Substructure Searching on Very Large Files by Using Multiple Storage Techniques

Alexander Bartmann, Helmut Maier, and Dirk Walkowiak*

Softron GmbH, Rudolf-Diesel Strasse 1, D8032 Gräfelfing, F.R.G.

Bernard Roth

Chemplex GmbH, Varrentrappstrasse 40-42, 6000 Frankfurt 90, F.R.G.

Martin G. Hicks

Beilstein Institute, Varrentrappstrasse 40-42, 6000 Frankfurt 90, F.R.G.

Received August 27, 1992

Traditional substructure search systems use a two stage algorithm consisting of a preliminary screening which operates on (inverted) index files to determine a set of candidates to be processed by the atom-by-atom search (ABAS).[1-6] The screening stage is usually fast, the performance of the system being governed by the screening efficiency. If a large number of candidates is left after the screening, the result is often a very large increase in retrieval time. The ABAS becomes the time dependent stage due to the excessively large number of disk seeks required to get the randomly distributed structure records into memory. The new search algorithm described in this paper is based on a special preprocessed structure file. It contains multiples of each molecule's connection table organized in clusters forming contiguous portions of the search file. Each cluster can be characterized by a substructure contained in all its molecules. A molecule may be a member of several different clusters, or it may appear repeatedly in the same cluster. File generation and update is fast and simple, and the mass storage requirements are only about 1 kbyte/molecule. A substructure search is performed by finding the minimum set of clusters containing all candidates for a given query. The ABAS only has to scan *sequentially* through the relevant portions of the structure file. Furthermore, each single I/O-operation can read hundreds of structures into memory. Only the structures that are not already verified to be hits by the screening must be processed. The ABAS is CPU-bound. This architecture offers an extremely good performance on very large files for various computer platforms (e.g., IBM-PC, IBM-Mainframe, VAX) and even on slow storage devices like CD-ROMs.

INTRODUCTION

One-Stage Method. In its simplest form, a substructure search system consists solely of an atom-by-atom search. For a given query, it must check the subgraph isomorphism of each full-structure in the file. Mathematically, the ABAS has to solve an NP-complete problem, which, therefore, should be kept small in size. In the past sophisticated back-tracking algorithms have been developed which have performed extremely well in practice. Despite its desirable properties (simple, requires no file loading, no random access while searching and only a minimum amount of disc space), this method is only suited for small sized files because the search time increases linearly with the file size.

Two-Stage-Method. For large sized files, the traditional approach has been to introduce a preliminary screening step.[1-6] All structures which due to their structural properties definitely cannot be hits are discarded in this first step, thus reducing the number of ABAS calls. Typical screening implementations use fixed screen libraries or apply a screen-generating algorithm during file loading. All methods work essentially in the same way. On file loading, all structures are analyzed in terms of the screens that they contain. The screens are coded and stored in inverted lists. Thus each screen is associated with a list of registry numbers (RN). A query is processed by finding all (or a significant number of) the screenes that it contains and intersecting the corresponding RN-lists. This defines the set of candidate structures to be checked by the ABAS. The ABAS uses the RN as the access key to a structure record. Since the records are distributed randomly in the file, many disk seeks will be required (Figure 1).

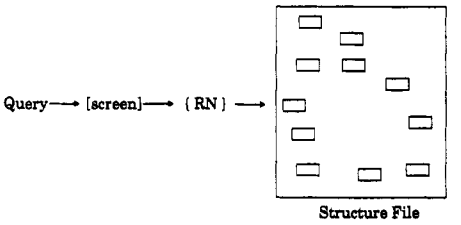

Figure 1. Screening of traditional two-stage methods leaves the ABAS with many random disk seeks.

If the number of candidates is "small", i.e., not more than a few thousand, these two-stage methods provide acceptable performance even on large files. However, as the number of candidates increases, the time spent for random disk seeks dominates. The system is I/O-bound.

With traditional two-stage methods certain queries can never or practically never be answered completely due either to time or system limitations. The exact solution of the problem is, however, often essential, e.g., if statistical structure–property studies are to be carried out on the whole of the file or if the result of the substructure search is to be used as input for further searches. Therefore, it is of great importance, that a structure search system be able to retrieve all answers with reasonable response times.

