Finding Consensus Patterns In Very Scarce Biosequence Samples From Their Minimal Multiple Generalizations

Yen Kaow Ng[1] and Takeshi Shinohara[2]

[1] Kyushu Institute of Technology,
Graduate School of Computer Science and Systems, Iizuka, 820, Japan
kalngyk@ai.kyutech.ac.jp

[2] Kyushu Institute of Technology,
Department of Artificial Intelligence, Iizuka, 820, Japan
shino@ai.kyutech.ac.jp

Abstract. In this paper we examine the issues involved in finding consensus patterns from biosequence data of very small sample sizes, by searching for so-called *minimal multiple generalization (mmg)*, that is, a set of *syntactically minimal* patterns that accounts for all the samples. The data we use are the *sigma regulons* with more conserved consensus patterns for the bacteria *B. subtilis.* By comparing between the mmgs found over different search spaces, we found that it is possible to derive patterns close to the known consensus patterns by simply making some reasonable requirements on the kinds of patterns to obtain. We also propose some simple measures to evaluate the patterns in an mmg.

1 Introduction

Finding frequently occurring patterns (called *consensus patterns*) within a set of biosequences is a common task in molecular biology. The kinds of patterns used in the various methods may differ very significantly [3], for example compare the patterns used in the PROSITE [9] database and the TEIRESIAS algorithm [7]. The patterns we consider are strings over a finite alphabet Σ and a variable symbol "*", where its language is the strings obtainable by (independently) replacing all variables in the pattern with strings over Σ [8].

Our approach to consensus pattern is based on finding so-called *minimal multiple generalizations (mmg)* [2] from subclasses of the pattern languages. Given any sample set S of strings over Σ, the mmgs are the most (syntactically) specific set of up to k patterns over the given class where their languages together contain S. Arimura *et. al.* [2] gave a polynomial time algorithm **MMG** for finding such mmgs. Others [1, 11, 10, 6] have studied how well the **MMG** discovered mmgs perform in predicting gene functions. Here, we consider instead the resemblance between the discovered mmgs and well established consensus patterns.

As samples, we use the sigma factor dependent promoter sequences in *Bacillus Subtilis*, which has well established consensus patterns [4]. The difficult part in using these samples is that they are not very abundantly available, and hence it is difficult for candidate patterns to achieve sufficient levels of statistical significance. We show that by constraining our search to only patterns of slightly more specific forms, some patterns in the discovered mmgs match the known consensus patterns fairly accurately. We also note some qualities that can be found in the better matching patterns.

2 Preliminaries

Let Σ be a finite set of alphabets. A *regular pattern* is a string over $\Sigma \cup \{*\}$, where the '$*$' symbol is called a *variable*. A *substitution* θ for a pattern p is a set of replacements for variables in p with regular patterns. $p \preceq q$ iff p is obtainable from q via some substitution θ; $p \prec q$ iff $p \preceq q$ but not $q \preceq p$. Given two sets of patterns P and Q, $P \sqsubseteq Q$ iff for each $p \in P$, $p \preceq q$ for some $q \in Q$, and $P \sqsubset Q$ iff $P \sqsubseteq Q$ but not $Q \sqsubseteq P$. The language of a pattern p, $L(p) = \{w \in \Sigma^* \mid w \preceq p\}$. For a set of patterns P, $L(P) = \bigcup_{p \in P} L(p)$. Note that $P \sqsubseteq Q \Rightarrow L(P) \subseteq L(Q)$ [8].

Given $k \in N$, a finite set $S \subseteq \Sigma^*$ and a class of patterns $\mathcal{P} \subseteq (\Sigma \cup \{*\})^*$, a set of patterns $P \subseteq \mathcal{P}$ is a *k-minimal multiple generalization* (or k-mmg) for S over $\mathcal{P}$ if $\sharp P \leq k$, $S \subseteq L(P)$, and there are no other set of up to k patterns Q in $\mathcal{P}$ where $Q \sqsubset P$ and $S \subseteq L(Q)$. We use the efficient algorithm **MMG** in [2] for finding such mmgs.

Using different classes of patterns in **MMG** may change its output very significantly. Originally the regular patterns were studied in [2]. Then, the regular patterns with restricted number of variable occurrences were studied in [1]. In this work we study those classes, with the addition of so-called *range specifiers* (as in the PROSITE patterns) where a variable is written "$*(x, y)$" for some x,

$y \in N$. A variable written this way, when replaced with constant strings, can only be replaced with strings of length at least x and at most y.

