Beilstein-Institut

Glyco-Bioinformatics – *Bits 'n' Bytes of Sugars*
October 4th – 8th, 2009, Potsdam, Germany

Mining Patterns from Glycan Structures

Ichigaku Takigawa[1,3], **Kosuke Hashimoto**[1,2], **Motoki Shiga**[1,3], **Minoru Kanehisa**[1] **and Hiroshi Mamitsuka**[1,3,*]

[1]Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji 611 – 0011, Japan

[2]DHHS/NIH/NLM, National Center for Biotechnology Information (NCBI), 8600 Rockville Pike, Bethesda, MD 20894, U.S.A

[3]Institute for Bioinformatics Research and Development (BIRD), Japan Science and Technology Agency (JST), Kawaguchi-shi, Saitama 332 – 0012, Japan

E-Mail: [*]mami@kuicr.kyoto-u.ac.jp

Received: 20th February 2010 / Published: 10th December 2010

Abstract

Glycans can be directed trees in which nodes are monosaccharides and edges are linkages extending between monosaccharides in the direction from the amino acid-connecting monosaccharide. We present an efficient method for mining frequent and statistically significant subtrees from glycan trees. The two key points of the method are: (1) It can reduce the number of redundant subtrees obtained by usual frequent subtree mining techniques and (2) can keep significant subtrees only by removing frequent but insignificant subtrees, like those with only one or a few nodes. We confirmed the efficiency of the approach in various manners, including biological significance. Our approach would be useful for mining unknown conserved patterns in larger glycan datasets to be obtained by a high-throughput manner in the future.

Introduction

Glycans or carbohydrate sugar chains are a major class of cellular macromolecules, working for a lot of important biological functions including antigen-antibody interactions [1] and cell fate controlling [2]. Like twenty types of amino acids for proteins and four types of bases for nucleic acids, building blocks of glycan structures can be monosaccharides, such as fructose, galactose, glucose and mannose [3]. Glycan structures are formed in that monosaccharides are connected to each other, allowing more than two branches extended from one monosaccharide but without any cycle. Glycans can be then *trees* in a computer science sense which consist of two parts: *nodes* and *edges* connecting nodes without forming any cycle. In addition, glycans can be *directed trees*, in which edges are directed, meaning that an edge is directed from a *parent* to a *child* [1]. This unique feature of glycans makes them different from other macromolecules such as nucleic acids and proteins: they are simple, non-branched, left-to-right type sequences of their building blocks. Furthermore this feature has made hard to determine the glycan structures experimentally and keep the size of structural databases on glycans relatively small and the speed of increasing the database size slow. However, due to long-term experimental efforts, the current number of structurally different glycans being accumulated in major glycan databases reaches about ten thousand, which might increase more by high-throughput techniques in the future [4]. This situation allows conducting data-driven techniques in bioinformatics for analyzing glycan structures and finding embedded biological significance [5, 6].

Our approach is based on techniques in data mining (or machine learning) to find rules, patterns or hypotheses from a large number of given examples. Machine learning techniques or problem settings can be classified into roughly three types: classification, clustering and pattern mining [7, 8]. Classification is to capture rules for distinguishing examples of each label (or class) from others, where labels are assigned to all examples, while clustering and pattern mining are conducted in each class. That is, clustering is to partition given examples into some groups and pattern mining is to find patterns common to given examples. Originally these techniques were applied to simple tables, where a row and a column correspond to an example and its attribute, respectively. A typical example in biology is a dataset of gene expression measured by cDNA microarray, where a row corresponds to some experimental condition while a column is a gene, each element in a table showing an expression value of the corresponding gene under the corresponding experimental condition [9]. Given a gene expression table, pattern mining means capturing rules which are genes, each being over- (or under-) expressed through given experimental conditions. The techniques to obtain this type of rules from a table have already been developed in a wide variety of manners and matured in some sense. The current focus in data mining and machine learning is to capture rules or patterns in more complex data, such as sequences, trees and graphs. Glycans can be directed trees, meaning that mining patterns from glycans can be along with the recent research topic of machine learning. The latest techniques for trees in machine learning are support vector machines with tree kernels [10] for classification, probabilistic models of trees [11 – 15] for clustering, and frequent subtree mining [16, 17] for pattern mining. Some conserved patterns are already known in glycans [18], and thus out of the three problem settings in machine learning, we focus on the third issue in this work, i. e. mining frequent subtrees from given trees.

