Journal of Bioinformatics and Computational Biology

BREADTH-FIRST SEARCH APPROACH TO ENUMERATION OF TREE-LIKE CHEMICAL COMPOUNDS

YANG ZHAO[†,§], MORIHIRO HAYASHIDA[†,§§,*], JIRA JINDALERTUDOMDEE[†,||], HIROSHI NAGAMOCHI[††,‡] and TATSUYA AKUTSU[†,¶,*]

[†]*Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 6110011, Japan*

[††]*Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 6068501, Japan*

[§]*tyoyo@kuicr.kyoto-u.ac.jp*
[§§]*morihiro@kuicr.kyoto-u.ac.jp*
[||]*jira@kuicr.kyoto-u.ac.jp*
[‡]*nag@amp.i.kyoto-u.ac.jp*
[¶]*takutsu@kuicr.kyoto-u.ac.jp*

Molecular enumeration plays a basic role in the design of drugs, which has been studied by mathematicians, computer scientists and chemists for quite a long time. Although many researchers are involved in developing enumeration algorithms specific to drug design systems, molecular enumeration is still a hard problem to date due to its exponentially increasing large search space with larger number of atoms. To alleviate this defect, we propose efficient algorithms, *BfsSimEnum* and *BfsMulEnum* to enumerate tree-like molecules without and with multiple bonds, respectively, where chemical compounds are represented as molecular graphs. In order to reduce the large search space, we adjust some important concepts such as *left-heavy*, *center-rooted* and *normal form* to molecular tree graphs. Different from many existing approaches, BfsSimEnum and BfsMulEnum firstly enumerate tree-like compounds by breadth-first search order. Computational experiments are performed to compare with several existing methods. The results suggest that our proposed methods are exact and more efficient.

Keywords: enumeration; chemical graphs; breadth-first search; drug design; tree structure.

1. Introduction

Analysis of chemical compounds has gained more attentions recently in bioinformatics because drug design is one of the major goals of bioinformatics and chemical compounds play an important role in metabolic networks. Among various topics on chemical compound analysis, molecular enumeration is a fundamental one that has

*Corresponding author

many applications in the field of drug design [5], and has been studied by mathematicians, computer scientists and chemists for more than one century. Especially, molecular enumeration algorithms have been developed for molecular design, molecular classification and structure elucidation using spectrometric techniques such as mass-spectrum (MS) and nuclear magnetic resonance (NMR) [14]. Although such approaches are not yet as active as marketing to present pharmaceutical service, they still play an essential role in pharmaceutics and therapeutics such that many researchers are involved in developing tools for drug design. It is to be noted that molecular enumeration has also been used as an engine of data mining and knowledge discovery from chemical compound data [3,9,12].

Since the first system for enumerating molecules, DENDRAL [15] came out, more researchers in academia have focused on developing computer-aided technology to study this crucial problem. Approaches for enumerating molecules are based on the representation of chemical compounds as molecular graphs. Conceptually, a molecular graph is defined as a connected multi-graph with vertices and edges colored by the atomic symbols and chemical bonds, where the degree of a vertex represents the atomic valence and the multiplicity of an edge represents the bond order [6]. Given molecular formula together with specific restrictions, desired molecules for biological system are enumerated by constructing all distinct graph structures. As proved by Dobson [4], the solution space for enumerating the desired molecules is estimated to be exponentially increasing as the size of molecules grows. The large search space leads to the foremost barrier for marketing these types of systems to the real-world service.

To date, some algorithms have been developed such as Molgen [8,6] and Enumol [11,7,16,2], etc. Molgen is known as the popular and useful tool that has been developed since 1985. This integrated project produces a fundamental structure generator that can enumerate desired molecules by given molecular formula with optional further restrictions, e.g. presence or absence of particular substructures. Although functional for constructive structure generation and application-oriented for molecular structure elucidation are continually upgraded by Faulon et al. [6], its enumeration algorithm still requires a vast amount of computational expense.

Recent approaches showed that it is possible to infer trees from feature vectors under constant levels in polynomial time [10,1]. However, these algorithms are not practical or cannot perform enumeration. One other constructive approach, Enumol was recently proposed [7,11,16] that enumerates tree-like molecular graphs by depth-first search (DFS) order. Herein, tree-like compounds have simple structures that can be represented by molecular tree graphs. They defined unique centroid and used the concept of *left-heavy* for labeling of a molecular tree graph such that these utilizations are then shown to be useful for reducing the search space. Although the results showed their competitive advantage to some existing approaches (e.g. Molgen), the growth of computational consumption during the constructive generation step is still needed to be retained.

In this study, we firstly propose efficient algorithms BfsSimEnum and BfsMulEnum for tree-like molecular enumeration by breadth-first search (BFS) order. Unlike algorithms by DFS order [7,11,16], our methods enumerate tree structures by keeping their balance which can efficiently save CPU time. For further reduction of the search space, we adjust some important concepts such as *center-rooted*, *left-heavy* and *normal form* when labeling a tree-structured molecular graph. It is interesting to note that target tree graphs are enumerated by BFS order while family tree is searched by DFS order in this study. Finally, computational experiments are performed whose results indicate that our methods are exact and more efficient than state-of-the-art ones. Although we focus on enumeration of tree-like chemical compounds in this paper, there are substantial possibilities of extensions of the methods to deal with more general structures. In the conclusion section, we discuss these potential extensions of our algorithms for the future step.

2. Preliminaries

Molecular formulas can be represented as molecular graphs whose vertices are labeled with the kinds of the corresponding atoms and edges are labeled with the types of bonds. In this section, we provide some elementary definitions that will be used later in our algorithms.

2.1. *Molecular trees*

We call acyclic connected molecular graphs without multi-edges *simple trees* which represent chemical compounds without multiple bonds. Conversely, *multi-trees* are allowed to have multi-edges.

Let $\Sigma = \{l_1, l_2, \ldots, l_s\}$ be the set of labels of atomic symbols. The degrees of vertices in such a molecular graph are restricted by a valence function $val : \Sigma \rightarrow Z^+$ that links each l_i ($l_i \in \Sigma$) with a positive integer. It is noted that only double and triple bonds are taken into account in this study. Herein, we give the label set Σ an order so as to distinguish atoms having the same valence. A *rooted* tree is defined as a tree with one vertex chosen as its root. Then, a molecular tree can be represented as a rooted ordered multi-tree $T(V, E)$, where V is a nonempty finite set of vertices that correspond to atoms, E is a nonempty finite set of edges that correspond to bonds (see an example in Figure 1). Let $num(l_i)$ be the number of vertices labeled as l_i in T. Let $height(T)$ be the height of T, and $maxpath(T)$ be a set of the longest paths in T, respectively. Let $T(v)$ denote the subtree rooted at vertex v. Let $l(v)$, $depth(v)$ and $degree(v)$ be the label, depth and degree of vertex v in T, respectively. Let $mul(u, v)$ be the multiplicity of edge (u, v), where u and v are distinct vertices in T.

