INFERRING A CHEMICAL STRUCTURE FROM A FEATURE VECTOR BASED ON FREQUENCY OF LABELED PATHS AND SMALL FRAGMENTS

TATSUYA AKUTSU *

Bioinformatics Center, Institute for Chemical Research, Kyoto University
Gokasho, Uji, Kyoto 611-0011, Japan
E-mail: takutsu@kuicr.kyoto-u.ac.jp

DAIJI FUKAGAWA

National Institute of Informatics
Chiyoda-ku, Tokyo 101-8430, Japan
E-mail: daiji@nii.ac.jp

This paper proposes algorithms for inferring a chemical structure from a feature vector based on frequency of labeled paths and small fragments, where this inference problem has a potential application to drug design. In this paper, chemical structures are modeled as trees or tree-like structures. It is shown that the inference problems for these kinds of structures can be solved in polynomial time using dynamic programming-based algorithms. Since these algorithms are not practical, a branch-and-bound type algorithm is also proposed. The result of computational experiment suggests that the algorithm can solve the inference problem in a few or few-tens of seconds for moderate size chemical compounds.

1. Introduction

Drug design is one of the important targets of bioinformatics. For designing new drugs, classification of chemical compounds is important and thus a lot of studies have been done. Recently, *kernel methods* have been applied to classification of chemical compounds[1,2,3,4]. In most of these approaches, chemical compounds are mapped to *feature vectors* (i.e., vectors of reals) and then *support vector machines* (SVMs)[5] are employed to learn classification rules. Though several methods have been proposed, feature vectors based on *frequency of small fragments*[1,2] or *frequency of labeled paths*[3,4] are widely used, where other chemical properties such as molecular weights, partial charges and logP are sometimes combined with these, and weights/probabilities are sometimes put on paths/fragments.

On the other hand, a new approach was recently proposed for designing and/or optimizing objects using kernel methods[6,7]. In this approach, a desired object is computed as a point in the feature space using suitable objective function and optimization technique and then the point is mapped back to the input space, where this mapped back object is

*Work partially supported by Grants-in-Aid "Systems Genomics" and #16300092 from MEXT, Japan, and by the Kayamori Foundation of Information Science Advancement.

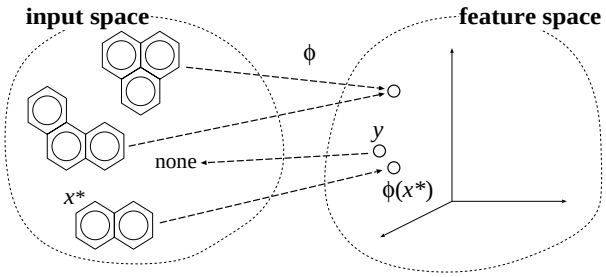


Figure 1. Inference of a chemical compound from a feature vector. Multiple compounds may be mapped to the same point in a feature space.

called a *pre-image*. Let ϕ be a mapping from an input space to a feature space. Then, the problem is, given a point y in the feature space, to find a pre-image x in the input space such that $y = \phi(x)$. It should be noted that ϕ is not necessarily injective or surjective. If ϕ is not surjective, we need to compute the approximate pre-image x^* defined by $x^* = \arg\min_x \ dist(y, \phi(x))$ (see Fig. 1).

The pre-image problem has potential application to drug design[7] by using a suitable objective function reflecting desired properties. For example, suppose that we have two chemical compounds x and y having different functions and we want to design a new chemical compound having both functions. In such a case, we may develop a new compound by computing a pre-image of the middle point of feature vectors corresponding to x and y (i.e., a pre-image of $(\phi(x) + \phi(y))/2$). Though this example is too simple, several ways can be considered for applications of the pre-image problem to drug design and other bioinformatics problems.

