ELSEVIER

Carbohydrate Research 304 (1997) 61–67

CARBOHYDRATE RESEARCH

Representation and searching of carbohydrate structures using graph-theoretic techniques

Ian J. Bruno [a,1], Nick M. Kemp [a,b], Peter J. Artymiuk [b], Peter Willett [a,*]

[a] *Krebs Institute for Biomolecular Research, Department of Information Studies, University of Sheffield, Western Bank, Sheffield S10 2TN, UK*

[b] *Krebs Institute for Biomolecular Research, Department of Molecular Biology and Biotechnology, University of Sheffield, Western Bank, Sheffield S10 2TN, UK*

Received 28 January 1997; accepted 20 June 1997

Abstract

This paper describes how the carbohydrate structures in the Complex Carbohydrate Structure Database (CCSD) can be represented by labelled graphs, in which the nodes and edges of a graph are used to denote the residues and the inter-residue linkages, respectively, of a carbohydrate. These graph representations are then searched with a subgraph-isomorphism algorithm. We describe the use of one such algorithm, that due to Ullmann, and demonstrate that it provides a very precise way of searching the structures in CCSD. We also describe the use of screening techniques that can eliminate many of the CCSD structures from the subgraph-isomorphism search, with a consequent increase in the speed of the search. © 1997 Elsevier Science Ltd.

Keywords: Graph theory; Complex Carbohydrate Structure Database; Sequence comparison; Subgraph-isomorphism algorithm

1. Introduction

There is increasing interest in the use of database techniques for the storage and retrieval of information pertaining to the structures of carbohydrates [1]. The Complex Carbohydrate Structure Database (CCSD) was started following the Carbohydrate Structure Database Workshop that took place in Auburn, New York in 1986 [2–4]. At this Workshop, mechanisms were established for collecting, evaluating, and organizing the large amount of published structural data and for acquiring new data as it was generated. The Complex Carbohydrate Research Centre, University of Georgia has developed software to permit rapid searching of the resulting database of sequences. This search program is referred to as CarbBank and is designed for use with PC-compatible equipment. The structure-searching facilities in CarbBank use pattern-matching routines that permit the identification of substrings of residues, exact matches, and chains with some minimum number of residues in common. In this paper, we discuss the use of an alternative approach to the searching of CCSD that uses techniques from the branch of mathematics called *graph theory*.

* Corresponding author. E-mail: p.willett@sheffield.ac.uk.

[1] Current address: Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge CB2 1EZ, UK.

PII S0008-6215(97)00196-1

In order to accomplish this, we propose the use of *graphs* to represent the carbohydrate structures in the CCSD. The *nodes* and *edges* of the graphs correspond to the saccharide residues and the linkages between them, respectively. The nodes and edges are *labelled* with the residue types and linkage types, respectively. Full details of this representation are given in the Experimental section below. Each edge in a carbohydrate graph has an associated direction corresponding to the direction of the glycosidic linkage between two residues, and is thus represented as a *directed graph*. The use of a graph-theoretic description means that searching operations on databases of carbohydrates can be implemented using isomorphism algorithms, which compare one graph with another to determine the equivalence relationships that exist between them [5–8]. Specifically, we discuss the use of a *subgraph-isomorphism* algorithm for *substructure searching*, which is the ability to identify all of the molecules in a database that contain a user-defined partial structure [9]. In what follows, we shall use the term *residue-by-residue* searching to refer to the identification of a query substructure in a carbohydrate structure. Analogous graph-theoretic procedures have been described for the representation and searching of two-dimensional (2D) small molecules [10], of three-dimensional (3D) small molecules [11], and for the detection of motifs in 3D protein structures [12].