We define the tree-like compound enumeration problem as follows.

Problem 1. Given a set Σ of s labels representing atoms, number n_i of each label

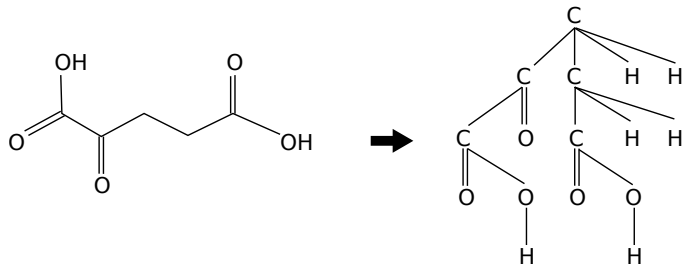


Fig. 1. An example of transforming 2-oxo-glutarate into a rooted ordered multi-tree T. This transformed molecular tree has $depth(T) = 4$, $maxpath$ = {HOCCCCCOH} and label set {C, O, H}, where $num(\mathrm{C}) = 5$, $num(\mathrm{O}) = 5$ and $num(\mathrm{H}) = 6$.

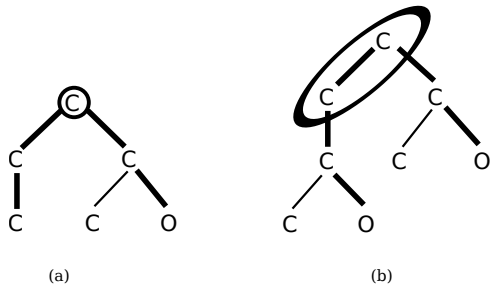


Fig. 2. Illustration of two kinds of center-rooted trees. The thick lines represent one of the longest paths, and the vertices in circles represent the center.

l_i, a valence function $val : \Sigma \to Z^+$, enumerate all molecular multi-trees T such that $n_i = num(l_i)$ for all l_i in Σ and $degree(v) = val(l(v))$ for all vertices $v \in T$.

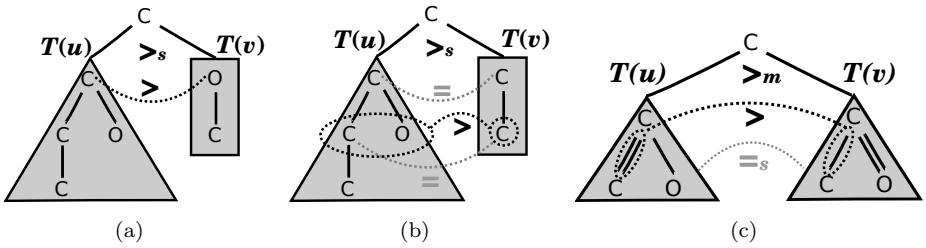
2.2. *Center-rooted*

We define a unique *center* to a rooted tree T as the center of any path in $maxpath(T)$, where such a center should be a single vertex (Figure 2(a)) or an edge (Figure 2(b)). It is obvious that such a center in T is unique regardless of the number of elements in $maxpath(T)$. Thus T is called *center-rooted* if its root is the center or one endpoint of the center.

2.3. *Left-heavy*

We introduce two inequalities $>_s$ and $>_m$ for ordered simple and multi-trees, which are recursively defined as in Definition 1 and Definition 2, respectively. We say that $T(u)$ is heavier than $T(v)$ if $T(u) >_s T(v)$ or $T(u) >_m T(v)$.

Definition 1. Let $(u_1, u_2, \ldots, u_h)$ and $(v_1, v_2, \ldots, v_k)$ be the children of u and v in simple or multi-tree T, respectively. We define $T(u) >_s T(v)$ if



(a) (b) (c)

Fig. 3. Illustration of inequality $T(u) >_s T(v)$ and $T(u) >_m T(v)$. Substructures in gray areas represent comparative subtrees rooted at u and v in each subfigure. The label set of these examples is $\Sigma = \{C, O\}$ whose order is $C > O$. (a) $T(u) >_s T(v)$ holds since the root of $T(u)$ has a greater label than that of $T(v)$. (b) Comparison of their corresponding descendants in BFS order shows that $T(u) >_s T(v)$. (c) The special case of $T(u) >_m T(v)$ with $T(u) =_s T(v)$, for which further comparison is needed to check their corresponding edge multiplicity in BFS order. Since the edge multiplicity in dot circles of $T(u)$ is greater than the corresponding location of $T(v)$, $T(u) >_m T(v)$ holds.

(1) $l(u) > l(v)$ (see Figure 3(a)), or
(2) $l(u) = l(v)$ $\exists\, i$, $\forall j \leq i$ $T(u_j) =_s T(v_j)$, and
   (a) $i < \min\{h, k\}$, $T(u_{i+1}) >_s T(v_{i+1})$, or
   (b) $i = k < h$ (Figure 3(b)).

   Specially, we define $T(u) =_s T(v)$ if $l(u) = l(v)$ and $T(u_j) =_s T(v_j)$ for all $j \leq h = k$.

Definition 2. We define $T(u) >_m T(v)$ for multi-tree T if

(1) $T(u) >_s T(v)$, or
(2) $T(u) =_s T(v)$ (see also Figure 3(c)), and
   $\exists i$, $(\forall j \leq i)$ $mul(e_j) = mul(e'_j)$ and $mul(e_{i+1}) > mul(e'_{i+1})$, where $e_1, e_2, \ldots, e_m$ (resp., $e'_1, e'_2, \ldots, e'_m$) be all the edges in $T(u)$ (resp., $T(v)$) in the BFS order.
   Specially, we define $T(u) =_m T(v)$ if $T(u) =_s T(v)$ and $mul(e_j) = mul(e_j)$ for all $j \leq m$.

For reducing the search space in the generation process, we utilize the definition of *left-heavy* for rooted trees, where the definition is slightly modified from that by Fujiwara et al. [7] and Nakano and Uno [13].

Definition 3. A molecular tree T is left-heavy if $T(v_i) \geq_m T(v_{i+1})$ $(i = 1, \ldots, k-1)$ holds for the children $(v_1, \ldots, v_k)$ of each vertex v in T.