SCREENS AS KEYS TO STRUCTURES

System performance would be improved by reducing the number of disk seeks. In systems using inverted lists of screenes, the conversion step, [screen] → {RN}, is required because the RN is the only key to the original structure. Reorganization of the structure file, so that each key points

0095-2338/93/1633-0539$04.00/0

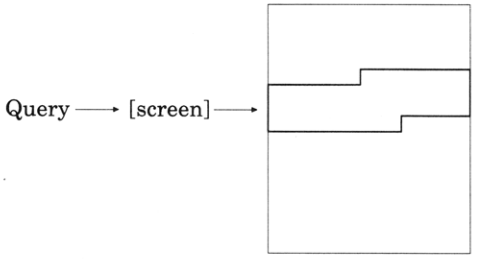

Figure 2. Search file provides access to the structures directly from the screens, reducing the number of disk seeks during the ABAS.

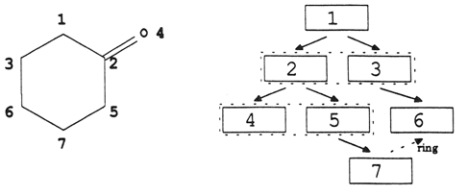

Figure 3. Unique numbering of cyclohexanone and the corresponding bundles (those consisting of several atoms are marked with dashed boxes).

directly to a set of candidate structures, has made the conversion unnecessary. The benefit is a marked decrease in the number of keys for the ABAS requiring less random disk seeks in favor of sequential reads. In contrast to the original structure file, this reorganized file will be referred to as the search file.

This kind of structure storage required a new data format. The access key to a data record must be closely related to the information in it. The well-known mass storage formats like SDF (Beilstein),[7,8] Registry III (CAS), or the various CTfile formats (MDL)[9] are not adequate for this purpose.

BITSTRINGS

Concentric Environments. The development of the new structure storage format was influenced by the concept of concentric environments (FREL) introduced by Dubols in the DARC system as an alternative to traditional fragment screens.[10] The FRELs, which consist of two sphere fragments built around a central atom or bond focus, are stored in a tree form for searching.

In this work the concept of concentric environments has been extended to yield a very compact linear representation of complete connection tables, the "bit strings".

Unique Numbering. The radius of perception is chosen to be large enough to include all atoms in a molecule. Starting with a fixed center, all atoms are uniquely numbered, based on atom and bond characteristics, such that (a) the center atom is indexed 1, (b) all atoms of the same sphere span a contiguous interval of indexes, (c) similarly, for all atoms with the same root atom and (d) atom X has a lower index than Y, if X has the lower indexed root atom (Figure 3).

Bundle. All atoms spreading from the same root atom are defined as a "bundle". The center atom is defined as bundle 0. Bundles play an important role in substructure searching on the search file (Figure 3).

Bit String. Ordered by the unique numbering, all atoms are written as a linear string containing all atom and bond attributes, the complete connectivity including rings and stereochemistry and tautomer information. Compressing the string with Huffman coding[11] techniques and adding a 3-byte registry number yields the final bit string. Nonstructural features like coordinates, factual data, chemical names, etc., are not included.

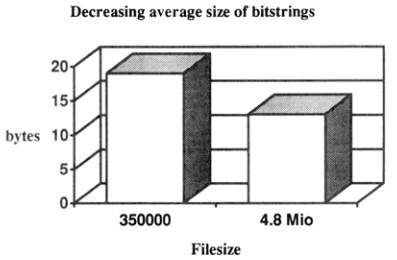

Figure 4. Decreasing average size of bit strings with growing file size.

Decoding. A bit string can be decoded directly into a connection table by scanning from left to right. Although a partial bit string uniquely defines a partial connection table, the opposite is not true because there is no unique numbering of atoms in substructures.

Exact Hash Code. A bit string may be used as an exact hash code for a structure. This has allowed the development of a very efficient full-structure search system. For this purpose, only one bit string per structure, e.g., the "smallest", needs to be stored in the search file.