Subgraph-isomorphism algorithms typically make use of a backtracking, depth-first tree search [5,6]. In the context of a residue-by-residue search of a carbohydrate database, this involves matching a pair of residues (one from the query substructure and one from a database structure) and then attempting to match their neighbours. If the neighbours can be matched then the process is repeated for their neighbours; alternatively, if a match is not obtained, then it is undone and another possible mapping tried. The procedure terminates when the complete set of query residues (and hence the entire query substructure) has been matched or when every possible mapping has been tried. Improvements in performance can be achieved using *relaxation* techniques [9]. Relaxation involves assigning a value to a residue and then iteratively refining it by examining the values of neighbouring residues; at each stage, a note is made of those residues in the matching structure that have equal values. This procedure results in a database structure and a query substructure being iteratively partitioned into sets of potentially matching residues, thus reducing the numbers of comparisons that need to be made during the subsequent backtracking search. The subgraph-isomorphism algorithm used here is that described by Ullmann [13], which forms the basis for several operational substructure-searching systems for searching databases of 2D or 3D small molecules [9].

There is, however, a problem with the use of such algorithms. Subgraph isomorphism belongs to the class of NP-complete problems, for which no efficient algorithms are known to exist [14]. Rapid database searching can hence be achieved only if some means can be found of eliminating many of the molecules in a database from the time-consuming subgraph-isomorphism search [10]. This is done in small-molecule database systems by means of an initial *screen search*, which involves a preliminary check that all of the substructural fragments specified in the query substructure are present in the database structure that is being inspected, but without taking account of the detailed inter-connections of these features. Only those few molecules that contain all of the query fragments, which are referred to as *screens*, are then passed on for the detailed subgraph-isomorphism search, which takes account of the fragments' detailed inter-connections. We have therefore implemented the use of a screen search that uses fragments called *augmented residues* and generates a screenset using the algorithm of Cringean et al. [15] in order to overcome this problem with our graph-theoretic approach.

2. Results and discussion

Representation of carbohydrate structures by connection tables.—A carbohydrate graph is represented by a tabular data structure, called a *connection table*, in which each line describes one of the residues in a carbohydrate. Each line consists of two parts preceded by an integer label that identifies the residue in question: the first part describes the residue itself, while the second part identifies the residues that are attached to the central residue described in the first part. Specifically, the first part contains the residue type, the ring size, the anomeric configuration, and the absolute configuration, and the second part contains the integer labels of the residues to which the central residue is connected, together with the bond type for each such connection. An example of a structure from the CCSD and the corresponding connection table are illustrated in Fig. 1. This shows, for example, that node number 2 is a Glc residue that is joined by a $1 \rightarrow 4$ linkage to node number 3, which

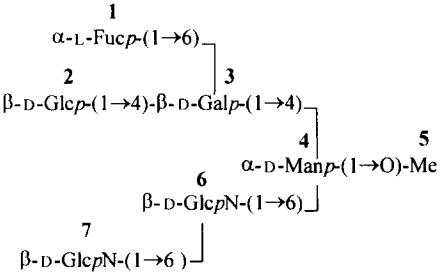
the next line of the connection table reveals to be a Gal residue. 33,174 (89%) of the structures contained in CCSD14 (released June 1995) were successfully converted into connection tables which were then used in the searching experiments discussed below.

The 3,953 structures that were not converted into connection tables in our present analysis contain repeat sequences. It would seem possible that the information present in repeating sequences could be fully represented by storing a structure that contains the sequence twice, or as many times as are necessary to cope with the largest query a system can handle. Additional structural information, for example, non-glycosidic components and modified residues not distinguished in our representation, could be represented in an additional field, by including other residue types or by representing substituents as additional nodes in the graph.

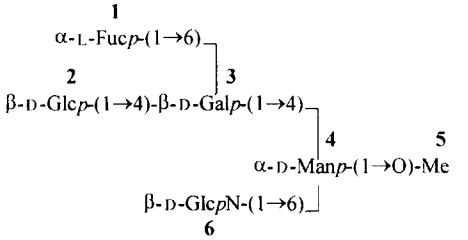
The set of 33,174 connection tables from CCSD14 constituted the dataset used for evaluating the graph-theoretic techniques. The connection tables for these structures were stored in a direct-access file that was accessed by means of an index. The index contained the identifying number of each structure, the number of the record in the main file at which that structure's connection table started, and the number of residues in the structure. It should be emphasised that the connection table is the internal representation of a carbohydrate that forms the basis for the computer processing, and that this representation is invisible to the user who wishes to carry out database searches.

