Chemical Compound Enumeration and Host-Pathogen Protein Interaction Prediction by Graph-Based Approaches

グラフ構造に基づく手法による化合物の列挙および宿主・病原体間のタンパク質相互作用予測

Jira Jindalertudomdee

September 2016

Abstract

The rapid development of technology and experiment leads to the large amount of data to be analyzed and studied. The representation of data is important because it has an affect on the complexity of data analysis. Because of its versatility and representation power, a graph has been used as a data structure to represent several biological and chemical data such as protein interaction data, cell metabolism data, and a chemical compound in the form of a protein-protein interaction network, a metabolic network, and a molecular graph, respectively. Moreover, representing data by a graph is one method to apply mathematical knowledge, such as graph algorithm, to examine those data. This thesis makes use of a graph representation of biological and chemical data to solve two chemoinformatics and bioinformatics problems, which is the first part and the second part of this thesis, repectively.

In the first part of this thesis, we propose two efficient chemical compounds enumeration methods from a chemical formula using a tree structure, an acyclic connected graph, to represent a chemical compound. The enumeration algorithm can be applied to several problems including drug discovery by filtering the unnecessary compounds out and returning only all non-redundant compounds with the desired properties. The first proposed method is the enumeration method for tree-like chemical compounds containing benzene rings and naphthalene rings from a chemical formula. The second enumeration method allows users to input a chemical formula and desired cyclic substructures and generates all non-redundant chemical compounds containing no cycles except for the input substructures. Two proposed algorithms can help the chemists spend less time to discover compounds with the desired number of atoms and substructures. Both of them use a tree structure to represent a chemical compound, where nodes and edges denote atoms and chemical bonds, respectively. One cyclic structure (a benzene ring in the first method and an input substructure in the second method) is compressed into one node with an additional attribute to keep how atoms in the cyclic structure bond with adjacent atoms, while a naphthalene ring in the first method is denoted by two benzene nodes bonding together with a special bond. The results and computational time of proposed methods are compared with those of a commercial general purpose structure generator named *MOLGEN*. Giving the same number of the enumerated compounds, proposed methods are significantly faster than the existing tool. This result shows both the reliability and the efficiency of two proposed methods.

The second part is the prediction of the interaction between host proteins and pathogen proteins using a feature extracted from a protein-protein interaction network, called a graphlet degree vector. It has been shown that: 1) different pathogens tend to target proteins involving in the same biological pathway, and 2) the proteins with similar function are not always close to each other but share common topological structure in the protein-protein interaction network. Accordingly, we hypothesize that pathogen proteins interact with host proteins that have similar topology in the protein-protein interaction network and used a graphlet degree vector as a representation of a topological structure of the protein-protein interaction network to test that hypothesis. We used stochastic gradient descent method with logistic loss and soft confidence-weighted learning method to train the prediction model. Then, we predicted the protein-protein interaction between human and four pathogens and used F-score of 10-fold cross validation to evaluate the accuracy of the models. The average accuracy of the proposed method is better than the existing method, that does not use a graphlet degree vector as a feature, for all pathogens. The results suggest that a graphlet degree vector is an excellent feature for solving host-pathogen protein interaction prediction problem.

Acknowledgments

First and foremost, I would like to express my sincere thankfulness to my supervisor, Professor Dr. Tatsuya Akutsu, for the continuous support throughout 5 years and a half in this laboratory since I was a research student until Ph.D. course. His beneficial advice, encouragement, and inestimable guidance have assisted me conducting the research and writing this dissertation, improved my critical thinking skill, and inspired me to pursue research as my future career. He has provided me several research-related opportunities, which lets me gain a lot of valuable experience and become a better researcher.

Besides my supervisor, I would like to thanks Professor Dr. Hiroshi Nagamochi. With his precious comments and suggestion, my research regarding the enumeration method for tree-like compounds containing benzene rings and naphthalene rings was significantly enhanced.

My thanks also goes to Assistant Professor Dr. Morihiro Hayashida for the academic support and advice, which help me conducting my research smoothly. He also helped me gained a lot of confidence in my first conference by proofreading my presentation, which was in Japanese. I thank Assistant Professor Dr. Takeyuki Tamura and secretary Ms. Tamami Fukushiro for their kind support in scientific and administrative issues.

I must also acknowledge the Japanese Ministry of Education, Culture, Sports, Science, and Technology (MEXT) for supporting my daily expense and tuition fee during my graduate school life. MEXT also provided me a scholarship for purchasing a structure generator software, which is necessary for my research.

I would also like to thanks the committee of this disseration: Professor Dr. Tatsuya Akutsu, Professor Dr. Akihiro Yamamoto, and Professor Dr. Yasuo Okabe for the significant review and suggestion to ensure the quality of this work.

I wish to thank Kyoto for being a nice and beautiful city to live for almost six years. Having an opportunity to visiting a plenty of marvellous temples and shrines as well as attending wonderful cultural events always makes me enjoys my Ph.D. life and refreshes me from working stress.

Last but not the least, I would like to thank my family for supporting me spiritually throughout the completion of this work as well as all of my friends for encouraging me and sharing good memories together.

Publication Notes

Chapter 2 is based on the paper [38], which is published in *BMC Bioinformatics.*
Chapter 3 is based on the paper [37], which is published in *Journal of Computational Biology.*
Chapter 4 is based on the paper accepted for IEEE 16th International Conference on BioInformatics and BioEngineering.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Graph is an abstract data type utilized in various fields of computer science including bioinformatics and chemoinformatics because it can capture the structure of the data it representing and make the data easier to be analyzed. An example of the utilization of graph in bioinformatics is a representation of a metabolic network, a network of chemical reactions of compounds in a living cell. A metabolic network is represented by a Boolean network, where a node denotes either a reaction or a compound and an edge connecting a compound and a reaction together if that compound is either a reactant or a product of that reaction.

For chemoinformatics, a chemical compound can be simply converted to a graph by regarding an atom as a node colored by a chemical element and connecting two nodes by an edge if there is a bond between two corresponding atoms. Colored graph generating algorithm can be modified to solve chemical compound enumeration algorithm, which is a problem of generating non-redundant chemical compounds from given constraints, by treating a color of nodes as a chemical element. The frequently used constraint in the enumeration problem is a chemical formula, which corresponds to the number of nodes for each color. The enumeration problem is a fundamental problem in chemoinformatics since it can be applied to many practical problems including drug design as well as structure elucidation of an unknown compound by generating a library of candidate compounds from the desired number of atoms.

Another graph representation of biological data is a protein-protein interaction network (PIN). A PIN is a graph of interacting proteins, where a node denotes a protein and an edge connecting two interacting proteins. Some properties of proteins can be implied from a PIN using graph algorithms or graph properties. For instance, the higher the degree of a protein node in a PIN, the more other proteins it interacts with, which reflects how important it is. There are several researches study the relation of PIN and other biological problems such as host-pathogen protein interaction prediction, which is a prediction of the interaction of proteins between a host and a pathogen.

Infectious diseases, such as HIV, lower respiratory infections, and diarrhoeal, are caused by various pathogens such as bacteria, viruses, and fungi. Because infectious diseases can spread from one person to another person, they have been the main cause of death in human since 1990 [60] and are also predicted to be one of the leading causes in 2030 [50]. Accordingly, discovery of drugs being able to cure those diseases or prevent the transmission of infectious diseases can significantly decrease human death and illness.

Because most drug targets are host proteins interacting with pathogen proteins [21], such as cell membrane receptors and enzymes, several computational methods to predict the protein-protein interaction between human and pathogens have been developed to find a drug target. Development of a prediction method with high accuracy remains a challenge and is important to find the correct and efficient drug targets. PIN is introduced into this problem because it was studied that pathogens tend to target hubs protein in a human PIN [23].

After a drug target is known, we have to design a molecule which can interact with the drug target to inhibit the mechanism involved in the disease. Treating the receptors interacting with the drug target as molecular fragments of the desired compounds, we can generate a list of candidate drugs using a structure enumeration tool and find an optimal drug from that list. Therefore, an efficient structure enumeration tool, which can generate a compound library from receptors using low computational time, decreases the time spent to search for an optimal drug and leads to the improvement of drug design process.

