

Substructure Search with Tree-Structured Data

Keiichiro Ozawa,* Toshimasa Yasuda, and Shinsaku Fujita†

Ashigara Research Laboratories, Fuji Photo Film Co., Ltd., Minamiashigara, Kanagawa, 250-01 Japan

Received April 30, 1996®

A new screening process based on tree-structured data has been constructed for an in-house chemical substance database. The data structure and algorithm have improved the selectivity of screening and reduced search time. Thus, a prescreening step selecting an appropriate key is introduced to accelerate the total screening process. Its effectiveness has been proved by the comparison with the search without such prescreening. The present tree-structured system has been compared with a fragment-oriented screening system along with the retrieval of 87 query structures. The result shows excellent performance of the present screening system; it is because the tree-structured data include structural information of all nodes of the structures.

INTRODUCTION

Substructure search is now drawing more and more attention in the chemical information field as chemical substance databases become popular among researchers. It requires the effective manipulation of chemical structure data, and the methodology is one of the most significant subjects which is still stimulating information chemists.

The first generation of substructure searching systems¹ was based on a back-tracking algorithm and later adopted partitioning procedures and relaxation processes.² A fragment-oriented screening process was finally introduced in order to make such systems much less time-consuming. Thus most of the systems both for in-house purpose and for on-line use employ a two-step procedure which consists of a screening process and the following atom-by-atom matching process.

Recently, a new innovative strategy has been developed and applied to several major database systems. Their common technique is the implementation of tree-structured data on substructure searching; they do not use the traditional fragment-oriented screening method. For example DARC system,^{3,4} which has been used online to search the CAS Registry file (EURECAS) since 1981, uses that sort of data structure in a screening stage. DARC analyzes structures in terms of FRELs (Fragment Reduced to an Environment which is Limited). The FRELs describe two sphere fragments around a focus that is an atom with at least three non-hydrogen neighbors. The concept of concentric environments adopted by FRELs implies that structural data are stored in a tree form for screening. After the screening process with tree-formed data, a bit screen match is conducted mainly to characterize ring systems. A subsequent atom-by-atom matching eliminates noise data and acquires hits from the candidates.

More recently, major improvements have been achieved in a few systems. Thus, HTSS (the Hierarchical Tree Substructure Search system^{5,6}) is used for the Beilstein database on the Orbit online system. Another system is S4 (Softron's Substructure Search system⁷⁻⁹) which was developed for Beilstein and has been used on the Dialog online system. In particular, searching performance of the S4

implemented in a CD-ROM product is good enough to be used even by personal computers.¹⁰⁻¹³ Neither of these systems uses fragment-oriented screening. Besides they employ so elaborate trees that they do not need any atom-by-atom matching in many cases.⁸ These features make the whole retrieval process of the system very fast.^{15,16}

Hicks⁷ has reported that DARC searches faster than MACCS, which is one of the most popular in-house systems using fragment screening, as well as that S4 does much faster than DARC. It is also reported that the screening efficiency of DARC is generally higher than fragment-oriented screening processes like MACCS and CAS. All these advantages essentially stem from the tree structure of their data.

Despite of all these studies some questions remain unsolved about what is the best choice in all situations and about which combination of techniques gives further efficiency. The present paper deals with our system (SPHINCS Light) which is constructed with ORACLE as a main DBMS. It adopts a new screening process. The process is based on tree-structured data that contain information of all nodes in the structure; the single-key screening introduced here accelerates substructure search. Eighty-thousand structure data selected from the in-house database (SPHINCS¹⁷) are converted and stored in the present system. Using the structure data, the present searching process is compared with the traditional fragment screening process in SPHINCS in terms of search times and screening efficiency.

FILE AND DATA STRUCTURES OF A TREE

Atom Codes and Tree Files. File and data structure are among the most important concepts of database system. The search files consist of tree-structured data converted from all structures. Node attributes in the connection table of each structure are transformed into double-byte atom codes in the tree-structured data. The procedure of constructing tree files is illustrated in Figures 1 and 2 by using urea as an example.

1. A root node is selected randomly among all atoms of the structure except hydrogen. Structural information on each node (atom or functional group) is converted into a double-byte atom code in terms of elements, connectivity, the number of single bonds, the size of the smallest ring, the largest ring, charges, and so on. Although the code is not a full representation of a connection table, it includes ring information as a new matter. Thereby, the screening process can finish in one stroke without extra procedures

† Present address: Dept. of Chem. Materials Technol., Kyoto Inst. of Technol., Matsugasaki, Kyoto, 606 Japan.

® Abstract published in *Advance ACS Abstracts*, May 1, 1997.

