An Isomorphism Algorithm for Chemical Graph Searching

Hanxiong Chen

Abstract

The chemical structure searching is one of the most important subjects in many fields. On the other hand, structure searching is a graph isomorphism problem of testing subgraph, which has been proved to be NP-complete and should be avoided if possible. For this purpose, our proposal is to organize the chemical structure by pre-processing to a semi-order structure (a DAG, Directed Acyclic Graph), where each chemical structure has all its sub-structures as its super nodes in the DAG. With this DAG, subgraph searching is simplified to exact graph determination. When a graph is found to be isomorphic to the query graph, then the sub-DAG rooted at the found graph is the answer to the query. In this paper, we propose an improved polynomial-time algorithm for the graph isomorphism problem.

Keywords algorithm, chemical structure, isomorphic, retrieve.

1 Introduction

The chemical structure searching is one of the most important subjects not only in the field of chemical information documentation and processing, but also in other areas such as synthesis planning, drug design and so on. In graph theoretical terms, structure searching is a graph isomorphism problem which involves testing a series of topological graphs in the database for the existence of the query graph, and substructure searching is a problem of testing for the existence of the graph which contains the query graph. Although polynomial-time algorithms for graph isomorphism problem have been found in certain restricted types of graph[1], the subgraph isomorphism problem has been proved to be NP-complete. Therefore, to search substructure efficiently, the subgraph isomorphism operation should be avoided as possible[6].

The main approach for substructure searching with acceptable average time is to use heuristics and carry out the most time consuming operations in a pre-processing of the database. Many systems using screening approach have been developed for substructure searching[2][3]. They use invited file of some structural feature (fragments) of graph to eliminate the graphs which can not match the query. However, they still have to use the subgraph isomorphism operation for the final decision. The main reason is that only the present or absent of some structural feature is taken into account in the invited file, but the topological information (the relationship or connection among these fragments) is omitted. However, the storage and processing ability of computer system increased dramatically in recent years. It is becoming possible to represent and deal with chemical structures systematically with certain restrictions. That is, in stead of the fragmental index of the structural feature, a semi-ordered conceptual hierarchy (a DAG) of chemical graph should be constructed according to the substructure relationship in advance. The node in the hierarchy stands for the chemical structure or substructure, the link stands for the substructure or superstructure relationship between nodes. Taking advantage of the conceptual hierarchy, substructure searching can be implemented efficiently in two steps: find the entrance node standing for the query substructure by graph isomorphism (not subgraph isomorphism) operation, and then retrieve all candidate compounds by following the links directly.

2 Construct the conceptual hierarchy

The chemical structure is represented in a form of connection table in our system. A connection table contains a list of the non-hydrogen atoms, together with bond information that describes the exact manner in which individual atoms are connected together.

The basic idea of the construction algorithm[7] can be described as follows: for each structure in the database, generate its substructures by cutting off bond, atom or meta-atom (ring) and organize the hierarchy recursively. In the cutting strategy, the semantics of chemical structure is used to make the operation efficient and non-exponential.

In construction, SSSR algorithm [4] is used to detect meta-atom (rings), which finds the smallest set of smallest rings in a structure. And the graph isomorphic determination subroutine *ISOM* as well helps here. Figure 1 shows a part of the result hierarchy.