2.4. *Normal form*

In order to avoid duplications, we utilize a notion of *normal form* to molecular trees as formalized in Definition 4. The normal form includes the ideas of center-rooted

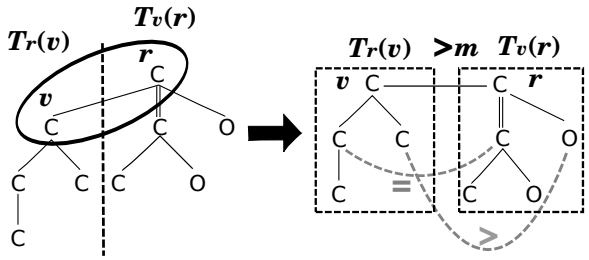


Fig. 4. Illustration of determining a normal form. Since the definition of normal form is based on the ideas of left heavy and center-rooted, the root is one endpoint of the center, further comparison is needed between subtree $T_r(v)$ and $T_v(r)$, only if r is the root and v is the other endpoint of the center. The given tree is determined as a normal tree because $T_r(v) >_m T_v(r)$.

and left-heavy. We call a tree in the normal form a *normal tree*.

Definition 4. Let T be a left-heavy center-rooted ordered tree rooted at r.

(1) If the center is a single vertex, then T is a normal tree.
(2) If the center is an edge (r, v) and $T_r(v) \geq_m T_v(r)$, where $T_r(v)$ ($T_v(r)$) denotes the subtree induced by v (r) and its descendants when r (v) is root, then T with root r is a normal tree (see also an example in Figure 4).

2.5. *Family tree*

Our approach searches a special tree structure called *family tree*. Let $\mathcal{C}_n$ denote a set of left-heavy center-rooted simple trees with at most n vertices, where $n = \sum_{i=1}^{s} n_i$. Suppose that a tree $T \in \mathcal{C}_n$ has k vertices ($0 < k \leq n$), numbered as $(v_1, v_2, \ldots, v_k)$ in BFS order. Let $P(T)$ be a tree generated from T by removing the last vertex v_k of T. We call $P(T)$ the *parent* of T.

Theorem 1. *If a given tree T is left-heavy center-rooted, then its parent $P(T)$ is left-heavy center-rooted as well.*

Proof. Suppose that v_k is the last vertex of T in BFS order. In terms of Definition 3, for any $T(v_j)$ in T such that $v_k \in T(v_j)$, we have $T(v_h) \geq_m T(v_j) >_m T(v_l)$, where v_h denotes any left sibling of v_j and v_l denotes any right sibling of v_j, respectively. It is noted that $T(v_j) \neq T(v_l)$ always holds, because $height(T(v_j)) > height(T(v_l))$ to keep v_k the last vertex. Then, $T(v_h) >_m T(v_j) - v_k \geq_m T(v_l)$ holds for such $T(v_j)$ (see also an illustration in Figure 5). Together with the definition that $P(T)$ is a tree with one less of the last vertex than T, $P(T)$ is also left-heavy.

The removed vertex v_k is in a path of $maxpath(T)$ because v_k is one of the deepest vertices in T. Let u be another endpoint of the path. Then, $depth(u) = depth(v_k)$ or $depth(u) = depth(v_k) - 1$ holds because T is center-rooted. Correspondingly, we obtain that $depth(u) = depth(parent(v_k)) + 1$ or $depth(u) = depth(parent(v_k))$. If

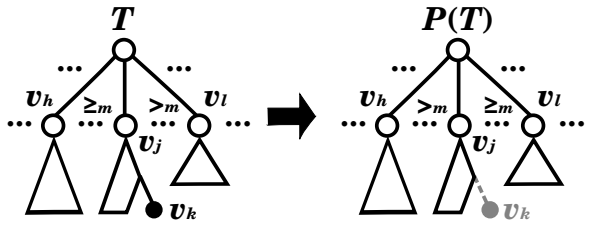


Fig. 5. Illustration of $P(T)$ being left-heavy. Suppose that v_k is the last vertex of T in BFS order, $T(v_j)$ is any tree such that $v_k \in T(v_j)$, and v_h and v_l are any left and right siblings of v_j in T. Since T is left-heavy and $T(v_h) \geq T(v_j) > T(v_l)$ holds, $T(v_h) \geq T(v_j) - v_k \geq T(v_l)$ always holds (the right-side).

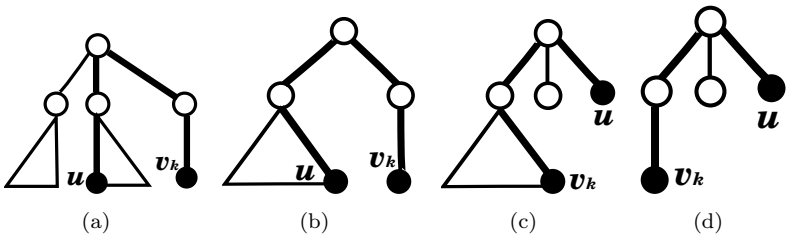
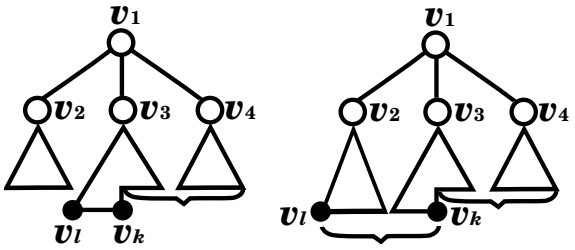


Fig. 6. Illustration of 4 kinds of center-rooted trees with their longest paths including the deepest rightmost vertex v_k (i.e., last vertex in BFS order). The black vertices denote the deepest rightmost vertices and the other endpoints of the longest paths in these trees.

the path between u and $parent(v_k)$ belongs to $maxpath(P(T))$, then $P(T)$ is center-rooted because the difference between $depth(u)$ and $depth(parent(v_k))$ is at most 1 (Figures 6(b) and 6(d)). Otherwise, $P(T)$ is still center-rooted because there exist other paths than the path between u and v_k in $maxpath(T)$ and these paths also belong to $maxpath(P(T))$ (Figures 6(a) and 6(c)). Thus, $P(T)$ is center-rooted as well. □

Theorem 1 implies that for any $T \in \mathcal{C}_n$, its parent tree $P(T)$ belongs to $\mathcal{C}_n$. Similarly, we can generate $P(P(T))$ by removing the vertex v_{k-1}, the deepest rightmost leaf of $P(T)$, from $P(T)$. Thus, a unique sequence T, $P(T)$, $P(P(T))$, $P(P(P(T)))$, $\ldots$, ϕ of trees in $\mathcal{C}_n$ can be generated by repeatedly removing the deepest rightmost leaf for each T in $\mathcal{C}_n$. A *family tree* of $\mathcal{C}_n$, denoted by $\mathcal{F}_n$, is defined by merging all these sequences. Obviously, each vertex in such $\mathcal{F}_n$ represents a tree in $\mathcal{C}_n$.