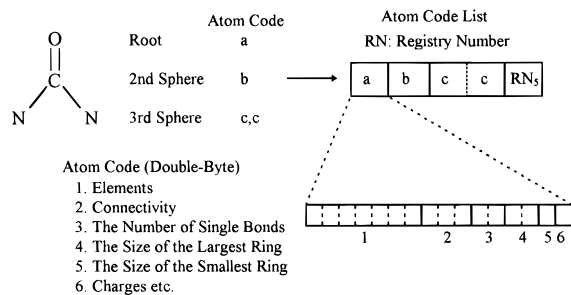


Figure 1. Conversion from a structure to an atom code list.

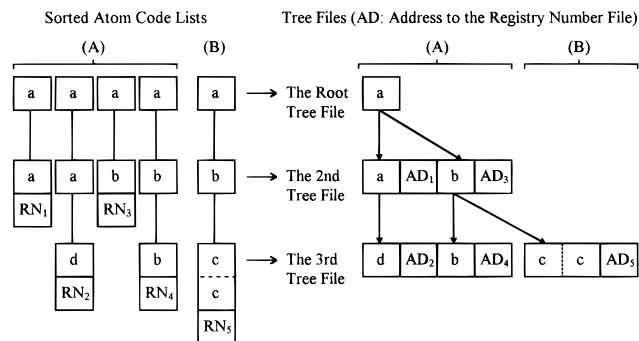


Figure 2. Data format of atom code lists and tree files.

on ring information. This is an advantage of the present system, since DARC needs such an extra procedure on ring information.^{1,7} In Figure 1, the oxygen atom is selected as a root node and "a" is obtained as the atom code.

2. The second sphere nodes neighboring the root node are traced. Hydrogen atoms are not counted. Thus the atom codes of the second sphere are obtained in a similar way as shown in the first step. In Figure 1 the carbon atom is in the second sphere and "b" indicates the atom code.

3. In the same way the node information of each sphere is converted into an atom code one after another. After the conversion is finished, all the atom codes are gathered in sequential data in order, and a unique registry number (RN₅) is added to the end of the data. As shown in the right hand side of Figure 1, a second sphere atom code comes after the root's atom code, and a set of two atom codes "c" are distinguished from the others because they are in the same sphere. The sequential data are referred to as "an atom code list". This conversion is iterated until all nodes except hydrogen are specified as a root node. Hence atom code lists can be generated as many as the structure's nodes at the best. For the case of Figure 1 three atom code lists are generated. The reason why there are not four is that the two nitrogen atoms are equivalent.

4. The above conversion process is applied to all structures in the data file. All the atom code lists are sorted in terms of the atom codes. The tree files (Figure 2, right) are generated by the transformation of the atom code lists (Figure 2, left). Each tree file contains each sphere atom codes. Thus, the root tree file stores all root atom codes of all structures. The second tree file stores all second sphere atom codes in all structures. In this respect the present tree-structured searching system is different from S4.¹³ Each tree file of S4 consists of data of atoms spreading from the same parent atom. On the other hand, each tree file in this system contains addresses to the atom codes in its child tree. Such an address is represented by a downward arrow (Figure 2, right).

Let us consider a case in which four preceding lists (A) are present before the data of urea (BV) are treated (Figure 2, left). Figure 2 (right) illustrates how the atom code list in Figure 1 (B) is put into the preceding tree files (A). First the root atom code a of (B) is compared with the one in the root tree file (A). There is already the atom code a in it. Hence it is not added in the root tree file. The second sphere atom code b of (B) is found not to match the first code a in the second tree but to match the second code b. The b in the second tree file has an address arrow only to b in the third tree file at the stage of A. After the registration of (B), the new atom codes "c,c" are added at the end of the third tree file, and a downward arrow (address) is drawn to show the parent-child relationship. The whole conversion job for the 80 000 structure data totally makes 105 tree files, which include spheres up to the 105th sphere at most.

Registry Number File. During the transformation of the atom code lists, a registry number file is also made. It is a sequential data file of registry numbers. After each atom code list is stored in the tree files, its registry number is written at the end of the registry number file. At the same time the file address to the registry number file is obtained and added to the tree file. In Figure 2 the file address (AD₅) is written next to the atom codes c,c. The atom codes are also accompanied with the occurrence number of records in the lower spheres of the tree; the number describes how many registry numbers should be accessed in the registry number file. If a query matches with such atom codes, all registry numbers corresponding to the query structure are obtained by accessing the registry number file through the address and the occurrence number. Each total search time is shortened drastically with the registry number file. This file system is characteristic only of the present searching system.

Query Structure. Query structure data are prepared in the same way as an atom code list is generated. Thus node attributes are converted to an atom code, and the series of atom codes makes an atom code list which is used as a searching key. Free sites have to be specified as substitution points. Since all nodes are taken as a root node, searching keys are generated as many as the nodes of the query structure.

