Enumerating Naphthalene Isomers of Tree-like Chemical Graphs

Fei He[1], Akiyoshi Hanai[1], Hiroshi Nagamochi[1] and Tatsuya Akutsu[2]
[1] *Discrete mathematics, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*
[2] *Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto, 611-0011, Japan*

Keywords: Isomers Enumeration, Naphthalene, Chemical Graph.

Abstract: In this paper, we consider the problem of enumerating naphthalene isomers, where enumeration of isomers is important for drug design. A chemical graph G with no other cycles than naphthalene rings is called *tree-like*, and becomes a tree T possibly with multiple edges if we contract each naphthalene ring into a single virtual atom of valence 8. We call tree T the *tree representation* of G. There may be more than one tree-like chemical graphs whose tree representations equal to T, which are called *naphthalene isomers* of T. We present an efficient algorithm that enumerates all naphthalene isomers of a given tree representation. Our algorithm first counts the number of all the naphthalene isomers using dynamic programming, and then for each k, generates the k-th isomer by backtracking the counting computation. In computational experiment, we compare our method with MolGen, a state-of-the-art enumeration tool, and it is observed that our program enumerates the same number of naphthalene isomers within extremely shorter time, which proves that our algorithm is effectively built.

1 INTRODUCTION

Enumeration of chemical structures has been widely applied in drug discovery (Blum and Reymond, 2009), structure elucidation (Meringer and Schymanski, 2013), exploration of chemical universe (Fink and Reymond, 2007). It is to be noted that these are important not only in chemo-informatics but also in bioinformatics because one of the major targets of bioinformatics is development of novel drugs. Since DENDRAL was developed as an enumeration tool for structure elucidation using data from mass spectrometry (Buchanan and Djerassi, 1976), several enumeration tools have been developed to solve the Computer-Assisted Structure Elucidation (CASE) problem. MolGen, whose development was initiated in 1985, is one of the best enumeration tools so far, since it can not only enumerate all possible chemical compounds (Benecke and Wieland, 1995) (Benecke and Wieland, 1997) (Kerber and Meringer, 1998), but also allow users to add some restrictions to the input of the enumeration such as the number of multiple bonds. Recently, OMG (Open Molecule Generator) has been newly developed as the first open source enumeration tool (Peironcely, 2012) based on the canonical augmentation path strategy which grows intermediate chemical structures by adding bonds with checking the uniqueness, with allowing the use of prescribed substructures to enumerate more chemically adequate structures. Although OMG is a useful tool, it is reported that OMG is not faster than MolGen (Peironcely, 2012), especially when dealing with large-sized chemical graphs.

Another important enumeration tool is Enumol (Fujiwara and Akutsu, 2008) (Shimizu and Akutsu, 2011). This tool enumerates chemical structures with tree-like graphs efficiently. Akutsu and Nagamochi initiated the development of Enumol after studying computational complexity of inference of a chemical graph from its feature vector given as a labeled path frequency vector (Akutsu and Nagamochi, 2011). Then, Fujiwara et al. (Fujiwara and Akutsu, 2008) proposed a branch-and-bound algorithm to enumerate tree-like chemical graphs from a given path frequency vector. Shimizu et al. (Shimizu and Akutsu, 2011) gave an algorithm for enumerating structures with a set of feature vectors. These algorithms can only enumerate tree-like chemical graphs.

A benzene is a chemical compound with the molecular formula C_6H_6 and the six carbon atoms in a benzene form into a hexagon. A naphthalene is a compound with the molecular formula $C_{10}H_8$, and its molecule is composed of two benzene rings with

He, F., Hanai, A., Nagamochi, H. and Akutsu, T.
Enumerating Naphthalene Isomers of Tree-like Chemical Graphs.
DOI: 10.5220/0005783902580265
In *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2016) - Volume 3: BIOINFORMATICS*, pages 258-265
ISBN: 978-989-758-170-0

one common edge. Among all ten carbon atoms in a naphthalene ring, only eight of them are attached with hydrogen atoms. A substitution of a proton by other atoms (or atom groups) is one of the most common reactions of naphthalene. In these substitution reactions, such as sulfonation, chlorination and nitration, the hydrogen atoms of a naphthalene ring is substituted by other atoms (or atom groups). Thereby a naphthalene ring may have bonds with other atoms (or atom groups) (Hardinger, 2005). The difference of substitutions in the relative positions of the substituted hydrogen atoms results in non-isomorphic structures, which are called the naphthalene isomers. Given a tree-like chemical graph, Li (Li and Akutsu, 2013) proposed a different method with a guaranteed time complexity to enumerate the benzene isomers by the two following steps: (1) count the number f of all the benzene isomers of the input tree representation using dynamic programming; and (2) for each $k = 1, 2, \ldots, f$, generate the k-th benzene isomer by backtracking the previous computation using the dynamic programming. In this paper, we generalize the algorithm by Li (Li and Akutsu, 2013) into one for enumerating the naphthalene isomers from a tree-like chemical graphs. The major difference is that the combinatorial complexity of arrangements of atom groups around a naphthalene ring is much higher than that around a benzene ring. Yet, we managed to generate all isomers in a running time per isomer similar with the result by Li (Li and Akutsu, 2013) by generating all distinct arrangements of atom groups around a naphthalene ring in advance and storing them in a table. As discussed later, it is an important step towards extension of enumeration of benzene isomers to general chemical structures. Our experimental result shows that our algorithm runs much faster than MolGen for enumeration of naphthalene isomers of tree-like chemical compounds.

In this paper, we consider chemical graphs with naphthalene rings. A chemical graph G possibly with naphthalene rings is called *tree-like* if (i) no two naphthalene rings share any atoms; (ii) no multiple edges exist between a naphthalene ring and an atom (or a naphthalene ring); and (iii) the graph can be viewed as a multi-tree (a tree with multiple edges) if we contract each naphthalene ring into a single vertex of an atom of a virtual element `na` of valence 8. We call the tree in the above (iii) the *tree representation* of G. Note that for a given tree representation T, there may be more than one tree-like chemical graphs whose tree representations are given by T. All tree-like chemical graphs whose tree representations equal to T are called the *naphthalene isomers* of T. Distinct naphthalene isomers of T are caused by different arrangements of the atom groups around a naphthalene ring when we replace each virtual atom in T with a naphthalene ring to obtain a naphthalene isomer.