Screenset generation.—Augmented residues were represented by linear strings that are analogous in format to individual lines in a residue-based connection table (such as that illustrated in Fig. 1). The representation was *canonicalised* such that each type of augmented residue had one, unambiguous, representation (Fig. 2).

A FORTRAN 77 program was written that systematically generated all the augmented-residue occurrences from the file of connection tables, and from this created the screenset. There were 157,444 fragment occurrences in the dataset of which 4,074 were distinct; the list of unique fragments was partitioned



(a)



(b)

Number	Residue					Connections					
	RT	*r*	*a*	*S*	*C*	b_1	res_1	b_2	res_2	b_3	res_3
1	Fuc	*p*	α	L	1	(1→6)	3				
2	Glc	*p*	β	D	1	(1→4)	3				
3	Gal	*p*	β	D	3	(1←6)	1	(1←4)	2	(1→4)	4
4	Man	*p*	α	D	3	(1←4)	3	(1→O)	5	(1←6)	6
5	Me	-	-	-	1	(1←O)	4				
6	GlcN	*p*	β	D	2	(1→6)	4	(1←6)	7		
7	GlcN	*p*	β	D	1	(1→6)	6				

(c)

Fig. 1. (a) and (b) CCSD representations and (c) connection-table representation of a complex carbohydrate (a). *RT* is the residue type, *r* the ring size, *a* the anomeric configuration, *S* the absolute configuration, *C* the connectivity, res_n the residue number of neighbour *n*, and b_n the corresponding bond type. If structure (b) is a query and screens are assigned to each of the six residues then the screen search will only retrieve structure (a) if the residue labelled 6 (at least) is considered to be partially specified.

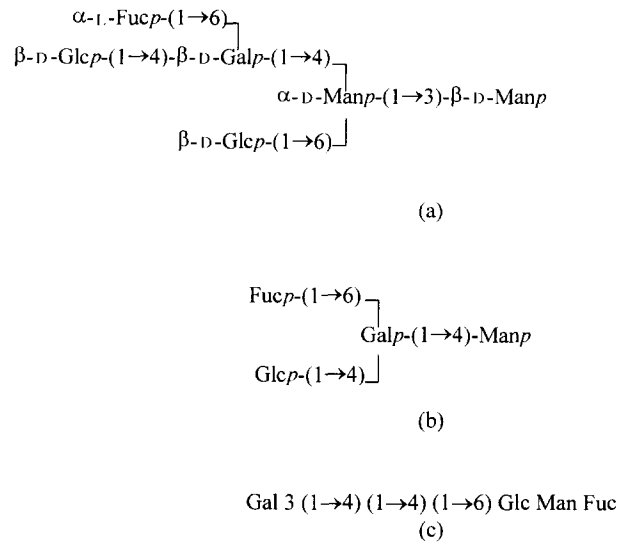


Fig. 2. An example of an augmented-residue fragment (b) extracted from structure (a), and its canonicalised representation (c). Canonicalisation is achieved by ordering the connections in increasing numerical value of linkage descriptor. The canonicalised representation includes, in order, parent residue type, connectivity, all linkage types of connections to the parent residue, and connected residue types in the same order as the linkages connecting them to the parent residue.

to give a screenset containing 128 screens.

The assignment of screens for all the structures in the dataset are stored in a *bit-map*. The bit-map is an array of $P \times N$ where P is the number of screens in the screenset and N is the number of structures. Here P and N are 128 and 33,174, respectively. An element $[i,j]$ of the bit-map is set to TRUE if structure i has been assigned screen j. The row i of a bit-map is referred to as the *bit-string* of structure I.

Implementation of the searching procedures.—A query substructure will normally contain two types of query residue: we shall refer to these as *fully specified* and *partially specified* residues. A fully specified residue is one for which all of its connected residues are defined: this will typically be a residue that is near to the centre of the query. Residues at the edge of the query will typically have only some of their linkages specified explicitly, with the remaining linkages being connected to other, undefined residues. The search procedures allowed residues in the query to be defined as either fully or partially specified. However, the screen search was performed by considering all residues in a query to be partially specified in order to ensure that no potential hits were missed.