SCREENING ALGORITHM

Two screening methods were examined to testify the present process based on tree-structured data. The one is the all-key screening that contains iterative screening processes using all searching keys. The other is the single-key screening that consists of two steps, i.e., prescreening and screening with a single key, where the prescreening selects an appropriate searching key by which the smallest candidates are retrieved in the screening process.

All-Keying Screening. The all-key screening searches the tree files by iterative screening processes using all searching keys. The screening process is an application of a simple tree-walking algorithm. First of all the root atom code in the query key is compared with one in the root tree file. Figure 3 shows an example in which they match with each other because both atom codes are a. Thereafter their child nodes in the second sphere are compared. The atom code a in the root tree file has an address to the atom code a in the second tree file. It does not match with b in the second sphere of the query key as shown by a dashed arrow.

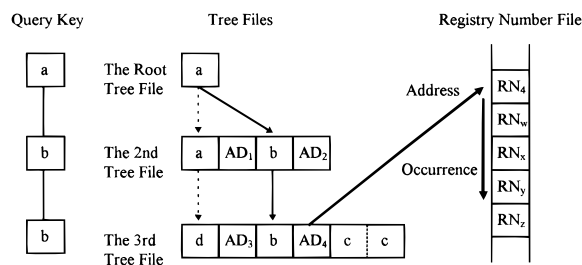


Figure 3. A schematic diagram of a query key, tree files, and the registry number file.

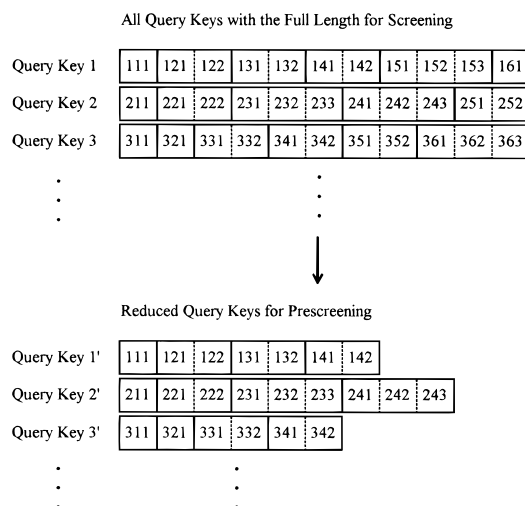


Figure 4. Data format of query keys for the prescreening process.

Further comparison reveals that the second b in the tree file matches the counterpart of the query as shown by a solid arrow. Finally the atom code b in the third sphere of the query key matches with the b in the third tree file. The tree files are searched by a preorder traversal in this way. The atom code b in the third tree file is accompanied with an address in the registry number file and an occurrence number. Thereby, the registry numbers of the all structures which partly include the query key as a substructure can be obtained by accessing the registry number file along with the address and the occurrence number (Figure 3). These overall procedures are referred to as the screening process. It is repeated until all atom codes in the tree files are compared, and it results in bitmap data of candidate structures from the registry numbers.

The screening process is iterated until all query keys are examined. In order to reduce the number of candidates, the final data are obtained by the Boolean product of all bitmap data.

Single-Key Screening. Single-key screening consists of two procedures. The first procedure (prescreening) selects the most efficient query key from the all keys. The selected key is expected to retrieve the fewest candidates and finish the screening process in the shortest time. The subsequent screening procedure retrieves the candidates with the selected query key as a single key.

In the prescreening stage, atom codes up to the fourth sphere are selected from all query keys in the searching. Figure 4 illustrates the way to reduce the query keys. Each number shows an atom code. The query key 1 has atom codes up to the sixth sphere. Therefore atom codes of the fifth and sixth spheres (151, 152, 153, and 161) are eliminated. And it makes the query key 1' for the prescreen-

ing procedure. In the same way atom codes 251 and 252 in the query key 2 are eliminated to make the query key 2'. By using the reduced query keys the prescreening searches the tree files by a preorder traversal, the retrieval algorithm of which is similar to the one of all-key screening. If all the atom codes in the query are assigned, the occurrence number for the registry number file is obtained. However, it is not necessary to get the registry numbers from the registry number file. Only the occurrence number is accumulated during the searching. The accumulated number is referred to as an index of candidates for the query key. In other words it is considered to indicate the relative number of candidates. After the retrieval with all other keys, they are sorted in terms of their indices, and then the query key with the smallest index is selected as the most effective one giving the smallest candidates. The selected special key with its full length is employed in the following screening procedure. Registry numbers are obtained by the same algorithm as the all-key screening.

The prescreening process does not always provide a correct key giving the smallest candidates. The possibility in which the prescreening process gives the correct key was measured with 87 test query structures (Figure 5). The query structures are selected in the experiment as queries frequently used in usual retrieval and roughly categorized into two groups, ring compounds (nos. 1–49) and chain compounds (nos. 50–87). The asterisks indicate free sites. Most of the free sites are placed instead of hydrogen atoms in order to testify the feasibility of fuzzy queries. The searching file contains the 80 000 structures, and the tests were carried out on a workstation computer (SPARCStation IPX).