Given a set of atoms of each kind (including the virtual atom `na`) possibly with constraints on the numbers of specified paths, all possible tree-like chemical graphs can be enumerated by efficient algorithms created by Fujiwara et al. (Fujiwara and Akutsu, 2008) and Shimizu et al. (Shimizu and Akutsu, 2011). When we replace each virtual atom in a tree representation with a naphthalene ring, different chemical structures may be generated due to different relative positions of the substituents (atom groups) around each of the restored naphthalene rings. Let $\mathcal{G}(T)$ denote the set of all naphthalene isomers of a tree representation T. In this paper, we consider the following problem of enumerating naphthalene isomers.

Input: A tree representation T of a tree-like chemical graph.
Output: All naphthalene isomers $G \in \mathcal{G}(T)$ of T.

In this paper, we design an efficient algorithm that enumerates all the naphthalene isomers $G \in \mathcal{G}(T)$ of a given tree representation T. We use dynamic programming to first count the number $|\mathcal{G}(T)|$ of all the naphthalene isomers of T, and then generate all naphthalene isomers by tracing back the computation process done for counting the total number $|\mathcal{G}(T)|$ of naphthalene isomers. Thus our enumeration algorithm consists of the following two phases.

Phase 1: Count the number $|\mathcal{G}(T)|$ of the naphthalene isomers of a tree representation $T(v_0)$ by a dynamic programming.
Phase 2: For each $k = 1, 2, \ldots, |\mathcal{G}(T)|$, generate the k-th isomer $G_k \in \mathcal{G}(T)$ by a procedure of backtracking the computation in Phase 1.

The paper is organized as follows. Section 2 defines isomorphism of rooted graphs. Section 3 designs a dynamic programming algorithm for counting the number of isomers of rooted subtrees and that of T. Section 4 presents an algorithm for generating the k-th isomer among all isomers of T. Section 5 reports on experiments to an implementation of our algorithm for counting and generating isomers. Section 6 makes some concluding remarks.

2 PRELIMINARIES

We introduce isomorphism in tree-like graphs.

A chemical graph is given by a connected undirected graph $G = (V, E)$ with a set V of vertices and a set E of edges possibly with multiple edges, where

$\mathrm{col}(v)$ denotes the atom assigned to each vertex $v \in V$. For a subgraph H of G, let $V(H)$ and $E(H)$ denote the sets of vertices and edges in H. For a vertex subset X or a subgraph X of G, let $N_G(X)$ denote the set of neighbors of X (the vertices in $V-X$ or $V-V(X)$ adjacent to some vertex in X). A vertex v with $\mathrm{col}(v)=\ell$ is called an ℓ-vertex. Let $E_G(u,v)$ denote the set of edges between vertices u and v in G. The set $E_G(u,v)$ is called a *bridge* if the graph becomes disconnected by removing edges in $E_G(u,v)$. Two graphs $G=(V,E)$ and $G'=(V',E')$ are *isomorphic* if there is an isomorphism between them, i.e., a bijection $\psi : V \to V'$ such that $\mathrm{col}(v)=\mathrm{col}(\psi(v))$ holds for each vertex $v \in V$ and $|E_{G'}(\psi(u),\psi(v))| = |E_G(u,v)|$ for all vertex pairs $u,v \in V$. When two graphs $G=(V,E)$ and $G'=(V',E')$ have designated vertices $r \in V$ and $r' \in V'$ called the roots, we say that $G=(V,E)$ and $G'=(V',E')$ are *rooted-isomorphic* if there is an isomorphism $\psi : V \to V'$ such that $r'=\psi(r)$.

Let G be a tree-like chemical graph. A naphthalene ring H is a cycle of ten C-vertices eight of which has a neighbor in $N_G(H)$ joined by a single edge. For an ordered pair (v,w) such that $E_G(v,w)$ is bridge, we denote $G[v,w]$ the subgraph induced from G by the vertex subset $\{v\} \cup X_w$ such that X_w is the set of vertices reachable from vertex w by a path that does not pass through vertex v. We regard v as the root of $G[v,w]$.

For each vertex v not on a naphthalene ring in G, we define a *coterie* of the set $\{G[v,w] \mid w \in N_G(v)\}$ of subgraphs to be a maximal subset of $\{G[v,w] \mid w \in N_G(v)\}$ such that any two in the subset are rooted-isomorphic. Hence the set $\{G[v,w] \mid w \in N_G(v)\}$ of subgraphs is partitioned into several sets of coteries, which is called the *coterie-family* of G at vertex v. Fig. 1 shows the coterie-family $\{G[v,w_1],G[v,w_2]\},\{G[v,w_3]\}$ and $\{G[v,w_4]\}$ for the subgraphs of root v of graph G, where $G[v,w_i]$ denotes the subgraph composed of root v and its adjacent vertices w_i for each $i=1,2,3,4$ respectively.

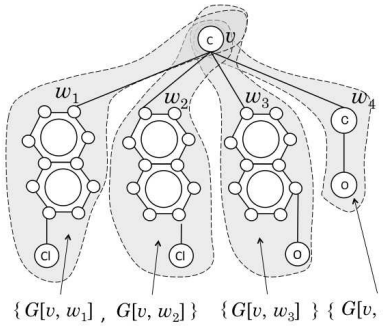

Figure 1: An example of coterie-family with three coteries.

For a naphthalene ring H with C-vertices $v_1,v_2,\ldots,v_8$ in G and $N_G(H)=\{w_1,w_2,\ldots,w_8\}$ where $w_i \in N_G(v_i)$, each C-vertex $v_i \in V(H)$ is contained in the subgraph $G[v_i,w_i]$. We define a *coterie* of the set $\{G[v_i,w_i] \mid v_i \in V(H)\}$ of subgraphs to be a maximal subset of $\{G[v_i,w_i] \mid v_i \in V(H)\}$ such that any two in the subset are rooted-isomorphic. Hence the set $\{G[v_i,w_i] \mid v_i \in V(H)\}$ of subgraphs is partitioned into several sets of coteries, which is called the *coterie-family* of G at naphthalene ring H.

