IPSJ SIG Technical Report

Vol.2013-MPS-96 No.15
Vol.2013-BIO-36 No.15
2013/12/11

Breadth-first Search Approach to Enumeration of Tree-like Chemical Compounds

Yang Zhao[1,a) Morihiro Hayashida[1,b)] Jira Jindalertudomdee[1,c)] Hiroshi Nagamochi[2,d)]
Tatsuya Akutsu[1,e)]

Abstract: Enumeration of chemical compounds plays a basic role in the design of drugs and the determination of chemical structures from mass spectrometry. Hence, it has been studied by mathematicians and computer scientists as well as chemists for quite a long time. In this technical report, we restrict enumerated structures to trees, and propose efficient algorithms, *BfsSimEnum* and *BfsMulEnum* for enumeration of tree-like chemical compounds without and with multiple bonds, respectively. Unlike many existing approaches, our methods iteratively add each atom to a balanced tree by breadth-first search order. In order to reduce large search space, we adjust some important concepts such as *left-heavy*, *center-rooted* and *normal form* to balanced trees. For evaluation of efficiency, we perform experiments for several instances, and compare our methods with existing methods. The results suggest that our proposed methods are exact and more efficient.

1. Introduction

Chemical compounds play important roles in living organisms and metabolic networks. Enumeration of chemical compounds is an essential tool for analysis of chemical compounds, drug design [1], molecular classification and structure elucidation using spectrometric techniques such as mass-spectrum (MS) and nuclear magnetic resonance (NMR) [2]. Hence, the enumeration has been studied by by mathematicians and computer scientists as well as chemists for more than one century. Molecular enumeration has also been used for data mining and knowledge discovery from chemical compound data [3], [4], [5].

More researchers have developed computer-aided technology for enumerating molecules starting with DENDRAL [6]. Many approaches for enumerating molecules represent a chemical compound as a molecular graph, which is defined as a connected multi-graph with vertices and multi-edges labeled by the atomic symbols and chemical bonds, respectively. Here, the degree of a vertex represents the atomic valence and the multiplicity of a multi-edge represents the bond order [7]. Given chemical formula together with specific restrictions, desired chemical structures for biological system are enumerated by constructing all distinct graph structures. The number of enumerated structures is estimated to be exponentially increasing with the total number of atoms as proved by Dobson [8].

Some algorithms such as MOLGEN [7], [9] and EnuMol [10], [11], [12], [13] have been developed. MOLGEN is known as the popular and useful tool, and can enumerate desired chemical structures by given chemical formula with optional further restrictions, e.g. presence or absence of particular substructures. Although functions for constructive structure generation have been continually upgraded by Faulon et al. [7], its enumeration algorithm still requires a vast amount of computational expense.

Recent approaches indicated that it is possible to estimate trees from feature vectors under constant levels in polynomial time [14], [15]. These algorithms, however, are not practical or cannot perform enumeration. One other constructive approach, EnuMol was recently proposed [10], [11], [12] 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. To avoid the generation of the same tree and to reduce the search space, they defined unique centroid and used the concept of *left-heavy*. Although they reported their competitive advantage to some existing approaches (e.g. MOLGEN), the computational consumption during the constructive generation step is still needed to be retained.

In this technical report, we propose efficient algorithms BfsSimEnum and BfsMulEnum for enumeration of tree-like chemical compounds by breadth-first search (BFS) order [16]. Unlike algorithms by DFS order [10], [11], [12], our methods enumerate tree structures by keeping their balance. For further reduction of the search space, we adjust some important concepts such as *center-rooted*, *left-heavy* and *normal form* when generating a balanced tree. It should be noted that target trees are enumerated by BFS order while family tree is searched by DFS order in this study. We perform computational experiments, whose results indicate

1 Bioinformatics Center, Institute for Chemical Research, Kyoto University, Gokasho, Uji, Kyoto 611–0011, Japan
2 Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 606–8501, Japan
a) tyoyo@kuicr.kyoto-u.ac.jp
b) morihiro@kuicr.kyoto-u.ac.jp
c) jira@kuicr.kyoto-u.ac.jp
d) nag@amp.i.kyoto-u.ac.jp
e) takutsu@kuicr.kyoto-u.ac.jp