The screen search and residue-by-residue search were implemented in FORTRAN77 using the University of Salford FTN77/386 compiler on a 75-MHz Pentium PC which runs under DOS. The program takes as input the connection table representing the query substructure and the screenset. The database bit-map is also read in as input. The screen search is performed by generating a bit-string from the query connection table and comparing it with each row of the database bit-map in turn. A particular database structure is passed on for the residue-by-residue search only if its bit-string contains all of the bits specified in the bit-string representing the query substructure. The Ullmann algorithm is then used to match the query connection table against the connection tables representing those structures which pass the screen search.

Query structures were formulated algorithmically from randomly selected structures in CCSD14. In all, 15 sets of queries were created, each set containing 100 queries. Within each set the query structures are defined by a fixed number of augmented-residue fragments. Augmented residues may differ in the number of residues they contain. If the query sets are numbered 1–15 then the mean number of residues contained in query set N is approximately $N + 1.5$.

The residue-by-residue search.—The mean time over all 15 query sets for comparing one query structure with all 33,174 structures in our data set from CCSD14 is 28.1 s with a standard deviation of 10.7. The mean times for each query set are in the range 16.7–58.6 s. These search times are broadly comparable with those of CarbBank 3.0 when run on the same hardware platform. In fact, our figures are dominated by the time taken to read in the connection tables, which accounts for over 90% of the total times.

Searches performed without the screen search were between 1.7 and 10.9 times slower. The time taken to perform the screen search is very small indeed, since it involves merely bit-level operations. However, its implementation results in a significant reduction or *screenout* in the number of structures that need to be read in from disk and processed by the Ullmann algorithm, and thus the total time taken to search. Over the 15 query sets the mean screenout was 89% with a standard deviation of 6.9, so that the time-consuming subgraph-isomorphism search needed to be applied to just 11% of the database on average.

Searches were performed with and without the comparison of anomeric configuration, absolute configuration, and ring form during the residue-by-residue matching stage of the search, which we shall refer to as higher- and lower-level specificity, respec-

tively. The mean time required just for the Ullmann search stage, when averaged over the 15 query sets without the additional information, was 1.56 s with a standard deviation of 0.42. However, when the search was performed with the additional information the mean time dropped to 1.26 s with a standard deviation of 0.30. With more structural information included during the search, the Ullmann algorithm can more quickly decide whether residues in the query and file structures do in fact match, and the overall search time will be quicker.

The methods described above were tested on a subset of 30 queries taken from the 15 query sets used previously. The results gained from using this set of queries were compared with those from searches performed with CarbBank Version 3.0 using the same queries and the same set of 33,174 structures from CCSD14. The latter searches were carried out using the standard CarbBank methods, which involve the specification of a series of *residue + complexes* that are linked using Boolean operators to form a query [16]. A residue + complex is defined as a 'core' residue plus linkages to other residues and information about the attached groups. The standard method of searching requires that a structure is loaded from the database and query residues are identified and stored in buffers as *residue complexes*. Searches using this method can lead to mismatches or *false-drops* when a structure is retrieved containing each specified *residue + complex*, but not connected in the manner described by the query substructure. In the method described in this paper there is no such limitation in searching for substructures. Fig. 3 gives an example of a query which leads to false-drops, together with the residue complexes used for the CarbBank search.

With the queries described at the lower level of specificity, that is, without comparison of anomeric configuration, absolute configuration, and ring form, 20 gave the same number of hits with both CarbBank and our method. For the remaining 10 query structures, CarbBank retrieved a total of 57 more struc-