There are several related works. Bakir, Weston and Scölkopf proposed a method to find pre-images in a general setting by using Kernel Principal Component Analysis and regression[6]. Bakir, Zien and Tsuda developed a stochastic search algorithm to find pre-images for graphs[7]. However, the pre-image problems are not studied from a computational viewpoint. Graphical degree sequence problems[8], graph inference from walks[9] and the graph reconstruction problem[10] are related to the pre-image problem for graphs. However, results on these problems are not directly applicable to the pre-image problem.

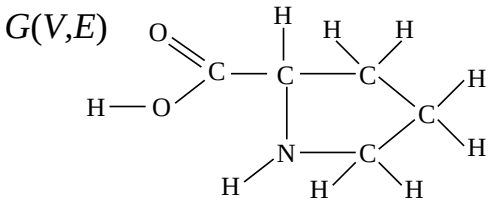
In our previous works[11,12], we studied a theoretical aspect of the pre-image problem on graphs. In Ref. 11, we studied the problem of inferring a graph from the numbers of occurrences of vertex-labeled paths. We showed that this problem can be solved in polynomial time of the size of an output graph if graphs are trees of bounded degree and the lengths of given paths are bounded by a constant, whereas this problem is NP-hard even for planar graphs of bounded degree. In Ref. 12, these results were further improved. We showed that the inference problem can be solved in polynomial time if graphs are outerplanar of bounded degree and bounded face size and the lengths of given paths are bounded by a constant, whereas this problem is NP-hard even for trees of bounded degree if the lengths of paths are not bounded.

In this paper, we extend our previous algorithms so that constraints on valences of atoms

are taken into account. Moreover, we modify and extend these so that feature vectors based on frequencies of small fragments can be treated. These modifications are important because major feature vectors are based on either frequencies of small fragment structures[1,2] or frequencies of labeled paths[3,4]. The modified algorithms have another application: *elucidation of chemical structures* from mass/NMR spectra data. This elucidation problem has a long history and many methods have been developed[13,14]. However, to our knowledge, no polynomial time algorithm was known for the problem. A more important result of this paper is that a heuristic algorithm is developed for inference of tree-like chemical structures. It works within a few or few-tens of seconds for inference of moderate size chemical compounds with tree or tree-like structures.

2. Problem Definitions

First, we review the definition of the problem on inference of graph from path frequency[11]. Let $G(V, E)$ be an undirected vertex-labeled connected graph and Σ be a set of vertex labels, where all results can be modified for including edge labels. Since we are considering chemical structures, we reasonably assume in this paper that the maximum degree of vertices and the size of Σ are bounded by constants. Let $\Sigma^{\leq k}$ be the set of label sequences (i.e., the set of strings) over Σ whose lengths are between 1 and k. Let $l(v)$ be the label of vertex v. For a path $P = (v_0, \ldots, v_h)$ of G, $l(P)$ denotes the label sequence of P. For graph G and label sequence t, $occ(t, G)$ denotes the number of paths P in G such that $l(P) = t$. Then, the feature vector $\boldsymbol{f}_K(G)$ of level K for $G(V, E)$ is defined by $\boldsymbol{f}_K(G) = (occ(t, G))_{t \in \Sigma^{\leq K+1}}$. See Fig. 2 for an example. It should be noted that the size (i.e., number of vertices) n of the original graph can be obtained from $\boldsymbol{f}_K(G)$. In this paper, we assume for simplicity that *tottering paths* (paths for which there exists some i such that $v_i = v_{i+2}$) are not counted in feature vectors.



$\boldsymbol{f}_1(G)$	C	N	O	H	CC	CN	CO	CH	NC	NN	• • •	OH	• • •
	5	1	2	9	8	2	2	7	2	0	• • •	1	• • •

$\boldsymbol{f}_F(G)$	C	N	O	H					• • •
	5	1	2	9	3	0	1	1	• • •

Figure 2. Examples of feature vectors $\boldsymbol{f}_K(G)$ for GIPF/CIPF ($K = 1$) and $\boldsymbol{f}_{\mathcal{F}}(G)$ for CIFF.

