Efficient Tree-Matching Methods for Accurate Carbohydrate Database Queries

Kiyoko F. Aoki kiyoko@kuicr.kyoto-u.ac.jp
Atsuko Yamaguchi atsuko@kuicr.kyoto-u.ac.jp
Yasushi Okuno okuno@kuicr.kyoto-u.ac.jp
Tatsuya Akutsu takutsu@kuicr.kyoto-u.ac.jp
Nobuhisa Ueda ueda@kuicr.kyoto-u.ac.jp
Minoru Kanehisa kanehisa@kuicr.kyoto-u.ac.jp
Hiroshi Mamitsuka mami@kuicr.kyoto-u.ac.jp

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

Abstract

One aspect of glycome informatics is the analysis of carbohydrate sugar chains, or glycans, whose basic structure is not a sequence, but a tree structure. Although there has been much work in the development of sequence databases and matching algorithms for sequences (for performing queries and analyzing similarity), the more complicated tree structure of glycans does not allow a direct implementation of such a database for glycans, and further, does not allow for the direct application of sequence alignment algorithms for performing searches or analyzing similarity. Therefore, we have utilized a polynomial-time dynamic programming algorithm for solving the *maximum common subtree* of two trees to implement an accurate and efficient tool for finding and aligning maximally matching glycan trees. The KEGG Glycan database for glycan structures released recently incorporates our tree-structure alignment algorithm with various parameters to adapt to the needs of a variety of users. Because we use similarity scores as opposed to a distance metric, our methods are more readily used to display trees of higher similarity. We present the two methods developed for this purpose and illustrate its validity.

Keywords: pairwise tree alignment, glycobiology, carbohydrate sugar chain, glycan, tree matching

1 Introduction

To the field of glycobiology, we can apply bioinformatics to analyze the structures of carbohydrate sugar chains, or glycans. This will lead us to glycome informatics, which, with the continuing progress of technological resources, is becoming more of a reality than when glycobiology first appeared some twenty years ago. Glycobiology is the study of the structure, biosynthesis, and biology of saccharides (sugar chains or glycans) that are widely distributed in nature. Since most glycans are on the outer surface of cellular and secreted macromolecules, they assist in the mediation of a wide variety of events in cell-cell and cell-matrix interactions crucial to the development and function of a complex multicellular organism, and they also serve as regulatory switches. Therefore, the understanding behind the structure and into the function of these sugar chains are indeed vital in enabling biologists to further understand the development and functioning of higher level organisms.

The difficulty in working with glycans in terms of informatics is that their structures are not simple like sequences; glycans are branched tree structures with various types of linkages. The nodes of the trees correspond to monosaccharides (a representative set is listed in Table 1) and the edges contain information such as to which hydroxyl (carbon number) of which nodes they are linked and their linkage type, or anomer (i.e., α or β). Figure 2 is an example of a glycan chain. The root is normally

represented on the right end, and each monosaccharide is linked to its "parent" (to its right), to one of its parent's (usually six) hydroxyls in an α or β type linkage.

Currently there are many facets of these glycan structures which are barely understood, especially in terms of their biosynthesis and function [6]. A better understanding of these tree structures would lead glycobiologists towards insights enabling them to pursue their studies more effectively. The recent release of a glycan database called KEGG Glycan [14, 27] should enable such research. However, in the development of such a database, effective methods for representing and utilizing the data is necessary. For example, even the seemingly simple action of querying a database becomes complex when the data structures become trees (as opposed to sequences). In such a query, many possibilities exist to establish a "match." However, it is easy to develop a greedy algorithm that would quickly become inefficient with larger and larger data sizes. Therefore, the necessity for an efficient and accurate tree-matching algorithm brought about the implementation of a *maximum common subtree (MCST)* algorithm for carbohydrate chains. By structuring glycans in the database as trees whose nodes are labeled with monosaccharides, we can utilize the MCST problem by structuring a query as a tree, comparing it to the glycans in the database, and returning a list of those that match most closely with the query (i.e., those whose matching subtrees are largest).