© 2013 Information Processing Society of Japan

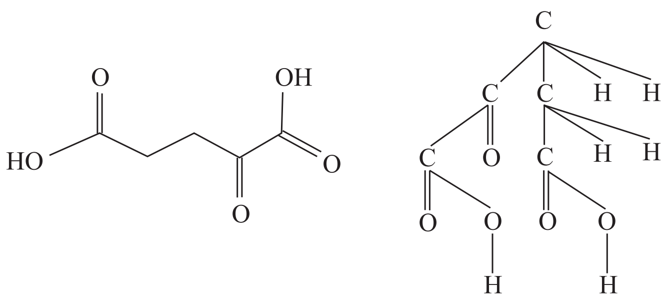

Fig. 1 Example of transforming 2-oxo-glutarate into a rooted ordered multi-tree T. This transformed molecular tree has $depth(T) = 4$, $maxpath(T) = \{\text{HOCCCCCOH}\}$ and label set {C, O, H}, where $num(\text{C}) = 5$, $num(\text{O}) = 5$ and $num(\text{H}) = 6$.

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 technical report, 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

Chemical structures 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 connected acyclic molecular graphs without multi-edges *simple trees* that 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 $val : \Sigma \to Z^+$ 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 with the same valence and to define a normal form for molecular trees. 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 Fig. 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 in T. Let $l(v)$, $depth(v)$, $degree(v)$ and $parent(v)$ be the label, depth, degree and the parent vertex 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 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

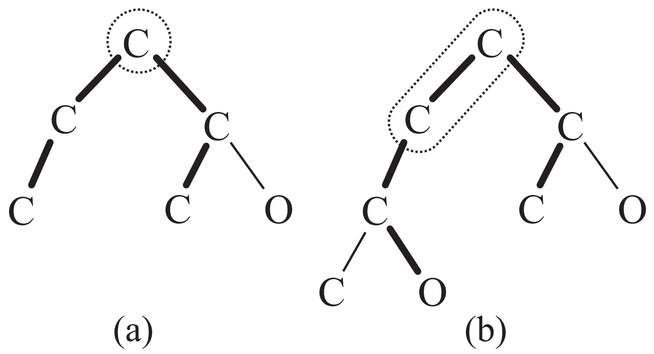

Fig. 2 Illustration of two kinds of center-rooted trees. The thick lines represent one of the longest paths, and the vertices in dotted circles represent the center.

$degree(v) = val(l(v))$ 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 (Fig. 2(a)) or an edge (Fig. 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 Definitions 1 and 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

(1) $l(u) > l(v)$, or

(2) $l(u) = l(v)\ \exists\, i, \forall j \le i\ T(u_j) =_s T(v_j)$, and

  (a) $i < \min\{h, k\}$, $T(u_{i+1}) >_s T(v_{i+1})$, or

  (b) $i = k < h$.

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

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

(1) $T(u) >_s T(v)$, or

(2) $T(u) =_s T(v)$, and

  $\exists i$, $(\forall j \le i)\ mul(e_j) = mul(e'_j)$ 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 $j \le m$.

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. [11] and Nakano and Uno [17].

Definition 3 A molecular tree T is *left-heavy* if $T(v_i) \ge_m T(v_{i+1})$ $(i = 1, \ldots, k-1)$ 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*

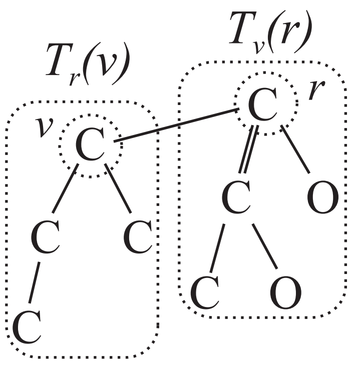

Fig. 3 Illustration of determining a normal form. Since the definition of normal form is based on the ideas of left heavy and center-rooted, the root is one endpoint of the center, further comparison is needed between subtree $T_r(v)$ and $T_v(r)$, only if r is the root and v is the other endpoint of the center. The given tree is determined as a normal tree because $T_r(v) >_m T_v(r)$.

form to molecular trees as formalized in Definition 4. The normal form includes the ideas of center-rooted 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 Fig. 3).