Graph Inference from Path Frequency (GIPF)[11] Given a feature vector $\boldsymbol{v}$ of level K, output a graph $G(V, E)$ satisfying $\boldsymbol{f}_K(G) = \boldsymbol{v}$. If there does not exist such $G(V, E)$, output "no solution".

For the case of "no solution", we can consider the problem (**GIPF-M**) of finding $G(V, E)$ which minimizes the L_1 distance between $\boldsymbol{v}$ and $\boldsymbol{f}_K(G)$ (see also Fig. 1)[11].

We sometimes omit K from $\boldsymbol{f}_K(G)$ if K is obvious or is not relevant. We may also use $\boldsymbol{f}$ to denote a feature vector if G and K are not relevant. For a vector $\boldsymbol{v}$, $(\boldsymbol{v})_i$ denotes the i-th element of $\boldsymbol{v}$ (i.e., the value of i-th coordinate of $\boldsymbol{v}$). For vectors $\boldsymbol{v}$ and $\boldsymbol{u}$, $\boldsymbol{v} \preceq \boldsymbol{u}$ means that $(\boldsymbol{v})_i \leq (\boldsymbol{u})_i$ for all i.

In order to treat chemical compounds, constraints on *valences* of atoms must be taken into account. For example, a carbon atom can be connected to at most four other atoms. If double bounds are used, it can be connected to at most two other atoms. Let Σ be the set of atom types. For each $a \in \Sigma$, the maximum valence $val(a)$ is associated. We also assume that each edge e has multiplicity $m(e)$ where $m(e)$ is usually 1 (single bond), 2 (double bond) and 3 (triple bond). When treating aromatic structures, aromatic bond may be modeled as an edge with multiplicity 1.5. In this paper, we use "degree" to mean the number of edges connected to a vertex and "valence" to mean the sum of multiplicities of edges connected to a vertex, respectively. Then, we define chemical compound inference problem from path frequency as follows:

Chemical compound Inference from Path Frequency (CIPF) Given a feature vector $\boldsymbol{v}$ of level K, output a graph $G(V, E)$ satisfying $\boldsymbol{f}_K(G) = \boldsymbol{v}$ and $\sum_{w:\{v,w\}\in E} m(\{v, w\}) \leq val(l(v))$ for all $v \in V$. If there does not exist such $G(V, E)$, output "no solution".

For the case of "no solution", **CIPF-M** is defined in the same way as in GIPF-M. Next, we define pre-image problems for feature vectors based on frequencies of fragments. Let $\mathcal{F} = \{F_1, \ldots, F_M\}$ be a set of graphs (chemical substructures) satisfying the valence conditions. Since information on the number of occurrences of each atom type is usually included in feature vectors, we assume that a graph consisting of each single atom is contained in $\mathcal{F}$. We also assume that the size of each F_i is bounded by a constant K because small fragments are usually employed. Let $occ(F_i, G)$ denote the number of subgraphs of G that are isomorphic to F_i, where we assume that subgraphs consisting of the same vertices are counted only once for each F_i. Then, a feature vector $\boldsymbol{f}_{\mathcal{F}}(G)$ for G is defined by $\boldsymbol{f}_{\mathcal{F}}(G) = (occ(F_i, G))_{F_i\in\mathcal{F}}$. The pre-iamge problem from a feature vector based on fragments is defined as below (see also Fig. 2).

Chemical compound Inference from Fragment Frequency (CIFF) Given a feature vector $\boldsymbol{v}$ based on a set of fragments $\mathcal{F}$, output a graph $G(V, E)$ satisfying $\boldsymbol{f}_{\mathcal{F}}(G) = \boldsymbol{v}$ and $\sum_{w:\{v,w\}\in E} m(\{v, w\}) \leq val(l(v))$ for all $v \in V$. If there does not exist such $G(V, E)$, output "no solution".