Furthermore, a family tree for molecular multi-trees can be similarly defined. Let $\mathcal{S}_m$ denote a set of left-heavy center-rooted multi-trees with at most m multi-edges. Suppose that a tree T belongs to $\mathcal{S}_m$ with h multi-edges ($0 < h \leq m$), and $(e_1, e_2, \ldots, e_h)$ is a sequence of multi-edges of T in BFS order. Let $P(T)$ be a tree generated from T by changing the multi-edge e_h to be a single edge. Then $P(T)$



(a) (b)

Fig. 7. Illustration for determining the possible positions where a new leaf should be added to a current tree. (a) If the deepest leftmost vertex v_l and the deepest rightmost vertex v_k are in the same subtree, a new leaf can be added to only vertices from $parent(v_k)$ to v_{l-1} (within the brace). (b) If v_l and v_k are included in distinct subtrees, a new leaf can be added to vertices from $parent(v_k)$ to v_k (within the brace).

is called the *parent* of T with one less multi-edges of T. We can easily prove that Theorem 1 can be extended to multi-trees and $P(T)$ for each T in $\mathcal{S}_m$ is in $\mathcal{S}_m$ as well. Similarly, $P(P(T))$ can be generated by changing the multi-edge e_{h-1} to a single edge from $P(T)$. Thus, a unique sequence T, $P(T)$, $P(P(T))$, $P(P(P(T)))$, ..., of trees in $\mathcal{S}_m$ can be generated by repeatedly changing the last multi-edge in BFS order to a single edge for each T in $\mathcal{S}_m$. A family tree of $\mathcal{S}_m$, denoted by $\mathcal{F}_m^M$, is defined by merging all these sequences. Obviously, each vertex in $\mathcal{F}_m^M$ represents a tree in $\mathcal{S}_m$ and root of such $\mathcal{F}_m^M$ is a simple tree. It should be noted that both $\mathcal{F}_m$ and $\mathcal{F}_m^M$ are searched by DFS order.

3. Methods

In this section, we propose BfsSimEnum and BfsMulEnum for enumerating molecular simple and multi-trees by breadth-first search order. As Ishida et al. [11] and Shimizu et al. [16] pointed out, the number of enumerated solutions exponentially increases with the increasing number of atoms. To reduce the large search space, concepts of center-rooted, left-heavy and normal form are taken in use as restrictions for avoiding duplicates.

3.1. *BfsSimEnum for simple tree enumeration*

Given a molecular formula with valence function, while trying to enumerate all possible simple trees, BfsSimEnum searches a family tree that: each vertex is a left-heavy and center-rooted simple tree, and specially, each leaf is in normal form. Notice that each vertex in such a family tree represents a tree with at most n' vertices, where n' is the total number of atoms whose valences are greater than 1, since atoms with valence one such as hydrogen atoms can be easily added as leaves at last.

This algorithm tries to search a family tree which starts with an empty tree, and grows up by repeatedly adding a new vertex to a current tree T in BFS order at every turn until all n' vertices are added. Algorithm 1 gives an introduction of BfsSimEnum (see also an illustration in Figure 9).

Firstly, BfsSimEnum constructs a tree T with one single vertex whose valence is greater than 1. As an addition step, this algorithm repeatedly adds a new vertex to possible positions of T according to left-heavy and center-rooted to construct a new tree. BfsSimEnum outputs a generated tree if and only if n' vertices are added and it is in normal form. The correctness of BfsSimEnum can be seen as follows. This algorithm searches all possible left-heavy center-rooted simple trees of a family tree which can cover all solutions of Problem 1. Then duplicates are excluded by checking the normal forms. Therefore, this algorithm correctly outputs all structures without any repetition.

Input An ordered set of labels $\Sigma = \{l_1, \ldots, l_s\}$, where $l_1 > l_2 \ldots > l_s$, number n_j of each label l_j, a valence function $val : \Sigma \rightarrow Z^+$
Output A set of all possible simple trees $\mathcal{R}$ which are *normal trees*
BfsSimEnum(Σ, val, $\{n_i\}$)
  $\mathcal{R} := \emptyset$
  for each $l_j \in \Sigma$ such that $val(l_j) > 1$ **do**
    $T :=$ a tree consisted of a root with l_j
    AddNode(Σ, val, $\{n_j\}$, T, $\mathcal{R}$)
  return $\mathcal{R}$
end
AddNode(Σ, val, $\{n_j\}$, T, $\mathcal{R}$)
  $n' :=$ the number of given atoms whose valences are greater than 1
  if $|T| = n'$ **and** T holds *normal form* **then**
    $\mathcal{R} := \mathcal{R} \cup \{T\}$
  else
    $v_k, v_l :=$ the deepest rightmost and leftmost vertices in T, respectively
    if v_k and v_l are included in the same proper subtree **then**
      $v_e := v_{l-1}$
    else $v_e := v_k$
    for each v_i from $parent(v_k)$ to v_e in BFS order **do**
      if $degree(v_i) < val(l(v_i))$ **then**
        $l_g :=$ the largest possible label for v_{k+1} (see Figure 8)
        for each $l_j \in \Sigma$ such that $l_j \leq l_g$ **and** $val(l_j) > 1$ **and** $num(l_j) < n_j$ **do**
          $T' := T$; add a new vertex with l_j as the last child of v_i in T'
          AddNode(Σ, val, $\{n_j\}$, T', $\mathcal{R}$)
end

Algorithm 1: BfsSimEnum for simple tree enumeration.

The point of this algorithm is how to keep a new constructed tree left-heavy and center-rooted at an addition step. Let v_k and v_l be the deepest rightmost and leftmost vertices in T, respectively. To keep a new tree center-rooted, the candidate positions to be added are determined by checking whether or not vertices v_l and v_k are in a subtree: (i) if they are in a subtree, a vertex can be added to only the positions ranging from $parent(v_k)$ to v_{l-1} (see an illustration in Figure 7(a)); (ii) otherwise, a vertex can be added to the positions ranging from $parent(v_k)$ to v_k (see also the Figure 7(b)). To keep a new tree left-heavy, candidate labels

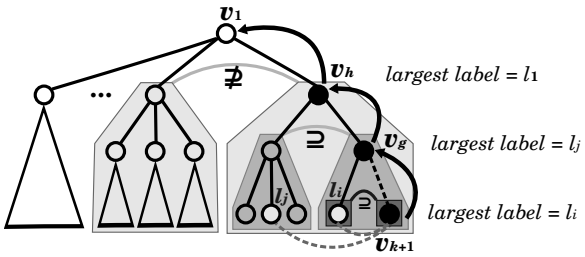


Fig. 8. Illustration for determining the largest possible label for v_{k+1}. The black circles are vertices having left siblings in the path from v_{k+1} to the root v_1. BfsSimEnum separately compares all of the subtrees rooted at such vertices with the subtrees rooted at their left siblings to determine the largest possible label. For instance, since $T(v_{k+1}) \subseteq T(v_k)$ and $T(v_g) \subseteq T(v_{g-1})$, the largest possible label at these comparison steps are l_i and l_j, which are the labels of corresponding vertices (light gray circles) of v_{k+1} in $T(v_k)$ and $T(v_{g-1})$, respectively; while since $T(v_h) \not\subseteq T(v_{h-1})$, no comparison is done for v_h and the largest possible label is l_1. The largest possible label for v_{k+1} is then determined by using l_i, l_j and l_1.