2.5 Family tree

Our approach searches a special tree structure called *family tree*. Let 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 $T \in C_n$ 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. Then, we can prove the following [16].

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

This implies that for any $T \in C_n$, its parent tree $P(T)$ belongs to 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, $P(T)$, $P(P(T))$, $P(P(P(T)))$, …, ϕ of trees in C_n can be generated by repeatedly removing the deepest rightmost leaf for each T in C_n. A *family tree* of 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 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. A family tree of $\mathcal{S}_m$, denoted by $\mathcal{F}_m^M$, is similarly defined. 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.

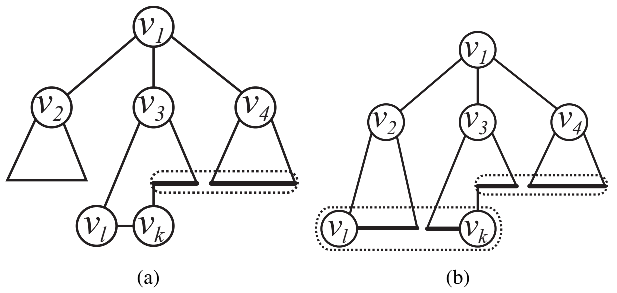

Fig. 4 Illustration for determining the possible positions (within the dotted curves) where a new leaf should be added to a current tree. (a) If the deepest leftmost vertex v_l and the deepest rightmost vertex v_k are in the same subtree, a new leaf can be added to only vertices from $parent(v_k)$ to v_{l-1}. (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.

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. [10] and Shimizu et al. [12] 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 that 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. Fig. 5 shows a pseudo code of BfsSimEnum (see also Fig. 7).

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 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 confirming 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 Fig. 4(a)); (ii) otherwise, a vertex can be added to the positions ranging from $parent(v_k)$ to v_k (see also Fig. 4(b)). To keep a new tree left-heavy, candidate labels for a new added vertex v_{k+1} are determined by confirming subtrees including v_{k+1}. Since the label

```
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 $val : \Sigma \to Z^+$
Output A set of all possible simple trees $\mathcal{R}$ which are normal trees
BfsSimEnum($\Sigma$, $val$, $\{n_i\}$)
  $\mathcal{R} := \emptyset$
  for each $l_j \in \Sigma$ such that $val(l_j) > 1$ do
    $T$ := a tree consisted of a root with $l_j$
    AddNode($\Sigma$, $val$, $\{n_j\}$, $T$, $\mathcal{R}$)
  return $\mathcal{R}$
end
AddNode($\Sigma$, $val$, $\{n_j\}$, $T$, $\mathcal{R}$)
  $n'$ := 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 $T$, resp.
    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 $v_{k+1}$
        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($\Sigma$, $val$, $\{n_j\}$, $T'$, $\mathcal{R}$)
end
```

Fig. 5 Pseudo code of BfsSimEnum for simple tree enumeration.

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}.

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.

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:

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

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 Fig. 7, 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 $2\mathcal{M}_2 + 4\mathcal{M}_3 = 2\cdot4+2\cdot2-2\cdot(2+2)-2+2 = 4$, which means that target molecular trees of this example should have two double bonds or one triple bond such that $(\mathcal{M}_2, \mathcal{M}_3) = (2, 0), (0, 1)$.