In this thesis, our goal is to provide three efficient computational methods using a graph data structure in the algorithm. The first two methods are two novel efficient chemical compound enumeration methods representing a chemical compound by a tree structure, a connected graph without cycle. These proposed methods optimize the discovery of a target compound in term of execution time. The last method is a host-pathogen protein interaction prediction method based on a graph-based feature of PIN to discover a protein which is suitable to be a drug target. These methods can be applied to search for a novel therapeutics, which can decrease the consequence of disease. In order for audience to understand the content without referring to other documents, we give knowledge related to this work in this chapter.

1.2 Enumeration problem

Chemoinformatics is a field of applying knowledge and techniques in computer science to analyze and solve problems in chemistry. The main goal of chemoinformatics is to save resources in terms of time and labor spent to solve that problem. One of the fundamental problems in chemoinformatics is the enumeration problem. Enumeration is a problem of generating all non-redundant chemical compounds satisfying the provided constraints.

Those constraints are properties of chemical compounds that users would like to find, which can be either physical properties, such as the number of atom types, the number of fragments and molecular weight, or chemical properties, such as hydrophobicity and polarity. An example of enumeration using the number of atom types is illustrated in Figure 1.1.

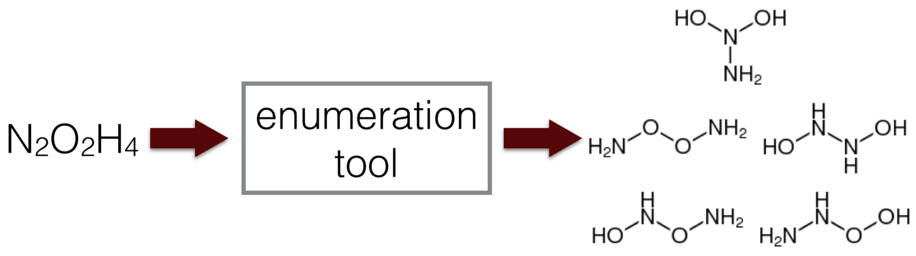
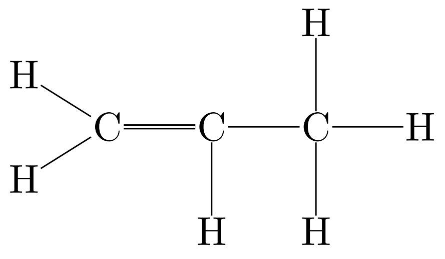


Figure 1.1: Illustration of input and output of the enumeration problem where constraint is the number of atom types

The main point of an enumeration algorithm is to reduce the size of chemical space from all possible compounds to only compounds satisfying the given constraints and guarantee that all such compounds are considered. Decreasing the size of chemical space shortens the time used to find the desired compound significantly. Depending on the constraints and the desired compound, an enumeration algorithm can be used to solve various kinds of problem [67], such as drug design by constructing chemical compounds from a set of atoms and/or fragments that have desired pharmacological properties [74], molecular design using chemical properties as the input [58], and structure identification by constructing a collection of molecules satisfying a given spectral data of unknown compound and search for the exact molecule from that collection [59, 72, 79, 91].

One of the example of the application of enumeration problem is the inverse quantitative structure-activity relationship (inverse QSAR) problem [17, 49], which is a problem of discovering a collection of compounds that are predicted to have given properties. First, those algorithms implement a QSAR model using an *atomic signature* as a molecular descriptor of a compound to reduce the chemical search space. An atomic signature defined in these works is the number of all fragments in a chemical compound consisting of the specified atom and all atoms within a given distance from that specified atom. An example of an atomic signature is shown in Figure 1.2, where the given distance is one. After that, they examine whether the predicted atomic signature satisfies two constraint equations or not to ensure that the compound can be reconstructed from the solution of equations. Then, the enumeration tool is utilized to enumerate chemical compounds from the validated atomic signature. Finally, the QSAR is applied one more time to ensure that all of the results satisfy given properties.

Because of its versatility, a numerous number of enumeration algorithms have been



Atomic Signature	
1	C(HHH)
1	C(CH=C)
1	C(HH=C)
6	H(C)

Figure 1.2: A chemical compound and its corresponding atomic signature with distance one.

developed for several decades since DENDRAL project [9] such as CONGEN [12] a structure generator, which is a component of DENDRAL project, GENOA [13] a later version of CONGEN, SMOG [56] an enumeration software based on a graph-theoretical algorithm, MOLGEN [7, 30, 40, 92] a commercial general-purpose structure generator, EnuMol [28, 35, 78] an exact enumeration algorithm for treelike chemical graphs, OMG [63] an open source structure generator, and a novel method for tree-like chemical graphs with naphthalene nodes [33].

These existing algorithms use various kinds of technique to generate compounds without redundancy as listed below.

- CONGEN enumerates chemical compounds by three main steps. First, it generates a composition list and a constraint list from the input. A composition list is a list of atoms and fragments of the desired compound and a constraint list is a list of undesired and desired substructures. Then, it enumerates compounds via the repetition of two steps, the generate step and the imbed step. The generate step generates intermediate structures by adding available atoms or desired fragments (called superatoms). If the added component is a superatom, the imbed step restores the superatom back to its original structure, connects atoms in the superatom with other atoms in all possible ways, and removes the duplicate ones. The complete structures are checked for the constraints in a constraint list at the final step.

- GENOA allows only one additional constraint, which is a fragment and the range of the number of that compound. It generates intermediate structures by adding one fragment/atom to the compound at a time and uses the concept of constructive substructure search to find all structures resulting from the overlap between the added fragment and fragments in the intermediate structures. For each overlapped structure found, it extends the structure in that many ways and keeps adding new fragments/atoms until all fragments and atoms are added to the structure.

- SMOG uses an adjacency matrix to store the information of a molecular graph. It starts from an empty adjacency matrix, and fills the entries corresponding to the input superatoms, which are fixed throughout the enumeration. Then, it fills the elements in the adjacency matrix row by row and checks whether it is a canonical matrix or not based on the lexicographical order of the entries. At the last step, it checks whether the molecular graphs satisfy the input constraints, e.g. connectivity, good lists, and bad lists.

- MOLGEN represents a compound by two data structures, a non-hydrogen adjacency matrix and a hydrogen vector. To enumerate chemical compounds, first, it generates all possible hydrogen vectors from the input chemical formula. Then, it generates all canonical matrices for each hydrogen vector in the previous step using the orderly generation approach to prune the redundant matrices as soon as possible.

- EnuMol focuses on tree-like chemical compounds containing benzene rings, because it uses a tree structure to represent a chemical compound for the efficiency of the enumeration. The input of enumol is not the number of atoms but the number of paths represented by a feature vector. It starts from an empty tree and adding a node to the tree using the proposed branch-and-bound algorithm. In branching step, a new leaf node is attached to the tree such that the left-heavy condition is preserved. In bounding step, it prunes the trees, which violate at least one of three constraints: 1) the center-rooted constraint, 2) the feature vector constraint, and 3) the valence constraint. These two steps are repeated until the number of paths in the tree is the same as that in the feature vector.

- OMG starts from a set of fully disconnected atoms and continuously connects atoms until the number of bonds of each atom equals to its valence. The enumeration process is considered as a family tree of molecular graphs, where a node of a family tree is a molecular graph.The child node in a family tree can be obtained by adding a bond to a pair of atoms in its parent node or adding the multiplicity of an existing bond by one. After a bond or the multiplicity of a bond is added, it checks for the canonicity by a graph canonizer which calculates the canonical labeling of a multigraph and compares the current molecular graph with the result of the canonizer.

- The combination of BfsSimEnum and BfsMulEnum uses a tree to represent a chemical compound. It consists of two main parts, BfsSimEnum and BfsMulEnum. First, BfsSimEnum generates compounds using the concept of a family tree similar to OMG except that it starts from an empty tree instead of a fully disconnected graph and the child node of a family tree is obtaned by adding an atom to the tree of its parent. After adding an atom, it checks for the canonicity using the proposed concept called a center-rooted left-heavy condition. If all atoms are added, BfsSimEnum passes the chemical compounds to BfsMulEnum, which adds the proper multiplicity of the existing bonds such that the degree of atoms does not exceed their valence. The final result of this method is the tree structures representing acyclic chemical compounds.

- An efficient algorithm for the enumeration of naphthalene isomers of tree-like chemical graphs proposed in [33] enumerates tree-like compounds containing naphthalene rings from a tree structure containing naphthalene nodes in two steps. First, it counts the number of all isomers obtained from the input tree based on dynamic programming. Next, it applies the backtracking technique to the counting computation to generate each isomer counted in the first step.