There is a slew of work underway in comparing tree structures, particularly in relation to RNA secondary structures and evolutionary trees. Tai [22] was the first to generalize the edit model from strings to tree structures, where he used the concept of an "edit distance" to compare the difference between two tree structures. His model was improved and applied to bioinformatics through the years by Zhang [16, 26], Shapiro [18], and the like. A theoretical approach to tree alignment was proposed by Jiang [13], and as of late, similarity measures for RNA secondary structures have been popular topics of research [12, 17].

Interestingly, despite the fact that in sequence alignment methods, the bioinformatics mainstream has used a certain "similarity score" where a higher value is used to indicate a better alignment, in approaching tree alignment from an algorithmic point of view, "distance" has been used primarily as the measure for comparing trees where lower distance values indicate higher similarity. Although there is an equivalency of similarity and distance algorithms for sequences where simple scoring schemes are used, the conversions are more complicated for more sophisticated scoring schemes, such as a PAM scoring matrix for scoring amino acid similarities. Only recently, with the development of local similarity measures for RNA secondary structures [11], has this difference between distance and similarity been clarified in comparing trees, independent of our work.

Evolutionary trees are also often studied in bioinformatics, where the *leaves* are uniquely labeled as distinct species (from a set of species) and the *maximum agreement subtree (MAST)* calculated to find the trees whose set of matching leaves is largest [5, 9, 10]. We note that evolutionary trees are often concerned with the similarity of the leaves in the trees as opposed to the structure of the trees themselves. Therefore, MAST would be considered a different problem from the MCST problem.

To summarize the contributions of this paper,

- we have developed and implemented a tree alignment methodology for carbohydrate structures that outputs similarity score values, and in addition, provides both local and global tree alignment variations.

- Our methods utilize an MCST algorithm that has been shown to find the maximum common subtree of two trees in polynomial time [8]. Although it was defined for unlabeled trees, it has easily been extended to the labeled trees of glycan structures. Our methodology is currently utilized for querying the KEGG Glycan web-based database server [14, 27].

- Finally, because we use similarity scores, it is possible to use similar statistical analysis techniques to Vingron [24], Smith [25], and Altschul [3], in analyzing our tree-matching algorithms.

2 Preliminaries

This section will define the concepts used in the rest of this paper, which entails descriptions of concepts in graph theory, computational geometry, bioinformatics, and glycobiology. We attempt to give concise definitions of the concepts necessary to understand our motivation and our work.

2.1 Graph Theoretic Basics

A *tree* is defined as an acyclic connected graph, whose vertices we refer to as *nodes*. A *rooted tree* is a tree having a specific node called the root, from which the rest of the tree extends. Nodes that extend from a node x by one edge are called the *children* of x, and conversely, x would be called the *parent* of these children. A node with no children is a *leaf*. A *subtree* of tree T is a tree whose nodes and edges are subsets of those of T, an *ordered tree* is the rooted tree in which the children of each node are ordered, and a *labeled tree* is a tree in which a label is attached to each node. Finally, a *forest* is a set of trees. Note that all trees considered in this paper are labeled and rooted trees. We also note that because we are working with trees whose ordering may not be specifically defined (or even known), all trees considered in this work are unordered.

2.2 The Maximum Common Subtree Problem (MCST) and Algorithm

The maximum common subtree problem can be defined as the following.

Input: Two unordered, labeled, rooted trees T_1 and T_2.

Output: The tree T_c which is a subtree of both tree T_1 and T_2 and whose number of edges is the maximum among all such possible subtrees.

We hereafter refer to unordered, labeled, rooted trees simply as trees when no ambiguity occurs.

