Molecular Substructure Similarity Searching: Efficient Retrieval in Two-Dimensional Structure Databases

T. R. Hagadone

The Upjohn Company, Kalamazoo, Michigan 49001

Received April 10, 1992

Molecular substructure searching and full-structure similarity searching have become standard structure retrieval techniques in chemical database systems. While these search methods often provide useful results, they do not always meet the needs of the chemical database searcher. This paper describes the development of a substructure similarity (subsimilarity) search system that finds molecules containing a substructure that is similar to a query structure. It is argued that subsimilarity searching fulfills a retrieval need not met by existing search techniques. A major goal in the design of the system, the ability to execute a majority of subsimilarity searches interactively, was realized through a two-step search method that employs a unique subsimilarity screening step followed by an approximate maximal common subgraph (MCS) matching step. User acceptance of subsimilarity searching and its application in the research environment are discussed within the context of Upjohn's Cousin chemical database system.

INTRODUCTION

Development of chemical structure databases for use in research and development is a well-established activity in the pharmaceutical and chemical industries. Interactive search and retrieval of structures in such databases is accomplished through a variety of methods including full-structure searching, substructure searching, and more recently, full-structure similarity searching and 3D substructure searching. Each type of structure search produces useful results within its area of application; however, some searching needs are not met by the currently available search techniques. For example, consider what happens when a substructure search, either 2D or 3D, is performed and no hits occur or the hits that are found are not of interest. In this case, the searcher would often like to find any molecules in the database that contain a substructure that nearly matches the original substructure query. The usual solution to this problem is to repetitively remove portions of the original query and reexecute the search until either a satisfactory set of hits is found or the query becomes too small to be of interest. This technique, however, is a time consuming, hit or miss approach to the problem. A better solution would be to offer the user a search method that would automatically find any molecules that contain substructures that nearly match the original query. Full-structure similarity searching solves this problem if the goal is to find complete structures that closely resemble a query structure; however, existing similarity search systems cannot effectively find molecules that contain a substructure that approximates either a full-structure or substructure query because of limitations in the underlying fragment-based algorithms.

As an illustration of this searching need, consider the query and database structures shown in Figure 1. In most cases, a searcher would be interested in finding this database structure since it contains a substructure very similar to the query structure. However, a substructure search using this query would fail to match the database structure and the searcher would, in general, have no way to know that deleting a carbon on the query would result in a hit. A full-structure similarity search would fail to find the database structure as well since the overall similarity of the two molecules is not high.

The type of search required to address this problem, and the topic of this paper, can be termed a substructure similarity

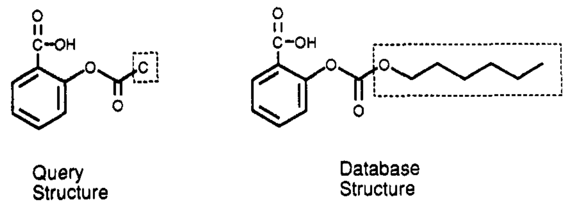


Figure 1. Sample subsimilarity query and a matching database structure.

(subsimilarity) search. Subsimilarity searching combines the features of substructure and similarity searching to allow successful retrieval in cases such as the one shown in Figure 1. It is a technique that can be applied to a wide range of chemical research problems including the selection of compounds for biological testing and the identification of reaction starting materials.

In this paper an approach to subsimilarity searching is proposed that uses the number of bonds in the maximal common substructure (MCS) of a query and database molecule as the similarity measure. The MCS of a pair of structures is the largest substructure that is present in both structures and is a measure that corresponds favorably to an "intuitive" notion of chemical similarity. We define the MCS similarity coefficient as the ratio of the number of bonds in the MCS to the number of bonds in the query structure, a value that ranges from zero to one.

A subsimilarity search system will be most useful if it can perform searches quickly, allowing a user to obtain search results for one or more queries in a single terminal session. Accordingly, a primary goal in our approach is to provide a subsimilarity searching capability that can interactively search a database of at least 100 000 structures. The largest obstacle standing in the way of this goal is the heavy computational burden imposed by the MCS calculation. Our attempt to overcome this computational complexity consists of a two-step search process in which an initial subsimilarity screening step is followed by an approximate MCS matching step.

The paper describes the development and use of a subsimilarity search system for searching 2D structure databases in the context of Upjohn's Cousin[1-3] drug discovery database system. We begin with a review of relevant existing techniques followed by descriptions of our approach to subsimilarity screening and the MCS algorithm. The implementation and performance of the system are then presented, and finally the user's view of and experience with the system are described.