These existing enumeration tools can be classified into two main groups based on the structure of the enumerated compounds. The first group has no limitation on the structure that can be enumerated, which are CONGEN, GENOA, SMOG, MOLGEN, OMG. However, enumeration tools in this group consume high computational time. Another group has a limitation on the enumerated structure such as only tree-like chemical compounds for BfsSimEnum and BfsMulEnum, only tree-like chemical compounds with benzene rings for EnuMol, or tree-like compounds with naphthalene rings for [33] but this group requires significantly less computational cost.

In this work, we proposed two novel efficient enumeration methods. The first one is an enumeration method for tree-like chemical compounds with benzene rings and naphthalene rings, a bicyclic aromatic compound, whose detail is explained in Chapter 2. Another method explained in Chapter 3 allows users to input desired cyclic substructures and enumerate tree-like chemical compounds without cycles except for the specified substructure. Both of them use a molecular graph explained in Section 1.3.1 to represent a chemical compound. These methods are proposed with the main objective of combining the advantages of both groups of the existing enumeration tools together. Compared with the first group, the proposed methods can be executed in a short period of time. At the same time, the proposed methods are able to enumerate chemical compounds with more complex cyclic structures than the tools in the second group. Although BfsBenNaphEnum and the one proposed in [33] can enumerate the same scope of chemical compounds, which are tree-like compounds containing naphthalene rings, BfsBenNaphEnum has been developed before [33]. Moreover, BfsBenNaphEnum is simpler than [33], which makes it can be extended to more complex structures.

1.3 Molecular graph and its redundancy

1.3.1 Molecular graph

To use a computer to enumerate chemical compounds, a data structure representing a chemical compound must be introduced. The most popular representation is a graph because its structure is similar to that of a chemical compound. A graph whose nodes represent atoms and edges represent bonds is called a *molecular graph* [52]. Thus, nodes and edges of a molecular graph are labeled by atom types and bond types, respectively.

In other words, letting Σ be a set of atom types, e.g. {C,N,O,H}, a molecular graph is defined as a graph $G(V, E)$, where V and E are a set of nodes and a set of edges of G, respectively. Label of any nodes v in V, $l(v)$, must be an element of Σ ($l(v) \in \Sigma$). Let $val(l_i)$ denote a valence of an atom type l_i in Σ. A covalent bond is a sharing of electron between two atoms. A covalent compound is a compound whose all atoms bonding with covalent bonds only. It is stable when the number of covalent bond of atoms in that compound equals to their valence. Therefore, degree of a node with atom type l_i equals to $val(l_i)$ in order for a compound to be stable. An example of a molecular graph representing ethane is illustrated in Figure 1.3, where C and H denote carbon and hydrogen atoms, respectively, $val(C) = 4$ and $val(H) = 1$.

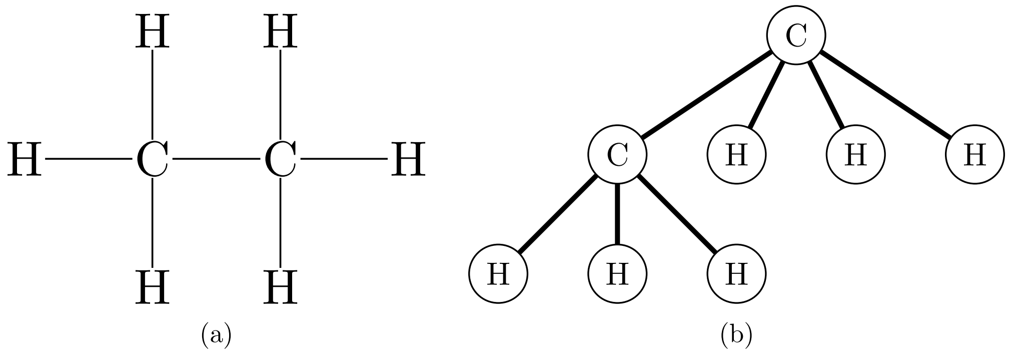


Figure 1.3: Ethane (A) and a molecular graph representing ethane (B)

A tree structure is a connected graph containing no cycles [94] which makes tree structures easier to detect the redundancy than a graph. Therefore, the enumeration of non-redundant tree structures is more efficient than that of non-redundant graphs. In order to optimize the algorithm, this work focuses on tree-like chemical compounds or chemical compounds without cycles so that a chemical compound can be represented by a tree structure. We call a tree structure whose nodes are labeled by atom type and edges are labeled by bond type as “a molecular tree”. The molecular tree is firstly used in the enumeration to count the number of alkane compounds (compounds with chemical formula C_nH_{2n+2}) by counting the number of non-redundant trees with n nodes and all nodes have degree four or less [15]. Those nodes represent carbon atoms of the compound, while hydrogen atoms are ignored because there is only one way to add hydrogen atoms to the compound such that the compound is valid (the degree of an atom equals to its valence). An example of this molecular tree in this enumeration is illustrated in Figure 1.4.

In this work, we use a rooted-ordered molecular tree to represent a chemical compound. A rooted tree is a tree, $G(V, E, r)$, where r is *a root node*. The root node is a node distinguished from other nodes [93] and located at the top of the tree. Given a node v in

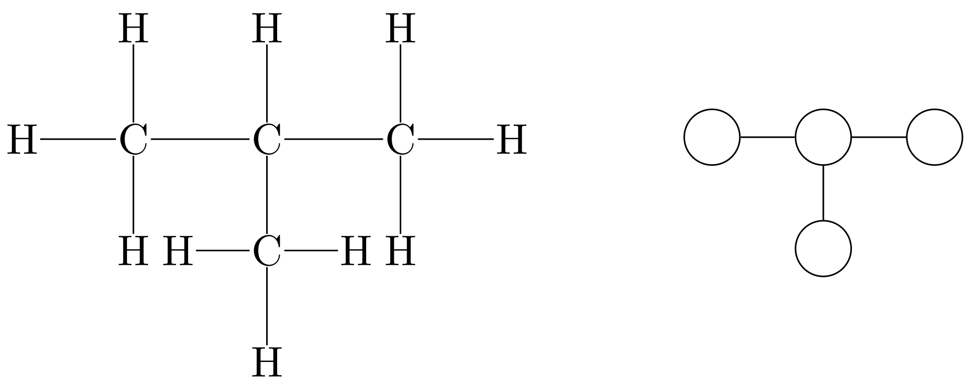


Figure 1.4: An alkane compound and its corresponding molecular tree

G, an adjacent node of v which lies in the path from v to r is *a parent node* of v and all other adjacent nodes are *the child nodes* of v. Each node, except the root node, in the tree must have exactly one parent node unless there are two paths from that node to the root node, which implies that there is a cycle in a graph. Nodes whose parent node is the same node are *sibling nodes*.

A rooted-ordered tree is a rooted tree, $G(V, E, r)$, such that for all nodes $v \in V$ there exists a function $child(v)$ that return a list of child nodes of v from the leftmost child node to the rightmost child node. If u is a child node of v and u is the i^{th} element in $child(v)$, u is the i^{th} child node of v.

1.3.2 Detecting graph redundancy by isomorphism

Isomorphism is known as a way to compare the equivalence of two graphs. Given two graphs, $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, isomorphism between G_1 and G_2 is a one-to-one mapping function $\theta : V_1 \to V_2$ that preserves the adjacency [31]. In other words, θ must satisfy the following properties.

1. If G_1 and G_2 are labeled graphs, v must have the same label as $\theta(v)$.

2. For any two nodes v_1 and v_2 in V_1, if $(v_1, v_2) \notin E_1$, then $(\theta(v_1), \theta(v_2)) \notin E_2$. Otherwise, $mul(v_1, v_2) = mul(\theta(v_1), \theta(v_2))$ must hold, where $mul(u, w)$ is the multiplicity of edge connecting node u and node w.

Two graphs G_1 and G_2 are isomorphic if there is an isomorphism between G_1 and G_2. The example of isomorphism is shown in Figure 1.5, where the red edges are the mapping of nodes corresponding to the isomorphism between two rooted trees.

In the case of rooted trees, given two rooted trees, $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$, they are redundant if T_1 and T_2 are isomorphic and that isomorphism also maps r_1 with r_2. For example, trees in Figure 1.6 have the uppermost node whose label is A as their root nodes. Those trees are the same rooted trees because they are isomorphic and have the same corresponding root node.