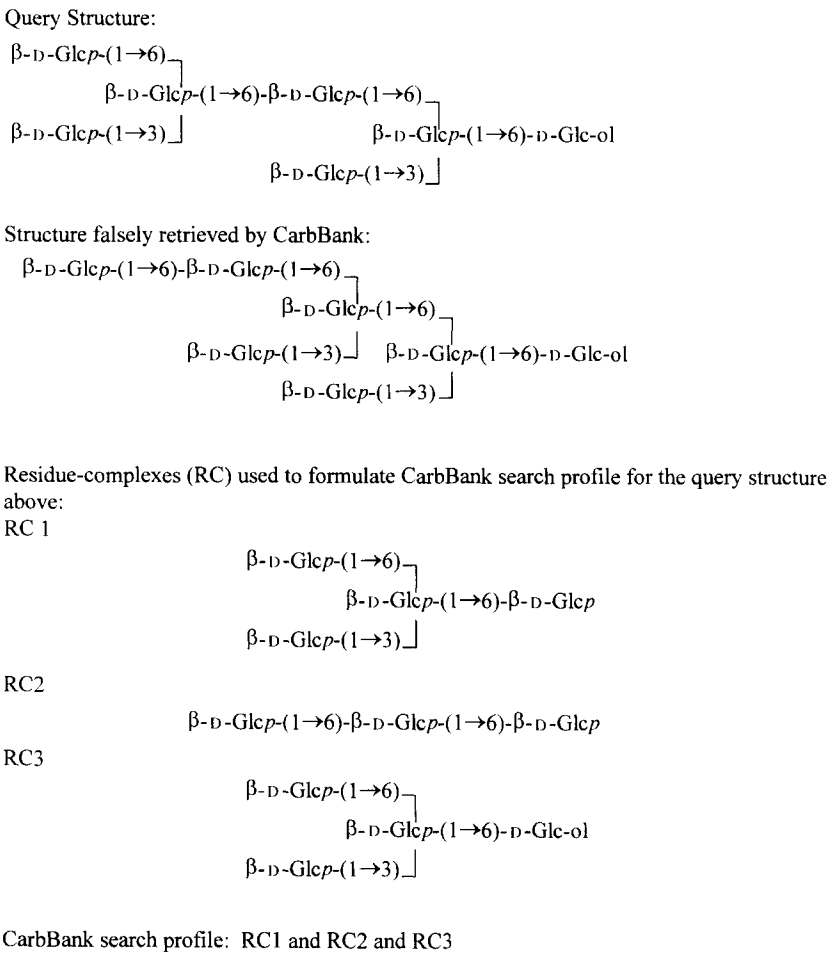


Fig. 3. An example of a query which leads to false-drops when a search is performed with CarbBank. Residue complexes used to construct the search-profile are shown for a search which assumes residues to be fully specified, including residue type, nature of linkages, and additional structural information. For the search our method gave 9 hits and CarbBank gave 10.

tures than our program. The excess structures retrieved by CarbBank are false-drops that do not contain the query as a substructure. For queries with the higher level of specificity, the same 10 CarbBank searches still retrieved a total of 26 excess structures.

The graph-theoretic methods may therefore have certain operational advantages over existing methods. In the first place a complex query may be submitted in its entirety to the search procedure without the necessity of splitting the query into smaller fragments. In addition it appears that the precision of the graph-theoretic procedure is higher without any concomitant loss of information. The incidence of false-drops in the CarbBank searches is greater the larger the substructures that are being searched for. We would thus expect that the differences in search performance between the two approaches will tend to increase as more complex structures start to be included in CCSD.

3. Experimental

Representation of sequences in the CCSD.—Structural features are represented in the CCSD in a manner similar to that used by *Carbohydrate Research* [17]. This follows the recommendations of the IUPAC–IUB Joint Commission on Biochemical Nomenclature for the abbreviated nomenclature of oligosaccharide chains [18]. The root (or type) of the monosaccharide is given a three-letter abbreviation, e.g., glucose and fructose are represented by Glc and Fru, respectively. Three-letter abbreviations for trivial names established by usage are also included in the standard, e.g., 6-deoxygalactose has the trivial name fucose, abbreviated to Fuc. Ring size and anomeric and absolute configuration are displayed with the root abbreviation in the form *a-C-Xxxr* where *a* represents the anomeric configuration (α or β), *C* the absolute configuration (D or L), *Xxx* the residue type, and *r* the ring size (*f*, *p*, or *s* for furanose, pyranose, and septanose, respectively). The root structure may be substituted and rules exist for systematically describing such derivatives, so that, for example, uronic acids, 2-amino-2-deoxy sugars, and alditols are represented by appending A, N, or ol, respectively, i.e. *Xxxr*A, *Xxxr*N, or *Xxx*-ol. Glycosidic linkages are denoted by the notation ($x \rightarrow y$), where *x* and *y* are the integer labels of the atoms involved in the linkage.