BACKGROUND

The need to find features common to a set of two or more molecules has appeared frequently in chemical research problems. An often-used approach to the problem involves a transformation of 2D or 3D structures into labeled graph form with nodes representing atoms and edges representing bond types or distances between atoms.[4] Maximal common subgraph algorithms are then used to find the maximal common substructures of the associated molecules. This approach has been applied to 2D structure representations in the areas of mass spectral search,[5] quantitative structure–activity relationships (QSAR), reaction center perception,[6] and synthetic starting material selection.[8] In the 3D arena, the MCS approach has been used in QSAR[9,10] and in comparing databases of 3D structures for fit with known receptors and active 3D model compounds.[11]

Because of the computational demands of the MCS algorithms that underlie the above applications, most researchers have sought ways to optimize the performance of their programs by developing techniques to prune the MCS search tree, thereby reducing the size of the search space that must be explored. Cone et al.[5] use Levi's[12] "compatibility table" as a way to eliminate unproductive atom pairings. McGregor and Willet[6,7] prune by attempting to find large common substructures early in a match and then calculate an upper bound on the size of later common substructures, pruning branches that lack the potential to reach the largest size common substructure already found. Wipke and Rogers[8] organize a database of structures in a storage tree that effectively parallelizes the search process and reduces the search effort by as much as 95% in some cases. Brint and Willett[13,14] apply Bron and Kerbosch's[15] efficient clique detection algorithm to simplified and full "correspondence graphs" derived from pairs of 3D structures. Kuntz et al.[11] reduce the complexity of the search by taking a sample of the possible atom pairings between a receptor model and 3D database structures in an approach that approximates the results of a complete MCS analysis. Barakat and Dean[10] use the simulated annealing technique to sample the set of possible MCSs in pairs of 3D structures.

While all of the above techniques provide significant performance improvements over the approach of simply enumerating all possible MCSs, it was not clear that direct application of any of these methods would provide the interactive 2D subsimilarity search response we desired. Therefore, we developed and evaluated the alternative two-step search technique described below.

SUBSIMILARITY SCREENING

Screening systems are widely used in substructure search software as a method for improving performance. For each molecule included in the database, a list of selected structural fragments present in the molecule is prepared and stored in the database along with the connection table of the molecule. At search time, a list of the fragments present in the query is prepared in a similar fashion and compared to the fragment list for each database compound. Database molecules that lack any of the fragments in the query's fragment list can be discarded from further consideration. Molecules that pass the screen are sent on to the relatively much slower atom by

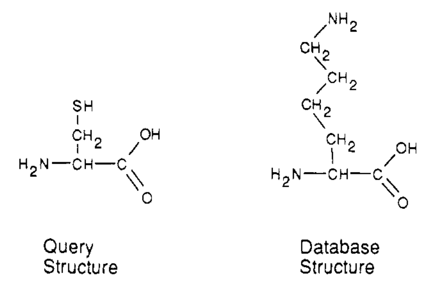


Figure 2. Query structure with 13 simple pairs and database structure with 23 simple pairs.

atom matching phase of the search. The screening step typically eliminates 99% or more of the database, thereby dramatically improving search speed.

It's not feasible to use the substructure screening technique directly in subsimilarity searching since the possibility exists that the MCS of a query and database structure will be large even though one or more query screening fragments are not present in the database molecule's fragment list. However, it is possible to use screening in a more limited way to establish an upper bound on the MCS coefficient value and to use this upper bound to eliminate structures from further consideration. This possibility arises from the fact that the searcher is not normally interested in knowing the MCS coefficient for every database molecule, but rather only wants to find the molecules with the largest MCS coefficient values. Typically, the searcher is only interested in molecules that bear a reasonably close similarity to the query structure, say 0.75 or above, or alternatively may only want to see the best 10 or 100 matches. If a minimum acceptable MCS coefficient value and/or a maximum number of hits can be established, then a subsimilarity screening technique can be used as described below.

Imagine a set of screens that contains the collection of fragments that consist of all bonded pairs of atoms in a molecule and their connecting bond. With this screen set the number of fragments for a structure will be the same as the number of bonds in the structure. For example, in Figure 2, the query structure contains 13 simple pair fragments, including bonds to hydrogen, and the database structure contains 23 simple pairs. At search time, if we check a database structure for the presence of each simple bonded pair present in the query structure, and find some of them missing, we have gained some information that allows us to reduce the upper bound on the size of the MCS for the pair of structures. Examining Figure 2 we find that the query contains a C–S and S–H pair that are not present in the database structure. Therefore, we know that an upper bound on the MCS for these two structures is $13 - 2 = 11$ with an upper bound on the similarity coefficient of $11/13 = 0.85$. If the user has specified a minimum acceptable similarity coefficient of 0.90, then we can immediately discard this database molecule.