Now, we will review the algorithm to solve the MCST problem, originally shown to run in polynomial time in [8]. For the sake of simplicity, we will compare nodes in describing this algorithm, but note that we can easily compare edges as well. The formalized procedure is as follows. Let T_1 and T_2 be rooted trees. Let $\{u_1, \ldots, u_n\}$ (resp. $\{v_1, \ldots, v_m\}$) be the set of nodes in T_1 (resp. T_2). For a node u, $sons(u)$ denotes the set of child nodes of u. For a node v in tree T, $T(v)$ denotes the subtree of T induced from v and its descendants. For each pair of u_i, v_j, we compute $R[u_i, v_j]$, which denotes the size of a maximum common subtree of $T_1(u_i)$ and $T_2(v_j)$ under the condition that u_i corresponds to v_j. Assuming without loss of generality that $|sons(u)| \leq |sons(v)|$, $\mathcal{M}(u, v)$ denotes the set of one-to-one mappings from $sons(u)$ to $sons(v)$. Then, $R[u_i, v_j]$ can be computed by the following dynamic programming procedure:

$$\begin{aligned} R[u, 0] &= 0, \\ R[0, v] &= 0, \\ R[u, v] &= 1 + \max_{\psi \in \mathcal{M}(u,v)} \left\{ \sum_{u_i \in sons(u)} R[u_i, \psi(u_i)] \right\} \end{aligned}$$

As such, the score for the maximum common subtree can be found from the value of $R[u_i, v_j]$ with the maximum score. The resulting maximum common connected subtree correlated with this score can then be retrieved by backtracking to find the matching nodes that contributed to this score.

2.3 Global and Local Sequence Alignment Algorithms

Because the methods that we have developed utilize the dynamic programming procedures for sequence alignment, we review these procedures in this subsection. There are two variations to the procedure for aligning sequences with gaps, global and local. We present both procedures here.

Global sequence alignment with gap costs can be computed by the following dynamic programming procedure [19, 20]:

$$\begin{aligned} S[i,0] &= d \cdot i, \\ S[0,j] &= d \cdot j, \\ S[i,j] &= \max \begin{cases} S[i,j-1]+d, \\ S[i-1,j]+d, \\ S[i-1,j-1]+w(x_i,y_j) \end{cases} \end{aligned}$$

where $x_1 \ldots x_n$ and $y_1 \ldots y_m$ are two input sequences, $d < 0$ is a penalty for a gap, and $w(x,y)$ denotes the score between residues x and y. $S[n,m]$ gives the score of an optimal global alignment.

The local sequence alignment procedure can be computed by the following dynamic programming procedure:

$$\begin{aligned} S[i,0] &= 0, \\ S[0,j] &= 0, \\ S[i,j] &= \max \begin{cases} 0, \\ S[i,j-1]+d, \\ S[i-1,j]+d, \\ S[i-1,j-1]+w(x_i,y_j), \end{cases} \end{aligned}$$

where $\max_{i,j} S[i,j]$ gives the score of an optimal local alignment.

2.4 Glycans

As an amino acid is to a protein and a nucleotide is to a DNA sequence, the basic component of glycans is the monosaccharide, of which a handful are most common in higher animal oligosaccharides (listed in Table 1). Each unit is linked to one or more other monosaccharides by various types of linkages, depending on the configuration (i.e., α or β) and the carbon number which connects them, so we must deal with complex, branched, tree structures, unlike the linear structure of proteins or DNA.

Simply analyzing the existing tree structures of glycans is important, especially considering that there does exist patterns in these structures that are used for recognition by various agents such as pathogens as well as by proteins that enable the development and functioning of the organism. For example, it has been shown in the literature that lectins recognize glycans via certain monosaccharide configurations (patterns) on the outer-most portion of their tree structures; sialic acids as ligands have been shown to be recognized by proteins of animal, plant and microbial origin, or more specifically, sialic acid binding lectins [23]. It seems that recognition can be affected by specific structural variations and modifications of certain monosaccharides, *their linkage to the underlying sugar chain, and the structure of these chains* [23]. Not only would an understanding of structural patterns in glycans be used to further support studies in sugar recognition, but such work would be helpful in unraveling their biological functions [6, 7]. Therefore, our work is an effort to enable pattern matching in known glycan structures to not only reveal possible motifs in glycans, but to also lead to conjectures into their functions.

3 KCaM Methods

We now present our methods for tree-matching in this section, which we call KCaM, for KEGG Carbohydrate Matcher.