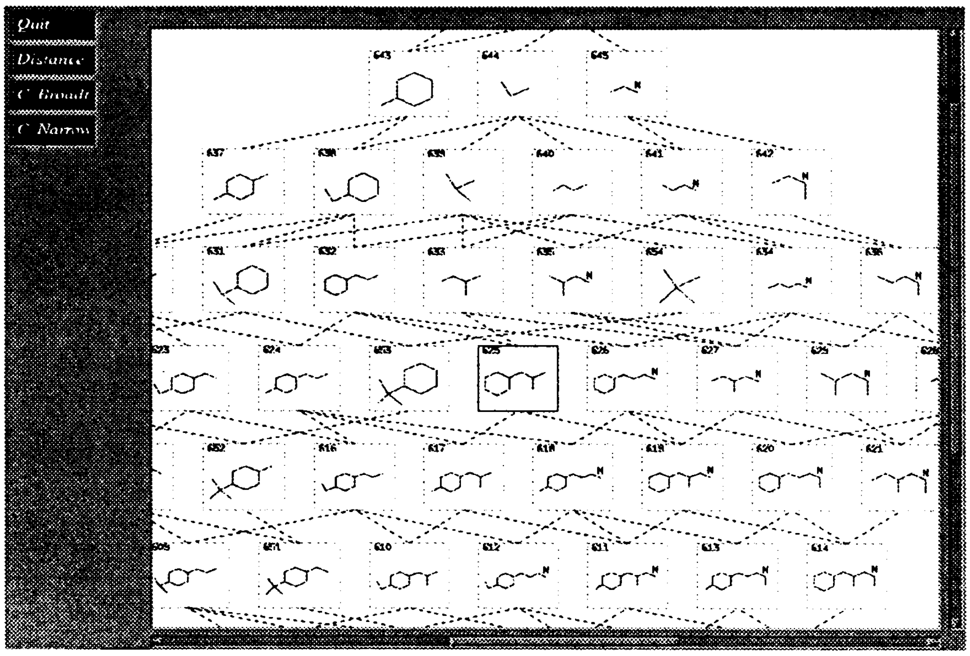


Fig. 1: An DAG of Chemical Structure

3 The graph isomorphism algorithm ISOM

An efficient algorithm is significant here because that in our approach to substructure searching, the graph isomorphism operation is used not only in the pre-processing of the database, but also in the searching step to find the entrance of the hierarchy.

Definition (molecule): $\mathcal{M} = (\mathcal{A}, \mathcal{B})$ is a chemical molecule. Where $\mathcal{A} \subset \mathcal{N} \times \mathcal{C}$ is the set of nodes which represents the set of elemental atoms and $\mathcal{B} \subset \mathcal{A} \times \mathcal{A} \times \mathcal{C}$ is the set of bonds. "Color" $\mathcal{C}$ represents the set of atom names (Carbon, Oxygen, etc.) and the set of types of bonds (single, double, and so on) in $\mathcal{A}$ and $\mathcal{B}$, respectively.

For convenience, because the node number identifies the node, we always use n, instead of c, to stand for node $a = (n, c)$. Similarly, a bond b is cited by an unique number.

Algorithm *ISOM*: Determine whether two chemical molecules, M_1 and M_2, are isomorphic. If their are isomorphic, output the correspondence of the atoms between them.

For the simplicity, we introduce a special list D which doesn't allow repeating elements. Let $D =< n_1, n_2, ..., n_k >$, $(\forall i \neq j,\ n_i \neq n_j)$, the notations in the following are used.

$|D| \triangleq |\{n_1, n_2, ..., n_k\}| = k$

$SET(D) \triangleq \{n_1, n_2, ..., n_k\}$

$n \in D \triangleq n \in SET(D)$

The algorithm either fails (i.e., terminates with output "not isomorphic") or outputs D_1, D_2, where $|D_1| = |A_1| = |D_2| = |A_2|$, and $SET(D_i) = A_i, i = 1, 2$. The implementation of the algorithm is described in the following steps.

3.1 Set generation

In this step, both nodes and edges of the molecules are scanned.