In our system, the user is allowed to specify both a minimum acceptable similarity coefficient (Minsim) and a maximum acceptable number of hits (Maxhits). At search time, an upper bound on the similarity coefficient is calculated and saved for each database structure using the similarity screening technique described above. Molecules are then passed to the detailed MCS algorithm described below beginning with the molecule with the largest MCS upper bound and progressing to the molecule with the smallest upper bound. As the search proceeds, the best actual MCS values calculated by the detailed MCS algorithm, up to the Maxhits limit, are maintained in a hit list. Either of two possible conditions can occur during the detailed matching that will allow all remaining molecules

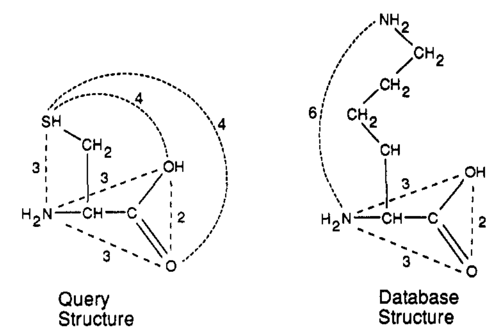
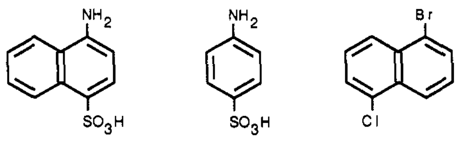


Figure 3. Query and database structures augmented with heteroatom bond distance pairs.



1 2 3

Figure 4. Structures 1 and 2 have a larger MCS than structures 1 and 3 when the chemical graphs are augmented with heteroatom bond distance pairs.

to be immediately eliminated. First, if the MCS upper bound for the remaining unprocessed molecules falls below the Minsim value we can terminate the search since we know that none of the remaining molecules can achieve a similarity greater than Minsim. Secondly, if the MCS upper bound for the remaining molecules falls below the MCSs of all of the Maxhits molecules in the hit list, we can similarly terminate the search since we know that none of the remaining molecules can have a MCS better than that of any of the molecules already in the hit list.

Using similarity screening in this way greatly speeds up the search by eliminating the need to perform the detailed MCS analysis on a large percentage of the database. However, we made a modification to our definition of similarity that provides improved screenout and at the same time results in a similarity measure that better corresponds to an "intuitive" notion of molecular similarity. In the modified definition we give relatively greater weight to relationships between heteroatoms than that given to relationships between carbons. This is accomplished by augmenting the chemical graphs of the query and database molecules with additional edges between heteroatoms. For all pairs of heteroatoms in a molecule, separated by a shortest path length of two to six bonds, an additional edge connecting the atoms and labeled with the length of the shortest path between them is added to the graph. Figure 3 shows the structures of Figure 2 augmented with these additional edges. The added edges are included in subsimilarity screening and the MCS calculation and contribute to the size of the MCS in the same fashion as normal molecular bonds. Because the added edges result in additional edge matches in common substructures containing multiple heteroatoms, their effect is to increase the similarity coefficient for common substructures that are rich in heteroatoms over the coefficient of common substructures with the same number of bonds that contain primarily carbon atoms. Since heteroatoms often play the key chemical roles in molecular recognition, binding, and reactivity, we feel this emphasis on their relationships is reasonable. Consider for example that if augmented graphs are used, structures 1 and 2 in Figure 4 have a larger MCS than structures 1 and 3, whereas with unaugmented graphs, structures 1 and 3 have a larger MCS than 1 and 2.[16]

The other effect of augmenting the chemical graphs is that a richer similarity screen set is created which emphasizes the heteroatom relationships and balances out the relatively large number of carbon–carbon pair screens. The enhanced screen set provides improved performance for queries that contain two or more heteroatoms and results in screenouts of 90% or better for most queries. This high screenout percentage makes it possible to perform the detailed MCS calculations on the remaining molecules in an interactive fashion. It should be noted that the augmented graphs are transparently built for the screen generation and MCS calculation processes and are not seen by the user.

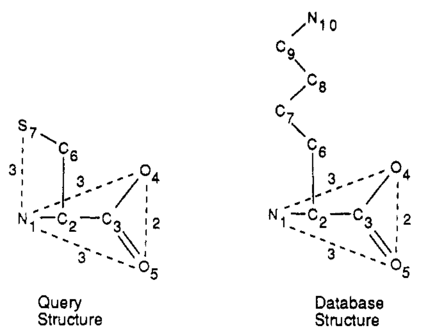
APPROXIMATE MCS ALGORITHM

As a result of the screening process, only a small percentage of the complete database is passed on for detailed MCS analysis; however, even for a few hundred or few thousand structures the MCS analyses put a considerable strain on computing resources. Although the MCS analysis step is similar to the atom by atom matching step in substructure searching, MCS algorithms are inherently more computationally complex than the subgraph isomorphism algorithms typically used in substructure searching. This can be seen intuitively by considering that a subgraph isomorphism algorithm simply checks for the presence of the query structure within the database structure whereas a MCS algorithm has to, in effect, check for the presence of all possible substructures of the query within the database structure.

MCS algorithms can be divided into two classes, those that always find the maximal common substructure and those that find a common substructure that is equal to or a close approximation to maximal. The first class is optimized through the application of heuristics that attempt to minimize the amount of the total search space that must be explored to find the MCS. The second class of algorithms, which are generally faster than the first class, attempt to provide an acceptable balance between the accuracy of the result and the time required to perform the search. Emphasis is placed on providing as fast a search as possible within an acceptable range of accuracy which is achieved by exploring a carefully selected sample of the total search space.