CIFF-M is defined in the same way as in GIPF-M and CIPF-M. In the case of elucidation of chemical structures from mass/NMR spectra[13], upper and lower bounds of the

number of occurrences of each fragment are specified. Let $\boldsymbol{ub}$ and $\boldsymbol{lb}$ be vectors corresponding to upper and lower bounds, respectively. Then, $CIULF$ is defined as:

Chemical compound Inference from Upper and Lower bounds of frequencies of Fragments (CIULF) Given feature vectors $\boldsymbol{ub}$ and $\boldsymbol{lb}$ based on a set of fragments $\mathcal{F}$, output a graph $G(V, E)$ satisfying $\boldsymbol{lb} \preceq \boldsymbol{f}_{\mathcal{F}}(G) \preceq \boldsymbol{ub}$ and $\sum_{w:\{v,w\}\in E} m(\{v, w\}) \leq val(l(v))$ for all $v \in V$. If there does not exist such $G(V, E)$, output "no solution".

It is worthy to note that CIFF is clearly a subproblem of CIULF. It should also be noted that CIPF is a subproblem of CIFF because each labeled path can be treated as a fragment.

3. Dynamic Programming Algorithms

In this section, we extend our previous algorithms[11,12] for CIPF, CIFF and CIULF.

3.1. *A Basic Algorithm for CIPF*

As in Ref. 11 we begin with a very simple case: we consider inference of chemical compounds with tree structures from a feature vector of level 1 (i.e., $K = 1$). For simplicity, we assume that only two kinds of atoms N and H, and single bonds (i.e., edges with multiplicity 1) can appear in chemical compounds. In this case, a feature vector for tree T has the following form: $\boldsymbol{f}_1(T) = (n_N, n_H, n_{NN}, n_{NH}, n_{HN}, n_{HH})$, where n_x denotes the number of atoms of type x and n_{xy} denotes the number of occurrences of a labeled path of (x, y).

We construct the dynamic programming table $D(\dots)$ defined by

$$D(n_{N1}, n_{N2}, n_{N3}, n_H, n_{NN}, n_{NH}, n_{HN}) =$$
$$\begin{cases} 1, \text{ if there exists a chemical compound (tree) } T \text{ such that} \\ \quad \boldsymbol{f}_1(T) = (n_{N1} + n_{N2} + n_{N_3}, n_H, n_{NN}, n_{NH}, n_{HN}, 0), \\ \quad \text{the number of nitrogen atoms with degree 1 is } n_{N1}, \\ \quad \text{the number of nitrogen atoms with degree 2 is } n_{N2}, \\ \quad \text{and the number of nitrogen atoms with degree 3 is } n_{N3}, \\ 0, \text{ otherwise.} \end{cases}$$

It should be noted that we ignore chemical compound of H_2 here and thus n_{HH} should be always 0. This table can be constructed by a dynamic programming procedure based the following recursion where the initialization part is straight-forward.

$$\begin{aligned} &D(n_{N1}, n_{N2}, n_{N3}, n_H, n_{NN}, n_{NH}, n_{HN}) = 1 \textbf{ iff.} \\ &\quad D(n_{N1}, n_{N2} - 1, n_{N3}, n_H, n_{NN} - 2, n_{NH}, n_{HN}) = 1 \quad \textbf{or} \\ &\quad D(n_{N1} - 1, n_{N2} + 1, n_{N3} - 1, n_H, n_{NN} - 2, n_{NH}, n_{HN}) = 1 \quad \textbf{or} \\ &\quad D(n_{N1} + 1, n_{N2} - 1, n_{N3}, n_H - 1, n_{NN}, n_{NH} - 1, n_{HN} - 1) = 1 \quad \textbf{or} \\ &\quad D(n_{N1}, n_{N2} + 1, n_{N3} - 1, n_H - 1, n_{NN}, n_{NH} - 1, n_{HN} - 1) = 1. \end{aligned}$$