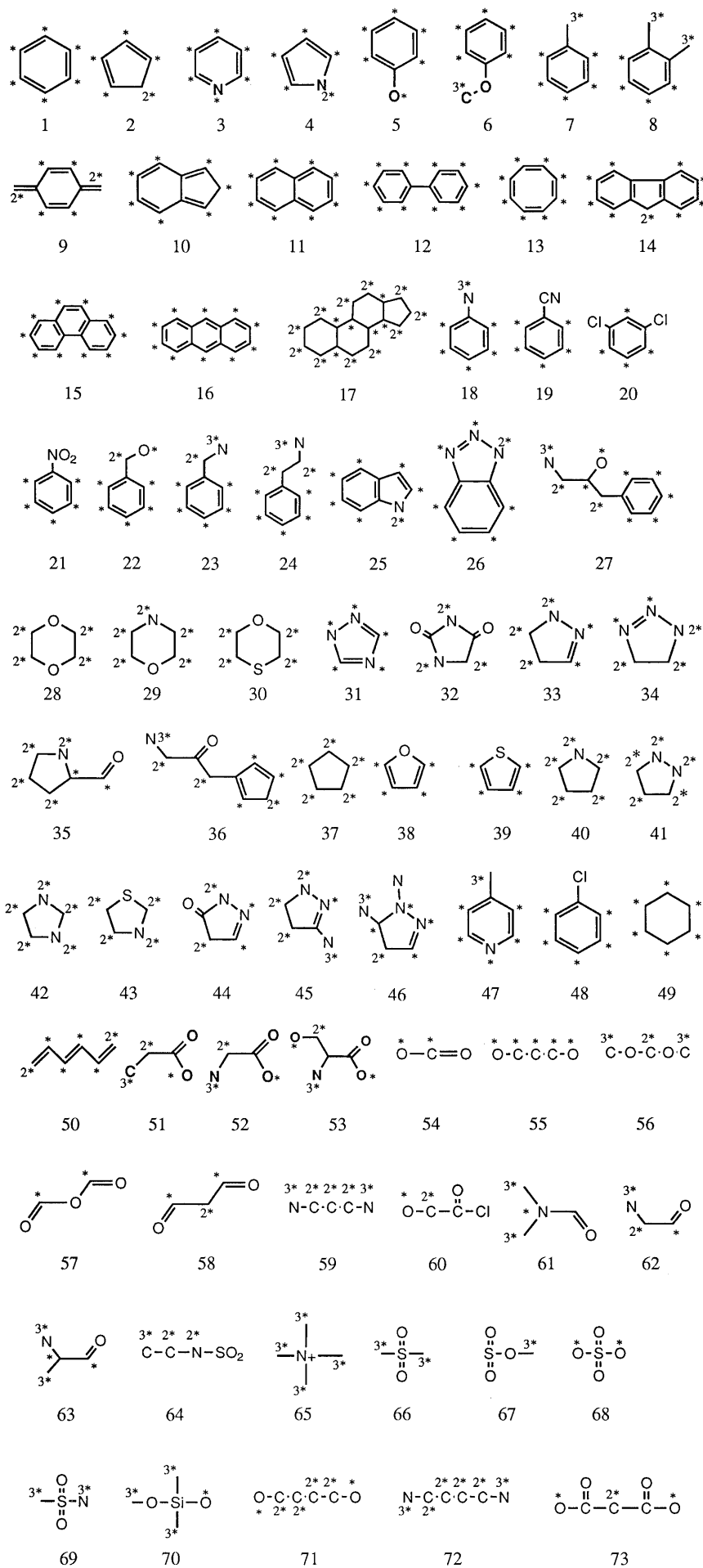
Figure 6 shows the relationship between the length of the query key and the possibility of correct estimation. Average searching time is also plotted. The possibility refers to the correctness of estimating the most efficient key. As more spheres are taken in a key, the possibility for the correct estimation increases. The figure also explains the delicate balance of the accuracy and the search time. As seen in the Figure 6 the possibility saturates in the fourth sphere with about 90%, while the search time increases even in the fifth sphere. This result indicates the keys with up to the fourth sphere are enough to estimate the smallest candidates.

Figure 7 shows the relationship between the sphere number of the tree file and the average number of branches of each set of atom codes. If there are too many branches in a certain node of the tree file, it is very difficult to estimate how many addresses there are under the node to get registry numbers. It is because the structures of the lower spheres are very complicated, but if there is only one branch on the average in a certain node, the number of candidates are expected to be almost equal to the nodes of the sphere.