3.1 Approximate Matching

In this section, we introduce the method for performing an approximate matching of carbohydrate structures. These methods are based on the global and local alignment algorithms for sequences and on the algorithm for finding a maximum common subtree of two trees (see Section 2.2). There are two versions for the approximate tree-matching algorithm: *global approximate matching* and *local approximate matching*. We introduce each of these algorithms individually.

3.1.1 Global Approximate Matching

We first describe the global approximate matching algorithm. Combining the global sequence alignment procedure (Section 2.3) with the maximum common subtree procedure (Section 2.2), we have developed the following dynamic programming procedure for finding the maximum common subtree of two trees T_1 and T_2:

$$
\begin{aligned}
Q[u,0] &= \sum_{u_i \in T_1(u)} d(u_i), \\
Q[0,v] &= \sum_{v_i \in T_2(v)} d(v_i), \\
Q[u,v] &= \max \left\{ \begin{array}{l}
\max_{v_i \in sons(v)} \left\{ Q[u,v_i] + d(v) + \sum_{v_j \in sons(v)-\{v_i\}} Q[0,v_j] \right\}, \\
\max_{u_i \in sons(u)} \left\{ Q[u_i,v] + d(u) + \sum_{u_j \in sons(u)-\{u_i\}} Q[u_j,0] \right\}, \\
w(u,v) \ + \ \max_{\psi \in \mathcal{M}(u,v)} \left\{ \sum_{u_i \in sons(u)} Q[u_i,\psi(u_i)] \ + \sum_{v_i \in sons(v)-\psi(sons(u))} Q[0,v_i] \right\}.
\end{array} \right.
\end{aligned}
$$

In the above, $d(u)$ denotes the cost for deleting a node u, and $w(u,v)$ represents the similarity between nodes u and v. The similarity of incoming edges to u and v can also be taken into account in the following way. For a node u, $p(u)$ denotes the parent of u. Then we can define $w(u,v)$ by

$$
w(u,v) = \max \left\{ \begin{array}{l}
0, \\
\alpha \cdot \delta(label(u), label(v)) - \beta \cdot (1 - \delta(ulabel(p(u),u), ulabel(p(v),v))) \\
\qquad\qquad - \beta \cdot (1 - \delta(dlabel(p(u),u), dlabel(p(v),v))),
\end{array} \right.
$$

where $\delta(x,y) = 1$ iff. $x = y$, $label(u)$ denotes the name of the monosaccharide unit, or sugar, and $ulabel(p(u),u)$ (resp. $dlabel(p(u),u)$) indicates the carbon number of sugar $p(u)$ (resp. sugar u) to which the edge $(p(u),u)$ is connected. In the current implementation, we use $\alpha = 100.0$ and $\beta = 25.0$ without further tuning.

3.1.2 Local Approximate Matching

Next, we describe the local approximate matching algorithm using the local sequence alignment procedure from Section 2.3. We modify the global approximate matching algorithm and obtain the following procedure for local approximate matching:

$$
\begin{aligned}
Q[u,0] &= 0, \\
Q[0,v] &= 0,
\end{aligned}
$$

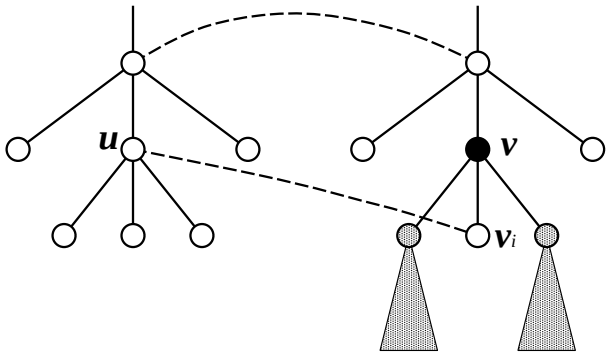


Figure 1: Deletion costs for black and shaded nodes are taken into account in global approximate matching, whereas the deletion cost for only the black node is used in local approximate matching.

$$Q[u,v] \;=\; \max \left\{ \begin{array}{l} 0, \\ \max_{v_i \in sons(v)} \{Q[u,v_i] + d(v)\}, \\ \max_{u_i \in sons(u)} \{Q[u_i,v] + d(u)\}, \\ w(u,v) \;+\; \max_{\psi \in \mathcal{M}(u,v)} \left\{ \displaystyle\sum_{u_i \in sons(u)} Q[u_i, \psi(u_i)] \right\}. \end{array} \right.$$

It should be noted that $\max_{u,v} Q[u,v]$ gives the score of an optimal matching.