Our choice of an approximate MCS algorithm was determined by the goal of providing fast searches. We felt that the use of an acceptably accurate approximate algorithm was appropriate considering the already imprecise relationship between molecular similarity measures and chemical and biological reality. The algorithm we developed attempts to find large common substructures early in the MCS analysis in a fashion similar to that described by McGregor.[7] It uses the compatibility of the neighbors of potential atom pairings to direct the growth of common substructures. Approximation is introduced by limiting backtracking to the initial atom pair assignment.

The two-phase matching process that is used is best illustrated through an example. The first phase is shown in Figure 5 in which hydrogens have been removed and distance pairs have been limited to a range of two to three bonds for clarity. In this phase each query atom and its neighboring atoms and bonds are compared for match potential with each database atom and its neighbors. The matrix, M, that is produced as a result contains n rows and m columns where n is the number of query atoms and m is the number of database structure atoms. Each element of M contains the number of neighbors of the specified query atom that have neighbors on the corresponding database structure atom that match on atom



Query Atom	Database Atom									
	1	2	3	4	5	6	7	8	9	10
1	3	0	0	0	0	0	0	0	0	1
2	0	3	1	0	0	2	2	2	2	0
3	0	1	3	0	0	1	1	1	1	0
4	0	0	0	3	2	0	0	0	0	0
5	0	0	0	2	3	0	0	0	0	0
6	0	1	1	0	0	1	1	1	1	0
7	0	0	0	0	0	0	0	0	0	0

Figure 5. Matrix of match potentials for query database atom pairs.

Step	Pair matched	Subgraph size	Next pair candidates
1	(1,1)	0	(2,2)
2	(2,2)	1	(3,3), (3,6), (6,3), (6,6)
3	(3,3)	2	(4,4), (5,5), (4,5), (5,4), (6,6)
4	(4,4)	4	(5,5), (6,6)
5	(5,5)	7	(6,6)
6	(6,6)	8	None

Figure 6. Common subgraph growth path for structures in Figure 5. Parenthesized numbers represent a query atom, database atom pair.

and bond type. For example, Q_1, the first query atom, has four neighbors, including the distance edges, three of which have matching neighbors on D_1, the first database structure atom; therefore, $M_{1,1}$ is given a value of 3. Elements of **M** that are assigned relatively high values represent query-database atom pairs that have the potential of contributing significantly to the size of a common substructure. These high-value pairings are given preference over lower value pairings when growing common subgraphs.

In the second phase of the analysis, a number of growth cycles are executed in which each cycle is started by the selection of a high-value element from **M** as an initial atom pairing. Pairs of neighbors of this initial pair, with compatible atom and bond types, are used to create a list of potential additional pairings. A selection is made from this candidate list and is used to generate an updated candidate list. Growth continues in this fashion by selecting additional pairings and generating updated candidate lists until the set of candidate pairs is exhausted.

A sample growth cycle is shown in Figure 6 in which the common subgraph is started with the initial pairing Q_1, D_1. This produces a candidate set containing only the pair Q_2, D_2 since we only allow growth along real bonds and not along the distance pair edges. Selecting Q_2, D_2 generated four new candidate pairs. Q_3, D_3 is matched next since it is the candidate pair with the highest value in M. This eliminates all previous candidate pairs except for Q_6, D_6, which is still a possible pairing, and adds four new candidate pairs for a total of five. Two of these pairings have a M value of 3, two have a value of 2, with the remaining pair having a value of 1. At this point, as often occurs, multiple maximum value candidate pairs exist. When this happens, the algorithm randomly selects one of the maximum value pair candidates, in this case Q_4, D_4. In the example, the two high-value candidates are compatible with each other, and selecting them in either order results in the growth of the same common subgraph; however, in the general case, different selections can result in the growth of different common subgraphs. In the last two steps of Figure 6 growth of the common fragment completes by matching Q_5, D_5 and then Q_6, D_6.

In this example, only a single common substructure fragment was grown. In general however, when the set of candidate atom pairs becomes empty, the growth cycle continues by selecting an available high-value pair from **M** which is used to grow the next common fragment. Available pairs are those for which neither the query nor database structure atoms are part of the already grown fragment(s). Random selection is used here as well if multiple candidates have the maximum value. Growth of disconnected common fragments continues either until the set of available initial pairs in **M** is exhausted or until the number of fragments grown reaches the limit of a user defined parameter named Maxfrags. This parameter, which has a default value of 2, allows the user to control the amount of fragmentation allowed when growing common substructures. It should be noted that although in the example the query and database structure atoms were similarly numbered, the algorithm is insensitive to atom numbering, and any alternative atom numbering would have resulted in essentially the same growth path.