In frequent subtree mining, a subtree is defined to be *frequent* if it appears more than a certain (pre-specified) number of times, which is called *minimum support* or *minsup*. For example, if we have ten trees and a subtree appears five times out of the ten trees, it is frequent if the minimum support is less than five. In this setting, there already exist efficient algorithms for finding frequent subtrees from a given set of trees, and any of these algorithms can be applied to glycans. However, a simple setting of frequent subtrees has two serious issues to which we address in this work. Firstly, resultant frequent subtrees are very redundant, because if a frequent subtree is found, all subtrees of the frequent subtree are frequent. Secondly, frequent subtrees are not necessarily found in a given set of trees significantly, because small subtrees, such as single nodes, are likely to be frequent, meaning that smaller subtrees are frequent in any dataset. In this work, we present an approach for dealing with these two problems.

Our approach has two parts, each responding to one of the two problems raised in the last paragraph (Note: originally our method appeared in [19]). In the first part, to reduce the redundancy of frequent subtrees, we developed a concept called α-closed frequent subtrees and an algorithm for mining α-closed frequent subtrees efficiently. In the second part, we prepare a dataset, generated randomly according to the distribution of a given set of trees, and employ hypothesis testing on both these two datasets to remove frequent but insignificant subtrees from the frequent subtrees obtained in the first part. With these two parts, we can relax the two serious issues in mining frequent subtrees.

We obtained frequent patterns of glycans by applying our approach to a real dataset of glycans. The frequent patterns generated include well-known, important patterns in glycobiology, the top pattern being peculiar to Type 2 oligosaccharides. We further evaluated our approach by the framework of classification, where we used patterns obtained by our approach as binary attributes of each glycan, an attribute value taking one if the corresponding pattern is in the glycan; otherwise zero. In this framework, a set of glycans can be a table of the attributes generated from the patterns, by which we can compare the discriminative performance of our approach with the most recent methods in machine learning, i. e. support vector machines with tree kernels. From this experiment, we showed that our approach could outperform competing methods by controlling the number of attributes or frequent patterns obtained.

Method

Our method has two parts, which can be clearly separated. However, we merged these two parts into one algorithm for efficiency. This point will be described in Part 3.

Part 1: Mining α-closed frequent patterns

The goal of the first part is to reduce the redundant outputs, i. e. frequent subtrees, and for this issue, we first need to introduce two important concepts: *closed* and *maximal* frequent subtrees [20]. Before that, we briefly define an important notion in frequent pattern mining: The number of appearances of subtree T in a given set of trees is called the *support* of T. Here, a frequent subtree T can be defined to be closed unless the support of any supertree of T is equal to that of T, while it can be maximal unless any supertree of T is frequent. More concretely, the idea of closed frequent subtrees allows to remove a redundant subtree if its support is equal to that of any of its supertrees. A problem of closed frequent subtrees is, however, that in general the support of a frequent tree is close to that of its supertree but not necessarily equal to. On the one hand, this means that in most cases closed frequent subtrees still suffer the redundancy problem of frequent patterns. On the other hand, the idea of maximal frequent subtrees outputs the largest frequent subtrees only, resulting in that many frequent subtrees can be summarized into only one maximal frequent subtree. Thus, maximal frequent subtrees can reduce the number of all frequent subtrees drastically, while a problem is that the support of each frequent subtree is lost. This means that maximal frequent subtrees might be too coarse to analyze all frequent subtrees. Thus, to solve these problems, we present an idea called α-closed frequent subtrees, which are a natural extension of closed frequent subtrees, relaxing the relatively strong restriction of closed frequent patterns to reduce the number of redundant patterns. The α-closed frequent subtrees can be defined in the following manner [19].

Definition 1
A frequent tree T is defined to be α-closed unless the support of any of its frequent supertrees is larger than or equal to $\alpha \times \text{support}(T)$, where α takes a value between zero and one.