The approximate matching algorithm is useful for the case when a core structure is known, and we are looking for a specific pattern near the leaves. In this case, we can enter the query with a single tree containing both the core and the pattern, and the gapped alignment will return all similar trees, with the core matched at the root, any pattern matched near the leaves, and gaps allowed in the intermediate section.

However, it may be the case that the user is only looking for a single specific connected component with known anomer and linkage information. That is, a match with absolutely no gaps may be preferred, which may be actually more biologically sensible. Thus, we developed the exact matching algorithm, described in the next section.

3.2 Exact Matching

The exact matching algorithm uses the same methodology for matching tree structures as introduced in the previous sections, except that we match *edges* as opposed to *nodes*, and we do not allow for gaps. In addition to global and local exact matching, there is also another parameter taken into consideration in performing the match operation; the anomer and carbon values may or may not be ignored to determine whether the edges match.

3.2.1 Local Exact Matching

We can adapt the dynamic programming procedure for local approximate matching from the previous section to local exact matching by letting u and v represent edges, and letting $w(u,v)$ equal one (1) when edges u and v match, and zero (0) otherwise. The definition of a match between edges u and v would be determined by the parameters selected. That is, if we are considering only the names of the monosaccharides as the criteria for a match, then regardless of the anomer value or the carbon numbers by which the monosaccharides are linked (which may be unknown), $w(u,v)$ would be given a value of one (1) as long as the nodes on each end of the linkage correspond to the same monosaccharides. However, if we were given the stricter criteria of ensuring that the monosaccharides, anomer value, and carbon numbers must be equal to be considered a match, then the value of one (1) would be applied less generously, where all three factors must be equal in order to receive a score. In this case, a so

called "ordering" of the edges may be taken into consideration, seeing as in some cases, the ordering of the children may be important in a query (i.e., the carbon number/hydroxyl to which a child is attached may be considered important). Since the exact matching algorithm does not allow for gaps, we also set the gap penalty and score for a mismatch a value of $-\infty$ for the exact match case. Thus, the local exact matching algorithm will return the single largest maximum common connected subtree between two trees rooted at u and v where $Q(u, v)$ is maximized.

3.2.2 Global Exact Matching

The global exact matching algorithm was developed for the case when not only do we want the single largest common subtree, but we want to see all possible connected matches between the two input trees. Thus, the global exact matching algorithm recursively calls the local exact matching procedure for all subtrees of the two input trees that have not been matched. That is, after the single largest matching tree has been found, the resulting, unmatched portions of both trees are then used as the input trees for another round of the local exact matching algorithm. The recursion ends when either no matches are found, or one of the input trees has been matched in its entirety to the other. Accordingly, the score returned is the summation of the scores for each recursive call to the local exact matching procedure, and the set of matched subtrees returned make up the forest of matching subtrees.

Table 1: Common monosaccharide names and their symbols.

Monosaccharide name	Sym.
Glucose	▲
Galactose	●
Mannose	○
N-acetyl neuraminic / sialic acid	◆
N-acetylglucosamine	■
N-acetylgalactosamine	□
Fucose	△
Xylose	▽
Glucuronic acid	◈
Iduronic acid	◈