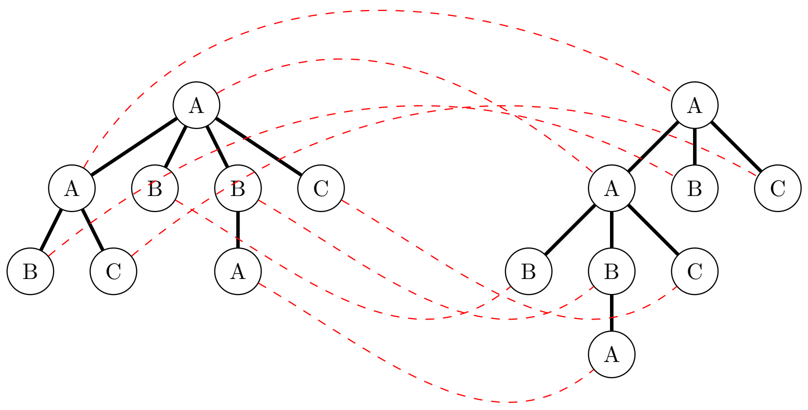


Figure 1.5: Examples of two isomorphic labeled graphs where dashed lines indicate the correspondences between nodes of two graphs

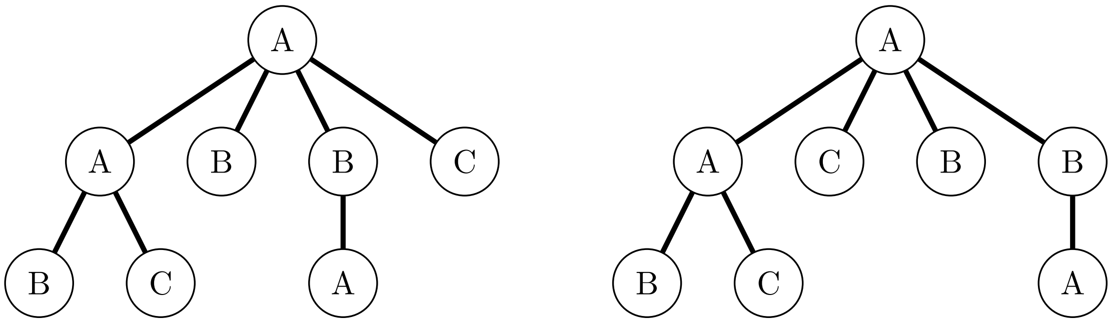


Figure 1.6: Examples of rooted trees that are different ordered trees but the same unordered trees

For rooted-ordered trees, the function *child*(v), which returns a list of child nodes of v from the leftmost child node to the rightmost, is taken into account in the comparison. Two rooted-ordered trees, $T_1(V_1, E_1, r_1)$ and $T_2(V_2, E_2, r_2)$, are redundant if there is an isomorphism θ such that three following consitions are satisfied.

1. θ maps V_1 to V_2,

2. $\theta(r_1) = r_2$, and

3. $\forall u \in V_1$, θ maps the i^{th} element of $child(u)$ to the i^{th} element in $child(\theta(u))$.

For example, trees in Figure 1.6 are different rooted-ordered trees because θ maps the second element of $child(r)$ to the third element of $child(\theta(r))$, where r is the root node of the left tree.

We use a rooted-ordered tree, instead of a graph, to represent a chemical compound because it takes significantly less time to examine the redundancy of the rooted-ordered trees than the redundancy of graphs. To check for the redundancy of two graphs, we have to utilize a graph canonizer to find a canonical form of the structure every time a graph is modified, e.g. a node is added, which consumes a lot of computational time. On the other hand, we can eliminate the redundancy of rooted-ordered trees by defining a rule to select which node to be the root node and a rule to arrange the order of sibling nodes. The redundancy of the tree can be examined only by checking whether the generated trees violate the defined rule or not.

We convert a molecular graph into a rooted-ordered tree by three main steps. First, we generate the corresponding tree structure from a graph by replacing each cyclic substructure with a single node and connecting that node with nodes bonding with any atoms in the corresponding substructure (step (1) Figure 1.7). These nodes contain the information of how atoms in that substructure bond with their adjacent nodes. Then, we generate the corresponding rooted tree by selecting a node to be the root node (step (2) Figure 1.7). Based on the center-rooted condition, the center node of the longest path is the root node. Finally, we sort the sibling nodes in the rooted tree according to the left-heavy rule, which results in a canonical rooted-ordered tree (step (3) Figure 1.7).

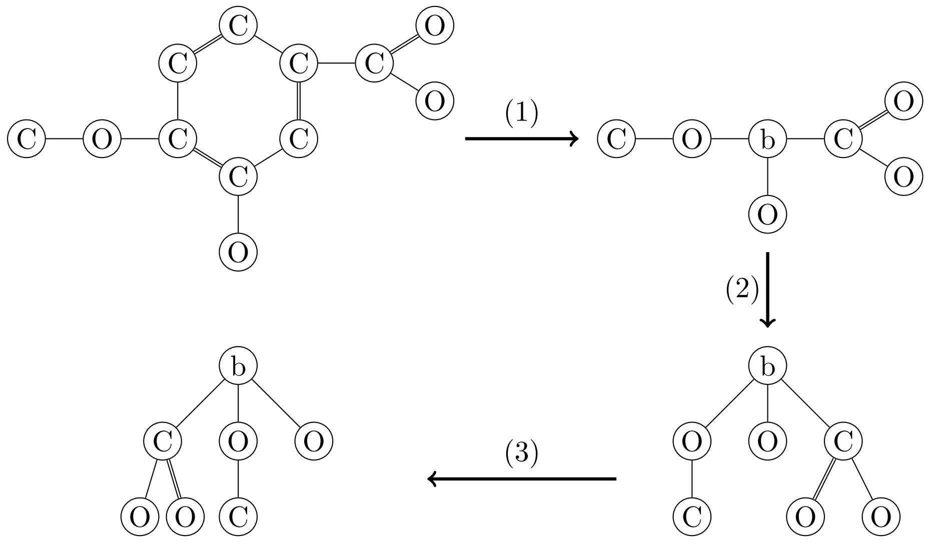


Figure 1.7: A graph to a rooted-ordered tree conversion process

Let C_1 and C_2, be two chemical compounds represented by two isomorphic molecular trees, T_1 and T_2, respectively. C_1 and C_2 are the same compound because the isomorphism guarantees that the number of nodes with the same atom types in T_1 and T_2 and the way they bond with each other are the same. An example of two isomorphic molecular trees is given in Figure 1.8. Because the enumeration must guarantee non-redundancy of chemical compounds, only one out of all isomorphic molecular trees must be generated.

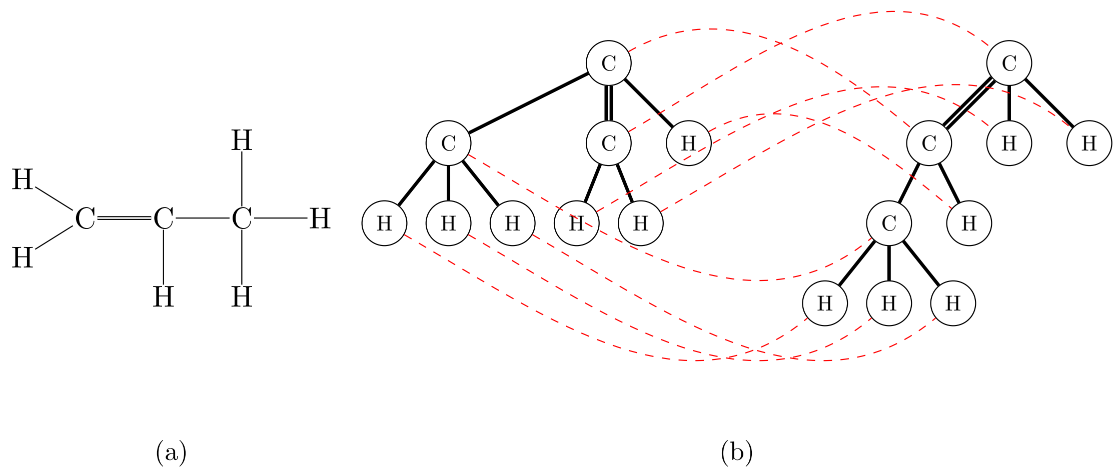


Figure 1.8: Examples of two isomorphic molecular trees (b) representing propene (a) where dashed lines indicate the correspondence between atoms of two molecular trees

1.4 Host-pathogen protein interaction

1.4.1 Fundamental of protein

Protein, the main component in a living cell, is a sequence of amino acids. More than half of dry weight of cell is proteins [18]. Because there are 20 kinds of amino acid as shown in Table. 1.1, there are several ways to combine those amino acids together, which results in the existence of abundant kinds of protein. A protein is synthesized from a DNA, a polynucleotide, via two processes. First, an RNA, which is also a polynucleotide, is generated from a DNA via a transcription process. Then, a protein is translated from that RNA via a translation process. Because there are only four kinds of nucleotide, one amino acid is encoded by three consecutive nucleotides, called a codon, in the RNA.