The actual MCS was found on the first try in this example; however, this will not be true in general. Once the algorithm has found a common substructure, it makes a record of the match and begins another growth cycle by clearing the current pair assignments and selecting a new high-value initial pairing from the **M** matrix. Growth cycles are performed for each of the highest value elements in each row of **M**. The algorithm keeps a record of the similarity coefficient of the largest common substructure that has been found and the number of occurrences of this size substructure. If the same maximum similarity coefficient continues to be found repeatedly, the remaining growth cycles will be skipped and processing will go on to the next database molecule. An important characteristic of this method is that no backtracking is performed to consider alternative atom pairings other than in the selection of the initial pair assignments. This policy of limiting backtracking to the initial pairing significantly reduces the time required for the search as discussed in the performance section.

While the example of Figure 6 illustrates the main points of the approximate MCS algorithm, some additional optimization opportunities are exploited during the MCS analysis. At the beginning of a growth cycle a variable named P_{coef}, which represents an upper bound on the similarity coefficient for this growth cycle, is initialized to a value of 1.0. As a query atom is matched to a database structure atom, each neighbor of the query atom, including the heteroatom distance edge neighbors, is checked against the neighbors of the matching database structure atom. For every query atom neighbor that does not have a corresponding database structure atom neighbor, P_{coef} is reduced by $1/n$, where n is the number of edges in the augmented query graph. A growth cycle can

Subsimilarity screening time	9.5 secs
Screenout	93%
MCS analyses	3780
MCS CPU time	60 secs
MCS CPU time per molecule	16 ms.
MCS faults	.7%

Figure 7. Average search statistics for 495 subsimilarity searches over a 57 000 molecule database.

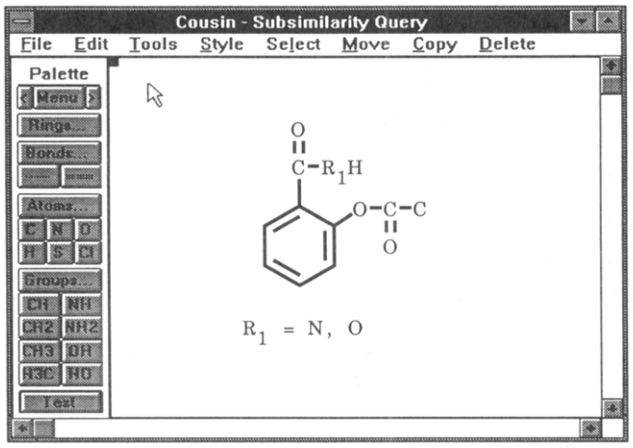


Figure 8. Cousin molecule editor with sample subsimilarity query.

be terminated if at any point P_{coef} falls below the coefficient for the largest common substructure found so far for this database molecule, below the Minsim value specified by the user, or below the lowest coefficient value in the hit list.

IMPLEMENTATION AND PERFORMANCE

Implementation, testing, and refinement of the subsimilarity screening and MCS algorithms described above were performed on an IBM 3090-400J CMS host machine. Testing was done on a database of 57 000 structures with an average molecule size of 22 atoms and 22 bonds. The subsimilarity screening file was organized in a semiinverted format with multiple superimposed bits assigned to each screen fragment.[17] Patterns of screen bits were assigned to bonded pair and distance pair fragments as well as to multiple occurrences of these. Screen bits were not assigned to pairs containing hydrogens since storing these pairs would require a significant amount of storage but would do little to improve screenout. The subsimilarity screening algorithm assumed that all hydrogen-containing pairs in the query were also present in each database structure. Connectivity information used in the detailed MCS analysis portion of the search was held in a second file.

Search tests were designed to provide insight into several aspects of the behavior of the system. We wanted to know how effective the screening algorithm was on average and in the worst cases. Similarly, we wanted to know how fast the approximate MCS algorithm was on average and in the worst cases, and since the MCS algorithm is approximate we needed to know how often and how badly it failed. During the course of the testing a number of improvements were made to the algorithms, and the test results presented here represent the current performance of the system.

In one large test, 495 searches were executed over the database using a uniformly selected sample of structures from the database as queries. The best 50 matches, with a similarity coefficient of at least 0.67 for the largest single-fragment common subgraph, were retrieved in each search. These 50 molecules were then analyzed by an exact MCS algorithm, and the actual MCSs found were compared to those found by the approximate MCS algorithm to detect any failures in the approximate algorithm. The results as shown in Figure 7 were encouraging. Searches completed in a little over 1 min on average, although several queries took up to 7 min to complete. In only 168 out of 495 × 50 = 24 750 cases did the approximate algorithm fail to find the MCS. Out of the 168 failures, the approximate MCS was one bond too small in 160 cases and two bonds too small in the remaining eight cases.

We tested the approximate MCS algorithm on several difficult searches in which both the query and database molecules consisted of large heavily branched carbon structures. In these cases the frequency of failure increased modestly, but the size of the approximate MCSs remained within two bonds of the actual MCSs. We executed searches that systematically tested a range of query sizes and search parameter values. The best search response was obtained for large queries with several heteroatoms. Large Minsim, small Maxhits, and small Maxfrags values helped to improve the search speed. Conversely, small queries with few heteroatoms, small Minsim values, large Maxhits values, and large Maxfrags values all contributed to producing relatively slow searches.