for a new added vertex v_{k+1} are determined by checking subtrees including v_{k+1}. Since the label set Σ is ordered as $l_1 > l_2 > \ldots > l_s$, our algorithm aims to seek the largest possible label for v_{k+1} such that all smaller ones are candidate labels for v_{k+1}. Suppose that v_h is a vertex in the path from v_{k+1} to the root v_1, which has left sibling. Let $T(v_h) \subseteq T(v_{h-1})$ if and only if there exists an injection mapping ψ of vertices from $T(v_h)$ to $T(v_{h-1})$ such that $l(v_i) = l(\psi(v_i))$ for all $v_i \in V(T(v_h))$ and $(\psi(v_i), \psi(v_j)) \in E(T(v_{h-1}))$ for all $(v_i, v_j) \in E(T(v_h))$, where $V(T)$ and $E(T)$ denote the vertex set and edge set of tree T, respectively. The largest possible label is determined by comparing the subtrees rooted at all such v_h with the subtrees rooted at their corresponding left siblings: (i) if $T(v_h) \subseteq T(v_{h-1})$, the largest label is l_j, where l_j is the label of corresponding vertex of v_{k+1} in $T(v_{h-1})$; (ii) if $T(v_h) \not\subseteq T(v_{h-1})$, the largest possible label is l_1. The largest possible label for v_{k+1} is thus determined as the smallest one obtained from these comparison steps (see an illustration in Figure 8).

3.2. *BfsMulEnum for multi-tree enumeration*

We propose BfsMulEnum for multi-tree enumeration, which starts with an output of BfsSimEnum and repeatedly changes a single edge to a multi-edge in BFS order at every turn until all possible multi-edges are added. As mentioned before, only edges with multiplicity 2 or 3 are taken into account as multiple bonds in this approach.

Let $\mathcal{M}_2$ and $\mathcal{M}_3$ denote the number of double bonds and triple bonds, respectively. $\mathcal{M}_2$ and $\mathcal{M}_3$ are computed so that they satisfy the following equation:

$$2\mathcal{M}_2 + 4\mathcal{M}_3 = \sum_{i=1}^{h} n_i \cdot val(l_i) - 2\sum_{i=1}^{h} n_i - \sum_{j=h+1}^{s} n_j + 2, \tag{1}$$

where two sets of labels $\{l_1, \ldots, l_h\}$ and $\{l_{h+1}, \ldots, l_s\}$ represent atoms whose valences are greater than 1 and whose valences are 1, respectively. In the example in Figure 9, as the molecular formula is given as $C_2O_2H_2$, feasible multiple bonds for $\mathcal{M}_2$ and $\mathcal{M}_3$ should satisfy that $2\mathcal{M}_2 + 4\mathcal{M}_3 = 2 \cdot 4 + 2 \cdot 2 - 2 \cdot (2+2) - 2 + 2 = 4$, which means that target molecular trees of this example should have two double bonds or one triple bond such that $(\mathcal{M}_2, \mathcal{M}_3) = (2, 0), (0, 1)$.

```
Input M2, M3 and output R of Algorithm 1
Output  a set of all possible multi-trees Rm
BfsMulEnum (M2, M3, R )
  Rm := ∅
  while |R| ≠ 0 do
    T := a tree of R
    remove T from R
    AddMultiedge (T, root of T, M2, M3, Rm)
  return Rm
end
AddMultiedge (T, vi, M2, M3, Rm)
    if M2 = 0 and M3 = 0 and T is a normal tree then
      Rm := Rm ∪ {T}
    else
      AddMultiedge( T, vi+1, M2, M3, Rm)
      miss(vi) := val(l(vi)) − degree(vi)
      miss(parent(vi)) := val(l(parent(vi))) − degree(parent(vi))
      if miss(vi) ≥ 1 and miss(parent(vi)) ≥ 1 and M2 > 0 then
        T2 := T
        set double bond to (parent(vi), vi) in T2
        AddMultiedge( T2, vi+1, M2 − 1, M3, Rm)
      if miss(vi) ≥ 2 and miss(parent(vi)) ≥ 2 and M3 > 0 then
        T3 := T
        set triple bond to (parent(vi), vi) in T3
        AddMultiedge( T3, vi+1, M2, M3 − 1, Rm)
end
```

Algorithm 2: BfsMulEnum for multi-tree enumeration.

From the output of BfsSimEnum together with $\mathcal{M}_2$ and $\mathcal{M}_3$ determined as above, BfsMulEnum aims to construct a set of target multi-trees $\mathcal{R}_M$ by BFS order, see also the details in Algorithm 2. BfsMulEnum recursively sets a multi-edge to feasible positions of T according to normal form to construct a new tree. Only if all feasible multiple bonds are set, BfsMulEnum outputs such a new tree. Different from BfsSimEnum, at a setting step, BfsMulEnum needs to check whether a new tree is in normal form by comparing the edge multiplicity together with checking the feasibility of setting other multi-edges when generation for such a new tree is still continued. The correctness of BfsMulEnum can be similarly validated as follows. This algorithm generates all possible multi-trees by keeping them left-heavy which can cover all of the solutions. Finally, all possible structures are correctly enumerated by checking the normal forms.

Although it consumes a little more computational expense for enumerating multi-tree structures than that for enumerating simple ones in this study, computation of BfsMulEnum is not complicated since it only deals with edges without