We note that α-closed frequent subtrees have the following properties.

Property 1
Given a set of trees and a minimum support, let $\mathscr{F}$ be the set of all frequent subtrees, $\mathscr{C}$ be the set of all closed frequent subtrees, $\mathscr{A}_\alpha$ be the set of all α-closed frequent subtrees, and $\mathscr{M}$ be the set of all maximal frequent subtrees. These sets satisfy that $\mathscr{M} \subseteq \mathscr{A}_\alpha \subseteq \mathscr{C} \subseteq \mathscr{F}$.

PROPERTY 2
$\mathscr{A}_\alpha$ is a monotone increasing family with respect to a, implying $\mathscr{A}_\alpha \subseteq \mathscr{A}_{\alpha'}$ ($\alpha \leq \alpha'$). If $\alpha = 1$, $\mathscr{A}_\alpha = \mathscr{C}$ and if $\alpha = 0$, $\mathscr{A}_\alpha = \mathscr{M}$.

PROPERTY 3
Once we are given a set of α-closed frequent subtrees, we can retrieve all frequent subtrees specified by the given α-closed frequent subtrees.

These properties show the nice features of α-closed frequent subtrees. PROPERTY 1 indicates the inclusive relationships among $\mathscr{M}$, $\mathscr{A}_\alpha$, $\mathscr{C}$ and $\mathscr{F}$, particularly the relationship among $\mathscr{A}_\alpha$, $\mathscr{C}$ and $\mathscr{F}$, being monotone, as shown in PROPERTY 2.

We further developed an efficient algorithm for enumerating all α-closed frequent subtrees from a given number of trees or glycans, keeping approximately the same computational burden of an algorithm for mining closed frequent subtrees. We do not go into the detail of the algorithm developed. Interested readers should refer [19], where we showed the idea behind the algorithm as well as the pseudo-code by which a software on mining α-closed frequent subtrees can be implemented.

Part 2: Ranking frequent subtrees by statistical testing

As mentioned in the Introduction, frequent patterns are not necessarily significant, since smaller patterns are likely to be frequent. Thus we checked the significance of each frequent pattern by using statistical testing over 'case' and 'control' datasets. The case dataset is a given set of trees, while the control dataset was generated by using the case dataset in the following two steps:

1. We compute the distribution of parent-child pairs in the case dataset.
2. We repeat the following procedure over all trees of the case dataset: For each tree, we replace each parent-child pair with another pair according the computed distribution, keeping the structure of the given tree, and save this new tree as a tree in the control dataset.

We then count, for each frequent subtree *T*, how many times *T* appears (and does not appear) in each of the case and control datasets, to make the following 2×2 contingency table.

Table 1. 2 × 2 contingency table.

	Case	Control	Total
T appears	N_{caT}	N_{conT}	N_T
T does not appear	N_{canT}	N_{connT}	N_{nT}
Total	N_{ca}	N_{con}	N

Finally, we can check the independence of the contingency table by one-sided Fisher's exact test. The probability that a contingency table is generated follows a hypergeometric distribution, which is given by using the counts in the contingency table as follows:

$$\mathrm{Pr} = \mathrm{N_T!\ N_{nT}!\ N_{ca}!\ N_{con}!/N_{caT}!\ N_{canT}!\ N_{conT}!\ N_{connT}!}$$

The *p*-value of the one-sided Fisher's exact test on this table can be computed by the sum of all probabilities of tables that are more extreme than this table. We can rank frequent subtrees by the *p*-values of the corresponding contingency tables, according to their significance.

Part 3: Merging two parts for efficiency

In reality, the final output will be the top ranked patterns, say the top *K* patterns. For this case, we can take the following procedure: whenever a frequent pattern is obtained in the mining procedure, i.e. Part 1, we compute its *p*-value by using Part 2 and if the *p*-value is smaller than the largest one of the current lowest *K* *p*-values, we can keep this pattern; otherwise this pattern is discarded. By doing this, we can save the memory space as well as avoid the computational load caused by running the above two parts separately: generating all frequent patterns once and then sorting them by their *p*-values.