One shortcoming discovered during testing involved the MCS upper bound ranking of structures by the screening step. In particular, large database structures such as peptides are often given high MCS upper bounds because they contain most or all of the query screening pairs. However, the pairs are usually scattered throughout the molecule and the actual MCS that is found is small. A similar problem occurs for

```
Initial similarity scan completed for 98034 molecules.
Performing atom by atom similarity matches ("best" candidates first).
50 most similar molecules will be retrieved, minimum required sim.= 0.67
 (Press the Enter key to interrupt the search)

Percent   Attempted  <---- Distribution of Matches by Similarity Value --->
Complete   Matches   1.00  .97  .93  .90  .87  .83  .80  .77  .74  .70  .67
--------  ---------      +----+----+----+----+----+----+----+----+----+----+
  60.1       152          9   15    9    5    0    4    4    0    0    0
  89.0       311         10   19    9    6    0    4    2    0    0    0
  97.0       459         12   24   11    3    0    0    0    0    0    0
  98.9       606         15   24   11    0    0    0    0    0    0    0
 100.0      1676         15   24   11    0    0    0    0    0    0    0
```

Figure 9. The search status table is updated every 10 s during a search to inform the user of search progress.

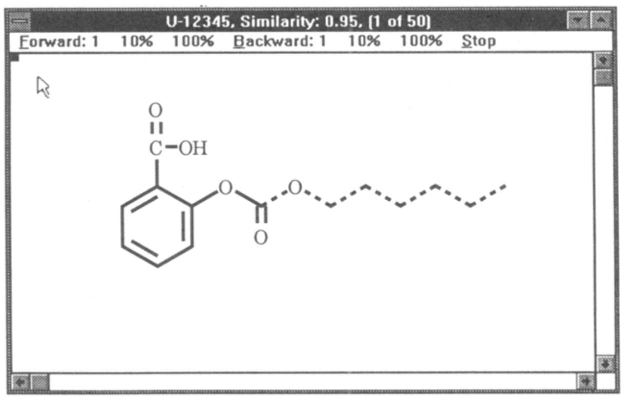

U-12345, Similarity: 0.95, (1 of 50)
Forward: 1 10% 100% Backward: 1 10% 100% Stop
O
C–OH
O
O
O

Figure 10. Display of a subsimilarity match with matching bonds shown solid and nonmatching bonds shown dashed.

queries that contain less than two heteroatoms since the carbon–carbon bonded pairs in the query are often not sufficient to allow for an accurate ordering of structures in the screening step. This limitation becomes important in the case of slow searches that are canceled by the user before completion. The user would like to assume that the best matches have already been found when the search is canceled; however, if the ranking of structures by the screening step is improper, this assumption may be invalid. An improvement to the ordering of structures by the screening step could be achieved by adding a set of larger fragments to the screen set to be used for ordering purposes but not to be included in the calculation of the MCS upper bound.

USER EXPERIENCE

Subsimilarity searching has been fully integrated into Upjohn's Cousin database system. Cousin currently logs approximately 5000 sessions per month by a user community of 1000 and provides a good environment for evaluating the usefulness of new database tools such as subsimilarity searching. The Cousin structure searching menu now includes subsimilarity search as a choice along with the more familiar substructure, full-structure, and full-structure similarity searches. When the subsimilarity search option is chosen, the user is presented with a submenu that allows a previously drawn query or a database molecule to be selected as a query template. The default values of search parameters can be modified at this point as well. Search parameters under user control include the minimum and maximum acceptable similarity coefficients (Minsim, Maxsim), the maximum number of hits (Maxhits), and the maximum number of fragments allowed in the MCS (Maxfrags). It is occasionally useful to specify a maximum acceptable coefficient of 0.999 to eliminate molecules that have previously been found in a substructure search. Query definition is performed using the Cousin molecule editor, shown in Figure 8, which is a local workstation component of the Cousin graphical user interface. Subsimilarity query definition is analogous to substructure query definition with convenient graphical tools available for defining the basic query structure, variable substituents, generic atom types, and variable bond types. One restriction for subsimilarity queries, required to keep the number of bonds in the query well defined, is that alternate R-group fragments