Proteins are essential for living organisms because they perform various functions in order for the cell to grow and develop smoothly. For example, enzymes stimulate the chemical reactions such as metabolism in a cell, transport proteins in the cell membrane control the transportation of particles inward and outward the cell, etc. In nature, a protein does not stay as a long thin sequence but folds into a specific shape as a result of weak noncovalent bonds, such as hydrogen bonds and van der Waals attraction [2]. The same kind of protein always folds into the same specific shape called a *conformation* of that protein with a minor change when it binds with another molecule. However, the mechanism underlying the protein folding has still not been discovered yet.

Although a conformation is the final structure of one protein, the structure of protein is not only the conformation but can be classified into four levels, which are primary, secondary, tertiary, and quaternary structure. The primary structure, which is the simplest level of protein, is the unfolded amino acid sequence. The secondary structure is the form-

Amino acid	Abbreviation	Corresponding codons
Isoleucine	Ile	AUU, AUC, AUA
Leucine	Leu	CUU, CUC, CUA, CUG, UUA, UUG
Valine	Val	GUU, GUC, GUA, GUG
Phenylalanine	Phe	UUU, UUC
Methionine	Met	AUG
Cysteine	Cys	UGU, UGC
Alanine	Ala	GCU, GCC, GCA, GCG
Glycine	Gly	GGU, GGC, GGA, GGG
Proline	Pro	CCU, CCC, CCA, CCG
Threonine	Thr	ACU, ACC, ACA, ACG
Serine	Ser	UCU, UCC, UCA, UCG, AGU, AGC
Tyrosine	Tyr	UAU, UAC
Tryptophan	Trp	UGG
Glutamine	Gln	CAA, CAG
Asparagine	Asn	AAU, AAC
Histidine	His	CAU, CAC
Glutamic acid	Glu	GAA, GAG
Aspartic acid	Asp	GAU, GAC
Lysine	Lys	AAA, AAG
Arginine	Arg	CGU, CGC, CGA, CGG, AGA, AGG

Table 1.1: Twenty amino acids and their corresponding codons

ation of weak bond between atoms in the local region, which results in α-helix structure or β-sheet structure, the two most frequent folding patterns. The final form of folded protein is its tertiary structure, which is a conformation of protein. This tertiary structure consists of one or more *protein domains*, which are parts of amino acid sequence whose foldings are independent to each other. The quaternary structure is a binding of several proteins to form a protein complex.

1.4.2 Protein interaction

A protein does not perform its function alone but usually binds with other proteins to form a protein complex. The binding between proteins is considered as a physical protein-protein interaction, which is the important mechanism of protein to perform its function. Most of the physical interaction is caused by weak non-covalent bonds because of the reversibility, flexibility, and stability of those bonds. However, a few proteins, such as collagen, interact with each other by the covalent bond. Apart from physical interaction, there is another type of interaction between proteins called a functional interaction. Functional interaction is the interaction such that proteins do not contact each other but are subunits of the same protein complex or are involved in the same biological pathway [18].

The protein complex can bind with other substances, which can be ions or molecules. The substance binding with protein is called *ligand*. Both protein-protein binding and protein-ligand binding are extremely specific because the binding regions of one protein must exactly match with those of another protein or ligand in order to interact with each other. This means a protein can bind to only one or a few specific proteins/ligands.

1.4.3 Interaction between host and pathogen

Pathogens are microbes that cause the infectious disease, which is one of the leading causes of human's death, by interrupting or damaging their hosts. The interaction between microbes and human does not always cause the disease to human, e.g. *E coli.* [14]. This kind of microbes is not classified as pathogens. The infectious process of pathogens consists of several steps. The first step of the infection is adhering to the host cell using pathogens' specialized organelles called *adhesins*. This adhesion is the interaction between adhesins and the host's membrane proteins. After a pathogen attaches to a host, it invades the host cell to damage or interrupt its host. How it invades the host cell depends on that host cell. If the host cell is a phagocyte, a pathogen enters the host cell via phagocytosis of the host itself. Otherwise, a pathogen invades host by either of two internalization mechanisms, which are zipper mechanism and trigger mechanism. Both of them are the rearrangement of cytoskeleton induced by the protein-protein interaction in the adhesion step [69]. Once a pathogen enters the host cell successfully, it secretes a protein toxin to disrupt or destroy a host cell [57]. There are various kinds of protein toxins depending on the pathogens, but it can be classified into two main types, which are: 1) toxin that disrupts cell membrane, and 2) toxin that modifies components within cell using enzyme. The first kind of toxins modifies the permeability of the cell membrane or the pathways involved in the cell membrane. The latter one is the enzyme that modifies or interrupts function of the host cell.

One common thing among all steps in the infectious process is that they are involved in the interaction between host proteins and pathogen proteins. Therefore, understanding the host-pathogen protein interactions can clarify how pathogens invade and damage host's cell as well as how to prevent those mechanisms, which results in the improvement of the infectious disease therapeutics. In this work, we introduce the first method for solving the host-pathogen protein interaction prediction problem using a graphlet degree vector of a host protein in the protein-protein interaction network as a feature. We explain the rationale behind this feature, how we train the model, as well as the prediction results in Chapter 4.

1.5 Thesis organization

The content of this thesis is organized as follows.

In chapter 2, a novel efficient enumeration algorithm for tree-like chemical compounds containing benzene rings and naphthalene rings is proposed. To optimize the computation time, a chemical compound is represented by a tree-structure instead of a graph, where a node denotes an atom, and an edge denotes a chemical bond. A benzene ring is compressed into a single node and a naphthalene ring is represented by two benzene nodes connected by a special bond. The results of the proposed method are compared with those of a well-known commercial structure generator *MOLGEN*, which can enumerate not only tree-like chemical compounds but any kind of chemical compounds. Giving the same number of the enumerated compounds, the computation time of the proposed method is significantly less than that of *MOLGEN*.

In chapter 3, we propose an enumeration algorithm that allows users to define desired cyclic substructures and enumerate all tree-like chemical compounds containing that substructures without redundancy. Before enumerating compounds, we analyze the input substructures to find their automorphisms. We enumerate molecular trees representing the chemical compounds, where one input substructure is denoted by a single node, and use their automorphisms to eliminate the redundancy. The computational experiment was conducted to confirm the reliability and efficiency of the proposed algorithm.

In chapter 4, we hypothesize that pathogen proteins tend to interact with host proteins with similar topology structure in the PIN. This hypothesis arises because there have been shown that: 1) different pathogens tend to target host proteins involved in the same biological pathway and have similar function, and 2) there is a relation between function and local topology structure in a protein-protein interaction network (PIN) of different proteins. A prediction model of protein-protein interactions between human and four pathogens is proposed using a graphlet degree vector (GDV) of the human PIN as a feature. The results show that the prediction accuracy of model using GDV is better than that not using GDV, which implies the importance of GDV as a feature for solving host-pathogen protein interaction prediction problem.

Finally, chapter 5 concludes the importance and empirical finding of this work and guides the direction of possible future work of all proposed methods.