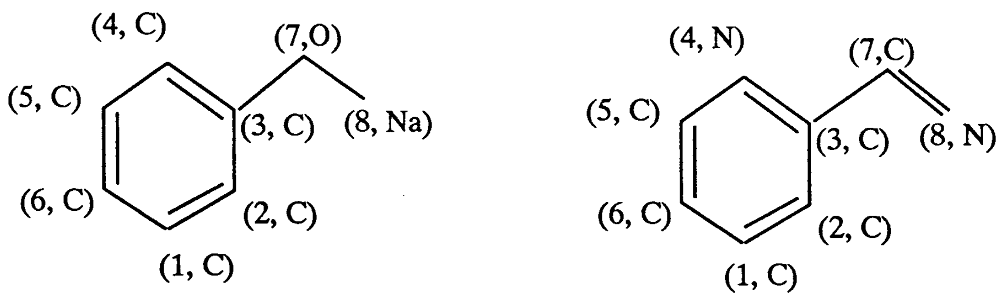
1. Scan A_i of M_i $(i = 1,\ 2)$, collect nodes with the same color, and nodes with the same degree. Whenever an un-equivalence is found then fails the whole algorithm. For example, suppose that the Carbon nodes of M_1 and M_2 have been collected into S_{11} and S_{21}, respectively, if S_{11} has two elements while S_{21} has three, then it is impossible that M_1 and M_2 are isomorphic. Further, unlike normal graphs, a bond of a molecule may be single, double, triple, and so on. According to different type of bonds, degrees are different.

2. Scan B_i of M_i $(i = 1,\ 2)$, collect nodes with same color. Different to normal graph, a bond may be double, triple as well as single. The start and end nodes of the same color bonds are collected together.

The semantics of the sub-sets generated in this step does not affect the correspondence and thus is ignored in the following discussion. We just number them as S_{ij} for the subsets, with $i = 1$ and $i = 2$ corresponding to M_1 and M_2, respectively. And for all j, $|S_{1j}| = |S_{2j}|$.

In the following, an example of set generation is illustrated for molecules in Figure 2, where "carbon nodes" are those nodes with color "C", that is, nodes (*, C) in the Figure. Although all sets are listed in this example, any of conditions 2, 3, 4, 6,7, 8 or 9 is enough to "fail" the algorithm.

Conditions	Query structure	Database structure
1. carbon nodes	$S_{11} = \{1,2,3,4,5,6\}$	$S_{21} = \{1,2,3,5,6,7\}$
2. sodium nodes	$S_{12} = \{8\}$	$S_{22} = \{\}$
3. oxygen nodes	$S_{13} = \{7\}$	$S_{23} = \{\}$
4. nitrogen nodes	$S_{14} = \{\}$	$S_{24} = \{4,8\}$
5. single-degree 1	$S_{15} = \{1,2,4,5,6,8\}$	$S_{25} = \{1,2,4,5,6,7\}$
6. single-degree 2	$S_{16} = \{3,7\}$	$S_{26} = \{3\}$
7. double-degree 1	$S_{17} = \{1,2,3,4,5,6\}$	$S_{27} = \{1,2,3,4,5,6,7,8\}$
8. single-bond	$S_{18} = \{1,2,3,4,5,6,7,8\}$	$S_{28} = \{1,2,3,4,5,6,7\}$
9. double-bond	$S_{19} = \{1,2,3,4,5,6\}$	$S_{29} = \{1,2,3,4,5,6,7,8\}$
Correspondence:	$D_1 = <>$	$D_2 = <>$



(a). The Query Structure **(b). A Database Structure**

Fig. 2: An Example.

3.2 Partition

In many cases, sets S_{ij} $(i = 1, 2)$ generated in set generation is not powerful enough to terminate the algorithm. This happens when no contradiction found, yet node correspondence D_1, D_2 do not cover A_1, A_2, that is, $\mathrm{SET}(D_i) \neq A_i$.