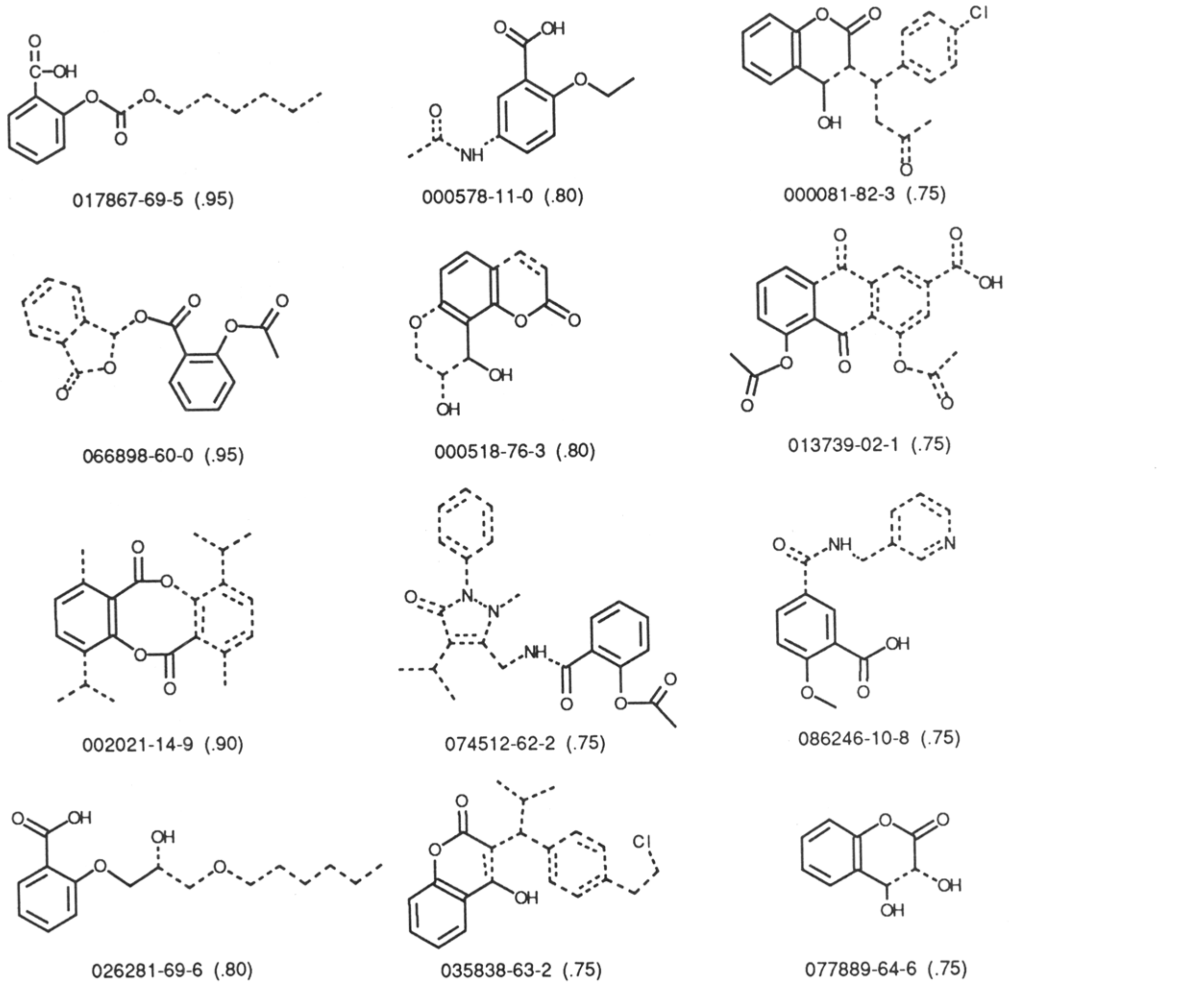


Figure 11. Sample search results with CAS registry numbers and subsimilarity coefficients. Bonds that are not part of the MCS are shown dashed.

are limited in size to a single atom; however, in practice this restriction has not posed a problem for users.

When query definition is complete, the search begins with the subsimilarity screening step which normally takes approximately 10 s. Following screening the detailed MCS analyses begin. Since this can be a potentially time-consuming process, a search status table, as shown in Figure 9, is displayed to keep the user informed of search progress. The table includes the percentage of the database processed so far, the number of detailed MCS analyses performed, and a count of the number of matches found, partitioned into 10 equal-sized intervals from the minimum to the maximum acceptable MCS coefficient. The display is updated every 10 s, and the user can cancel the search at any time and display the hits found up to that point.

Most searches fall into one of three categories. If the search is relatively fast, requiring less than 2 min, the user simply waits for the search to complete and then displays the results. In the case of slow searches the user has two options. During the early part of the search, the search status display normally changes rapidly as the first hits, if any, are found. After a few minutes, the number and distribution of hits will usually stabilize. At this point the user can cancel the search and display the hits that have been found so far; otherwise, if the full search is desired, the user will go on to another task while the search completes. The system's goal of performing the detailed MCS analysis on the molecules with the largest MCS upper bounds first usually results in the best hits being found early in the search, with the limitations previously noted. This is desirable for slow searches or for impatient users since the best hits will normally be retrieved even if the search is canceled before it finishes.

Following completion or cancelation of the search, the results are optionally displayed at the terminal with the best hits shown first. The hit number, registry number, similarity coefficient, and structure of each hit are displayed with the MCS atoms and bonds highlighted for identification as shown in Figure 10. It's particularly important to highlight the MCS region in the displayed structure since it's often difficult to visually locate the MCS otherwise. Forward and backward navigation through the hit list and display and/or printout of any additional database data asociated with the hits is accomplished through the on-screen command menu. Figure 11 shows a portion of the printed results for a subsimilarity search performed using the query in Figure 8.