Chapter 2

Enumeration method for tree-like chemical compounds with benzene rings and naphthalene rings by breadth-first search order

2.1 Background

Enumeration of chemical compounds is important in bioinformatics, and has been adapted to several applications such as drug discovery and design [8, 55, 90], structure elucidation [29, 44, 54], and analyses of chemical spaces [3, 10, 11, 25, 43, 51, 68]. It is defined as a problem of generating all non-redundant chemical structures satisfying some constraints. For example, a chemical formula, which consists of the number of each atom included in the compound, is given as an input. There are several algorithms for enumerating chemical compounds from a chemical formula and most of them use a molecular graph to represent a chemical compound, where the nodes and edges of the graph refer to atoms and bonds of the chemical compound, respectively. Some of those algorithms are claimed to be able to enumerate various chemical structures without restriction of the structure, such as MOLGEN [30] and Open Molecule Generator (OMG) [63]. It was reported that OMG is able to deal with different valences for a kind of atom, and was not efficient for several instances compared with MOLGEN. While the remaining ones, such as EnuMol [28, 35] as well as *BfsSimEnum* and *BfsMulEnum* [95], have a limitation of the structure of enumerated compounds, such as acyclic compounds for BfsSimEnum and BfsMulEnum and compounds with no cycle except for benzene rings for EnuMol, the methods consume significantly less computational time. There are also related application softwares, e.g. SmiLib [76] and CLEVER [81], that generate chemical compounds from given fragments. The limitation of these tools is that they require a library of desired chemical fragments, which can be generated by the enumeration tool.

The enumeration of chemical compounds consumes too high execution time to put into practice. The problem of high execution time can be relieved by limiting the structure of the enumerated compounds. Accordingly, alternative enumeration algorithms using a tree structure to represent a chemical compound have been developed.

Due to the limitation of a tree structure, the alternative algorithms can enumerate only acyclic chemical compounds or tree-like chemical compounds containing benzene rings. This work aims to decrease the limitation of a tree structure by developing an enumeration tool called *BfsBenNaphEnum*, which uses a tree structure to represent a chemical compound and, at the same time, can enumerate tree-like chemical compounds containing two kinds of cyclic structure, a benzene ring and a naphthalene ring.

Pólya proposed a group-theoretic method for isomer counting of single cyclic structures such as a benzene ring, a naphthalene ring, and an anthracene ring using the cycle index, from which many studies followed [88]. However, structures enumerated by these methods are restricted to certain types. Indeed, Meringer wrote that up to now the only way to calculate the number of isomers belonging to an arbitrary molecular formula is to use structure generators [53]. Suzuki et al. considered the problem of enumerating structures having monocyclic graph structures, each of which has exactly one cycle [84]. An enumeration method for tree-like chemical compounds containing only benzene rings as cyclic structures has been implemented on EnuMol web server [1]. On the other hand, our method can enumerate compounds containing naphthalene rings in addition to benzene rings. Moreover, the proposed algorithm can calculate the number of benzene rings and naphthalene rings from chemical formula, while users have to specify the number of benzene rings in EnuMol.

Chemical structures considered in this study can be represented by a molecular tree, where a benzene ring is converted to a node with valence six called a benzene node. We propose a new representation for a naphthalene ring. Instead of converting it into a single node with another atom label, we regard it as two benzene nodes fused together by a new kind of bond named *a merge bond*, which has never been used before. Since a merge bond merges two carbon atoms of two benzene rings together, it reduces the number of carbon atoms with free valence electron of two benzene rings by two so we represent a merge bond by a double-edge. Moreover, benzene nodes cannot have double bonds with other nodes because they bond with other non-benzene atoms by a single bond [32]. This means that a double-edge represents a double bond if it connects two non-benzene nodes, while it represents a merge bond if it connects two benzene nodes. Therefore, bonds in a benzene ring and a naphthalene ring are considered as the same bond and Kekulé representation is not included in this work. Besides, this work uses a two-dimensional molecular tree to represent a chemical structure so it cannot deal with stereoisomers. For tautomeric, this work considers two structures in a pair of tautomeric as non- redundant compounds and generates both of them.

Because we treat a benzene ring as a node and treat a naphthalene ring as two nodes, similar to other type of nodes, the number of benzene rings and naphthalene rings must be determined before enumerating the molecular trees. Then, BfsSimEnum and BfsMulEnum are modified to return a set of molecular trees as the output, given a chemical formula, the number of benzene rings, and the number of naphthalene rings. After that, an attribute called *carbon position list* is added into benzene nodes in a molecular tree to represent the way that benzene nodes bond with their adjacent nodes. This attribute is important because bonding with different carbon atoms in a benzene ring may result in different chemical structures. Finally, for each molecular tree from BfsSimEnum and BfsMulEnum, we generate a set of molecular trees whose nodes adjacent to benzene nodes are labeled with a carbon position such that all chemical structures are enumerated without redundancy based on normal form rule. The flowchart concluding the main steps of this work is given in Figure 2.1

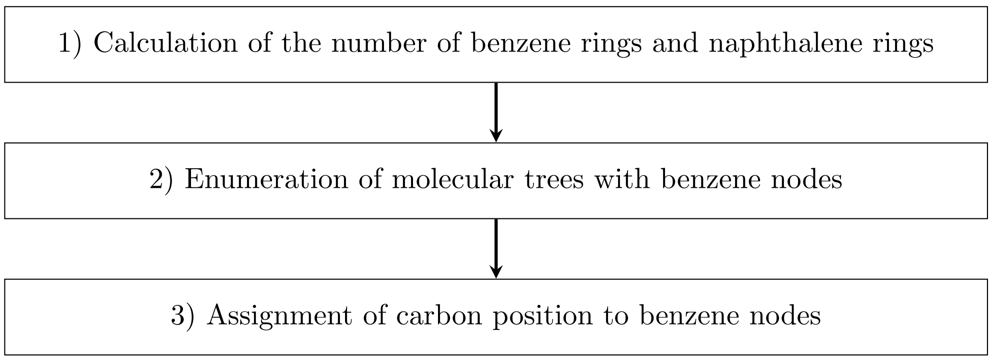

1) Calculation of the number of benzene rings and naphthalene rings
2) Enumeration of molecular trees with benzene nodes
3) Assignment of carbon position to benzene nodes

Figure 2.1: The flowchart concluding main processes of BfsBenNaphEnum

For evaluating our proposed method, we perform computational experiments for several instances, and compare the execution time by our method with that by MOLGEN. We show that our proposed method is efficient for enumerating chemical compounds containing benzene rings and naphthalene rings, and is from 50 times to 5,000,000 times faster than MOLGEN for several instances in our experiments.

2.2 Problem definition

Let we recall the definition of a set of atom labels (Σ), a molecular graph ($G(V, E)$), a label of a node v ($l(v)$), and a valence of an atom label l_i ($val(l_i)$) provided in Section 1.3.1. It should be noted that there exist different valences for a kind of atom, for example, carbon atoms of CO_2 and CO. For this case, it is sufficient to put two distinct labels C and $C_{(2)}$ in Σ, and to define $val(C) = 4$ and $val(C_{(2)}) = 2$. Let $deg(v)$ be the degree of a node v and $num(G, l_i)$ be the total number of nodes labeled with label l_i in a molecular graph G.

From the definition of isomorphism of Section 1.3.2, we define that two molecular graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are redundant with each other if G_1 and G_2 are isomorphic because they correspond to the same chemical compounds. Then, the enumeration problem is defined as follows.

Problem 1. *Given the numbers n_{l_i} of atoms for all labels $l_i \in \Sigma$, the number n_b of benzene rings, and the number n_n of naphthalene rings, enumerate all non-redundant connected molecular graphs G such that $num(G, l_i) = n_{l_i}$ for all $l_i \in \Sigma$, $deg(v) = val(l(v))$ for all nodes $v \in V(G)$, and G includes exactly n_b benzene rings, n_n naphthalene rings, and no other cyclic structures. It must be noted that n_b and n_n can be zero.*

If the input chemical formula contains five or less carbon atoms, BfsBenNaphEnum will enumerate only tree-like chemical compounds by specifying the number of benzene rings and the number of naphthalene rings to be zero because the number of carbon atoms is not enough for one benzene ring. Because we enumerate molecular trees such that degree of each node equals to valence of atom label of that node, charged molecules cannot be enumerated automatically. However, they can still be enumerated by specifying a charged atom as a new kind of atom type with appropriate valence value.

Since our enumeration methods deal with a chemical compound as a node-labeled rooted ordered tree for efficient enumeration, we contract cyclic structures appearing in a molecular graph to single nodes. Concretely, we contract a benzene ring to a node, called *benzene node*, labeled with a special label 'b', and contract a naphthalene ring to two benzene nodes connected by a special bond, called *merge bond*, represented by a double edge (see Figure 2.2).