Experimental Results

Mining significant subtrees

We used all records in the KEGG Glycan database [21], removing entries of monosaccharides only and those containing nodes labelled by phosphorus (P) or sulphur (S). The total number of glycans we used for the case dataset was 7,454, and we generated a control dataset, which keeps the same size as that of the case dataset.

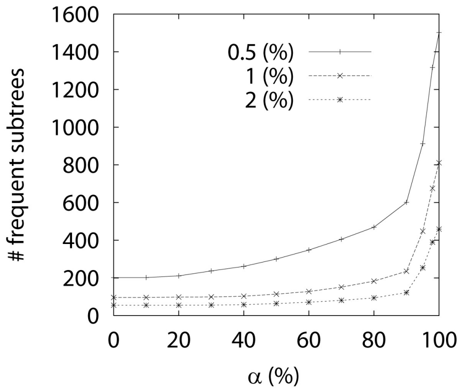
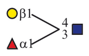
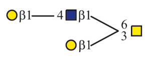
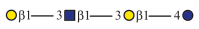
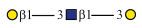
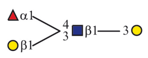


Figure 1. Mining significant subtrees in the KEGG Glycan Database: Frequent subtrees were obtained by our method, controlling the minimum support and the value of α.

Figure 1 shows the number of frequent subtrees obtained by our method, controlling the minimum support and the value of α. We note that here we did not use Part 2 or that the top *K* was set at a larger number than the number of frequent patterns obtained by the minsup in each setting. This figure indicates that the number of outputs was decreased by reducing the value of α (or increasing the *minsup*), meaning that redundancy of the output can be relaxed by the idea of α-closed frequent subtrees.

Rank		p-value	Support
1		1.6e-46	381
2		1.1e-40	164
3		5.0e-26	109
4		5.6e-26	233
5		8.6e-26	83

🔺 Fucose (Fuc)
🟡 Galactose (Gal)
🟨 N-acetylgalactosamine (GalNAc)
🔵 Glucose (Glc)
🟦 N-acetylglucosamine (GlcNAc)

Figure 2. The top five significant subtrees, obtained by our method under the setting that *minsup* is 0.5% and α is 0.4.

Figure 2 shows the top five significant subtrees, obtained by our method under the setting that *minsup* is 0.5% and α is 0.4. We first note that they are not ranked by the number of appearances but by *p*-values. In fact, the third-ranked subtree has the support of 109, which is smaller than that of the fourth-ranked subtree, but has a lower *p*-value than that of the fourth. We found that all these five subtrees are biologically well-known, significant glycan motifs. For example, the first subtree is a common subtree among Lewis X, Lewis Y and sialyl Lewis X, which are all Type 2 oligosaccharides, being attached to the membrane of red blood cells and used for the categorization of human blood types. The second subtree is famous as the core of O-glycans, a major class of glycans, and similarly the third and fourth subtrees are typical core parts of another major class of glycans, called glycosphingolipids. These results indicate that our approach detected typical patterns or motifs in glycans, implying that relatively lower ranked subtrees might include unknown patterns in the literature.

Applying to classifying glycans in their classes

We then applied the obtained frequent patterns to the problem of classifying glycans to validate the significance of the obtained subtrees. In fact, once we obtain a set of frequent patterns, we can generate attributes of an arbitrary glycan by checking whether this glycan has each of the frequent patterns: If the glycan has a frequent pattern, the corresponding attribute takes one; otherwise zero. This means, if 1,000 frequent subtrees are obtained, we can assign 1,000 binary attributes to an arbitrarily given glycan.

The problem setting is that we regard 485 O-glycans in KEGG as positives (which used as the case dataset) and randomly synthesized glycans as negatives (which are used for the control dataset as well). We performed 10×10 cross-validation over these glycans, meaning that we repeated the following ten times: we divided the examples into ten blocks and used nine out of ten blocks for training and the rest for testing. The performance was measured by AUC (the area under the ROC curve) [22], a current standard measure for classification, and the final result was averaged over all 100 runs of the 10×10 cross-validation.