3 Experimental Setup

We run our tests using a straight-forward implementation of the **MMG** algorithm (downloadable from www.daisy.ai.kyutech.ac.jp/~kalngyk). The program is given as input a set of biosequences S, a maximum number k of patterns allowed in a k-mmg, and finally, a class of patterns $\mathcal{P}$. The output k-mmg is then compared with known consensus patterns. This is repeated for $k = 5$ and $k = 10$ over all combinations of our candidates for S and $\mathcal{P}$, as follows.

(For S) We use the sigma factor dependent promoter sequences in *B. subtilis* (refer [4]) for samples. These sequences (and consensus patterns) are obtained from the on-line database DBTBS [5]. We use the regulons sigD (33/51/59), sigE (62/44/51), sigH (48/37/50), sigW (32/42/45) (numbers are for sample size, average sequence length and maximum sequence length), because their sample sizes are relatively large and there are relatively long contiguous conserved regions within their consensus patterns.

(For $\mathcal{P}$) We use subclasses of the regular patterns with restricted use of variables. We also consider patterns of the following form, the rational of which will be made clear in its description: Assume that each of the target patterns consists of exactly two segments. That is, they are of the form "$*_1 v *_2 w *_3$", where v, w are non-empty strings over Σ and the $*_i$s are variables. We further assume that v is not too close to the end of the sequences, while w is not too close to the beginning of the sequences. For this reason, we place upper bounds on the maximum length of the words that can be used to replace $*_1$ and $*_3$. In this study we use the same length x as the upper bound for both $*_1$ and $*_2$, and we set x to 0.5 times the length of the longest biosequence in the input sample. This gives us the pattern "$*(0,x)\ v\ *(0,\infty)\ w\ *(0,x)$". For comparison, we also test with other classes to observe the effects of the fixed pattern form and the use of the range specifiers. In total we have the following for candidates of $\mathcal{P}$:
1. patterns of fixed form "$*(0,x)\ v\ *(0,\infty)\ w\ *(0,x)$" where v, $w \in (\Sigma \cup \{*\})^*$
2. patterns of fixed form "$*\ v\ *\ w\ *$" where v, $w \in (\Sigma \cup \{*\})^*$
3. regular patterns with at most 3 variable occurrences

To evaluate how a pattern p in a k-mmg fits a sample S, the most intuitive measure is the number of sequences in S which matches the pattern p, i.e. $\sharp(S \cap L(p))$. Another useful measure is that of *coverage*, introduced in [6], which is the total number of sequences in $L(p)$ of length up to the longest sequence in S, i.e. $\sharp(\{w \in L(p) \mid w \text{ is not longer than the longest sequence in } S\})$ which we shall simply refer to as the *coverage of* p. Intuitively, the coverage of p should be related to the amount of generalization committed by $L(p)$ based on the sample S, and this has been observed in [6].

4 Results

Table 1 show the mmgs obtained from the different pattern classes $\mathcal{P}$. We report only patterns in the k-mmg with either very high number of matching sequences or with very low coverage, with emphasis on the former. Each pattern listed is followed by two values (x, y), where x is the number of sequences in S which matches the pattern, and y is the coverage of the pattern (for simplicity, range specifiers have been ignored in the computation of these coverages).

Here we see that the use of range specifiers enables **MMG** to not mix up between the two segments in the pattern, which is most evident in the sigD patterns. Without the use of range specifiers, the patterns obtained from fixed form patterns (pattern class 2) are often not much better than those obtained from regular patterns (pattern class 3). We also note that good correspondence with the known consensus pattern almost always come from patterns with "high" number of matching sequences and "low" coverage, even though we currently lack a method to decide the ideal range of these values.

5 Discussions

In summary, we found that using more restrictive classes of languages in the **MMG** algorithm gave us consensus patterns that matches the known consensus patterns fairly reasonably. (Though not included in this shorter version of the paper, we also found that patterns using so-called *composite symbols*, as in the PROSITE patterns, tend to yield less resembling patterns.) Of course, this does not mean that more restrictive classes necessarily produce better patterns. For example the class generated by "$*(0,x)\ v\ w\ *(0,x)$" where v, $w \in \Sigma^*$, though more restrictive than the class "$*(0,x)\ v\ *(0,\infty)\ w\ *(0,x)$", would not produce the consensus patterns we want in our test cases.