The correctness of the algorithm follows from the fact that any tree can be constructed incrementally by adding a vertex (leaf) one by one. Since the value of each element of the

feature vector is $O(n)$, the table size is $O(n^7)$ and thus the computation time is $O(n^7)$. Since it is straight-forward to extend this algorithm for a fixed number of atom types and a fixed number of bond types, we have:

Theorem 3.1. *CIPF for trees is solved in polynomial time in n for $K = 1$.*

As in Ref. 11, we can modify the algorithm for CIPF-M (since we only need to examine polynomial number of items in the DP table).

Corollary 3.1. *CIPF-M for trees is solved in polynomial time in n for $K = 1$.*

Recently, Nagamochi[15] developed a much more efficient algorithm for GIPF/CIPF for general graphs and $K = 1$. But, his algorithm can not be extended for cases of $K > 1$, which are much more important. Our algorithm can be extended for cases of $K > 1$ as to be shown below, and the idea in our algorithm is also used in a practical algorithm in Sec. 4.

3.2. *Algorithm for CIPF*

In this subsection, we show that CIPF can be solved in polynomial time for fixed K. For that purpose, we modify our previous algorithm[11] for GIPF as below, where details are omitted here. As in the original algorithm, we maintain the current degrees and valences of vertices of subtrees. When adding a new leaf u to an existing vertex w in a subtree, we check the constraint on valences and update information about degrees and valences. Clearly, this part can be done in constant time per addition of a leaf and thus does not affect the order of the time complexity.

Proposition 3.1. *CIPF (and CIPF-M) for trees can be solved in polynomial time in n if K and Σ are fixed.*

3.3. *Algorithms for CIFF and CIULF*

We develop algorithms for CIFF and CIULF by modifying the algorithm for GIPF. Since CIULF is more general than CIFF, we only consider CIULF here. We modify the table $D(\boldsymbol{v}, \boldsymbol{e}, d)$ in Ref. 11 as follows. Let $\boldsymbol{h} = (h_1, h_2, \ldots, h_M)$ be a vector of non-negative integers, where h_i corresponds to the number of occurrences of fragment F_i. Then, we define the table $D'(\boldsymbol{h}, \boldsymbol{e}, d)$ by

$D'(\boldsymbol{h}, \boldsymbol{e}, d) = 1$ iff. there exists a tree T such that $\boldsymbol{f}_{\mathcal{F}}(T) = \boldsymbol{h}, \boldsymbol{g}_K(T) = \boldsymbol{e},$ and $d(T) = d$.

Based on this table, we can develop a dynamic programming algorithm, where details are omitted here.

Theorem 3.2. *CICF and CIULF (and CICF-M) for trees of bounded degree can be solved in polynomial time in n if K, M and Σ are fixed.*

As a negative result, it was shown[12] that GIPF (a subproblem of CIFF) is NP-hard for trees of bounded degree if K is not fixed. This suggests that the time complexity increases non-polynomially in K. Here, we show another hardness result for CIULF (proof omitted), which suggests that the time complexity increases non-polynomially in M.

Theorem 3.3. *CIULF can not be solved in polynomial time unless P=NP even if the maximum degree is bounded by 2, where the size and number of fragments are not bounded by a constant.*

3.4. *Extensions to Outerplanar Graphs*

Though many chemical compounds have tree structures, many other chemical compounds have rings such as benzene rings. Therefore, it is desirable to develop algorithms for more general structures than trees. In the case of GIPF, the algorithm for trees[11] was extended for outerplanar graphs[12]. The same technique can also be applied to CICF and CIFF.