The *size* of a coterie is defined to be the number of subgraphs that belong to the coterie. Among coteries with the same size, we introduce some total order, and we always denote by R_i^j the subgraph that belong to the j-th coterie of size i. Fig. 2 illustrates an example of coterie-family at a vertex v with a coterie with size 1, a coterie with size 3 and a coterie with size 4.

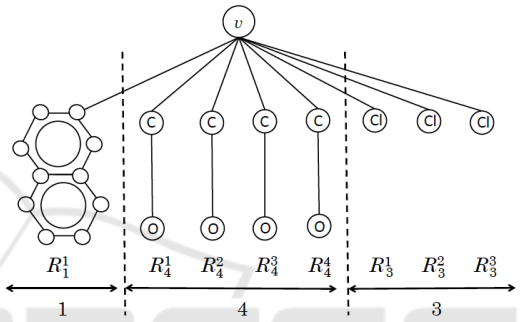

Figure 2: An example for a coterie-family at vertex v in a tree T where the cf-type at vertex v is $\mathbf{c}=(1,0,1,1,0,0,0,0)$.

A *coterie-family-type* (cf-type) of G at a vertex v or a naphthalene ring H is defined to be the set of the number of coteries of each size, and is denoted by the *occurrence vector* $\mathbf{c}$ whose i-th entry indicates the number of coteries of size i. In Fig. 2, $\mathbf{c}=(1,0,1,1,0,0,0,0)$.

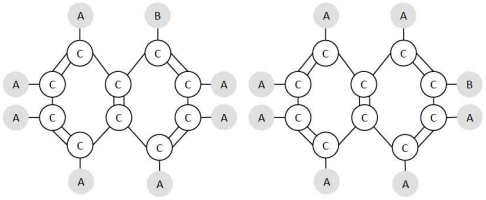

Figure 3: The two layout-types in $\mathcal{L}(\mathbf{c})$ of cf-type $\mathbf{c}=(1,0,0,0,0,0,1,0)$.

For a fixed cf-type $\mathbf{c}$ of G at a naphthalene ring H, there may be different ways of placing the subgraphs in $\{G[v_i,w_i] \mid v_i \in V(H)\}$ such that the resulting graphs are not isomorphic. The set $\mathcal{L}(\mathbf{c})$ of *layout-types* consists of such different ways of placing the subgraphs in $\{G[v_i,w_i] \mid v_i \in V(H)\}$. For example, cf-type $\mathbf{c}=(1,0,0,0,0,0,1,0)$ has two layout-types in $\mathcal{L}(\mathbf{c})$, which is illustrated in Fig. 3. Let $\lambda(\mathbf{c})=|\mathcal{L}(\mathbf{c})|$.

Table 1 shows the number $\lambda(\mathbf{c})$ of layout types for each possible cf-type $\mathbf{c}$ of a naphthalene ring. Among the layout-types in $\mathcal{L}(\mathbf{c})$ for each cf-type $\mathbf{c}$, we introduce some total order, and denote by $L_i(\mathbf{c})$ the i-th layout-type in $\mathcal{L}(\mathbf{c})$. In this paper, we assume that a complete list of all layout types of each cf-type for a naphthalene ring is available, where the total number of layout types is 23,911. An efficient method for generating all layout types of a geometrical object with two axial symmetries including naphthalene ring has been discussed by He and Nagamochi (He and Nagamochi, 2015).

Table 1: The number $\lambda(\mathbf{c})$ of layout types for each possible cf-type $\mathbf{c}$ of a naphthalene ring.

$\mathbf{c}$	$\lambda(\mathbf{c})$	$\mathbf{c}$	$\lambda(\mathbf{c})$
(0,0,0,0,0,0,0,1)	1	(1,0,0,0,0,0,1,0)	2
(0,1,0,0,0,1,0,0)	10	(2,0,0,0,0,1,0,0)	14
(0,0,1,0,1,0,0,0)	14	(1,1,0,0,1,0,0,0)	42
(3,0,0,0,1,0,0,0)	84	(0,0,0,2,0,0,0,0)	22
(1,0,1,1,0,0,0,0)	70	(0,2,0,1,0,0,0,0)	114
(2,1,0,1,0,0,0,0)	210	(4,0,0,1,0,0,0,0)	420
(0,1,2,0,0,0,0,0)	140	(2,0,2,0,0,0,0,0)	280
(1,2,1,0,0,0,0,0)	420	(3,1,1,0,0,0,0,0)	840
(5,0,1,0,0,0,0,0)	1680	(0,4,0,0,0,0,0,0)	648
(2,3,0,0,0,0,0,0)	1260	(4,2,0,0,0,0,0,0)	2520
(6,1,0,0,0,0,0,0)	5040	(8,0,0,0,0,0,0,0)	10080

In the rest of this section, we discuss the structure of isomorphism in tree representations.

Let T be a tree representation with virtual na atoms of valence 8, which we restore naphthalene rings to obtain a naphthalene isomer of T. Note that $E_T(v,w)$ is a bridge of T for each pair of adjacent vertices $v,w \in V(T)$, and $T[v,w]$ is the induced subtree of T rooted at v such that $V(T[v,w])$ consists of v and the vertices reachable from w without visiting v. Every vertex v in T is shared by $|N_T(v)|$ subtrees $T[v,w]$ with $w \in N_T(v)$. For a subtree $T' = T[v,w]$ for vertices $v \in V$ and $w \in N(v)$ in T, *restoration* of T' is an operation of expanding the na-vertices z $(\neq v)$ in T' into naphthalene rings H_z to obtain a chemical graph G_v rooted at v, where v may be a na-vertex. Let $\mathcal{G}(T')$ be the set of tree-like graphs G_v obtained by restoring a subtree T' of T.