Graphs and their use for the representation of chemical structures.—A *graph* is a mathematical construct that describes a set of objects, called *nodes* or *vertices*, and the relationships, called *edges* or *arcs*, that exist between pairs of the objects [5,6,19,20]. More formally, a graph, *G*, consists of a set of nodes, *V*, together with a set of edges, *E*, connecting pairs of nodes ($E \subseteq V \times V$). Two nodes are referred to as *adjacent* if they are connected by an edge. A *labelled graph* is one in which labels are associated with the nodes and/or edges. A *directed graph* is one in which each of the edges specifies not only that a relationship exists between a pair of nodes but also the direction of that relationship.

Subgraph isomorphism.—A *subgraph* of *G* is a subset, *P*, of the nodes of *G* together with a subset, *F*, of the edges connecting pairs of nodes in *P* ($P \subseteq V$ and $F \subseteq P \times P$). Two graphs, *G* and *G'*, are said to be *isomorphic* if they have the same structure, i.e., if there is a correspondence or mapping between the nodes of *G* and of *G'* such that adjacent pairs of nodes in *G* are mapped to adjacent pairs of nodes in *G'*. A graph *G'* is said to be a subgraph of *G* if there exists a one-to-one correspondence between the nodes and edges in *G'* and a subset of the nodes and edges in *G* such that incidence of *G'* is preserved. *Subgraph isomorphism* is illustrated in Fig. 4 where the labelled graph (c) has a correspondence of nodes and edges with subsets of nodes and edges in (a) and (b) such that adjacency is preserved.

The Ullmann subgraph-isomorphism algorithm.—The algorithm starts by invoking a relaxation technique to provide an initial set of matching nodes and then proceeds with a depth-first backtrack search. At each step of the search, as a potential match is made, the relaxation procedure is invoked again, thus further reducing the amount of backtracking that is

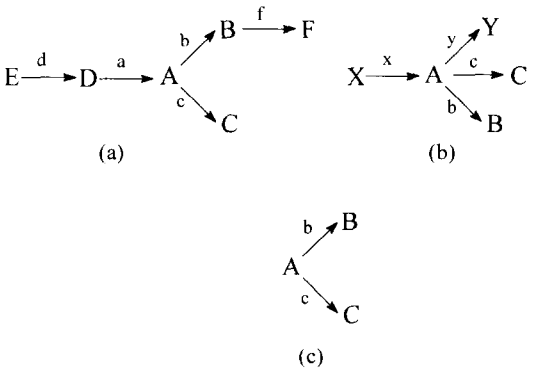


Fig. 4. Subgraph isomorphism: the labelled directed graph (c) has a correspondence of nodes and edges with subsets of nodes and edges in (a) and (b) such that adjacency is preserved. Graph (c) is thus a subgraph of graphs (a) and (b).

required. The relaxation procedure, which Ullmann refers to as refinement [13], is used to limit the number of levels of the search tree that have to be investigated before a mismatch is identified. When modified to handle residues, the algorithm makes use of the fact that if some query residue $Q(X)$ is bonded to another residue $Q(W)$, and if some database-structure residue $S(Z)$ matches with $Q(W)$, then there must also be some residue $S(Y)$ that is bonded to $S(Z)$ and that matches with $Q(X)$: this is a necessary, but not sufficient, condition for a subgraph isomorphism to be present (except in the limiting case of all of the query residues having been matched, when the condition is both necessary and sufficient). The refinement procedure is called before each possible assignment of a database-structure residue to a query residue; and the matched substructure is increased by one residue if, and only if, the condition holds for all residues W, X, Y, and Z.