SEARCH FILE

Bit strings are produced starting at each atom in turn for each structure. The average size of a molecule in the 4.8 million structure Beilstein Online File is ca. 20 atoms (not including the implicit hydrogen atoms), yielding 96 million bit strings. The bit strings are sorted and written to the search file storing the repetitive bytes of successive bit strings and identical bit strings with the same RN only once. For example, identical bit strings would be produced using either atom 1 or 5 in the molecule of Figure 3, as center atoms, similarly for atoms 3 and 7. The efficient use of the structure similarities means that the average size of a bit string in the search file decreases with growing file size.

The average size of a bit string including the 3-byte RN is only ca. 13 bytes in the Beilstein File. Thus, a block of 23 kbytes holds on average 1800 full-structure connection tables.

Clusters. The search file is inherently made up of clusters of similar structures. The degree of similarity is expressed in terms of the biggest common concentric environment or—equivalently—by the longest identical partial bit string. In other words, a partial bit string is the access key to a cluster of similar structures. A partial bit string can thus be described as being a special kind of substructure.

SUBSTRUCTURE SEARCHING

A query is processed in two steps, a preliminary screening and a final ABAS. For a given query, the screening determines the "best" set of (partial) bit strings that are to be used as direct access keys by the ABAS to the candidate sections in the search file.

Bundle Files. As mentioned above, the keys (i.e., partial bit strings) cannot be derived directly from the query due to the lack of a unique numbering in substructures. Instead, all bit strings contained in the search file must be decoded and reorganized in a single rooted tree. The bit strings are split on the bundle boundaries, which has proved to be a good compromise between atom and sphere boundaries. The bundles of different layers are stored in physically different files. For a good screening, the first 11 bundles proved to be sufficient.

Iterative Key Refinement. The screening is a process of iterative key refinement. First, using a heuristic algorithm

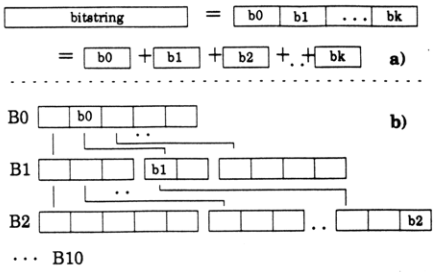

Figure 5. (a) Decomposition of a bit string into its bundles. (b) Bundle files: a single rooted tree used for iterative key refinement.

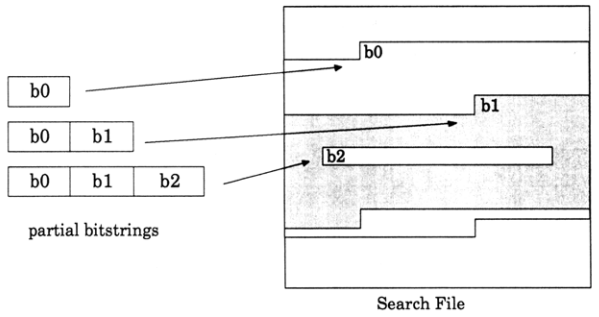

Figure 6. Iterative key refinement. Longer partial bit strings correspond to smaller sections of the search file.

the best starting atom in terms of the most selective concentric environment is determined in the query; then all O-bundles matching the starting atom are picked from the bundle file 0. (The bundle file 0 contains all different atoms of the structure file).

Each 0-bundle is the key to a section in the search file containing candidates for the ABAS. This section is very large if many bit strings start with this 0-bundle.

A 0-bundle is the root of a subtree in the bundle files. By branching from the 0-bundle, all 1-bundles, that match the corresponding atoms in the query, can be selected. Due to the way in which the bit strings are constructed, it is always known which atoms of the query correspond to the actual bundle. A matching 1-bundle is concatenated with the 0-bundle, thus yielding a more specific key to the search file. This procedure is iterated until the last bundle file is reached or until all atoms of the query have been mapped. Each iteration reduces the number of candidates for the ABAS.

Since the screening has the complete connection table information up to the current bundle level available, it can often be determined at this stage that the mapping has found an exact match for a query, thus making an ABAS match unnecessary. This gives an enormous increase in performance and often occurs in the cases where the query molecule is relatively small and unspecific.