Our aim is to generate all isomers in $\mathcal{G}(T)$ from a given tree representation T, supposing that we know the cf-type $\mathbf{c}_v$ of the coterie-family for the set $\{G[v_i,w_i] \mid v_i \in V(H_v),\ w_i \in N_G(v_i)\}$ of subgraphs around the naphthalene ring H_v restored from each na-vertex v in T. Since distinct isomers are caused by different layout-types around naphthalene rings after restoration of T, an isomer $G \in \mathcal{G}(T)$ can be specified by a layout-type $L[v] \in \mathcal{L}(\mathbf{c}_v)$ of each na-vertex v in T. Throughout the paper, we assume that non-rooted-isomorphic subgraphs in $\{G[v_i,w_i] \mid v_i \in V(H_v),\ w_i \in N_G(v_i)\}$ can be arranged in some total order based on their topological structure, and we assign indices $R_i^1, R_i^2, \ldots$ to the coteries with the same size i according to the order so that choosing a layout-type in $\mathcal{L}(\mathbf{c}_v)$ uniquely determines the relative positions of all the subgraphs in $\{G[v_i,w_i] \mid v_i \in V(H_v),\ w_i \in N_G(v_i)\}$ around the naphthalene ring H_v. However, when we change a layout-type $L[v] \in \mathcal{L}(\mathbf{c}_v)$ for some na-vertex v in T, the cf-type $\mathbf{c}_u$ of some other na-vertex u may change. To avoid such inconvenience in generating all isomers in $\mathcal{G}(T)$, we designate a vertex in T as the root and systematically compute the number of isomers of some subtrees of T by a dynamic programming, where all isomers of T will be generated by backtracking the computation.

Let $\mathcal{G}(T[v,w])$ denote the set of isomers obtained from the tree $T[v,w]$ by restoring all na-vertices z $(\neq v)$ in $T[v,w]$ into naphthalene rings H_z. Clearly if $T[v,w]$ and $T[v,w']$ are rooted-isomorphic for $w,w' \in N_T(v)$, then $\mathcal{G}(T[v,w]) = \mathcal{G}(T[v,w'])$. Conversely $T[v,w]$ and $T[v,w']$ are not rooted-isomorphic for $w,w' \in N_T(v)$, then $\mathcal{G}(T[v,w]) \cap \mathcal{G}(T[v,w']) = \emptyset$.

Let us treat a given tree representation T as a tree rooted at a vertex v_0, where the root is chosen as a topologically unique vertex in T such as "center" or "centroid." It is known that every tree has either a vertex or an edge removal of which leaves no component with more than $\lfloor n/2 \rfloor$ vertices. The former vertex is called *unicentroid* and the latter *bicentroid*. We choose the unicentroid of T as the root v_0 of T, where if T has the bicentroid (v',v''), then we remove the edge connecting v and v' and insert a virtual vertex v with the two neighbors v' and v'' as the unicentroid in the resulting tree. For each vertex v in a rooted tree representation $T = (V,E)$, let $\mathrm{Ch}(v) \subseteq N_T(v)$ denote the set of children of v, where $\mathrm{Ch}(v) = \emptyset$ if v is a leaf vertex of T. When v is not the root of T, let $\mathrm{p}(v) \in N_T(v)$ denote the parent of v. By the topological uniqueness of the root v_0, we see that for each non-root vertex v, tree $T[v,w]$ with $w \in \mathrm{Ch}(v)$ is not rooted-isomorphic to tree $T[v,\mathrm{p}(v)]$.

The set $\mathrm{Ch}(v)$ of children of a vertex v in T is decomposed into sets $\mathrm{Ch}_1(v), \mathrm{Ch}_2(v), \ldots, \mathrm{Ch}_t(v)$ such that the subtrees $T[v,u]$ and $T[v,u']$ are rooted-isomorphic if and only if $u,u' \in \mathrm{Ch}_i(v)$ for some i. Fig. 4 illustrates the rooted-isomorphic decompositions of child sets of a tree presentation T. Note that for a given tree presentation T rooted at a vertex v_0, we can compute the rooted-isomorphic decomposition $\{\mathrm{Ch}_i(v) \mid i = 1,2,\ldots,t_v\}$ for all vertices $v \in V$ in polynomial time in the size of T by expressing all subtrees $T[v,u]$, $u \in \mathrm{Ch}(v)$ for each $v \in V(T)$ as canoni-

cal forms such as left-heavy trees (Nakano and Uno, 2005).

In what follows, we assume that an input of the problem is the set

$$(T, v_0, \{\{\mathrm{Ch}_i(v) \mid i = 1,2,\ldots,t_v\}, v \in V\})$$

of a tree representation T rooted at the unicentroid v_0 and the rooted-isomorphic decompositions of child sets of T.

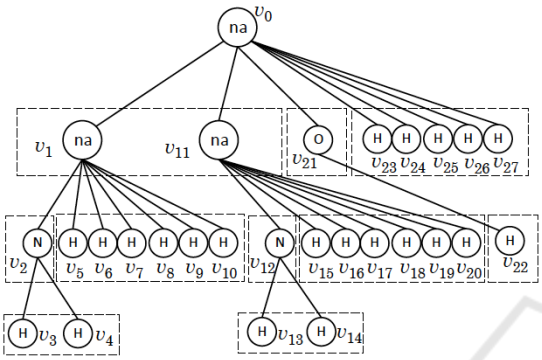

Figure 4: The rooted-isomorphic decompositions of child sets of tree representation T, where the dotted rectangles show the rooted-isomorphic decompositions for all non-leaf vertices in T.

3 COUNTING THE NUMBER OF ISOMERS

Let $\mathrm{nis}(v_0) = |\mathcal{G}(T)|$ for the root v_0 of T, and $\mathrm{nis}(v) = |\mathcal{G}(T[\mathrm{p}(v), v])|$ for each non-root vertex v in T. We derive recursive formulas over $\{\mathrm{nis}(v) \mid v \in V(T)\}$ such that $\mathrm{nis}(v)$ can be computed from $\{\mathrm{nis}(w) \mid w \in \mathrm{Ch}(v)\}$.

Example 1. Before we present details, we show how the numbers $\mathrm{nis}(v)$, $v \in V(T)$ of the example T in Fig. 4 will be computed.

1. Clearly for the leaf vertices $v \in V(T) - \{v_0, v_1, v_2, v_{11}, v_{12}, v_{21}\}$ in the example T, we have that $\mathrm{nis}(v) = 1$.

2. For the non-na-vertex v_2, the rooted-isomorphic decomposition of the child set $\mathrm{Ch}(v_2)$ is $\mathrm{Ch}_1(v_2) = \{v_3, v_4\}$ since $T[v_2, v_3]$ and $T[v_2, v_4]$ are rooted-isomorphic, where $\mathrm{nis}(v_3) = \mathrm{nis}(v_4) = 1$, and we know that $\mathcal{G}(T[\mathrm{p}(v_2), v_2])$ contains only one isomer, implying that $\mathrm{nis}(v_2) = 1$.

