jain, R. Reischuk, F. Stephan, and T. Zeugmann. Learning a subclass of regular patterns in polynomial time. In R. Gavaldà, K. P. Jantke, and E. Takimoto, editors, *Algorithmic Learning Theory: Fourteenth International Conference (ALT' 03)*, volume 2842 of *Lecture Notes in Artificial Intelligence*, pages 234–246, 2003.
9. C. Chan, M. Garofalakis, and R. Rastogi. RE-tree: an efficient index structure for regular expressions. *The VLDB Journal*, 12(2):102–119, 2003.
10. J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
11. S. Kannan, Z. Sweedyk, and S. Mahaney. Counting and random generation of strings in regular languages. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 551–557. Society for Industrial and Applied Mathematics, 1995.
12. Y. K. Ng and T. Shinohara. Inferring unions of the pattern languages by the most fitting covers. In *Algorithmic Learning Theory: Sixteenth International Conference (ALT' 05)*. to appear.
13. H. Ono and Y. K. Ng. Best fitting fixed-length substring patterns for a set of strings. In *Proceedings of The Eleventh International Computing and Combinatorics Conference (COCOON'05)*. to appear.
14. A. Shinohara. String pattern discovery. In Shai Ben-David, John Case, and Akira Maruoka, editors, *Algorithmic Learning Theory: Fifteenth International Conference (ALT' 04)*, volume 3244 of *Lecture Notes in Artificial Intelligence*, pages 1–13, 2004.
15. T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147 of *Lecture Notes in Computer Science*, pages 115–127. Springer-Verlag, 1982.
16. T. Shinohara and Y. K. Ng. Strong biases for the minimal multiple generalization algorithm on samples of very small sizes. In *The Proceedings of the 57th Meeting of SIG-FPAI (November 2004)*. The Japanese Society of Artificial Intelligence, 2004.
17. T. Takae, T. Kasai, H. Arimura, and T. Shinohara. Knowledge discovery in biosequences using sort regular patterns. Workshop on Applied Learning Theory, 1998.

18. J. Uemura and M. Sato. Compactness and learning of classes of unions of erasing regular pattern languages. In *Algorithmic Learning Theory: Thirteenth International Conference (ALT' 02)*, volume 2533, pages 293–307. Springer-Verlag, 2002.
19. M. Yamaguchi, S. Shimozono, and T. Shinohara. Finding minimal multiple generalization over regular patterns with alphabet indexing. In *Proceedings of the Seventh Workshop on Genome Informatics*, volume 7, pages 51–60. Universal Academy Press, 1996.