Theorem 3.4. *CIPF, CIPF-M, CICF, CICF-M and CIULF for chemical compounds with outerplanar structures can be solved in polynomial time in n if K, M and Σ are fixed and the number of edges of each face is bounded by a constant.*

4. A Branch-and-Bound Type Algorithm

Though the algorithms in the previous section work in polynomial time, these are not practical. Thus, we develop a branch-and-bound type algorithm (called BB-CIPF) for CIPF, where this algorithm can be modified for inferring more general classes of chemical compounds and/or for feature vectors based on frequency of small fragments.

Before presenting BB-CIPF, we need several definitions. Let $\boldsymbol{f}^{target}$ be the given feature vector for which a pre-image should be computed. Since information on paths of length 0 is included in $\boldsymbol{f}^{target}$, we know the number of occurrences of atom types in the pre-image of $\boldsymbol{f}^{target}$. Let $atomset(\boldsymbol{f})$ be the multi-set of atom types in the pre-image of a feature vector $\boldsymbol{f}$. Let $ATOMBONDPAIRS$ be a set of possible atom-bond pairs. For example, if we only consider C, N, O, H and do not consider aromatic bonds, the set is defined as $\{(\mathrm{C},1),\ (\mathrm{C},2),\ (\mathrm{C},3),\ (\mathrm{N},1),\ (\mathrm{N},2),\ (\mathrm{N},3),\ (\mathrm{O},1),\ (\mathrm{O},2),\ (\mathrm{H},1)\}$. It should be noted that $(\mathrm{C},4)$ is not included since it is not necessary.

The basic idea of BB-CIPF is similar to that of the algorithm in Sec. 3.1: beginning from a small tree, a leaf is added one by one. Though trees are not explicitly constructed in Sec. 3.1, BB-CIPF maintains trees.

When adding a leaf u, BB-CIPF basically examines all combinations of an atom-bond pair (a, b) and a vertex w in the current tree. However, we do not need to examine the following cases, where T^{cur} be the current tree and T^{next} be the next candidate tree obtained by adding a leaf to T^{cur}:

(i) Addition of a leaf with atom label a violates the condition on the numbers of atoms,

(ii) Connection of a leaf to $w \in T^{cur}$ by bond type b violates the condition on the valence of w,

(iii) Connection of a leaf to $w \in T^{cur}$ violates the condition on feature vectors (i.e., $\boldsymbol{f}(T^{next}) \preceq \boldsymbol{f}(T^{target})$ must hold since T^{next} must be a subgraph of T^{target}).

Therefore, we do not examine T^{next} further in these cases. These conditions significantly contribute to reducing the search space and are implemented in BB-CIPF.

BB-CIPF employs a kind of distance defined by:

$$dist(\boldsymbol{f}, \boldsymbol{f}^{target}) = \begin{cases} \infty, & \text{if } (\boldsymbol{f}^{target})_i < (\boldsymbol{f})_i \text{ for some } i, \\ \sum_i c^{k(i)}((\boldsymbol{f}^{target})_i - (\boldsymbol{f})_i), & \text{otherwise.} \end{cases}$$

where summation is taken over all elements (i.e., dimensions) of feature vectors, c is a constant (currently, $c = 10$) and $k(i)$ denotes the length of a path corresponding to the i-th element of a feature vector. Though we use the word "distance" for the sake of convenience, this measure is not symmetric and thus does not satisfy the conditions on usual distances. The meaning of weighting factor $c^{k(i)}$ is that priorities are put on longer paths when calculating distances. The pseudocode of BB-CIPF is given below.