3. We next consider the na-vertex v_1 in the example T. The rooted-isomorphic decomposition of the child set $\mathrm{Ch}(v_1)$ is given by $\mathrm{Ch}_1(v_1) = \{v_2\}$ and $\mathrm{Ch}_2(v_1) = \{v_5, v_6, v_7, v_8, v_9, v_{10}\}$ where we know that $\mathrm{nis}(v_2) = 1$ and $\mathrm{nis}(w) = 1$, $w \in \mathrm{Ch}_2(v_1)$. For any vertex $w \in \mathrm{Ch}(v_1)$, there is only one isomer in $\mathcal{G}(T[\mathrm{p}(w), w])$. The corresponding cf-type $\mathbf{c}_{v_1}$=(2,0,0,0,0,1,0,0) has $\lambda(\mathbf{c}_{v_1}) = 14$ layout-types (see Table 1). Therefore we have $\mathrm{nis}(v_1) = 14$.

4. We now consider the root na-vertex v_0, where the rooted-isomorphic decomposition of $\mathrm{Ch}(v_0)$ is $\mathrm{Ch}_1(v_0) = \{v_1, v_{11}\}\mathrm{Ch}_2(v_1) = \{v_{21}\}$ and $\mathrm{Ch}_3(v_1) = \{v_{23}, v_{24}, v_{25}, v_{26}, v_{27}\}$ and we see that $\mathrm{nis}(w) = 14$, $w \in \mathrm{Ch}_1(v_0)$, $\mathrm{nis}(w) = 1$, $w \in \mathrm{Ch}_2(v_0)$ and $\mathrm{nis}(w) = 1$, $w \in \mathrm{Ch}_3(v_0)$, respectively. For $\mathrm{nis}(w) = 14, w \in \mathrm{Ch}_1(v_0)$ and $|\mathrm{Ch}_1(v_0)| = 2$, there are two different coterie-families at v_0: i) one when $G' \in \mathcal{G}(T[v_0, v_1])$ and $G'' \in \mathcal{G}(T[v_0, v_{11}])$ are rooted-isomorphic; and ii) the other when they are not isomorphic.

i) The former cf-type is $\mathbf{c}_{v_0} = (1,1,0,0,1,0,0,0)$, where $R_1^1 \in \mathcal{G}(T[v_0, v_1])$, $R_2^1 \in \mathcal{G}(T[v_0, v_{21}])$ and $R_3^1 \in \mathcal{G}(T[v_0, v_{23}])$. Note that there are $\mathrm{nis}(v_1) \times \mathrm{nis}(v_{21}) \times \mathrm{nis}(v_{23}) = 14 \times 1 \times 1 = 14$ combinations of isomers in $\mathcal{G}(T[v_0, w])$ with $w \in \mathrm{Ch}(v_0)$ that give cf-type $\mathbf{c}_{v_0} = (1,1,0,0,1,0,0,0)$ at v_0. Since there are $\lambda(\mathbf{c}_{v_0}) = 42$ layout-types (see Table 1), we have $14 \times 42 = 588$ isomers in $\mathcal{G}(T)$ whose cf-type at v_0 is $(1,1,0,0,1,0,0,0)$.

ii) For the latter cf-type is $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$, where $R_1^1 \in \mathcal{G}(T[v_0, v_1])R_1^2 \in \mathcal{G}(T[v_0, v_{11}])R_2^1 \in \mathcal{G}(T[v_0, v_{21}])$ and $R_3^1 \in \mathcal{G}(T[v_0, v_{23}])$. Note that there are $\binom{\mathrm{nis}(v_1)}{2} \times \mathrm{nis}(v_{21}) \times \mathrm{nis}(v_{23}) = 91 \times 1 \times 1 = 91$ combinations of isomers in $\mathcal{G}(T[v_0, w])$ with $w \in \mathrm{Ch}(v_0)$ for cf-type $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$ at v_0. Since there are $\lambda(\mathbf{c}_{v_0}) = 84$ layout-types (see Table 1), we have $84 \times 91 = 7644$ isomers in $\mathcal{G}(T)$ whose cf-type at v_0 is $(3,0,0,0,1,0,0,0)$.

In total, the number $\mathrm{nis}(v_0)$ of isomers in $\mathcal{G}(T)$ is $588 + 7644 = 8232$.

In what follows, we generalize the above idea into recursive formulas from which we can systematically compute the number $\mathrm{nis}(v)$ from $\mathrm{nis}(w), w \in \mathrm{Ch}(v)$. The point is how each subset $\mathrm{Ch}_i(v)$ in the rooted-isomorphic decomposition of $\mathrm{Ch}(v)$ can be further partitioned into coteries. Different partitions of subset $\mathrm{Ch}_i(v)$, $i = 1,2,\ldots,t$ give rise to distinct cf-types. Let $n_i = |\mathrm{Ch}_i(v)|$ and f_i denote the number $\mathrm{nis}(w) = |\mathcal{G}(T[v, w])|$ of isomers of the subtree $T[v, w]$ rooted at a vertex $w \in \mathrm{Ch}_i(v)$. Denote the number of coteries partitioned from $\mathrm{Ch}_i(v)$ by h_i. Then $1 \le h_i \le \min\{n_i, f_i\}$ for each $i \in \{1,2,\ldots,t\}$. Let $\mathcal{H}_v$ be the set of all possible vectors, the i-th entrance of which is the number of coteries in $\mathrm{Ch}_i(v)$, then $\mathcal{H}_v = \{(h_1, h_2, \ldots, h_t) \mid 1 \le h_i \le \min\{n_i, f_i\}, i = 1,2,\ldots,t\}$, where $|\mathcal{H}_v| = \prod_{1 \le i \le t} \min\{n_i, f_i\}$.