Each node has 391 branches on the average in the root tree file and 28.4 branches in the second tree file. Those numbers are too many to estimate the number of addresses to the registry number file in the lower tree files, but it is rapidly decreasing until it counts under 2 in the fourth sphere. Therefore here again, a key with up to the fourth sphere is appropriate to estimate the relative number of candidates.

COMPARISON OF PERFORMANCE

Single-Key Screening and All-Key Screening. The performances of two methods described above were com-



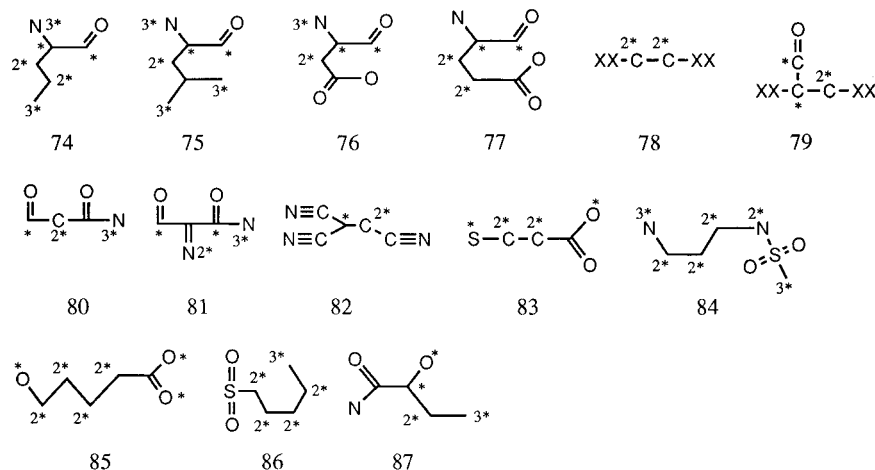


Figure 5. Eighty-seven query structures used in the experiment.

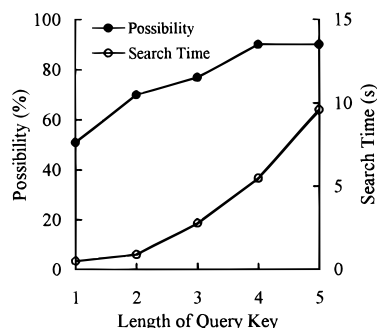


Figure 6. Relationship between length of query key and possibility of selecting the key with the smallest candidates, average searching time.

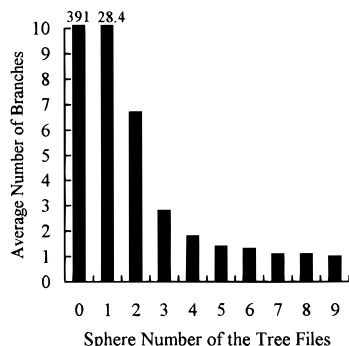


Figure 7. Relationship between the sphere number of the tree file and the average number of branches.

pared in terms of search times and screening efficiency. The results are shown in Table 1 where the screening efficiency of each entry is described as the quotient of the number of hits divided by the number candidates. Note that the screening efficiency of 1 stands for the fully selective screening performance.

(a) Screening Efficiency. From the data collected in Table 1, average values of screening efficiency and searching elapse times were calculated for the two category of compounds, as shown in Table 2. Although the queries (Figure 5) contain many nodes of unspecified valencies, both of the methods attain good values of screening of efficiency over 0.6. This value indicates that the two methods provide the comparable performance to DARC⁷ when the fuzziness of the queries is taken into consideration. The all-key screening exceeds the single-key screening by only 11% in the efficiency values. It should be noted that the former

method calculates the Boolean product among the candidates obtained from all keys, whereas the latter refers to the candidates from a single key. This suggests that other keys contribute very little to reducing the number of candidates in the all-key screening. This is attributed to the similarity of all keys stemming from one structure. They have the same atom codes although their orders are different.

The efficiency of searching chain compounds is generally higher than the one of searching ring compounds in both methods. This is because an atom code contains only two attributes of ring information, i.e., the size of smallest rings and the largest one. Since about 90% of the files includes one or more ring substructures, introducing more ring information is expected to increase the performance.

When the single-key screening is compared with the all-key screening, some retrievals of former screening are found to give far lower values, as shown in entry nos. 27, 62, and 70, and so on. Their common feature is the presence of two heteroatoms. At least two query keys have heteroatom as a root, which provide more distinct Boolean products than those with carbon atoms as roots.

(b) Search Time. Elapse times were measured, since they reflect cpu times well for a single test user. The mean search time of the single-key screening is much less than the one of the all-key screening, as shown in Table 2. A comparison of the search time as a whole system needs the development of an atom-by-atom matching process which eliminates noise records from the screening process. This subject is now under investigation.