Procedure $BB-CIPF(n, \boldsymbol{f}^{target})$
  Let T^{cur} be an initial tree constructed from a longest path appearing in $\boldsymbol{f}^{target}$;
  Compute feature vector $\boldsymbol{f}^{cur}$ from T^{cur};
  if $DFS-CIPF(T^{cur}, \boldsymbol{f}^{cur}, n, \boldsymbol{f}^{target})$=false **then** output "no solution";

Procedure $DFS-CIPF(T^{cur}, \boldsymbol{f}^{cur}, n, \boldsymbol{f}^{target})$
  if $|V(T^{cur})| = n$ **then**
    if $\boldsymbol{f}^{cur} = \boldsymbol{f}^{target}$ **then** output T^{cur}; **return** true;
    else return false;
  for $(a, b) \in ATOMBONDPAIRS$ **do**
    $L \leftarrow \emptyset$;
    if $\{l(v)|v \in V(T^{cur})\} \cup \{a\} \not\subseteq atomset(\boldsymbol{f}^{target})$; (set means multiset)
    then continue (i.e., examine the next pair in $ATOMBONDPAIRS$);
    for all $w \in V(T^{cur})$ **do**
      Let T^{next} be a tree got by connecting new leaf u with label a to w by bond b;
      if w does not satisfy the valence constraint **then continue**;
      Compute $\boldsymbol{f}^{next}$ from T^{next} and $\boldsymbol{f}^{cur}$;
      $dist^{next} \leftarrow dist(\boldsymbol{f}^{next}, \boldsymbol{f}^{target})$;
      if $dist^{next} \neq \infty$ **then** Add $(T^{next}, \boldsymbol{f}^{next}, dist^{next})$ to L;
    while L is not empty **do**
      Remove $(T^{next}, \boldsymbol{f}^{next}, dist^{next})$ from L such that $dist^{next}$ is the minimum;
      if $DFS-CIPF(T^{next}, \boldsymbol{f}^{next}, T^{target}, \boldsymbol{f}^{target})$ =true **then return** true;
  return false;

The details of the implemented code are slightly different from the above. The followings are the main differences: (i) Hydrogen atoms are added at the last stage of the search procedure: hydrogen atoms are added only if the frequencies of the other atoms are the same as those in the target feature vector. (ii) L does not contain T^{next} and $\boldsymbol{f}^{next}$ (in order to save memory space). These are reconstructed just before the recursive call of DFS-CIPF. (iii) When calculating $\boldsymbol{f}^{next}$ from T^{next} and $\boldsymbol{f}^{cur}$, paths beginning from and ending at a new leaf are only computed. (iv) Benzene rings can be added as if these were leaves, where structural information on benzene is utilized for calculating feature vectors. (v) A benzene ring is given as an initial structure when a compound is small and contains a benzene ring.

It should be noted that *BB-CIPF finds an exact solution* (i.e., an exact pre-image of a given feature vector) if it exists. BB-CIPF may be modified so that it can find a kind of approximate pre-image or it can enumerate all possible pre-images.

5. Computational Experiment

We performed computational experiment on BB-CIPF in order to evaluate practical computation time. We used a PC cluster with Intel Xeon 2.8GHz CPUs working under the LINUX operating system where only one CPU was used per execution.

We examined several chemical structures by varying K. As mentioned before, BB-CIPF can handle chemical compounds with tree structures where benzene rings can also appear in structures. In the experiment, a target feature vector is computed from a target chemical compound and is given as an input for BB-CIPF (a target chemical compound is not given to BB-CIPF). Then, BB-CIPF computes a chemical structure whose feature vector coincides with the target feature vector. We examined 8 chemical compounds with $K = 1, 2, 3, 4$. CPU times are shown in Table 1. CPU time is shown with underline if the same structure as the target compound was obtained. N/A means that search did not succeed in 10 minutes.

Table 1. Computation time of BB-CIPF for various chemical compounds.