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 Fig. 6. 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 confirm whether or not a new tree is in normal form by comparing the edge multiplicity together with confirming the feasibility of setting other multi-edges when generation for such a new tree is still continued.

```
Input $\mathcal{M}_2$, $\mathcal{M}_3$ and output $\mathcal{R}$ of BfsSimEnum
Output a set of all possible multi-trees $\mathcal{R}_m$
BfsMulEnum ($\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}$ )
  $\mathcal{R}_m := \emptyset$
  while $|\mathcal{R}| \neq 0$ do
    $T$ := a tree of $\mathcal{R}$
    remove $T$ from $\mathcal{R}$
    AddMultiedge ($T$, root of $T$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
  return $\mathcal{R}_m$
end
AddMultiedge ($T$, $v_i$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
    if $\mathcal{M}_2 = 0$ and $\mathcal{M}_3 = 0$ and $T$ is a normal tree then
      $\mathcal{R}_m := \mathcal{R}_m \cup \{T\}$
    else
      AddMultiedge( $T$, $v_{i+1}$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{R}_m$)
      $miss(v_i) := val(l(v_i)) - degree(v_i)$
      $miss(parent(v_i)) := val(l(parent(v_i))) - degree(parent(v_i))$
      if $miss(v_i) \geq 1$ and $miss(parent(v_i)) \geq 1$ and $\mathcal{M}_2 > 0$ then
        $T_2 := T$
        set double bond to $(parent(v_i), v_i)$ in $T_2$
        AddMultiedge( $T_2$, $v_{i+1}$, $\mathcal{M}_2 - 1$, $\mathcal{M}_3$, $\mathcal{R}_m$)
      if $miss(v_i) \geq 2$ and $miss(parent(v_i)) \geq 2$ and $\mathcal{M}_3 > 0$ then
        $T_3 := T$
        set triple bond to $(parent(v_i), v_i)$ in $T_3$
        AddMultiedge( $T_3$, $v_{i+1}$, $\mathcal{M}_2$, $\mathcal{M}_3 - 1$, $\mathcal{R}_m$)
end
```

Fig. 6 Pseudo code of BfsMulEnum for multi-tree enumeration.

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 any structural changes. Fig. 7 illustrates the process of both BfsSimEnum and BfsMulEnum.

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 Enu-

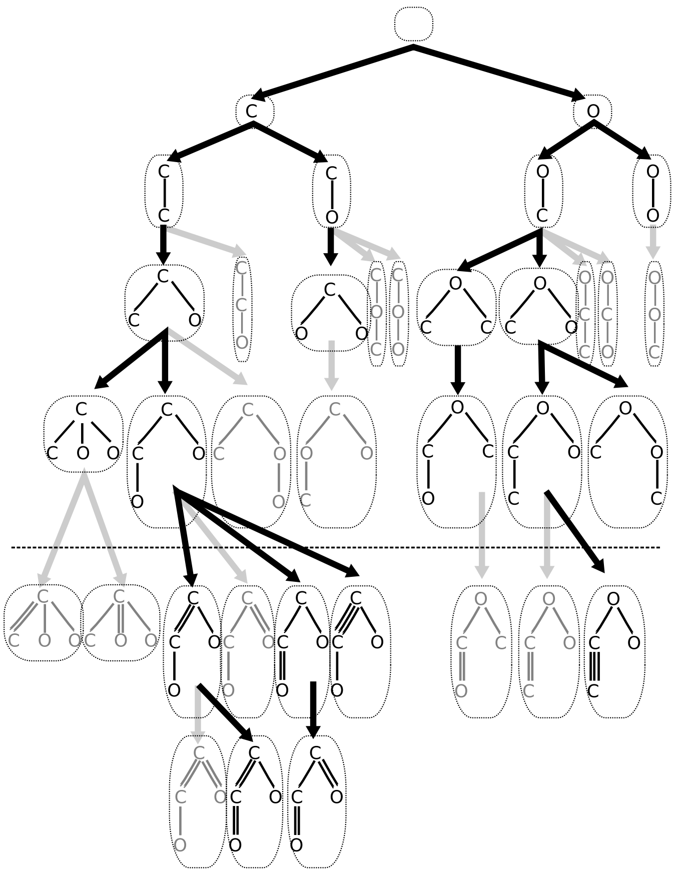

Fig. 7 Illustration of BfsSimEnum and BfsMulEnum. BfsSimEnum is processed above the dot line; BfsMulEnum is processed below the dot line. Graphs in gray color are considered as invalid by the algorithms and thus are not stored or proceeded any more. It should be noted that hydrogen atoms are added as leaves at last.