On the other hand, the results do show some promise for using the **MMG** algorithm in finding consensus patterns, provided that a suitable pattern class is given as input to the algorithm. Since each computation of the **MMG** algorithm runs very quickly (typically in a few seconds), it is conceivable for the algorithm to be employed interactively — where a user would iteratively specify a class of patterns to use, and decide when the output is suitable for consensus patterns.

It would be ideal if we can furthermore automate this process by running the algorithm with successively more restrictive classes of languages, and stop when successive iterations provide no further improvements in the patterns obtained. To do this, we need only (1) a criteria to evaluate if a set of patterns is better than another (for example the measures we have discussed herein may be adapted for this purpose), and (2) a measure to classify how restrictive is a class of languages. Problem (1) seems to be the more difficult problem, but we note that even the classification in (2) may not be simple. For the classes of languages used in this paper, proper inclusions between the classes allow us to estimate how restrictive the classes are. However, this will not work for classes that are incomparable with respect to inclusion. For example, we could not say if the class generated by patterns of the form "$*w$" is less, more, or just as restrictive as that generated by "$w*$", since they do not include each other — though assuming homogeneity in word generation, they should intuitively be considered equally restrictive. For this reason, more theoretical work on the complexity of subclasses of the pattern languages is perhaps needed to improve the use of the **MMG** algorithm.

Acknowledgements. We thank Prof. Satoru Kuhara (Kyushu University) for introducing us to the topic.

References

1. 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.
2. 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, pages 649–660. Springer-Verlag, 1994.
3. 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.
4. J. D. Helmann and C. P. Moran. *RNA Polymerase and Sigma Factors*, chapter 21, pages 289–312. American Society Microbiology, 2001.
5. Y. Makita, M. Nakao, N. Ogasawara, and K. Nakai. DBTBS: Database of transcriptional regulation in Bacillus Subtilis and its contribution to comparative genomics. *Nucl. Acids Res.*, 32:75–77, 2004.
6. Y. K. Ng, H. Ono, and T. Shinohara. Measuring over-generalization in the minimal multiple generalizations of biosequences. In *Proceedings of The Eighth Conference on Discovery Science (DS'05)*, volume 3735, pages 176–188. Springer-Verlag, 2005.
7. I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: the TEIRESIAS algorithm. *Bioinformatics*, 14(1):55–67, 1998.
8. T. Shinohara. Polynomial time inference of extended regular pattern languages. In *RIMS Symposia on Software Science and Engineering, Kyoto, Japan*, volume 147, pages 115–127. Springer-Verlag, 1982.
9. C.J. Sigrist, L. Cerutti, N. Hulo, A. Gattiker, L. Falquet, M. Pagni, A. Bairoch, and P. Bucher. PROSITE: A documented database using patterns and profiles as motif descriptors. *Brief Bioinform.*, 3:265–274, 2002.
10. T. Takae, T. Kasai, H. Arimura, and T. Shinohara. Knowledge discovery in biosequences using sort regular patterns. Workshop on Applied Learning Theory, 1998.
11. 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.