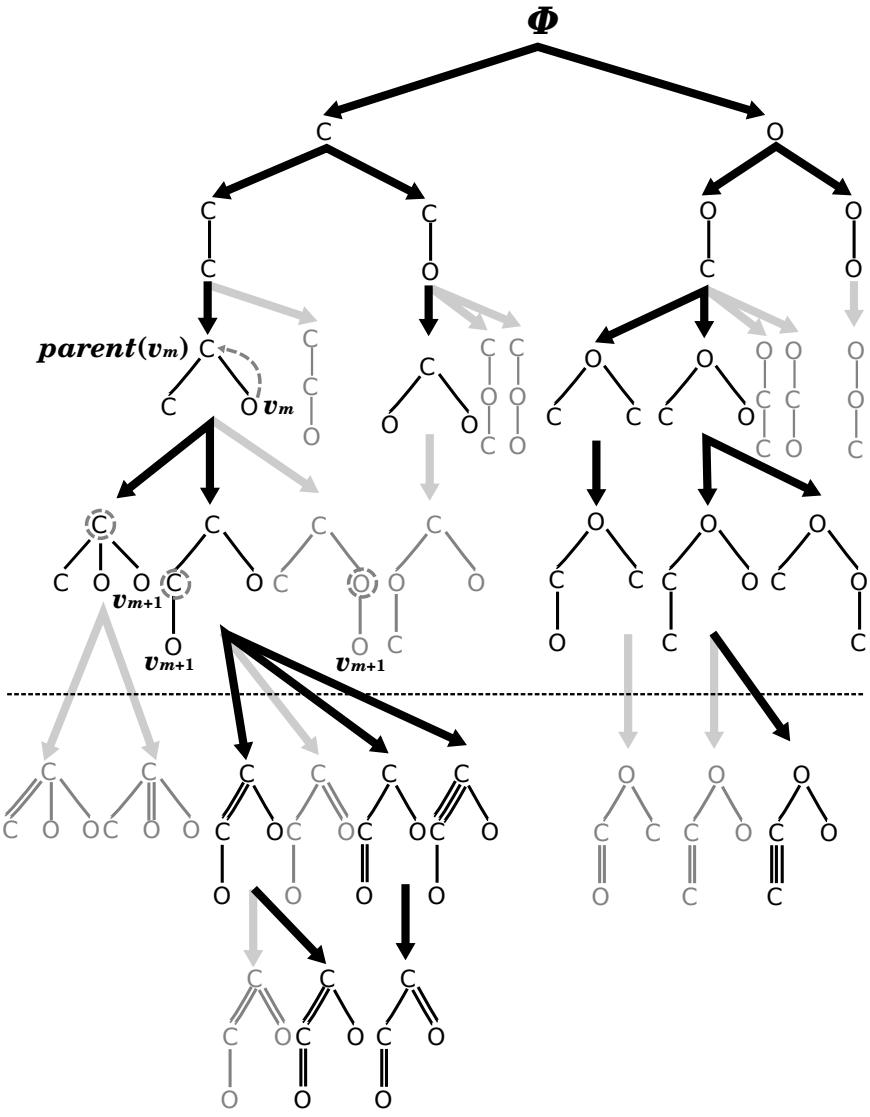


Fig. 9. Illustration of BfsSimEnum and BfsMulEnum. Algorithm 1 is processed above the dot line; Algorithm 2 is processed below the dot line. Graphs in gray color are considered as invalid by the algorithms and thus are not stored or proceeded any more. It should be noted that Hydrogen atoms are added as leaves at last.

any structural changes. Figure 9 illustrates the process of both BfsSimEnum and BfsMulEnum. It should be noted that atoms with valence 1 are added as leaves at last.

3.3. *Time complexity analysis*

From Algorithm 1, we can see that the size of $\mathcal{R}$ (which is used to store enumerated isomers) exponentially grows with the number of atoms increasing. Therefore both space and time complexities of BfsSimEnum are exponential. Since BfsMulEnum uses the outputs of BfsSimEnum, its space and time complexities are exponential as well. Whereas, the space complexity of our methods can be polynomial if we do not store the outputs.

Several researchers have focused on the output polynomial (which means to output one solution in polynomial time). Although the time complexity of our algorithms is exponential, development of output polynomial time algorithms will be our future work.

4. Results

Computational experiments were performed on BfsSimEnum and BfsMulEnum using a PC with Xeon CPU 3.47GHz and 24GB memory.

4.1. *Comparison with existing methods*

We assessed the computational performance by comparison with two state-of-the-art methods, Molgen (Version 3.5) and Enumol, under the same computational environment. The results are shown in Table 1 and Table 2. Unlike existing approaches whose molecular structures are generated by DFS order, the results successfully show that generating a solution by BFS order also performs well or even better for tree-like molecular enumeration, since all of these solutions are proceeded by keeping balance.

From Table 1, we can see that BfsSimEnum was faster than the other ones, which also implies that the employed and modified concepts of center rooted, left heavy and normal form are very useful for reducing the search space.

From the computational time shown in Table 2, we can see that BfsMulEnum is slightly less advantageous than Enumol when $\mathcal{M}_2 + 2\mathcal{M}_3 < 6$, which means that the number of double bonds is bounded by 5. As the number of multiple bonds increases (specially when $\mathcal{M}_2 + 2\mathcal{M}_3 \geq 6$), BfsMulEnum outperforms Enumol. The reason why BfsMulEnum is sometimes slower than Enumol is its dependence on BfsSimEnum. Due to this reason, there might be a large amount of simple trees computed by BfsSimEnum that cannot be expanded to multi-trees. On the other hand, our multi-tree enumeration method is significantly faster than Molgen.

4.2. *Extension to multivalent elements*

We extended our algorithms to deal with multivalent elements. Our methods allow one element occur with different valences in a generated molecular tree. Since every atomic label in Σ has a prescribed valence, we set n distinct labels to represent atom

Table 1. Comparison of BfsSimEnum with existing methods.

Molecular formula	# Enumerated results	Computational time (Sec.)		
		BfsSimEnum	Molgen	Enumol
$C_{18}H_{38}$	60523	0.016	3.04	0.025
$C_{19}H_{40}$	148284	0.036	5.93	0.060
$C_{20}H_{42}$	366319	0.086	8.18	0.15
$C_{22}H_{46}$	2278658	0.53	79.80	0.939
$C_{24}H_{50}$	14490245	3.284	733.12	6.153
$C_{26}H_{54}$	93839412	21.361	7367.48	41.292
$C_6O_3H_{14}$	772	0.001	0.01	0.001
$C_7O_3H_{16}$	2275	0.002	0.01	0.002
$C_{10}O_4H_{22}$	317677	0.072	1.19	0.108
$C_{12}O_4H_{26}$	3118708	0.691	15.31	1.088
$C_{16}O_4H_{34}$	278960984	60.16	2272.55	101.69
$C_{18}O_4H_{38}$	2567668160	533.84	-	965.4
$C_6N_2O_3H_{16}$	140014	0.031	0.36	0.049
$C_7N_2O_2H_{18}$	82836	0.019	0.17	0.029
$C_7N_3O_2H_{19}$	649970	0.135	1.48	0.216
$C_8N_3O_2H_{21}$	2361374	0.485	6.24	0.81
$C_9N_2O_2H_{22}$	893769	0.188	2.59	0.309
$C_9N_3O_2H_{23}$	8373347	1.683	25.52	2.839
$C_{10}N_3O_2H_{25}$	29105924	5.887	93.94	10.303
$C_{11}N_3O_2H_{27}$	99494345	20.110	367.72	35.139

with n multiple valences such that each of these labels is treated as an independent atom. For example, we set C and $C^{(2)}$ (whose valences are 4 and 2) to both represent carbon atom, while as defined in Σ, they are handled as two different labels with representing two different atoms. It is noted that this expansion requires the number of such multivalent elements being given in advance. However, we can remove this assumption by making exhaustive search on the numbers of C and $C^{(2)}$. Since the number of such combinations is bounded by the number of carbon atoms, it does not significantly increase the computational time. Although multivalent elements are not common for carbon atoms, they are common for sulfur and phosphorus atoms.