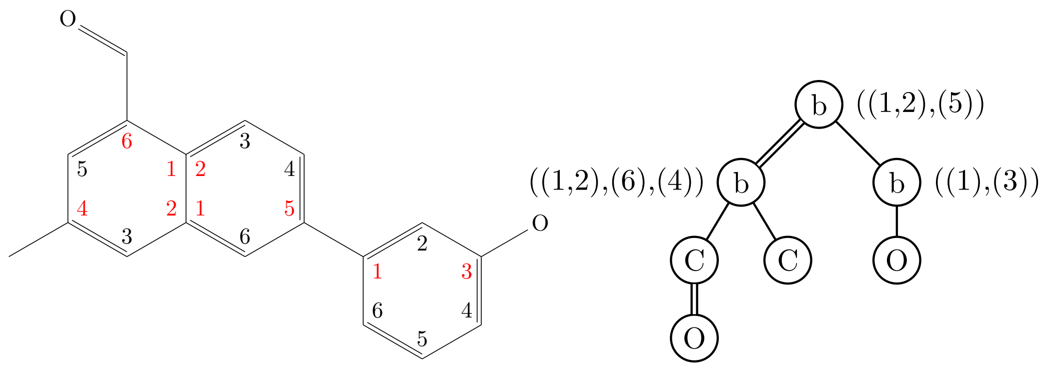


Figure 2.2: Example of a molecular graph including benzene rings and naphthalene rings. (a) A molecular graph including one benzene ring and one naphthalene ring. (b) A rooted tree contracted from the left graph. It is noted that hydrogen atoms are omitted.

Since six carbon atoms contained in a benzene ring are contracted into a benzene node, we need to remember which carbon atom in the benzene ring connects to its adjacent node in a molecular graph. Hence, we add an attribute called *carbon position list* to each benzene node. Figure 2.2 b shows examples of carbon position lists using numbers assigned to carbon atoms in benzene rings in Figure 2.2 a. We call such a node-labeled rooted ordered tree whose benzene nodes are attributed with carbon position lists *a carbon position-assigned molecular tree*. We enumerate carbon position-assigned molecular trees instead of molecular graphs. Because hydrogen atom has a valence one (it can be the leaf node of the molecular tree only) and there is only one way to assign hydrogen atoms to the compound, which is assign to all nodes whose degree less than their valence such that degree of those nodes is equal to their valence, we ignore hydrogen atoms during the enumeration step for the efficiency. Those hydrogen atoms are added to the molecular tree at the final step.

2.3 Preliminaries

2.3.1 Benzene ring and naphthalene ring

A benzene ring is a six-carbon atoms ring with alternating three single bonds and three double bonds. As a result of the delocalization of six electrons in *p* orbital of carbon atoms, the distant between all six carbon atoms is the same although the length of single bonds is expected to be longer than the length of double bond [80]. This suggests that a benzene ring is a resonance hybrid of two structures shown in Figure 2.3 a, which is usually represented by Figure 2.3 b. In other words, all bonds between two carbon atoms in a benzene ring are considered as one and a half bond. It is a prototype of aromatic compounds, a group of unsaturated cyclic hydrocarbon compounds such as naphthalene, anthracene, phenanthrene.

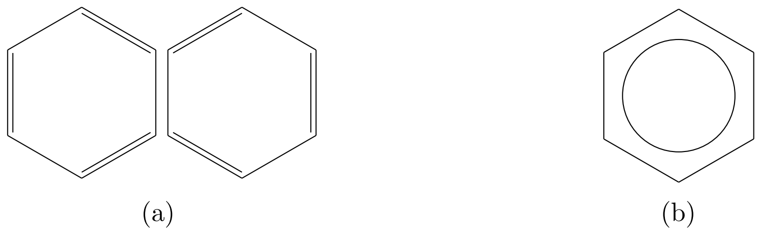


Figure 2.3: Illustration of a benzene ring. (a) Two structures which are the hybrid resonance of a benzene ring. (b) Representation of a benzene ring.

Two benzene rings can be fused together by sharing two carbon atoms and form a naphthalene ring, a bicyclic structure consisting of ten carbon atoms (see Figure 2.4) [48]. A naphthalene ring is the simplest polycyclic aromatic hydrocarbons. Similar to a benzene ring, all bonds between carbon atoms are considered as the same bond so a naphthalene ring is a hybrid of three structures (Figure 2.4), where the most important structure is shown in Figure 2.4 a [80]. Two carbon atoms shared by both rings are at the points of ring fusion. These atoms share all their electrons to adjacent carbon atoms so they do not bond with hydrogen atoms as all other atoms do.

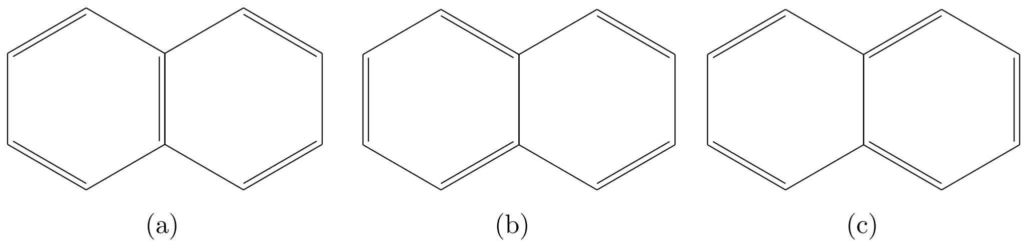


Figure 2.4: Three structures which are hybrid structures of a naphthalene ring.

Because bonds between carbon atoms in a benzene ring and a naphthalene ring are considered as the same kind of bond, a benzene ring and a naphthalene ring are symmetric. The redundancy of chemical compounds can occur due to their symmetry. Accordingly, we use the concept of an automorphism group as a method to discover such redundancy and eliminate it. An automorphism ϕ of a graph G is a one-to-one mapping function of a vertex set of G onto a vertex set of G itself that preserves the adjacency [94]. This means that for any two nodes u and v, $\phi(u)$ is adjacent to $\phi(v)$ if and only if u is adjacent to v and the label of edge between u and v must be the same as that of edge between $\phi(u)$ and $\phi(v)$. There always exists at least one automorphism called *identity mapping*, which is an automorphism such that $\phi(u) = u$ for all nodes u. An automorphism group of a graph G is a set of all automorphisms of G.

Next, we find automorphism groups of a benzene ring and a naphthalene ring when they are treated as two molecular graphs. Let Aut_b and Aut_n be the automorphism groups of a benzene ring and a naphthalene ring, respectively (see Figure 2.5).

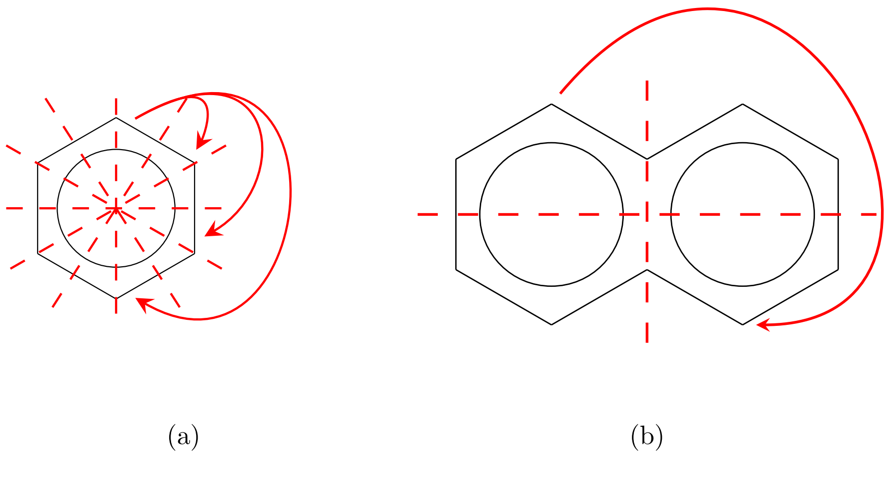


Figure 2.5: Illustration of automorphism of a benzene ring and a naphthalene ring. (a) A benzene ring. (b) A naphthalene ring. Dashed lines indicate reflections, curves indicate rotations, where all automorphisms are not shown.