For a subtree $T' \subseteq T$ rooted at v and the rooted-isomorphic decomposition $\{\mathrm{Ch}_1, \ldots, \mathrm{Ch}_{t_v}\}$, we define a subset $\mathcal{G}_{\mathbf{h}}(T')$ of $\mathcal{G}(T')$, $\mathbf{h} \in \mathcal{H}_v$ to be the

set of restored graphs of T' whose number of coteries in $\mathrm{Ch}_i(v)$ equals to h_i for each $i \in \{1,2,\ldots,t_v\}$. Let $g(\mathbf{h}) = |\mathcal{G}_{\mathbf{h}}(T')|$. For any two different vectors $\mathbf{h}_1, \mathbf{h}_2 \in \mathcal{H}_v$, restored subgraphs $G_{\mathbf{h}_1}(T') \subseteq \mathcal{G}_{\mathbf{h}_1}(T')$ and $G_{\mathbf{h}_2}(T') \subseteq \mathcal{G}_{\mathbf{h}_2}(T')$ are always non-isomorphic. Then we have the equation as follows.

$$\mathrm{nis}(v) = \sum_{\mathbf{h} \in \mathcal{H}_v} g(\mathbf{h}). \tag{1}$$

A specific form of function $g(\mathbf{h})$ of $\mathbf{h} \in \mathcal{H}_v$ can be derived in a similar manner with the method by Li et al. (Li and Akutsu, 2013). Due to a space limitation, we here omit deriving the form of function $g(\mathbf{h})$, which takes a different form in each of the following three cases:

1. Vertex v is a non-na vertex.
2. Vertex v is a non-root na-vertex.
3. Vertex v is a root na-vertex.

We analyze the time complexity of our algorithm. Since the valence of any atom is constant and $|N(H)| = 8$ for a naphthalene ring, the degree $|N_T(v)|$ of any vertex v in T is $O(1)$. Hence $|\mathcal{H}_v|$ is also $O(1)$, and computing $\{g(\mathbf{h}) \mid \mathbf{h} \in \mathcal{H}_v\}$ and determining $\mathrm{nis}(v)$ can be executed in $O(1)$ time (Li and Akutsu, 2013). Therefore the total running time of our algorithm is $O(n)$ for the input size $n = |T(V)|$.

4 CONSTRUCTION OF NAPHTHALENE ISOMERS

In the first phase of our algorithm, given a tree representation T, we compute $\mathrm{nis}(v)$ and $\{g(\mathbf{h}) \mid \mathbf{h} \in \mathcal{H}_v\}$ for all vertices $v \in V(T)$. In the second phase of our algorithm, we generate isomers in $\mathcal{G}(T)$ one by one.

For this, we fix a total order $\sigma_{v_0} : \mathcal{G}(T) \to [1, \mathrm{nis}(v_0)]$ over all isomers in $\mathcal{G}(T)$ such that $\sigma_{v_0}(G) = k$ means that G is the k-th isomers in $\mathcal{G}(T)$ in the total order. Similarly for each non-root vertex v in T, we fix a total order $\sigma_v : \mathcal{G}(T[\mathrm{p}(v), v]) \to [1, \mathrm{nis}(v)]$ over all isomers in $\mathcal{G}(T[\mathrm{p}(v), v])$ such that $\sigma_v(G) = j$ means that G is the j-th isomers in $\mathcal{G}(T[\mathrm{p}(v), v])$. Since the way of fixing total orders σ_v is same during the second phase, it holds $\sigma_u = \sigma_v$ for any two vertices $u, v \in V(T)$ such that $T[\mathrm{p}(u), u]$ and $T[\mathrm{p}(v), v]$ are rooted-isomorphic. Hence the k-th isomer $G_u \in \mathcal{G}(T[\mathrm{p}(u), u])$ and the k'-th isomer $G_v \in \mathcal{G}(T[\mathrm{p}(v), v])$ will be rooted-isomorphic if and only if $k = k'$.

Supposing such fixed total orders $\{\sigma_v \mid v \in V(T)\}$, we generate the k-isomer $G \in \mathcal{G}(T)$ of T (i.e., the isomer G such that $\sigma_{v_0}(G) = k$) for a specified integer $k \in [1, \mathrm{nis}(v_0)]$, where we call k the *search index* of vertex v. For the child set $\mathrm{Ch}(v) = \{w_1, w_2, \ldots, w_q\}$, the isomer $G \in \mathcal{G}(T)$ consists of q subgraphs $G_{w_1}, G_{w_2}, \ldots, G_{w_q}$ such that $G_{w_i} \in \mathcal{G}(T[v, w_i])$. We let $k[w_i] = \sigma_{w_i}(G_{w_i})$, i.e., G_{w_i} is the $k[w_i]$-th isomer in $\mathcal{G}(T[v, w_i])$. Hence we see that the isomer $G \in \mathcal{G}(T)$ consists of $k[w_i]$-th isomer in $\mathcal{G}(T[v, w_i])$ for each child $w_i \in \mathrm{Ch}(v)$, where we call $k[w_i]$ the *search index* of vertex w_i. When v is a na-vertex, we also identify the layout-type $L[v]$ around the naphthalene ring H_v.

In the rest of this section, we show how to compute search indices.

An isomer $G \in \mathcal{G}(T)$ can be determined by choices of cf-types c_z and layout-types $L[z] \in \mathcal{L}(\mathrm{c}_z)$ for all naphthalene rings H_z stored from the na-vertices z in T. In fact, we see that only the set $\{L[z] \mid \text{na-vertices } z \text{ in } T\}$ determines a cf-type c_z of each naphthalene ring H_z. Each leaf vertex v in T has a unique cf-type c_v. Recursively when an isomer $G_w \in \mathcal{G}(T[v, w])$ of each child $w \in \mathrm{Ch}(v)$ of a vertex v has been determined, the cf-type c_v of vertex v can be uniquely determined, and the isomer $G_v \in \mathcal{G}(T[\mathrm{p}(v), v])$ (or $G_v \in \mathcal{G}(T)$) consisting of these subgraphs $G_w \in \mathcal{G}(T[v, w])$, $w \in \mathrm{Ch}(v)$ can be uniquely determined with a specified layout-type when v is a na-vertex (or G_v can be uniquely determined only by G_w, $w \in \mathrm{Ch}(v)$ when v is not a na-vertex). Our task is to output a set $\{L[z] \mid \text{na-vertices } z \text{ in } T\}$ of layout-types as an isomer $G \in \mathcal{G}(T)$.