The searching of ring compounds is more time-consuming than that of chain compounds. This is simply because the numbers of candidates for ring queries are more than those for chain queries. Some specific queries are found to consume much time, e.g., nos. 7, 18, and 24. All of them are benzene derivatives that involve chain substituents. Further attributes concerning ring substructures should be introduced to increase the performance in both selectivity and search time; this is a future target of the present project.

Fragment Screening and Tree-Structured Screening. The fragment-oriented screening process evaluated in this experiment has been implemented in our in-house chemical substance database system (SPHINCS). It analyzes the structures in terms of 1110 fragments, and the analyzed data are stored in ADABAS files. Then searching is conducted comparing the data with those of the query structure. The

Table 1. Searching Performance of All Queries^a

query no.	all-key screening		single-key screening		fragment screening		query no.	all-key screening		single-key screening		fragment screening	
	screening efficiency	time(s)	screening efficiency	time(s)	screening efficiency	time(s)		screening efficiency	time(s)	screening efficiency	time(s)	screening efficiency	time(s)
1	0.993	9	0.993	9	1.000	210	45	0.891	4	0.802	3	0.188	171
2	0.084	2	0.069	1	0.002	225	46	(0)	3	(0)	1	(2299)	79
3	0.240	12	0.237	9	0.207	160	47	0.138	35	0.087	12	0.056	137
4	0.467	2	0.385	0	0.055	85	48	0.987	36	0.987	14	0.901	143
5	0.992	35	0.992	17	0.936	150	49	0.152	3	0.152	3	0.302	147
6	0.993	60	0.981	17	0.922	151	50	0.983	55	0.983	21	0.980	109
7	0.989	65	0.988	26	0.957	133	51	0.960	16	0.909	8	0.872	195
8	0.516	47	0.506	20	0.810	92	52	0.803	5	0.671	4	0.112	117
9	0.000	47	0.000	12	0.000	105	53	1.000	5	0.846	2	0.024	153
10	(0)	7	(2)	5	(4583)	41	54	1.000	3	1.000	1	1.000	160
11	0.912	20	0.912	5	0.876	51	55	0.503	8	0.498	5	0.115	119
12	0.516	153	0.516	16	0.056	81	56	0.992	6	0.992	2	0.107	88
13	0.000	10	0.000	10	1.000	13	57	1.000	2	1.000	1	0.023	245
14	0.950	7	0.950	2	0.200	28	58	0.899	6	0.663	4	0.292	104
15	0.667	28	0.667	4	0.877	13	59	0.472	13	0.460	7	0.311	149
16	0.082	20	0.072	8	0.514	6	60	1.000	3	1.000	2	0.233	160
17	0.600	8	0.600	3	0.600	5	61	1.000	5	1.000	3	0.557	88
18	0.983	43	0.982	20	0.934	223	62	0.460	12	0.354	3	0.163	247
19	0.957	36	0.957	12	0.303	110	63	1.000	4	0.999	2	0.852	118
20	0.981	12	0.977	8	0.919	43	64	0.999	15	0.978	5	0.888	107
21	1.000	40	1.000	14	0.967	94	65	1.000	2	1.000	0	1.000	96
22	0.810	75	0.663	19	0.367	156	66	1.000	2	1.000	1	0.968	131
23	0.720	79	0.696	21	0.304	147	67	0.999	2	0.999	1	0.097	73
24	0.205	155	0.168	25	0.056	197	68	0.998	1	0.998	1	0.098	57
25	0.926	13	0.809	6	0.198	41	69	1.000	2	1.000	1	0.999	114
26	0.987	11	0.982	6	0.813	43	70	1.000	2	0.731	1	0.267	168
27	0.891	138	0.152	19	0.051	94	71	0.143	15	0.142	7	0.045	110
28	1.000	1	0.101	1	0.129	80	72	0.043	24	0.042	9	0.014	156
29	0.673	4	0.555	2	0.788	123	73	0.914	3	0.812	1	0.025	103
30	0.006	2	0.006	2	0.016	69	74	0.976	41	0.587	10	0.441	105
31	0.334	3	0.331	1	0.197	92	75	1.000	13	0.955	4	0.386	111
32	1.000	3	1.000	1	0.960	77	76	0.896	9	0.700	3	0.068	168
33	0.635	3	0.630	1	0.406	111	77	0.917	18	0.172	5	0.012	151
34	0.028	2	0.028	1	0.006	117	78	0.981	1	0.981	1	0.848	116
35	0.917	5	0.398	2	0.031	88	79	0.996	3	0.996	1	0.826	108
36	(0)	28	(5)	8	(8776)	84	80	0.964	12	0.663	5	0.358	181
37	0.223	1	0.223	1	0.020	164	81	0.938	10	0.258	6	0.019	102
38	0.601	1	0.600	0	0.056	158	82	(0)	2	(0)	1	(130)	62
39	0.294	1	0.291	2	0.043	134	83	0.669	8	0.465	5	0.315	122
40	0.220	2	0.197	2	0.064	125	84	0.887	39	0.857	11	0.277	103
41	0.774	1	0.751	1	0.146	72	85	0.148	48	0.051	14	0.016	139
42	0.325	2	0.306	1	0.272	121	86	0.316	109	0.292	16	0.055	97
43	0.328	2	0.327	1	0.257	135	87	1.000	17	0.992	5	0.833	150
44	0.993	3	0.959	1	0.665	116							