Obviously, sets with single-element (such a set is called a single-set in the sequel) are appended to D_1 and D_2 directly, because they give unique correspondence. Otherwise, it is tried to find small sets or even single-sets from the sets generated by calculating the intersection. It is very hard to calculate all the 2^m intersections of m sets. Rather, we will divide these sets S_{ij} into disjoint sets, using the following recursion. The assumption is that at first calling of *dvd*, $2 \times m$ sets $S_{ij}(i = 1, 2,\ j = 1, 2, ..., m)$ have been generated in the previous step.

```
Procedure $dvd(1, m, S)$
begin
    find an appropriate set $S_{1k}$
    for each $j$
        append all single sets to $D_i$, delete single elements from each $S_{ij}$.
        group $S_{ij} \cap S_{ik}$ to $S'_{ij}$, $(i = 1, 2)$
        group $S_{ij} - S_{ik}$ to $S''_{ij}$, $(i = 1, 2)$
        group $S_{ik} - S_{ij}$ to $S''_{ij}$, $(i = 1, 2)$
        (fail whenever a $S'_{1j} \neq S'_{2j}$ or $S''_{1j} \neq S''_{2j}$ encountered)
    call $dvd(1, m', S')$
    call $dvd(1, m'', S'')$
end
```

Ideally, S_k which divide S_{ij} to two sets of S'_{ij} and S''_{ij} with $\cup S'_{ij} = \cup S''_{ij}$ and $m' = m''$ is appreciated. If this is difficult, S_k with $|S_k| = max\{|S_i|/2\}$, or more simply the first non-single set, might be used. The complexity of this *dvd* procedure is $O(m \log m)$ for the recursion.

Lemma 1 Procedure *dvd* is sound and complete. Let S_{ij}, $(i = 1, 2)$ be the sets obtained from set generation, and S'_{ij}, $(i = 1, 2)$ be sets obtained from S_{ij}, $(i = 1, 2)$ by *dvd*. Then by sound, S_{ij} represents the conditions of the chemical(s), and by complete, any conditions of the chemicals taken into S_{ij}, $(i = 1, 2)$ are also kept in S'_{ij}.

Sketch of Proof

If S_{1k} is generated from condition C_k and S_{1t} from condition C_j, then by *dvd*, a $S'_{1k_1} = S_{1k} \cap S_{1t}$, $S'_{1k_2} = S_{1k} - S_{1t}$ and $S'_{1k_3} = S_{1t} - S_{1k}$ are generated. They represent $C_k \wedge C_t$,$C_k \wedge \neg C_t$ and $C_t \wedge \neg C_k$, respectively.

Conversely, let us check the C_k and C_t above. As mentioned, there are S'_{1k_1}, S'_{1k_2} and S'_{1k_3} in S' which represent $C_k \wedge C_t$,$C_k \wedge \neg C_t$ and $C_t \wedge \neg C_k$, respectively. So, C_k is kept by $C_k \wedge C_t \vee C_k \wedge \neg C_t$. C_t is kept for the same reason.

Using the discussion above, it is easy to prove the lemma using induction on the level of recursion of *dvd*. □

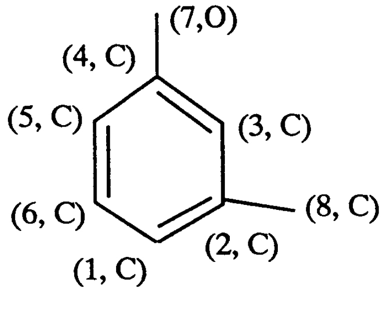
Lemma 1 guarantees the preservation of condition information. Having the sets till now as the base, new condition such obtained from the following steps are divided on this base without loss of information. Each new set introduced by a new condition is compared exactly with the m' ($m' \leq |A_1| - |D_1|$) sets generated so far, searching for further division. Once the comparison is done, the condition is discarded. Hence there is no necessity to backtrack.

3.3 Connectivity

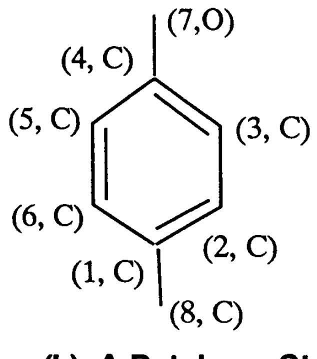
When a single-set is decided, the correspondence between the adjacent nodes around the unique node is taken into consideration. This step is described simply in the following.