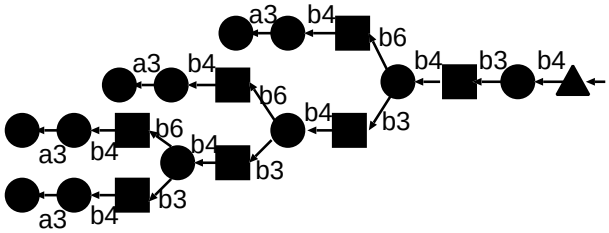
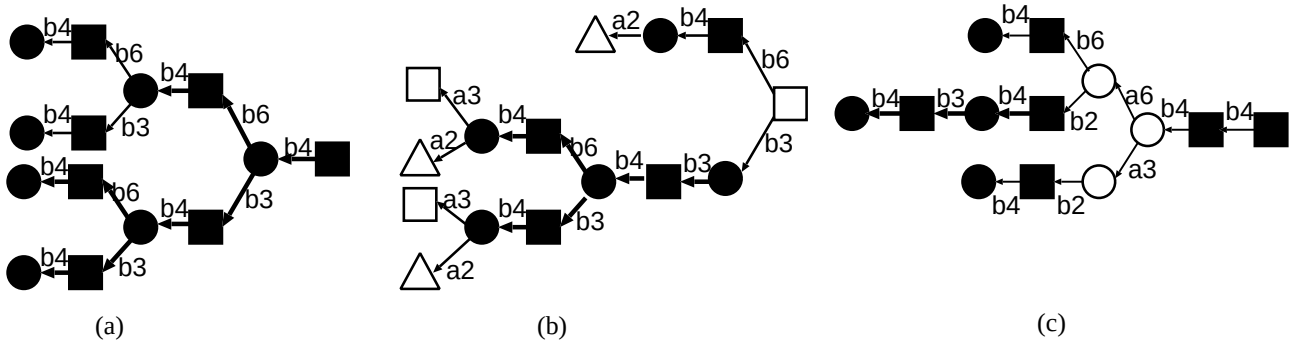


Figure 2: A glycan chain: Eicosasaccharide (KEGG Glycan ID G07296).



(a) (b) (c)

Figure 3: KEGG Glycan IDs (a) G02667 [score=9], (b) G02178 [score=6], and (c) G03993 [score=3]. The thicker edges correspond to the matching subtree.

4 Results

In this section, we will present an example of a match that was found by KEGG Glycan, and then we will take a look at the efficiency and accuracy of our algorithms. We will give a brief analysis of the running times of the algorithms presented in this paper, followed by a statistical analysis of the scores produced by our methodologies, based on the distributions of the outputted scores from a randomly selected set of tree structures from the KEGG Glycan database.

4.1 Matching Example

For our example, we selected KEGG Glycan ID G07296 (Figure 2) as the structure to use for our query. Of course, we could have easily used the Glycan structure editor to manually specify any arbitrary tree structure as a query as well. Using Local Exact Matching with Sugars only (disregarding carbon numbers and anomers), a listing of over 4000 hits were returned as a result of this query, sorted by similarity score. We arbitrarily selected a few results to illustrate the difference in similarity based on these scores. In each of these results, we illustrate the matching edges by thicker lines. Figure 3 illustrates a sample of the resulting matches. We can clearly see how the smaller similarity scores indeed return trees that look increasingly dissimilar from the query.

4.2 Algorithm Running Times

Because we utilized the dynamic programming technique for the basis of our tree matching methodologies, we can analyze the running times in a straightforward manner. With the exception of the Global Exact Matching algorithm, because the degree of each vertex is bounded (i.e., there are at most six hydroxyls for most monosaccharides, while the only exception may be sialic acid which has at most nine), the running times for all algorithms are on the order of $O(mn)$ where m and n are the number of nodes (or edges, for the exact matching case) in each of the trees being compared. Because of the recursion used in the Global Exact Matching algorithm, however, the running time is only slightly increased by a factor of $min(m, n)$, resulting in a running time of $O(mn\,min(m, n))$. Therefore, while maintaining efficiency, our algorithms are also able to produce meaningful measures of similarity to allow for the analysis in the following subsection.

4.3 Statistical Score Distributions

In analyzing the resulting matching algorithms in terms of accuracy, we considered the local approximate matching algorithm as our representative algorithm. Our hypothesis was that if our algorithm were accurate, then just as in sequence alignment algorithms such as the Smith-Waterman algorithm [19, 20], the distribution for a randomly selected set of carbohydrate structures should produce an extreme value distribution [3, 15, 21]. We took the classification of glycans provided by KEGG Glycan as listed in Table 2. We then randomly selected 100 glycan structures from within each class and performed an all-by-all round of ungapped, local matches.The probability density estimates of the score distributions indeed fit an extreme value distribution. The distributions of two of the major classes, N-Glycans and O-Glycans, are given in Figures 4 and 5.