^a (): No hit obtained. Number of screens is shown instead.**Table 2.** Performance Comparison between All-Key Screening and Single-Key Screening

structure	all-key screening			single-key screening		
	candidates	screening efficiency	time(s)	candidates	screening efficiency	time(s)
ring compds	463 005	0.564	26	482 826	0.521	8
chain compds	237 452	0.834	14	253 101	0.731	5
total	700 457	0.682	21	735 927	0.612	6

Table 3. Searching Performance of Fragment Screening

structure	fragment screening		
	candidate	screening efficiency	time(s)
ring compds	721 935	0.405	107
chain compds	484 992	0.392	128
total	1 206 945	0.399	116

retrieval was carried out on a FUJITSU M1800/20, a main frame computer, through a TSS terminal.

(a) Screening Efficiency. Table 3 shows elapse times of searching and screening efficiency for the fragment screening

method. The screening efficiency based on tree-structured data (Table 2) is far more excellent than those of the fragment screening. The advantage of searching with tree-structured data has been confirmed here again, and this tendency is consistent with the previous study.⁷ One of the reasons is the amount of structural information in which all nodes are taken into consideration, while the fragment screening takes only 1100 attributes.

The fragment screening surpasses the tree-structured screening in only nine query structures (Table 1). These results are attributed to the distinct fragments that specified the queries very well. For example the fragment screening

analyzes no. 8 query structure into a six-membered conjugated ring with ortho substitution. Query no. 13 generates a more specific fragment, i.e., an eight-membered conjugated ring, which is exactly the same structure as the query. There are four queries which retrieve no hits. These results are realized in the methods base on tree-structured data, while the fragment method gives incomplete results. Hence, the screening system based on tree-structured data is concluded to be more efficient than the fragment-oriented screening.

(b) Search Time. Strict comparison of search time is difficult since hardware configurations are different from each other. The fragment screening is conducted on a mainframe computer with 57 MIPS (MIPS = megainstructions per second). And the tree-structured screening is on a workstation computer with 28.5 MIPS. As Barth et al.¹⁴ reported in their paper, there is no practical measure available which can be used for different machine architectures. Then simple comparison is presented here.

The single-key screening on a workstation computer is appropriate as far as the searching time on the terminal concerns. As can be seen from the result (Table 1), the tree-structured screening methods gives a rapid search time. The mean time of the all-key screening is about six times faster than the fragment screening, while that of the single-key screening is about 19 times. In order to examine a fragment method on a workstation, a similar retrieval to the fragment screening of SPHINCS was conducted on a SPARCStation by using ORACLE as a relational DBMS, in which 100 attributes for 80 000 records were searched sequentially. It requires 20 s on the average. Much more time has to be consumed when 1110 attributes are taken into consideration. This implies that the tree-structured screening is equal or rather superior to the fragment method.











(c) Exact Match. Exact match retrieval refers to the searching with the query having no free sites. Performance comparison was investigated between the fragment and the tree-structured screening methods by using ten queries listed in Table 4. While the fragment screening gives enormous numbers of candidates, the two tree-structured screening methods acquire exactly the hit records without noise candidates. Table 4 shows that the query benzene and the third query pyridine retrieve more than one candidate. This is because some records in the data file include mixed structures. Thereby they retrieve the mixed structures including the query.

And the search times of tree-structured screening methods are much faster than those of fragment screening. Each search process is finished in almost 1 s which indicates exact queries do not require many accesses to the storage. Although the tree-structured screening methods have no such special functions as hash codes for the exact match, they are capable of obtaining small number of candidates sufficiently.

OTHER FEATURES

The screening with tree-structured data is applied to the search for the query with generic terms like A, X, Q, and M, where A corresponds to any, X to halogens, Q to heteroatoms, and M to metal atoms. The symbol XX is used to notify a terminal node of a polymer unit so that the nodes are distinguished from those of monomeric structures. The tree-structured screening is more selective and faster than

Table 4. Searching Performance of Exact Match Retrieval

query	all-key screening		single-key prescreening		fragment screening	
	candidates	time(s)	candidates	time(s)	candidates	time(s)
	8	1	*	*	35070	106
	0	1	0	0	10885	128
	69	1	69	0	2176	297
	1	1	1	1	1264	188
	1	1	1	1	36953	321
	1	2	1	0	23781	265
	0	1	0	0	615	107
	1	2	1	1	19777	196
	1	2	1	1	1977	192
	1	2	1	1	924	121
mean time		1		1		192

* No need for prescreening because of only one query key.

the fragment-oriented screening even when they include the generic atoms of broader meaning.