We also performed some experiments to verify this extension. Table 3 gives the number of generated trees and computational time. Since Molgen cannot deal with such extension, we only compared our generated results with that of Enumol. From the results, we can see that our extension is also correct and competitively faster than Enumol.

Table 2. Comparison of BfsMulEnum with existing methods.

Molecular formula	$\mathcal{M}_2 + 2\mathcal{M}_3$	# Enumerated results	Computational time (Sec.)		
			BfsMulEnum	Molgen	Enumol
$C_{18}H_{34}$	2	3218346	0.266	145.99	0.311
$C_{19}H_{34}$	3	31503100	2.727	3753.04	2.7
$C_{20}H_{34}$	4	250132215	23.689	98799.2	23.39
$C_{20}H_{28}$	6	1185277179	181.37	-	188.6
$C_{22}H_{36}$	5	5445565067	556.21	-	544.52
$C_{22}H_{34}$	6	10198151506	1185.27	-	1192.53
$C_{22}H_{30}$	8	19663780677	3255.08	-	3392.54
$C_{10}O_4H_{16}$	3	10003272	1.5	400.83	1.335
$C_{12}O_4H_{16}$	5	282338151	63.33	176186.1	56.352
$C_{12}O_4H_{10}$	8	49498872	78.91	1183717.4	90.82
$C_{16}O_2H_{20}$	7	1996919931	467.48	-	470.21
$C_{16}O_4H_{30}$	2	12880695359	1172.81	-	1137.07
$C_6N_2O_3H_{14}$	1	643197	0.1	5.13	0.1
$C_6N_2O_3H_{10}$	3	1499019	0.345	44.01	0.307
$C_7N_2O_2H_{10}$	4	1312737	0.360	83.95	0.317
$C_7N_2O_2H_6$	6	257531	0.380	329.75	0.41
$C_7N_3O_2H_9$	5	8360420	3.932	1855.59	3.836
$C_7N_3O_2H_7$	6	3282844	3.81	11166.89	4.21
$C_8N_3O_2H_{11}$	5	62066528	20.931	16791.67	20.141
$C_8N_3O_2H_9$	6	31421502	21.52	70591.54	23.36
$C_9N_2O_2H_{10}$	6	18780376	10.038	15779.98	10.478
$C_9N_2O_2H_8$	7	7205103	9.27	76774.18	11.33
$C_9N_3O_2H_{11}$	6	252761084	119.06	288470.42	123.68
$C_9N_3O_2H_9$	7	107205329	113.67	637866.1	138.33
$C_{10}N_3O_2H_{11}$	7	932854039	637.2	-	715.99
$C_{11}N_3O_2H_{21}$	3	7268812476	802.67	-	774.56
$C_{11}N_3OH_{15}$	6	956851032	247.52	-	250.02

4.3. *Structural matching*

We performed some other experiments that examine whether each generated structure is found in the database. Here we used PubChem[a] to process the matching experiments. The matching rates of structures generated by both BfsSimEnum and BfsMulEnum are shown in Table 4 and Table 5, respectively. Since the valence of Hydrogen atom is one which can be spontaneously added as a leaf at last, the number of enumerated isomers is not affected by the number of H atoms, but is strongly relevant to the number of atoms whose valences are greater than 1. From these numbers of enumerated isomers, we can clearly see that the number of enumerated structures increases with the number of atoms with valence greater than 1 growing.

[a]http://pubchem.ncbi.nlm.nih.gov

Table 3. Results for compounds with multivalent elements.

Molecular formula	# Enumerated results	Computational time (Sec.)	
		BfsSimEnum	Enumol
$C_{10}C_2^{(2)}O_2H_{22}$	1754152	0.39	0.6
$C_{16}C^{(2)}O_3H_{34}$	1068503824	228.8	403.16
$C_8C_2^{(2)}N_3O_2H_{21}$	602536450	122.82	229.21
$C_{11}C^{(2)}N_3OH_{27}$	195396579	40.45	72.11
$C_{10}C_2^{(2)}O_2H_{20}$	10917613	1.56	1.42
$C_{10}C^{(2)}O_2H_6$	288867	1.42	1.97
$C_{12}C_2^{(2)}O_2H_{10}$	289235948	444.81	517.13
$C_9C^{(2)}N_3OH_9$	212688117	221.31	270.83

From the mathcing results, we can also see that with increase of the number of generated structures, the matching ratio decreases. It indicates that a large amount of structures are exponentially generated, but only a small fraction of them has been explored. This finding is encouraging such that these unexplored regions provide a vast potentiality to improve our today's drugs by new compound design. Therefore, exploration of these unknown structures might be a crucial extension topic so as to discover new useful compounds, which will advance deep understanding of biology and lead to a new strategy to treat disease.

Further combination with chemical activity and biological property can help researchers to find useful molecules so as to solve real world problems such as molecular design and drug design, etc. For instance, let each element be weighted by its interacting ability with others so that we can predict each enumerated structure with a score. Using such scores, we can further predict new compounds and also reduce unreal isomers from the large unexplored chemical space. While that, BfsSimEnum and BfsMulEnum as rudimentary approaches provided an efficient route toward further combinations and applications.

5. Conclusion

In this study, we proposed BfsSimEnum and BfsMulEnum for enumerating tree-like compounds by firstly utilizing the breadth-first search order. Owing to the utilization of BFS order, both BfsSimEnum and BfsMulEnum only produce balanced intermediate trees during their search of a family tree without proceeding or storing any unbalanced ones, which can efficiently avoid duplicates. Together with the employed and modified concepts such as center-rooted, left-heavy and normal form, our proposed methods are successfully showed to be useful for reducing the large search space.

The results of computational experiments indicate that our algorithms are exact and faster than state-of-the-art ones for simple tree enumeration. But for multi-trees

Table 4. Matching results for chemical compounds only with single bonds to PubChem.