Example 2. We show how search indices and layout-types will be computed in the example T in Fig. 4

1. After counting phase, we have computed $\{g(\mathbf{h}) \mid \mathcal{H}_v\}$ for all non-leaf vertices v. In the example, $\mathcal{H}_{v_0} = \{\mathbf{h}_1 = (1,1,1), \mathbf{h}_2 = (2,1,1)\}$ with $g(\mathbf{h}_1) = 588$ and $g(\mathbf{h}_2) = 7644$ and $|\mathcal{H}_v| = 1$ for all the other non-leaf vertices v in T.

2. Suppose that we generate the k-th isomer in $\mathcal{G}(T)$ of the example. Recall that $\mathrm{nis}(v_0) = g(\mathbf{h}_1) + g(\mathbf{h}_2) = 588 + 7644 = 8232$, where we can regard the first 588 isomers in $\mathcal{G}(T)$ are constructed based on the first vector $\mathbf{h}_1 \in \mathcal{H}_{v_0}$ and the remaining 7644 isomers in $\mathcal{G}(T)$ are based on the second vector $\mathbf{h}_2 \in \mathcal{H}_{v_0}$. Hence when $k \leq 588$, we generate an isomer $G \in \mathcal{G}(T)$ based on $\mathbf{h}_1 \in \mathcal{H}_{v_0}$ and the corresponding cf-type $\mathbf{c}_{v_0} = (1,1,0,0,1,0,0,0)$; and when $k > 588$, we generate an isomer $G \in \mathcal{G}(T)$ based on $\mathbf{h}_2 \in \mathcal{H}_{v_0}$ and the corresponding cf-type $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$.

3. Assume $k = 600$, and consider how to determine the search indices $k[w]$ of the children w of v_0. Since $k = 600 > 588$, we construct the k-th isomer $G \in \mathcal{G}(T)$ based on $\mathbf{h}_2 = (2,1,1) \in \mathcal{H}_{v_0}$ and cf-type $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$; i.e., for $k' := k - 588 = 12$, we construct the k'-th isomer among such $g(\mathbf{h}_2) = 7644$ isomers based on $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$. Re-

call that

$$g(\mathbf{h}_2) = 7644 = 91 \times 84 \\ = \binom{\mathrm{nis}(v_1)}{2} \times \lambda((3,0,0,0,1,0,0,0)). \quad (2)$$

By $\mathbf{h}_2 = (2,1,1)$, the subset $\mathrm{Ch}_1(v_0) = \{v_1, v_{11}\}$ is partitioned into two sequences (v_1) and (v_{11}), indicating that we select an isomer $G' \in \mathcal{G}(T[v_0,v_1])$ and an isomer $G'' \in \mathcal{G}(T[v_0,v_{11}])$ which are not rooted-isomorphic. There are $\binom{\mathrm{nis}(v_1)}{2} = \binom{14}{2} = 91$ such pairs of isomers G' and G'', and each pair correspond with root v_0's 84 layout-types. By $k' = 12$, we have $0 \times 84 < 12 \leq 1 \times 84$. Then this means that in 600-th isomer, vertex v_0 has the first layout-type in occurrence vector $\mathbf{c}_{v_0} = (3,0,0,0,1,0,0,0)$, which means that the layout-type $L[v_0] = 1$ at na-vertex v_0 in the example T in Fig. 4.

4. Then we decide the search indices for the two na-vertices v_1 an v_{11} in $\mathrm{Ch}_1(v_0)$. The 91 pairs of non–isomorphic graphs G' and G'' rooted at v_1 and v_{11} have search indices of the two vertices listed from the smallest to the largest: (1,2), (1,3), …, (1,14), (2,3), …, (13,14), and we choose the 12th one, where $(k[v_1], k[v_{11}])$=(1,12). Then G' has the $k[v_1] = 1$-st isomer in $\mathcal{G}(T[v_0,v_1])$ and G'' has the $k[v_{11}] = 12$-th isomer in $\mathcal{G}(T[v_0,v_1])$. Since v_1 and v_{11} do not have any descendants that are na-vertex, this directly means that vertex v_1 has the 1-st layout-type, and v_{11} has the 12-th layout-type.

We designed an algorithm that determines the search indices $k[w]$ of children w of a vertex v with a search index k, and it runs in $O(\log|\mathcal{G}(T)|)$ time for each non-na-vertex v in T. Due to space limitation, we omit the algorithms and explanation of time complexities.

If we calculate the layout-types for all na-vertices v in tree T for a search index k, $k \in [1, \mathrm{nis}(v)]$, then we get our k-th isomer. The algorithms for determining the search indices and the layout-type for one vertex run in $O(\log|\mathcal{G}(T)|)$, and we can see that generating the k-th isomer runs in time $O(n\log|\mathcal{G}(T)|)$. Hence, all the isomers in $\mathcal{G}(T)$ can be generated in $O(n|\mathcal{G}(T)|\log|\mathcal{G}(T)|)$ time. It should be noted that the required space for executing our algorithm is $O(n)$, since we do not need to store any isomers generated before for a possible comparison with newly generated isomers to check duplication.

5 EXPERIMENTAL RESULTS

In this section, we attempt to show the correctness and efficiency of our algorithm. For this, we conducted two experiments over problem instances constructed based on some chemical compounds and databases. We implemented our algorithm in C language and executed it on a PC with Intel(R) Core(TM)i7 CPU 1.7 GHz and 8 GB memory. In this paper, we assume that a complete list of all layout types of each cf-type for a naphthalene ring is available as a table. In our program, we calculate the total number of isomers and generate each of them for a specific input using the table. The experimental results of computing time do not include the process of preparing this table.

In the first experiment, we solve small problem instances by our algorithm so that we can see whether all isomers of the instances are correctly and efficiently enumerated. For this, we used five relatively small instances, whose formulas are $\mathtt{na_1O_2H_8}$, $\mathtt{na_1C_2O_2H_{10}}$, $\mathtt{na_1C_3O_2H_{12}}$, $\mathtt{na_1C_4O_2H_{12}}$, $\mathtt{na_1C_5O_2H_{12}}$, to compare the results with those by MolGen. Here, chemical formula defines a set of chemical trees, from which isomers of the same formula are generated by our algorithm. For example, the first set consists of two tree representations all of which have the same formula $\mathtt{na_1O_2H_8}$. We used the algorithm due to Shimizu et al. (Shimizu and Akutsu, 2011) to generate all such trees.