We have found the usage pattern of subsimilarity searching to be typical of that seen in a newly introduced database tool. Approximately 40 searches per month were performed during the first few months following introduction. This compares to current rates of approximately 1000 substructure searches, 850 name searches, 90 full-structure similarity searches, and 20 full-structure dissimilarity searches per month. We find this degree of initial use of subsimilarity searching encouraging and expect the usage rate to increase as the scientific community becomes more aware of subsimilarity searching's role relative to the other types of structure search. We are also encouraged by the fact that subsimilarity searching has been used in a number of biological screening studies to select compounds containing substructures similar to known active molecules. Subsequent testing of these compounds has resulted in the identification of additional actives that would have been difficult to find using other structure search techniques.

CONCLUSIONS

We feel that subsimilarity searching as described here represents a useful new tool in the arsenal of chemical structure database retrieval methods. The molecules found in a subsimilarity search meet important retrieval needs that occur regularly in chemical research and that are not met by current substructure search and full-structure similarity search systems. The role of subsimilarity searching can be viewed as complementary to existing search techniques; however, for some users subsimilarity searching is seen as a tool that largely replaces substructure and full-structure similarity searching.

The widespread adoption of substructure searching as a retrieval tool occurred when software and hardware technology progressed to the point where interactive searching could be made available to a large user community through a convenient user interface. We hope this paper has demonstrated that subsimilarity searching represents another useful database retrieval tool that should also be made available to a broad audience.

REFERENCES AND NOTES

(1) Hagadone, T. R.; Lajiness, M. S. Capturing Chemical Information in an Extended Relational Database System. *Tetrahedron Comput. Methodol.* **1988**, *1*, 219–230.
(2) Hagadone, T. R.; Howe, W. J. Molecular Substructure Searching: Minicomputer-Based Query Execution. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 182–186.
(3) Howe, W. J.; Hagadone, T. R. Molecular Substructure Searching: Computer Graphics and Query Entry Methodology. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 8–15.
(4) Varkony, T. H.; Yossi, S.; Smith, D. H. Computer-Assisted Examination of Chemical Compounds for Structural Similarities. *J. Chem. Inf. Comput. Sci.* **1979**, *19*, 104–111.
(5) Cone, M. M.; Venkataraghavan, R.; McLafferty, F. W. Molecular Structure Comparison Program for the Identification of Maximal Common Substructures. *J. Am. Chem. Soc.* **1977**, *99*, 7668–7671.
(6) McGregor, J. J.; Willett, P. Use of a Maximal Common Subgraph Algorithm in the Automatic Identification of the Ostensible Bond Changes Occurring in Chemical Reactions. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 137–140.
(7) McGregor, J. J. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. *Software Pract. Exper.* **1982**, *12*, 23–34.
(8) Wipke, W. T.; Rogers, D. Tree-Structured Maximal Common Subgraph Searching. An Example of Parallel Computation with a Single Sequential Processor. *Tetrahedron Comput. Methodol.* **1989**, *2*, 177–202.
(9) Crandell, C. W.; Smith, D. H. Computer-Assisted Examination of Compounds for Common Three-Dimensional Substructures. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 186–197.
(10) Barakat, M. T.; Dean, P. M. Molecular Structure Matching by Simulated Annealing III. The Incorporation of Null Correspondences into the Matching Problem. *J. Comput.-Aided Mol. Design* **1991**, *5*, 107–117.
(11) Kuntz, I. D.; Blaney, J. M.; Oatley, S. J.; Langridge, R.; Ferrin, T. E. A Geometric Approach to Macromolecule-Ligand Interactions. *J. Mol. Biol.* **1982**, *161*, 269–288.
(12) Levi, G. A Note on the Derivation of Maximal Common Subgraphs of Two Directed or Undirected Graphs. *Calcolo* **1972**, *9*, 341–352.
(13) Brint, A. T.; Willett, P. Algorithms for the Identification of Three-Dimensional Maximal Common Substructures. *J. Chem. Inf. Comput. Sci.* **1987**, *27*, 152–158.
(14) Brint, A. T.; Willett, P. Upperbound Procedures for the Identification of Similar Three-Dimensional Chemical Structures. *J. Comput.-Aided Mol. Design* **1988**, *2*, 311–320.
(15) Bron, C.; Kerbosch, J. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* **1973**, *16*, 575–577.
(16) Ugi, I.; Wochner, M.; Fontain, E.; Bauer, J.; Gruber, B.; Karl, R. Chemical Similarity, Chemical Distance, and Computer-Assisted Formalized Reasoning by Analogy. In *Concepts and Applications of Molecular Similarity*; Johnson, M. A., Maggiora, G. M., Eds.; Wiley: New York, 1990; Chapter 9, p 249.
(17) Mooers, C. N. Zatocoding Applied to Mechanical Organization of Knowledge. *Am. Doc.* **1951**, *2*, 20–32.