We chose three competing methods, all using support vector machines but different three tree kernels. The first one is convolution kernel [23], which enumerates all possible subtrees of given two trees and checks the number of subtrees shared between the two trees. The second one is co-rooted subtree kernel [10], which checks the similarity between given two trees in a similar manner to the convolution kernel but by using part of their all subtrees. The third one is 3-mer kernel [23], which focuses on subtrees with only three nodes for computing similarity between two trees.

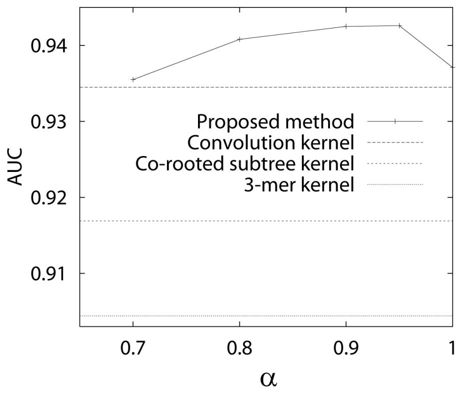


Figure 3. AUCs of the three competing methods and our proposed method for which *minsup* = 2.

Figure 3 shows the AUCs of the three competing methods and our proposed method for which *minsup* = 2. We note that the performance of our method can be controlled by changing α. For α of less than 0.7, the number of frequent patterns (or attributes) is too small to have a high AUC, while our method outperformed other three competing methods in AUC for α of no less than 0.7. In addition, we note that our method can show all attributes used in experiments as subtrees while other kernel-based methods cannot.

Concluding remarks

We have presented an approach for mining subtrees embedded in glycans which are frequent as well as statistically significant. We confirmed that the patterns ranked in the top five by our approach are consistent with the known literature in glycobiology. Experimental results further indicated the performance advantage of our method in classification. We did not check the scalability of our approach in our experiments, but the time- and space-scalability are also an advantage of our approach which would be more useful for larger-sized glycan databases to be obtained in the future. Our approach found the significant patterns removing redundancy. They are however still relatively similar to each other in some cases and needed to be summarized more in some form. Possible future work would be to develop a method for providing an overview of the significant patterns mined, showing their correlations with biological classification.

References

[1] Varki, A., Cummings, R., Esko, J., Freeze, H., Hart, G. and Marth, J. (1999) *Essentials of Glycobiology*. CSHL Press, New York (NY, USA).

[2] Stanley, P. (2007) Regulation of notch signaling by glycosylation. *Current Opinion in Structural Biology* **17**(5):530 – 535.
doi: http://dx.doi.org/10.1016/j.sbi.2007.09.007.

[3] Marth, J.D. (2008) A unified vision of the building blocks of life. *Nature Cell Biology* **10**(9):1015 – 1016.
doi: http://dx.doi.org/10.1038/ncb0908-1015.

[4] Haslam, S.J., North, S.J. and Dell, A. (2006) Mass spectrometric analysis of N- and O-glycosylation of tissues and cells. *Current Opinion in Structural Biology* **16**:584 – 591.
doi: http://dx.doi.org/10.1016/j.sbi.2006.08.006.

[5] Raman, R., Raguram, S., Venkataraman, G., Paulson, J.C. and Sasisekharan, R. (2005) Glycomics: an integrated systems approach to structure-function relationships of glycans. *Nature Methods* **2**(11):817 – 824.
doi: http://dx.doi.org/10.1038/nmeth807.

[6] Mamitsuka, H. (2008) Informatic innovations in glycobiology: relevance to drug discovery. *Drug Discovery Today* **13**(3 – 4):118 – 123.
doi: http://dx.doi.org/10.1016/j.drudis.2007.10.013.

[7] Bishop, C.M. (2006) *Pattern Recognition and Machine Learning*. Springer, New York (NY, USA).

[8] Han, J., Cheng, H., Xin, D. and Yan, X. (2007) Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery* **15**(1):55 – 86.
doi: http://dx.doi.org/10.1007/s10618-006-0059-1.

[9] Speed, T. (2003) *Statistical analysis of gene expression microarray data*. Chapman & Hall/CRC Press, Boca Raton (Florida, USA).
doi: http://dx.doi.org/10.1201/9780203011232.