For each $i = 1, 2$, for each node n_i newly add to D_i, collect n_i's adjacent nodes to a set $S_{i,m+1}$.

Indeed, such $S_{i,m+1}$ is used to divide the exist $S_{i,j}$'s immediately, thus to reduce the cost, and redundancy.



(a). The Query Structure



(b). A Database Structure

Fig. 3: Set connectivity is as powerful.

3.4 Set Connectivity

When the steps above give no more useful heuristics, and the algorithm does not terminate, then assignment is taken as the last measure in [5] For the example as shown in Figure 3, the steps above give.

Conditions	Query structure	Database structure
1. carbon nodes	S_{11} ={1,2,3,4,5,6,7,8}	S_{21} ={1,2,3,4,5,6,7,8}
2. single degree 1	S_{12} ={1,3,5,6,7,8}	S_{22} ={2,3,5,6,7,8}
3. single degree 2	S_{13} ={2,4}	S_{23} ={1,4}
4. double degree 1	S_{14} ={1,2,3,4,5,6}	S_{24} ={1,2,3,4,5,6}
5. single-bond	S_{15} ={1,2,3,4,5,6,7,8}	S_{25} ={1,2,3,4,5,6,7,8}
6. double-bond	S_{16} ={1,2,3,4,5,6}	S_{26} ={1,2,3,4,5,6}

By the partition step, these sets are reduced to (for convenience, S' is noted to S):

S_{11} ={1,3,5,6}	S_{21} ={2,3,5,6}
S_{12} ={2,4}	S_{22} ={1,4}
S_{13} ={7,8}	S_{23} ={7,8}

According to Sussenguth's proposal [5], elements in S_{12} are assigned to S_{22}, as

S'_{12} ={2}	S'_{22} ={1}
S'_{13} ={4}	S'_{23} ={4}

and

S'_{12} ={2}	S'_{22} ={4}
S'_{13} ={4}	S'_{23} ={1}

Then it has to lead to contradiction by connectivity for every case. This approach raises the combination problem. In contrast, our approach collects the adjacent nodes of S_{12} and S_{22}, this leads to

S'_{12} ={1,3,5}　　　　　　　　S'_{22} ={2,3,5,6}

Which fails the algorithm immediately.

Information for connectivity is retrieved from the connection table directly. So the complexity is $O(m \times |S_{ij}|)$.

Lemma 2 Set connectivity is as powerful as assignment.

[Proof] Let $S_1 = \{s_{11}, s_{12}, ...s_{1n}\}$ and $S_2 = \{s_{21}, s_{22}, ...s_{2n}\}$ are two sets generated with a condition C. Assume that s_{1i} is assigned to s_{2i} by assignment, then by connectivity, each S'_{1i} is generated for s_{1i}, and so is S'_{2i} for s_{2i}. On the other hand, by set connectivity, only one S'_1 is generated for S_1 and so is S'_2 for S_2. Clearly

$$S'_k = \bigcup_i S'_{ki}, \ k = 1, 2.$$

Now, if the lemma does not hold, then

$$S'_1 = S'_2$$

while $\exists\ j$,

$$S_{1j} \neq S_{2j}.$$

By the definition of connectivity, this means that the degree of s_{1j} is not equal to that of s_{2j}. However, by Lemma 1, since S_1, S_2 is produced from *dvd*, they should keep the conditions, including "same-degree". This contradicts the assumption that $s_{1j} \in S_1 \wedge s_{2j} \in S_2$. □

One advantage of assignment is that it leave the possibility of further searching. If one is not satisfied with the result from set connectivity, he may do assignment and calculating the connectivity (which is not good as set connectivity), then further assign over the sets got from connectivity, and calculating the connectivity again. Certainly, each such step raises one order of complexity.