Molecular formula	# Enumerated results	Matching to PubChem	
		# Matched isomers	Existence rate
$C_2O_2H_6$	5	5	1.0
$C_3O_2H_8$	11	11	1.0
$C_3O_3H_8$	28	22	0.786
$C_4O_2H_{10}$	28	28	1.0
$C_4O_3H_{10}$	88	59	0.670
$C_4O_4H_{10}$	255	35	0.137
$C_5O_2H_{12}$	69	68	0.986
$C_5O_3H_{12}$	258	96	0.372
$C_5O_4H_{12}$	869	60	0.069
$C_5O_5H_{12}$	2570	8	0.003
$C_6O_2H_{14}$	179	147	0.821
$C_6O_3H_{14}$	772	177	0.229
$C_7O_2H_{16}$	463	282	0.609
$C_7O_3H_{16}$	2275	206	0.091
C_8OH_{18}	171	142	0.830
$C_8O_2H_{18}$	1225	407	0.332
C_9OH_{20}	405	221	0.546
$C_9O_2H_{20}$	3246	407	0.125
$C_{10}OH_{22}$	989	254	0.257

enumeration, BfsMulEnum is often outperformed by Enumol only when $\mathcal{M}_2+2\mathcal{M}_3$ is bounded to 5. Although it is efficient for molecules which include large number of multiple bonds ($\mathcal{M}_2 + 2\mathcal{M}_3 \geq 6$), BfsMulEnum is possible to get a further extension to make it independent from BfsSimEnum. For this purpose, not only possible vertices but also possible multi-edges should be both taken into account when generating intermediate trees. Such an extension can significantly reduce search space to speed up BfsMulEnum because it aims to generate intermediate trees without expanding simple trees which no longer can be changed to multi-trees. To let our proposed methods be more practical and widely enployed, improvement of BfsMulEnum will be our imperative next future step.

By finding the generated structures in the PubChem database, we obtained low matching rates which suggest that a large amount of structures are generated with representing unknown compounds. These unexplored structures are expected to carry important chemical characteristics lack of which may cause dysfunction or diseases. Understanding of such unknown structures should be an essential future step so as to discover new compounds or/and even to design new drugs for diseases. We consider that combination of atomic chemical activity and biological property with these unknown structures should be useful to predict their potential functions.

BfsSimEnum and BfsMulEnum can also be extended to deal with some cyclic compounds such as benzenes by just representing these cyclic structures as spe-

Table 5. Matching results for chemical compounds with multiple bonds to PubChem.

Molecular formula	# Enumerated results	Matching to PubChem	
		# Matched isomers	Existence rate
$C_2O_2H_2$	4	4	1.0
$C_3O_2H_4$	19	14	0.737
$C_3O_2H_6$	19	15	0.789
$C_3O_3H_4$	43	10	0.233
$C_4O_2H_6$	80	32	0.4
$C_4O_2H_8$	66	44	0.667
$C_4O_3H_6$	241	43	0.178
$C_4O_3H_8$	212	66	0.311
$C_5O_2H_8$	308	87	0.282
$C_5O_2H_{10}$	204	102	0.5
$C_5O_3H_{10}$	798	137	0.172
$C_5O_4H_{10}$	2628	153	0.058
$C_6O_2H_{10}$	1139	169	0.148
$C_6O_2H_{12}$	641	206	0.321
$C_6O_3H_{12}$	2845	279	0.098
$C_7O_2H_{14}$	1946	277	0.142
$C_7O_3H_{14}$	9823	423	0.043
C_8OH_{14}	1863	247	0.133
C_8OH_{16}	790	228	0.289
C_9OH_{18}	2136	246	0.115
C_9OH_{16}	5626	231	0.041

cial vertices. For example, benzene can be represented as an atom with valence 6, together with further restrictions such as giving special constraints on the ordering of 6 positions so as to avoid duplicates. Such an extension has already been implemented in Enumol and is also under development for BfsSimEnum and BfsMulEnum. Further extensions to include more complex ring structures such as Naphthalenes are also under development. Therefore, the methodologies developed here are not limited to enumeration of tree-like chemical compounds but are useful to cover a wide range of chemical compounds.

Our proposed methods are fast and fundamental for molecular enumeration that has many useful applications. Extensions toward enumerating general compounds and combination with biological properties should be interesting future work.

Acknowledgments

This work was partially supported by Grants-in-Aid #22240009, #24500361, and #25-2920 from MEXT, Japan.

References

1. Akutsu T, Fukagawa D, Jansson J, Sadakane K, Inferring a graph from path frequency, *Discrete Applied Mathematics* **160**:1416–1428, 2012.
2. Akutsu T, Nagamochi H, Comparison and enumeration of chemical graphs, *Computational and Structural Biotechnology Journal* **5**(6):e201302004, 2013.
3. Deshpande M, Kuramochi M, Wale N, Karypis G, Frequent substructure-based approaches for classifying chemical compounds, *IEEE Trans Knowledge and Data Engineering* **17**:1036–1050, 2005.
4. Dobson CM, Chemical space and biology, *Nature* **432**:824–828, 2004.
5. Faulon JL, Bender A, *Handbook of Chemoinformatics Algorithms*, CRC Press, 2010.
6. Faulon JL, D P Visco J, Rose D, Enumerating molecules, *Reviews in Computational Chemistry* **21**:209–286, 2005.
7. Fujiwara H, Wang J, Zhao L, Nagamochi H, Akutsu T, Enumerating treelike chemical graphs with given path frequency, *Journal of Chemical Information and Modeling* **48**(7):1345–1357, 2008.
8. Gugisch R, Kerber A, Kohnert A, Laue R, Meringer M, Rucker C, Wassermann A, *Molgen 5.0, a Molecular Structure Generator*, Bentham Science Publishers Ltd., 2012.
9. Horváth T, Ramon J, Efficient frequent connected subgraph mining in graphs of bounded tree-width, *Theoretical Computer Science* **411**:2784–2797, 2010.
10. Imada T, Ota S, Nagamochi H, Akutsu T, Efficient enumeration of stereoisomers of outerplanar chemical graphs using dynamic programming, *Journal of Chemical Information and Modeling* **51**:2788–2807, 2011.
11. Ishida Y, Kato Y, Zhao L, Nagamochi H, Akutsu T, Branch-and-bound algorithms for enumerating treelike chemical graphs with given path frequency using detachment-cut, *Journal of Chemical Information and Modeling* **50**(5):934–946, 2010.
12. Jiang C, Coenen F, Zito M, A survey of frequent subgraph mining algorithms, *The Knowledge Enginering Review* **28**:75–105, 2013.
13. Nakano S, Uno T, Generating colored trees, *Lecture Notes in Computer Science* **3787**:249–260, 2005.
14. Pretsch E, Bühlmann P, Badertscher M, *Structure Determination of Organic Compounds*, Springer-Verlag Berlin Heidelberg, 2009.
15. Rouvray DH, The pioneering contributions of cayley and sylvester to the mathematical description of chemical structure, *Journal of Molecular Structure* **1**:54, 1989.
16. Shimizu M, Nagamochi H, Akutsu T, Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies, *BMC Bioinformatics* **12**(Suppl 14):1–9, 2011.