IMPLEMENTATIONS

S4. The above described algorithm has been implemented as the S4 (Softron) Substructure Search System. S4 is in use by DIALOG Online Information Services, Palo Alto, CA, to search the Beilstein Online File (currently ca. 5 million compounds) and S4-CD-ROM is used on the Beilstein Current Facts in Chemistry CD-ROM[12,13] (where each CD-ROM holds ca. 300 000 structures).

FILE GENERATION AND UPDATE

The larger the file size, the more critical are the times required to load and update the structures, especially for systems which require extensive preprocessing of the structures during file loading.

In the system described above, the most time consuming steps are the creation of the search and bundle files. On an IBM ES9121/480, the average task time to load one structure is only ca. 150 ms.

Mass Update. Adding a set of structures to file requires the generation of search and bundle files for the update structures followed by a merge of the two sets of files. Since all files contain sorted strings, the merge time can be kept to a minimum. The merge time as a function of the file size requires ca. 3.5 ms/structure. During the merge, it is possible to replace, add, and delete structure records.

Individual Update. The mass update procedure is not feasible if individual structures are required to be added and searched immediately. To overcome this problem the 1-stage method, using only an ABAS, described at the beginning of this paper can be added to the system. When the sequential file becomes too large, taking too much time to search, a mass update must be applied.

SUMMARY

A data structure for the storage of chemical compounds has been developed which gives extremely good response times for substructure searches on files with several million records. The number of time consuming disk seeks has been reduced in favor of fast sequential read operations. The retrieval time, normalized for one hit, is a decreasing function of the file size. The system is CPU-bound. It is ideally suited for substructure searches on very large files and on CD-ROMs.

ACKNOWLEDGMENT

We would like to thank the Beilstein Help Desk and especially Frau Gabi Ilchmann for their enthusiastic and thorough testing of the S4 Substructure Search System.

REFERENCES AND NOTES

(1) Hicks, M. G.; Jochum, C. Substructure Search Systems. 1. Performance Comparison of the MACCS, DARC, HTSS, CAS Registry MVSSS, and S4 Substructure Search Systems. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 191–199.

(2) Willett, P. A Review of Chemical Structure Retrieval Systems. *J. Chemom.* **1987**, *1*, 139–155.

(3) Willett, P. *Similarity and Clustering in Chemical Information Systems.* Research Studies Press Ltd.: Letchworth, Hertfordshire, England, 1987; pp 10–18.

(4) Ash, J. E.; Chubb, P. A.; Ward, S. E.; Welford, S. M.; Willett, P. Chemical Structure Search Systems and Services. In *Communication, Storage and Retrieval of Chemical Information*; Ellis Horwood: Chichester, U.K., 1985.

(5) Stobaugh, R. E. Chemical Structure Searching. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 271–275.

(6) Barnard, J. M. Problems of Substructure Searching and Their Solution. In *Chemical Structures*; Warr, W. A., Ed.; Springer-Verlag: Berlin, 1988; pp 113–126.

(7) Domokos, L. The Beilstein Structure Registry System. 1. General Design. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 320–326.

(8) Hicks, M. G. Reactions in the Beilstein Information System: Nonaporic Organic Synthesis. *J. Chem. Inf. Comput. Sci.* **1990**, 352–359.

(9) Dalby, A.; Nourse, J. G.; Hounsell, W. D.; Gushurst, A. K. I.; Grier, D. L.; Leland, B. A.; Laufer, J. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244–255.

(10) Dubois, J.-E.; Panaye, A.; Attias, R. DARC System: Notions of Defined and Generic Substructures. Fillation and Coding of FREL Substructure (SS) Classes. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 74–82.

(11) Huffman, D. *Proc. Inst. Radio Eng.* **1952**, *40*, 1008–1101.

(12) Hicks, M. G. Similarity and the Beilstein Information System: Searching for Concepts with Current Facts. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 631–638.

(13) Hicks, M. G. Beilstein Current Facts in Chemistry: a Large Chemical Database on CD-ROM. *Anal. Chim. Acta* **1992**, *265* (2), 291–300.