As mentioned before, we are not to design a formal isomorphic algorithm for normal graph. Our concentration is on the chemical molecule, the efficiency is important. Therefor the algorithm must not be too complicated nor takes too much time for calculation. That is the main reason why we designed set connectivity to take place assignment.

Nevertheless, we by no means refuse the assignment. The following example heuristic gives a slight modification to our approach, taking assignment into consideration. It should be noted that the graphs shown in Figure 4 is not realistic in chemistry and should better be left to isomorphism of normal graph which is out of the scope of this paper.

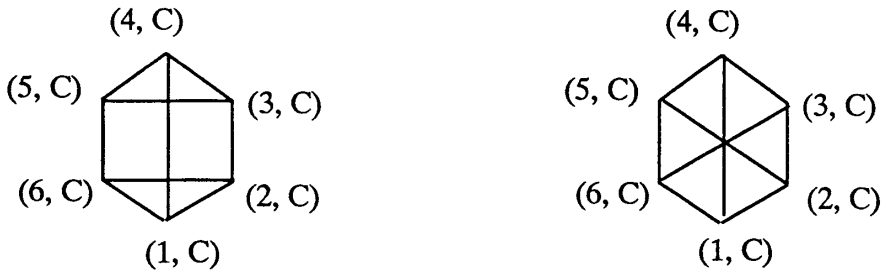


Fig. 4: Set Connectivity is as powerful as assignment.

In this figure, only one set including all the nodes is generated for each graph, $S_1 = S_2 = \{1, 2, 3, 4, 5, 6\}$. Even in such a case, one can easily check that following assignment, doing set connectivity gets to the conclusion "not isomorphic" immediately, while doing connectivity cannot terminate the algorithm.

4 Discussion

A chemical graph, as well as a set, is represented as a bit array. Thus the equivalence of two sets is simply determined by comparing a pair of integers. To get the entrance efficiently in the conceptual hierarchy, some indices such as size of the structure, atom type and number, bond type and number, ring size and number may be useful. Furthermore, to investigate the redundancy of the conceptual hierarchy, "density" of real node (a real node stands for a compound which really exists in the database) in the conceptual hierarchy should be evaluated although it is believed that the density will increase while the database augments.

（チン・カンユウ　産業情報学科）

References

[1] John M. Barnard: Substructure Searching Methods: Old and New, *J. Chem. Inf. Comput. Sci.* 1993, Vol. 33, pp. 532-538.

[2] Lynch, M. F.: Screening large chemical files, In *Chemical information systems*; Ash, J. E., Hyde, E., Eds.; Ellis Horwood: Chichester,1974; pp177-194.

[3] Robert E. Stobaugh: Chemical Substructure Searching , *J. Chem. Inf. Comput. Sci.* 1985, Vol. 25, pp. 271-275.

[4] Johann Gasteiger and Clemens Jochum: An algorithm for the perception of synthetically important rings, *J. Chem. Inf. Comput. Sci.* 1979, Vol. 19, pp. 43-48.

[5] Edward H. Sussenguth,Jr.: A graph-theoretic algorithm for matching chemical structures, *J. Chem. Doc.* 1965, Vol. 5, pp. 36-43.

[6] H. Chen, X. Yu, K. Yamaguchi, H. Kitagawa, N. Ohbo & Y. Fujiwara: Decomposition – An Approach for Optimizing Queries Including ADT Functions, *Information Processing Letters*, Vol. 43(6), pp. 327-333, Oct. 1992.

[7] J. An, H. Chen & Y. Fujiwara: Organizing the Chemical Graph for Substructure Searching. *ACS'95*, Aug. 95, Chicago.

要旨

化学構造の検索は様々な分野で重要な意味をもつ。一方、グラフ理論では、このような検索はNP問題として証明されたグラフの部分一致判定という操作になる。グラフの部分一致判定を極力さけるために化学構造を部分順序構造（DAG）に構築して、部分一致判定を同形判定に簡単化することを提案する。本論文では同形判定の効果的なアルゴリズムを提案する。