	regular pattern of fixed form with range specifiers		regular pattern of fixed form		regular pattern with at most 3 variable occurrences	
	$k = 5$	$k = 10$	$k = 5$	$k = 10$	$k = 5$	$k = 10$
sigD	*AGC***CG*** (14, 2.1×10^{35}) *CAC***CG*** (11, 2.0×10^{35}) ***AA**C***CCGATA*** (10, 1.8×10^{33}) ***TA**C***CCGA*** (10, 2.9×10^{34})	***AA**C***CCGATAT*** (8, 4.3×10^{32}) *AC*A**CGA*** (7, 6.4×10^{34}) *T**TA*****CCGA*** (5, 8.2×10^{33}) *ATAC***CCGATA*** (5, 5.0×10^{32})	***CCGA***C* (21, 8.2×10^{34}) *CG***CGATA*** (17, 1.7×10^{34})	***CCGATA*****GC*** (9, 4.1×10^{33}) *ACG***CGATAT*** (8, 1.8×10^{33}) ***GCCGATA***AC* (7, 1.0×10^{33})	***TA**C***CG*** (18, 2.1×10^{35}) ***AAA**C***CGATA*** (11, 2.1×10^{33}) *AGC***CGATA*** (9, 7.2×10^{33})	***AAA**C***CGATAT*** (8, 5.0×10^{32}) *A**TA**C**A*****CG*** (7, 1.7×10^{34}) *TTAC***CCG*** (6, 2.9×10^{33})
sigE	***CA*****CATA*** (42, 8.0×10^{29}) *TAAA***ATA*** (20, 3.2×10^{29}) ***CA***A**ATA*** (18, 7.9×10^{29})	*TAAA***ATA*** (20, 3.2×10^{29}) ***CA***A**ATA*** (18, 7.9×10^{29}) ***CA***T**CATA*** (17, 2.0×10^{29}) ***CAT*****CATA**T* (16, 8.3×10^{28}) *T**CATA*****CATA*** (12, 2.3×10^{28})	***CATA*****TA*** (48, 8.0×10^{29}) *A**ATA***AAG* (18, 3.3×10^{29}) *T**ATA*****TA**G* (8, 3.1×10^{29})	*T**CATA*****ATA*** (23, 7.9×10^{28}) *A**ATA***AAG* (18, 3.3×10^{29}) *G**CATA*****ATA*** (13, 7.9×10^{28}) ***CATA***TGTA* (13, 9.2×10^{28})	***CATA*****TA*** (48, 8.0×10^{29}) *A**ATA*****TA**G* (18, 3.3×10^{29})	*A**ATA***AG* (22, 7.9×10^{29}) ***CATA***TGTA* (13, 9.2×10^{28}) *G**CATA***T**ATA*** (13, 7.9×10^{28})
sigH	***AGG**A***GAAT*** (23, 2.2×10^{28}) *GA*T**AA*** (22, 6.7×10^{29}) ***AGG**A*A**AAT*** (16, 2.2×10^{28}) ***GG**G**AT*****GAAT*** (7, 5.4×10^{27})	***AGG**A*T**AA*** (16, 7.9×10^{28}) *A**AGG**A*A**AAT*** (12, 5.4×10^{27}) *A**AGG**A*C**GAAT*** (7, 1.3×10^{27}) ***AGG**A*T**GAAT*** (7, 5.4×10^{27}) ***GG**G**AT*****GAAT*** (7, 5.4×10^{27})	*AA*AAA* (32, 4.2×10^{29}) *GAA***GAAT*** (23, 8.0×10^{28}) *AA**AGG*****GAAT*** (11, 5.4×10^{27}) ***TA**A**T*****GAAT*** (8, 2.2×10^{28})	*AAAA*AAA* (12, 4.9×10^{28}) *AA**AGG*****GAAT*** (11, 5.4×10^{27}) *GAAG***GAAT*** (10, 2.2×10^{28}) ***TA**A**T*****GAAT*** (8, 2.2×10^{28}) ***GAATT***AAAT* (7, 5.4×10^{27})	*AA*AAA* (32, 4.9×10^{29}) *GAA***GAAT*** (23, 8.0×10^{28}) AA**AGG***AT***GAAT*** (8, 1.6×10^{26})	***GAA***AAAT* (16, 8.0×10^{28}) ***GG**A**A*****GAAT*** (13, 2.2×10^{28}) AA**AGG***AT***GAAT*** (8, 1.6×10^{26}) *GA**AGG*****GAAT*** (8, 5.4×10^{27})
sigW	***TGAAAC*****CGTA*** (21, 9.9×10^{23}) ***TGAAAC*****TA**G* (11, 3.7×10^{24}) ***GAAAC**TTT*ATAT* (8, 5.3×10^{22})	***TGAAAC**C*A**CGTA**T* (8, 1.4×10^{22}) *T**TGAAAC*****TA**G* (6, 8.9×10^{23}) ***TGAAAC***TA**CGTA**TA* (5, 3.2×10^{21}) ***GAAAC**TTTTT*ATAT* (5, 2.9×10^{21})	Same as patterns obtained from regular patterns of fixed form with range specifiers		***TGAAAC*****GTA*** (23, 3.7×10^{24}) ***TGAAAC***TA*G* (12, 2.4×10^{24}) ***GAAAC**TTT*ATAT* (8, 5.3×10^{22})	A***TGAAAC***TA*G (8, 5.7×10^{23}) ***TGAAAC**C*C**CGTA**T* (8, 1.4×10^{22}) ***TGAAAC***CTA*G (5, 8.9×10^{23}) ***GAAAC**TTTTT*ATAT* (5, 2.9×10^{21})

Consensus patterns sigD: *TAAA*GCCGATAT*; sigE: *CATAT*CATACA*, *ATATT*CATACA*;
sigH: *AGGTATT*GAATT*; sigW: *TGAAACN*CGTA*

Table 1. MMG results from different hypothesis space