The present system uses no full description of connection tables. Since the resulting connectivity data among the atoms are flexible, it is also possible to contain generic (Markush) structures. This feature is common to such tree-structured systems as the present one and the S4.

One of the important database functions is modification of data with registration and updating. As long as the analysis data are stored in conventional database, manipulation is easy for fragment screening. On the other hand the tree-structured screening methods are appropriate for the data that does not need frequent changes, because the tree files and registry number file have to be reconstructed on each modification. This subject should be solved in the future.

SUMMARY

A screening process was investigated using tree-structured data as searching files. This program is brought about by atom codes in the tree files. No additional screening process is necessary for ring information, because the atom codes include ring attributes of the nodes. Two screening methods were examined in terms of the screening procedure. In the single-key screening, the prescreening process selects the key which gives the smallest sets of candidates and is followed by the screening process with the selected key. On the other hand, the all-key screening is not so effective to reduce the candidates despite the longer searching time. Although both

of the methods show excellent screening efficiencies and short search times, the prescreening method has the advantage of short search times. Comparison between the fragment-oriented screening and the tree-structured screening indicates that the latter system is much more improved in both searching time and screening efficiency. And it also shows excellent features on the exact matching retrieval. This suggests the advantage of tree-structured data.

REFERENCES AND NOTES

- (1) Barnard, J. M. Substructure Searching Methods: Old and New. *J. Chem. Inf. Comput. Sci.* **1993**, 33, 532–538.
- (2) Von Scholly, A. A relaxation algorithm for generic chemical structure screening. *J. Chem. Inf. Comput. Sci.* **1984**, 24, 235–241.
- (3) Dubois, J.-E.; Panaye, A.; Attias, R. DARC System: Notions of Defined and Generic Substructures. Filiation and Coding of FREL Substructure (SS) Classes. *J. Chem. Inf. Comput. Sci.* **1987**, 27, 74–82.
- (4) Attias, R. DARC Substructures Search System: A New Approach to Chemical Information. *J. Chem. Inf. Comput. Sci.* **1983**, 23, 102–108.
- (5) Bruck, P.; Nagy, M. Z.; Kozics, S. Substructure search on hierarchical tree. In *Online information 87*; Proceedings of the 11th International Online Information Meeting, London; Dec 8–10, 1987; Learned Information: Oxford, 1987; pp 41–43.
- (6) Nagy, M. Z.; Kozics, S.; Veszpremi, T.; Bruck, P. Substructure Search on Very Large Files Using Tree-Structured Databases. In *Chemical Structures*; Warr, W. A., Ed.; Springer-Verlag: Berlin, 1988; pp 127–130.
- (7) 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.
- (8) Hicks, M. G.; Jochum, C.; Maier, H. Substructures search systems for large chemical data bases. *Anal. Chim. Acta* **1990**, 235, 87–92.
- (9) Hicks, M. G. Similarity and the Beilstein Information System: Searching for Concepts with Current Facts. *J. Chem. Inf. Comput. Sci.* **1992**, 32, 631–638.
- (10) Hicks, M. G. Beilstein Current Facts in Chemistry: a large chemical database on CD-ROM. *Anal. Chim. Acta* **1992**, 265, 291–300.
- (11) Maier, H.; Walkowiak, D. CHEMICAL SUBSTRUCTURE SEARCH ON CD-ROM. In *Software Development in Chemistry 4*; Gasteiger, J., Ed.; Springer-Verlag: Berlin, 1990; pp 1–9.
- (12) Welford, S. M. Chemical Structure Searching Using S4/MOLKICK on DIALOG. In *The Beilstein Online Database*; Heller, S. R., Ed.; American Chemical Society: Washington, DC, 1990; pp 64–79.
- (13) Bartmann, A.; Maier, H.; Walkowiak, D.; Roth, B.; Hicks, M. G. Substructure Search on Very Large Files by Using Multiple Storage Techniques. *J. Chem. Inf. Comput. Sci.* **1993**, 33, 539–541.
- (14) Barth, A.; Westermann, U.; Pasucha, B. Messenger and S4: A Comparison of Structure Search Systems. *J. Chem. Inf. Comput. Sci.* **1995**, 34, 714–722.
- (15) Warr, W. A. NEW CHEMICAL DATABASES ON CD-ROM. *DATABASE* **1993**, 16, 59–67.
- (16) Warr, W. A. More Chemistry On CD-ROM. *DATABASE* **1995**, 18, 60–64.
- (17) Hanai, S.; Miyakawa, M. Registry and Search of Chemical Compounds on Graphics Display. *Anal. Chim. Acta* **1987**, 194, 37–48.

CI960378+