Table 2: Glycan Classes and their sizes (number of glycan structures).

Class name	Size	Class name	Size	Class name	Size
GPI	75	Glycerolipid	30	Glycosaminoglycan	596
Glycoside	931	LPS	447	N-Glycan	2068
Neoglycoconjugate	111	O-Glycan	746	Oligosaccharide	34
Polysaccharide	344	Sphingolipid	914		

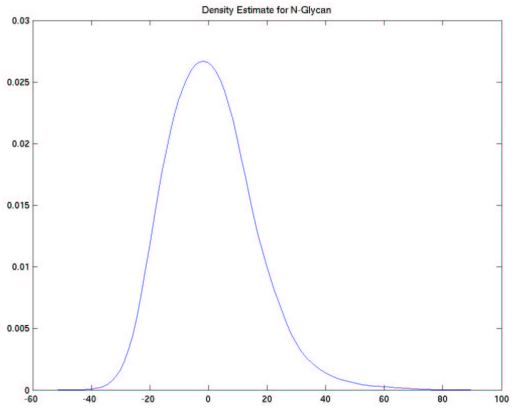


Figure 4: Score distribution of N-Glycans.

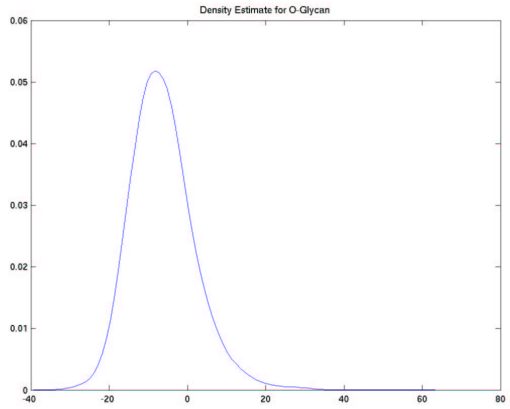


Figure 5: Score distribution of O-Glycans.

We can claim that these ungapped score distributions are homologous to the extreme-value distribution of scores as analyzed for local sequence alignment [2]. Much research has gone into characterizing score distributions and analyzing the significance statistics to sequence alignment scores to provide better results that have more biological meaning [15, 24, 25]. We believe that our algorithm is indeed a step forward in the same direction for tree-matching and alignment. Therefore, based on these promising results, we are confident that we will soon be able to provide confidence values to the resulting matches to give the user an idea of the significance of the results provided by KEGG Glycan.

5 Discussion and Future Work

In this paper, we presented an algorithm for aligning tree structures for use in querying a carbohydrate database. We can claim that our methodologies are efficient in that we use a known polynomial-time algorithm for finding the maximum common subtree of two trees. Further, we have also shown that the results of this algorithm on a real dataset of alignment scores seem to be accurate when the score distribution for a randomly selected set of glycans from a glycan tree database was analyzed.

Currently, work is continuing on an appropriate scoring mechanism for both the approximate matching and exact matching methods, similar to PAM or BLOSUM for amino acid similarities. That is, more work can be done in providing more biologically meaningful scores. The penalty costs may be tuned more appropriately, and the exact matching method may use various weights according to the parameters selected. In addition, based on the analysis of the score distributions, we will soon be able to provide significance metrics similar to BLAST [4] for each score, and we hope to move forward towards multiple tree alignments and prediction in the near future.

Acknowledgments

This work was partially supported by Grant-in-Aid for Scientific Research on Priority Areas (C) “Genome Information Science” from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

[1] Akutsu, T., An RNC algorithm for finding a largest common subtree of two trees, *IEICE Trans. Information and Systems*, E75-D:95–101, 1992.

[2] Altschul, S.F., Bundschuh, R., Olsen, R., and Hwa, T., The estimation of statistical parameters for local alignment score distributions, *Nucleic Acids Research*, 29(2):351–361, 2001.