Screensets.—The screensets developed here use fragments called *augmented residues*. An augmented residue is a fragment which consists of a residue, together with the residues directly linked to it. The approach advocated here involves the generation of all such fragments for the molecules in a dataset, and the sorting and cumulation of the resulting frequencies of occurrence to give a fragment dictionary from which the screenset is generated using the screenset-selection algorithm described by Cringean et al. [15]. This divides an input fragment dictionary into a number of sections such that each section contains approximately the same number of fragment occurrences. Each of the sections resulting from this partitioning procedure represents a single screen, which thus corresponds to the presence or absence in a carbohydrate of a range of types of augmented residue. Studies have shown [9,15,21] that the most useful fragments for screening purposes are those of intermediate, and approximately equal, frequencies of occurrence in the database that is to be searched; and the algorithm of Cringean et al. thus provides a simple, and computationally efficient, way of identifying an equifrequently occurring screenset.

Acknowledgements

We thank the Biotechnology and Biological Sciences Research Council Bioinformatics Initiative for funding, the Science and Engineering Research Council for the award of a Research Studentship to I.J.B., and the referees for comments on an earlier draft of this paper. The Krebs Institute is a designated Biomolecular Sciences Centre of the Biotechnology and Biological Sciences Research Council.

References

[1] T. Feizi and D. Bundle, *Curr. Opin. Struct. Biol.*, 6 (1996) 659–662.
[2] US Department of Energy, Office of Energy Research, *Summary Report of a Workshop on a Carbohydrate Structure Database*, Washington, DC, 1987.
[3] S. Doubet, K. Bock, D. Smith, A. Darvill, and P. Albersheim, *Trends. Biochem. Sci.*, 14 (1997) 475–477.
[4] K. Bock, *Towards a Carbohydrate Chemistry*, Report EUR12757, Commission of the European Communities, Luxembourg, 1990.
[5] R.C. Read and D.G. Corneil, *J. Graph Theor.*, 1 (1977) 339–363.
[6] G. Gati, *J. Graph Theor.*, 3 (1979) 95–109.
[7] R.E. Tarjan, *ACS Symp. Ser.*, 46 (1977) 1–20.
[8] J.J. McGregor, *Software — Pract. Exper.*, 12 (1982) 23–34.
[9] J.M. Barnard, *J. Chem. Inf. Comput. Sci.*, 33 (1993) 532–538.
[10] J.M. Barnard, *Structure representation and searching*, in J.E. Ash, W.A. Warr, and P. Willett (Eds), *Chemical Structure Systems*, Ellis Horwood, Chichester, 1991, pp 9–56.
[11] P. Willett, *Three-Dimensional Chemical Structure Handling*, Research Studies Press, Taunton, 1991.
[12] E.M. Mitchell, P.J. Artymiuk, D.W. Rice, and P. Willett, *J. Mol. Biol.*, 212 (1990) 151–166.
[13] J.R. Ullmann, *J. Assoc. Comput. Mach.*, 23 (1976) 31–42.
[14] M.R. Garey and D.S. Johnson, *A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
[15] J.K. Cringean, C.A. Pepperrell, A.R. Poirrette, and P. Willett, *Tetrahedron Comput. Methodol.*, 3 (1990) 37–46.
[16] *Manual and Tutorial for CarbBank — the Complex Carbohydrate Structure Database (CCSD) Program*, Version 2.9, pp II-12–20.
[17] Instructions to authors, *Carbohydr. Res.*, 297 (1997) III–IX.
[18] IUPAC–IUB Joint Commission on Biochemical Nomenclature, Nomenclature of Carbohydrates (Recommendations 1996), *Pure Appl. Chem.*, 68 (1996) 1919–2008; *Carbohydr. Res.*, 297 (1997) 1–92.
[19] R. Wilson, *Introduction to Graph Theory*, Oliver and Boyd, Edinburgh, 1972.
[20] N. Trinajstic (Ed.), *Chemical Graph Theory*, CRC Press, Boca Raton, 1983.
[21] A. Feldman and L. Hodes, *J. Chem. Inf. Comput. Sci.*, 15 (1975) 147–152.