Name	Chemical Formula	CPU time (sec.)			
		$K=1$	$K=2$	$K=3$	$K=4$
Methionine	$C_5H_{11}NO_2S$	9.08	<u>0.16</u>	0.019	<u>0.002</u>
Phenylalanine	$C_9H_{11}NO_2$	0.020	<u>0.010</u>	<u>0.010</u>	<u>0.014</u>
Arginine	$C_6H_{14}N_4O_2$	N/A	500.0	<u>19.9</u>	<u>1.51</u>
Aspirin	$C_9H_8O_4$	0.060	<u>0.001</u>	<u>0.002</u>	<u>0.003</u>
2-Ethylhexyl phthalate	$C_{16}H_{22}O_4$	N/A	4.29	6.04	<u>7.88</u>
Etidocaine	$C_{17}H_{28}N_2$	N/A	N/A	N/A	<u>0.470</u>
Esatenolol	$C_{14}H_{22}N_2O_3$	N/A	N/A	25.6	<u>1.46</u>
Trimethobenzamide	$C_{21}H_{28}N_2O_5$	N/A	N/A	N/A	30.7

It is seen from the table that the computation time decreases as K increases in general. It is reasonable that pruning operations are effectively performed if longer paths are employed. It is also seen that the same structures as the target ones are inferred when larger K is used. From this table, it is suggested that the algorithm can output a solution for

moderate size chemical structures (e.g., the number of carbon atoms is less than 20) if K is 3 or 4. Though further improvements are required, the result of computational experiment suggests that it is possible to solve the pre-image problem practically.

The current implementation of BB-CIPF can only output one solution. But, there are many possible solutions especially when K is small. Therefore, it is important future work to modify BB-CIPF so that it can efficiently output all possible solutions. It is also important future work to develop a method to select the best solution from the possible solutions.

References

1. E. Byvatov, U. Fechner, J. Sadowski and G. Schneider. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of Chemical Information and Computer Sciences*, 43:1882–1889, 2003.
2. M. Deshpande, M. Kuramochi, N. Wale and G. Karypis. Frequent substructure-based approaches for classifying chemical compounds. *IEEE Trans. Knowledge and Data Engineering*, 17:1036–1050, 2005.
3. H. Kashima, K. Tsuda and A. Inokuchi. Marginalized kernels between labeled graphs. *Proc. 20th Int. Conf. Machine Learning*, 321–328, 2003.
4. P. Mahé, N. Ueda, T. Akutsu, J-L. Perret and J-P. Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of Chemical Information and Modeling*, 45:939–951, 2005.
5. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge Univ. Press, 2000.
6. G. H. Bakir, J. Weston and B. Schölkopf. Learning to find pre-images. *Advances in Neural Information Processing Systems*, 16:449–456, 2003.
7. G. H. Bakir, A. Zien and K. Tsuda. Learning to find graph pre-images. *Lecture Notes in Computer Science*, 3175:253–261, 2004.
8. T. Asano. An $O(n \log \log n)$ time algorithm for constructing a graph of maximum connectivity with prescribed degrees. *Journal of Computer and System Sciences*, 51:503–510, 1995.
9. O. Maruyama and S. Miyano. Inferring a tree from walks. *Theoretical Computer Science*, 161:289–300, 1996.
10. J. Lauri and R. Scapellato. *Topics in Graph Automorphisms and Reconstruction*. Cambridge Univ. Press, 2003.
11. T. Akutsu and D. Fukagawa. Inferring a graph from path frequency. *Lecture Notes in Computer Science*, 3537:371–392, 2005.
12. T. Akutsu and D. Fukagawa. On inference of a chemical structure from path frequency. *Proc. BIOINFO 2005*, 96–100, 2005.
13. K. Funatsu and S. Sasaki. Recent advances in the automated structure elucidation system, CHEMICS. Utilization of two-dimensional NMR spectral information and development of peripheral functions for examination of candidates. *Journal of Chemical Information and Computer Sciences*, 36:190–204, 1996.
14. A. Korytko, K-P. Schulz, M. S. Madison and M. E. Munk. HOUDINI: a new approach to computer-based structure generation. *Journal of Chemical Information and Computer Sciences*, 43:1434–1446, 2003.
15. H. Nagamochi. A detachment algorithm for inferring a graph from path frequency. *Lecture Notes in Computer Science*, 4112:274–283, 2006.