Mol. 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.

5. Conclusion

In this technical report, 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. For multi-trees enumeration, however, 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 constructing intermediate trees. Such an extension can significantly reduce search space because we can decrease the possibility to expand simple trees which no longer can be replaced with multi-trees.

Another extension to our methods is to deal with more complex ring structures such as naphthalenes. 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.

Appendix

References

[1] Faulon, J. L. and Bender, A.: *Handbook of Chemoinformatics Algorithms*, CRC Press (2010).

[2] Pretsch, E., Bühlmann, P. and Badertscher, M.: *Structure Determination of Organic Compounds*, Springer-Verlag Berlin Heidelberg (2009).

[3] Deshpande, M., Kuramochi, M., Wale, N. and Karypis, G.: Frequent substructure-based approaches for classifying chemical compounds, *IEEE Trans. Knowledge and Data Engineering*, Vol. 17, pp. 1036–1050 (2005).

[4] Horváth, T. and Ramon, J.: Efficient frequent connected subgraph mining in graphs of bounded tree-width, *Theoretical Computer Science*, Vol. 411, pp. 2784–2797 (2010).

[5] Jiang, C., Coenen, F. and Zito, M.: A survey of frequent subgraph mining algorithms, *The Knowledge Enginering Review*, Vol. 28, pp. 75–105 (2013).

[6] Rouvray, D. H.: The pioneering contributions of Cayley and Sylvester to the mathematical description of chemical structure, *Journal of Molecular Structure*, Vol. 1, p. 54 (1989).

[7] Faulon, J. L., D. P. Visco, J. and Rose, D.: Enumerating molecules, *Reviews in Computational Chemistry*, Vol. 21, pp. 209–286 (2005).

[8] Dobson, C. M.: Chemical space and biology, *Nature*, Vol. 432, pp. 824–828 (2004).

[9] Gugisch, R., Kerber, A., Kohnert, A., Laue, R., Meringer, M., Rucker, C. and Wassermann, A.: *Molgen 5.0, a Molecular Structure Generator*, Bentham Science Publishers Ltd. (2012).

[10] Ishida, Y., Kato, Y., Zhao, L., Nagamochi, H. and Akutsu, T.: Branch-and-bound algorithms for enumerating treelike chemical graphs with given path frequency using detachment-cut, *Journal of Chemical Information and Modeling*, Vol. 50, No. 5, pp. 934–946 (2010).

[11] Fujiwara, H., Wang, J., Zhao, L., Nagamochi, H. and Akutsu, T.: Enumerating treelike chemical graphs with given path frequency, *Journal of Chemical Information and Modeling*, Vol. 48, No. 7, pp. 1345–1357 (2008).

[12] Shimizu, M., Nagamochi, H. and Akutsu, T.: Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies, *BMC Bioinformatics*, Vol. 12, No. Suppl 14, pp. 1–9 (2011).

[13] Akutsu, T. and Nagamochi, H.: Comparison and enumeration of chemical graphs, *Computational and Structural Biotechnology Journal*, Vol. 5, No. 6, p. e201302004 (2013).

[14] Imada, T., Ota, S., Nagamochi, H. and Akutsu, T.: Efficient enumeration of stereoisomers of outerplanar chemical graphs using dynamic programming, *Journal of Chemical Information and Modeling*, Vol. 51, pp. 2788–2807 (2011).

[15] Akutsu, T., Fukagawa, D., Jansson, J. and Sadakane, K.: Inferring a graph from path frequency, *Discrete Applied Mathematics*, Vol. 160, pp. 1416–1428 (2012).

[16] Zhao, Y., Hayashida, M., Jindalertudomdee, J., Nagamochi, H. and Akutsu, T.: Breadth-first search approach to enumeration of tree-like chemical compounds, *Journal of Bioinformatics and Computational Biology* (to appear).

[17] Nakano, S. and Uno, T.: Generating colored trees, *Lecture Notes in Computer Science*, Vol. 3787, pp. 249–260 (2005).

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

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