Table 2: Comparison of our algorithms with MolGen.

Instance	# rep.	# iso.	Time (s)	Mol-Time
$\mathtt{na_1O_2H_8}$	2	12	0.0001	0.222 s
$\mathtt{na_1C_2O_2H_{10}}$	26	294	0.0010	108.025 s
$\mathtt{na_1C_3O_2H_{12}}$	140	2458	0.0114	1064.696 s
$\mathtt{na_1C_4O_2H_{12}}$	623	13442	0.0594	$>$ 6 hours
$\mathtt{na_1C_5O_2H_{12}}$	2046	46354	0.2436	$>$ 4 days

Table 2 summarizes the results, where each row shows the chemical formula of each instance, the number of tree representations generated from the formula (abbreviated as # rep.), the total number of naphthalene isomers generated from these tree representations by our algorithm (abbreviated as # iso.), the CPU time of our algorithm (abbreviated as Time), and that of MolGen (abbreviated as Mol-Time). Note that MolGen runs on a workstation with Intel(R) Xenon(R) CPU 3.00 GHz and 32 GB memory. It was observed that the numbers of isomers generated by our algorithm completely match with those generated by MolGen, and that our algorithm runs much faster, even on a slower CPU, which shows the correctness and efficiency of our algorithm.

In the second experiment, we measured the computing time by our algorithms for additional eight sets of relatively large instances with multiple naphthalene rings in the same manner. Table 3 shows the results on these instances, with each row shows the chemical formula of each instance, the number of tree representations generated from the formula (abbreviated as # rep.), the total number of naphthalene isomers gener-

ated from these tree representations by our algorithm (abbreviated as # iso.), and the CPU time of our algorithm (abbreviated as Time), from which we can observe that our algorithm runs fast even for millions of isomers.

As we can see in Table 3, as the number of na-vertices in formulas increases, the number of the corresponding isomers goes up drastically. On the other hand, the average computing time per isomer remains the same level of 10^{-6} second, regardless of the large size of the input instance.

Table 3: Large Instances with Multiple Naphthalene Rings.

Instance	# rep.	# iso.	Time (s)
na_2COH_{16}	8	483	0.0017
$na_2C_2O_2H_{18}$	141	38,752	0.1261
na_3COH_{22}	22	14,276	0.0352
$na_3CO_2H_{22}$	95	172,516	0.6417
$na_3C_2O_2H_{24}$	582	2,219,180	8.9126
$na_3C_2O_3H_{24}$	2383	19,679,568	138.7940
na_4COH_{28}	60	441,050	1.8361
na_5COH_{34}	166	14,019,964	66.3542

6 CONCLUSION

In this paper, we have described our algorithm by a dynamic programming in a rather general way in the sense that not only benzene and naphthalene rings but also some other local chemical structure such as polycyclic aromatic hydrocarbons can be handled in a similar way. Thus we can design an algorithm that, given a tree representation where several local structures are contracted into virtual atoms, counts the number of isomers of the tree and generates all isomers one by one. Such an extended algorithm will be made available via the EnuMol web server (http://sunflower.kuicr.kyoto-u.ac.jp/tools/enumol/).

REFERENCES

Akutsu, T. and Nagamochi, H. (2011). Kernel methods for chemical compounds, from classification to design. In *IEICE Transactions on Information and Systems, Vol. E94-D, No. 10, pp. 1846-1853*.

Benecke, C., G. R.-H. R. K. A. L. R. and Wieland, T., M. (1995). A generator of connectivity isomers and stereoisomers for molecular structure elucidation. In *Analytica Chimica Acta, Vol. 314*.

Benecke, C., G. T.-K. A. L. R. and Wieland, T. (1997). Molecular structures generation with molgen, new features and future developments. In *Fresenius' Journal of Analytical Chemistry, Vol. 359, No. 1, pp. 23-32*.

Blum, L. C. and Reymond, J. L. (2009). 970 million drug-like small molecules for virtual screening in the chemical universe database gdb-13. In *J. Am. Chem. Soc., Vol. 131, No. 25, pp. 8732-8733*.

Buchanan, B. G., S.-D. H. W. W. G. R. F. E. L. J. and Djerassi, C. (1976). Applications of artificial intelligence for chemical inference. 22, automatic rule formation in mass spectrometry by means of the meta-dendral program. In *J. Am. Chem. Soc., Vol. 98, No. 20, pp. 6168-6178*.

Fink, T. and Reymond, J. L. (2007). Virtual exploration of the chemical universe up to 11 atoms of C,N,O,F, assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical and properties, compound classes, and drug discovery. In *Journal of Chemical Information and Modeling, Vol. 47, No. 2, pp. 342-353*.

Fujiwara, H., W. J.-Z. L. N. H. and Akutsu, T. (2008). Enumerating tree-like chemical graphs with given path frequency. In *Journal of Chemical Information and Modeling, Vol. 48, pp. 1345-1357*.

Hardinger, S. (2005). *Chemistry 14D thinkbook*. Hayden McNeil.

He, F. and Nagamochi, H. (2015). A method for generating colorings over graph automorphism. In *ISORA 2015, Luoyang, China, pp. 70-81*.

Kerber, A., L. R.-G. T. and Meringer, M. (1998). Molgen 4.0, match commun. math. comput. chem. In *Vol. 37, pp. 205-208*.

Li, J., N. H. and Akutsu, T. (2013). Enumerating benzene isomers of tree-like chemical graphs. In *Manuscript submitted to a journal*.

Meringer, M. and Schymanski, E. L. (2013). Small molecule identification with molgen and mass spectrometry. In *Metabolites, Vol. 3, pp. 440-462*.

Nakano, S. and Uno, T. (2005). Generating colored trees. In *Lect. Notes Comput. Sci., Vol. 3787, pp. 249-260*.

Peironcely, J. E., R.-C. M. F. D. R. T. C. L. F. J.-L. H. T. (2012). Omg: open molecule generator. In *Journal of Cheminformatics, Vol. 4, Ariticle 21*.

Shimizu, M., N. H. and Akutsu, T. (2011). Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies. In *BMC Bioinformatics, Vol. 12, Suppl 14, S3*.