[3] Altschul, S. and Gish, G., Local alignment statistics, *Methods Enzymol.*, 266:460–480, 1996.

[4] Altschul, S., Madden., T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D., Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[5] Amir, A. and Keselman, D., Maximum agreement subtree in a set of evolutionary trees: metrics and efficient algorithms, *SIAM J. Comput.*, 26(6):1656–1669, 1997.

[6] Bertozzi, C.R. and Kiessling, L.L., Carbohydrates and glycobiology review: chemical glycobiology, *Science*, 291:2357–2364, 2001.

[7] Drickamer, K., Two distinct classes of carbohydrate-recognition domains in animal lectins, *J. Biol. Chem.*, 263:9557–9560, 1988.

[8] Edmonds, J. and Matula, D., An algorithm for subtree identification, *SIAM Rev.*, 10:273–274, 1968.

[9] Farach, M., Przytycka, T.M., and Thorup, M., On the agreement of many trees, *Information Processing Letters*, 55(6):297–301, 1995.

[10] Finden, C.R. and Gordon, A.D., Obtaining common pruned trees, *J. Classification*, 2:255–276, 1985.

[11] Höchsmann, M., Töller, T., Giegerich, R., and Kurtz, S., Local similarity in RNA secondary structures, *Proceedings of CS Bioinformatics*, IEEE Computer Society Press, 159–168, 2003.

[12] Jannson, J. and Lingas, A., A fast algorithm for optimal alignment between similar ordered trees, *Lecture Notes in Computer Science*, 2089:232–240, 2001.

[13] Jiang, T., Wang, L., and Zhang, K., Alignment of trees - an alternative to tree edit, *Theoretical Computer Science*, 143:137–148, 1995.

[14] Kanehisa, M., *et. al.*, The KEGG databases at GenomeNet, *Nucleic Acids Res.*, 30:42–46, 2002.

[15] Karlin, S. and Altschul, S.F., Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes, *Proc. Natl. Acad. Sci.*, 87:2264–2268, 1990.

[16] Lin, G.H., Ma, B., and Zhang, K., Edit distance between two RNA structures, *Proceedings of the Fifth Annual International Conference on Computational Biology*, ACM Press, 211–220, 2001.

[17] Sczyrba, A., Krüger, J., Mersch, H., Kurtz, S., and Giegerich, R., RNA-related tools on the Bielefeld Bioinformatics Server, *Nucleic Acids Research*, 31(13):3767–3770, 2003.

[18] Shapiro, B.A. and Zhang, K., Comparing multiple RNA secondary structures using tree comparisons, *CABIOS (Computer Applications in the Biosciences)*, 6:309-318, 1990.

[19] Smith, T.F. and Waterman, M.S., Identification of common molecular subsequences, *J. Mol. Biol.*, 147(1):195–197, 1981.

[20] Smith, T.F. and Waterman, M.S., Comparison of biosequences, *Adv. Appl. Math.*, 2:482–489, 1981.

[21] Smith, T.F., Waterman, M.S., and Burks, C., The statistical distribution of nucleic acid similarities, *Nucleic Acids Research*, 13:645–656, 1985.

[22] Tai, K., The tree-to-tree correction problem, *Journal of the ACM*, 26:422–433, 1979.

[23] Varki, A., Sialic acids as ligands in recognition phenomena, *FASEB J.*, 11:248–255, 1997.

[24] Vingron, M. and Waterman, M.S., Statistical significance of local alignments with gaps, In *Bioinformatics: From nucleic acids and proteins to cell metabolism (extended abstract)*, pp. 75–84, eds. Schomburg D. and Lessel, U., Weinheim, 1995.

[25] Waterman, M. and Vingron, M., Sequence comparison significance and poisson approximation, *Statistical Science*, 9:401–418, 1994.

[26] Zhang, Z., Statman, R., and Shasha, D., On the editing distance between unordered labeled trees, *Information Processing Letters*, 42(3):133–139, 1992.

[27] `http://glycan.genome.ad.jp/`