[10] Shawe-Taylor, J. and Cristianini, N. (2004) *Kernel methods for pattern analysis*. Cambridge University Press, Cambridge (UK).

[11] Aoki, K.F., Ueda, N., Yamaguchi, A., Kanehisa, M., Akutsu, T. and Mamitsuka, H. (2004) Application of a new probabilistic model for recognizing complex patterns in glycans. *Bioinformatics* **20**(Supplement 1):i6-i14.
doi: http://dx.doi.org/10.1093/bioinformatics/bth916.

[12] Ueda, N., Aoki-Kinoshita, K.F., Yamaguchi, A., Akutsu, T., Mamitsuka, H. (2005) A probabilistic model for mining labeled ordered trees: capturing patterns in carbohydrate sugar chains. *IEEE Transactions on Knowledge and Data Engineering* **17**(8):1051 – 1064.
doi: http://dx.doi.org/10.1109/TKDE.2005.117.

[13] Aoki-Kinoshita, K.F., Ueda, N., Mamitsuka, H. and Kanehisa, M. (2006) Profile-PSTMM: Capturing tree-structure motifs in carbohydrate sugar chains. *Bioinformatics* **22**(14): e25-e34.
doi: http://dx.doi.org/10.1093/bioinformatics/btl244.

[14] Hashimoto, K., Aoki-Kinoshita, K.F., Ueda, N., Kanehisa, M. and Mamitsuka, H. (2006) A new efficient probabilistic model for mining labeled ordered trees. In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, (Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven and Dimitrios Gunopulos, eds.). pp. 177 – 186, ACM Press.

[15] Hashimoto, K., Aoki-Kinoshita, K.F., Ueda, N., Kanehisa, M. and Mamitsuka, H. (2008) A new efficient probabilistic model for mining labeled ordered trees applied to glycobiology. *ACM Transactions on Knowledge Discovery from Data* **2**(1):Article 6.

[16] Asai, T., Abe, K., Kawasoe, S., Arimura, H., Sakamoto, S. and Arikawa, S. (2002) Efficient substructure discovery from large semi-structured data. In: *Proceedings of the 2002 SIAM International Conference on Data Mining (SDM 2002)*, (Robert Grossman, Jiawei Han, Vipin Kumar, Heikki Mannila and Rajeev Motwani, eds.). pp. 158 – 174, SIAM.

[17] Zaki, M.J. (2002) Efficiently mining frequent trees in a forest. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, (David Hand, Daniel Keim and Raymond Ng, eds.). pp. 71 – 80, ACM Press.

[18] Lanctot, P.M., Gage, F.H. and Varki, A.J. (2007) The glycans of stem cells. *Current Opinion in Chemical Biology* **11**:373 – 380.
doi: http://dx.doi.org/10.1016/j.cbpa.2007.05.032.

[19] Hashimoto, K., Takigawa, I., Shiga, M., Kanehisa, M. and Mamitsuka, H. (2008) Mining significant tree patterns in carbohydrate sugar chains. *Bioinformatics* **24**(16):i167-i173.
doi: http://dx.doi.org/10.1093/bioinformatics/btn293.

[20] Chi, Y., Xia, Y., Yang, Y. and Muntz, R.R. (2005) Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *IEEE Transactions on Knowledge and Data Engineering* **17**(2):190 – 202.
doi: http://dx.doi.org/10.1109/TKDE.2005.30.

[21] Hashimoto, K., Goto, S., Kawano, S., Aoki-Kinoshita, K.F., Ueda, N., Hamajima, M., Kawasaki, T. and Kanehisa, M. (2006) KEGG as a glycome informatics resource. *Glycobiology* **16**: 63R-70R.
doi: http://dx.doi.org/10.1093/glycob/cwj010.

[22] Mamitsuka, H. (2006) Selecting features in microarray classification using ROC curves. *Pattern Recognition* **39**(12):2393 – 2404.
doi: http://dx.doi.org/10.1016/j.patcog.2006.07.010.

[23] Yamanishi, Y., Bach, F. and Vert, J.-P. (2007) Glycan classification with tree kernels. *Bioinformatics* **23**(10):1211 – 1216.
doi: http://dx.doi.org/10.1093/bioinformatics/btm090.