Aut_b is generated from rotation of $\pi/3$ radians and reflection. For $\phi_b \in Aut_b$, v_1 is adjacent to v_2 in a benzene ring if and only if $\phi_b(v_1)$ is adjacent to $\phi_b(v_2)$ in a benzene ring. Aut_n is generated from rotation of π radians and reflection. This results in 12 automorphisms and 4 automorphisms including identity mapping for a benzene ring and a naphthalene ring, respectively.

2.3.2 Center-rooted and left-heavy condition

In our previous work [95], we defined the normal form for molecular trees without any cyclic structures using *center-rooted* and *left-heavy* to avoid its redundant generation. In this work, we also utilize center-rooted and left-heavy for carbon position-assigned molecular trees, of which properties do not depend on carbon position lists.

A molecular tree T is called *center-rooted* if its root is the center node (see Figure 2.6 a) or one endpoint of the center edge of the longest path in T (see Figure 2.6 b). The center can be either a node or an edge depending on the length of the longest path.

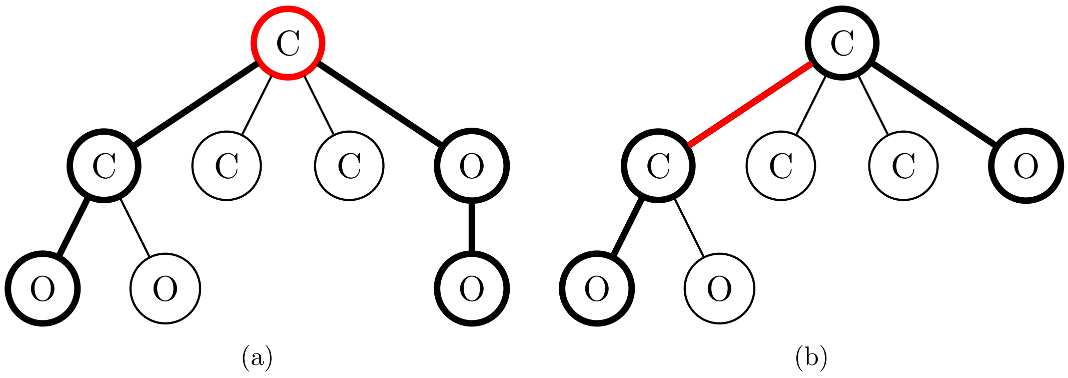


Figure 2.6: Illustration of center-rooted molecular trees. (a) Center of the longest path is a node. (b) Center of the longest path is an edge. The thick lines indicate one of the longest paths and the center node/edge is shown in red.

In order to define a left-heavy tree, atom-labels must be ordered so that they can be compared with each other, for example, b>C>N>O>H for $\Sigma = \{\text{b,C,N,O,H}\}$, where ‘b’ denotes a special atom representing a benzene ring. Let $T(u)$ be the ordered subtree rooted at u in T. Let u and v be two nodes in a molecular tree T, $(u_1, u_2, ..., u_h)$ and $(v_1, v_2, ..., v_k)$ be lists of child nodes of u and v, respectively. It is defined that $T(u) >_s T(v)$ if $l(u) > l(v)$ (Figure 2.7 a) or there exists an integer i such that $T(u_j) =_s T(v_j)$ for all $j < i$ and $(T(u_i) >_s T(v_i)$ (Figure 2.7 b) or $i = k + 1 \leq h$ (Figure 2.7 c). If $T(u) >_s T(v)$ or $T(v) >_s T(u)$ does not hold, it is said that $T(u) =_s T(v)$.

Let $mul(e)$ and $mul(u, v)$ be the multiplicity of edge $e = (u, v)$. Let $(e_1, e_2, ..., e_m)$ and $(e'_1, e'_2, ..., e'_m)$ be two lists of edges in $T(u)$ and $T(v)$ in breadth-first search (BFS) order (see Figure 2.8), respectively. $T(u) >_m T(v)$ if $T(u) >_s T(v)$, or if $T(u) =_s T(v)$ and there exists an integer i such that $mul(e_j) = mul(e'_j)$ for all $j < i$, and $mul(e_i) > mul(e'_i)$ (Figure 2.7 d). If $T(u) >_m T(v)$ or $T(v) >_m T(u)$ does not hold, it is said that $T(u) =_m T(v)$.

Let $child(v) = (v_1, v_2, ...)$ be a list of all child nodes of node v in BFS order. It is

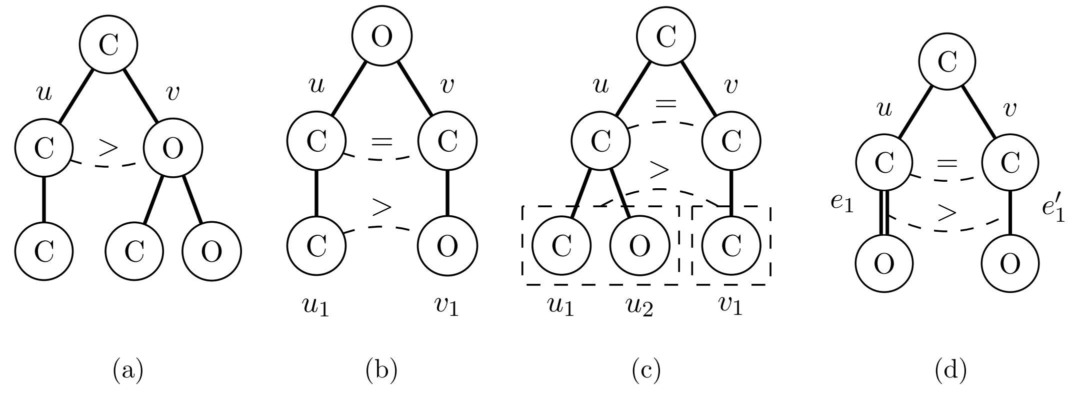


Figure 2.7: Illustration of three molecular trees such that $T(u) >_s T(v)$ or $T(u) >_m T(v)$. (a) $l(u) > l(v)$. (b) $l(u) = l(v)$, $T(u_1) >_s T(v_1)$. (c) $l(u) = l(v)$, $T(u_1) =_s T(v_1)$, $h = 2 > 1 = k$. (d) $T(u) =_s T(v)$, $mul(e_1) > mul(e'_1)$.

defined that a molecular tree T is *left-heavy* if $T(v_i) \geq_m T(v_{i+1})$ holds for all nodes v in T and all $i = 1, \ldots, |child(v)| - 1$.

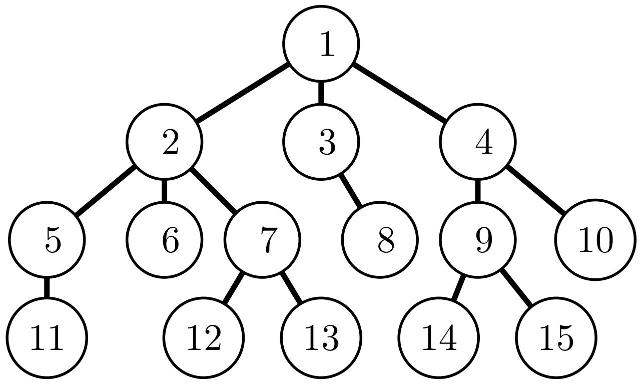


Figure 2.8: Illustration of breadth-first search (BFS) order. Numbers indicate BFS order for this example.

It should be noted that center-rooted and left-heavy are different from *centroid-rooted* and *left-heavy* defined by Fujiwara et al. [28], for example, the molecular tree in Figure 2.2 b is center-rooted and is not centroid-rooted because the number of nodes in the left subtree by removing the root, 4, is more than (total number of nodes $-1)/2 = (7-1)/2 = 3$. In addition, their left-heavy is defined using depth-first search order, not our breadth-first search order.

2.3.3 Carbon position list

Let $s = (v_1, v_2, \ldots, v_n)$ be a list of nodes, $|s|$ and $s[i]$ denote the size and the i-th element of s, respectively. Let $T^{sub}(v_1, v_2)$ be the left-heavy tree rooted at v_1 that consists of the connected component including v_1 when the edge (v_1, v_2) is deleted from T (see Figure 2.9). $T^{sub}(v_1, v_2) =_m T(v_1)$ if v_1 is a child of v_2 in T. Let $index(v, T)$ be the order of

$v \in V(T)$ by traversing a center-rooted left-heavy molecular tree T with BFS order, which is also denoted by $index(v)$ if T is clear.

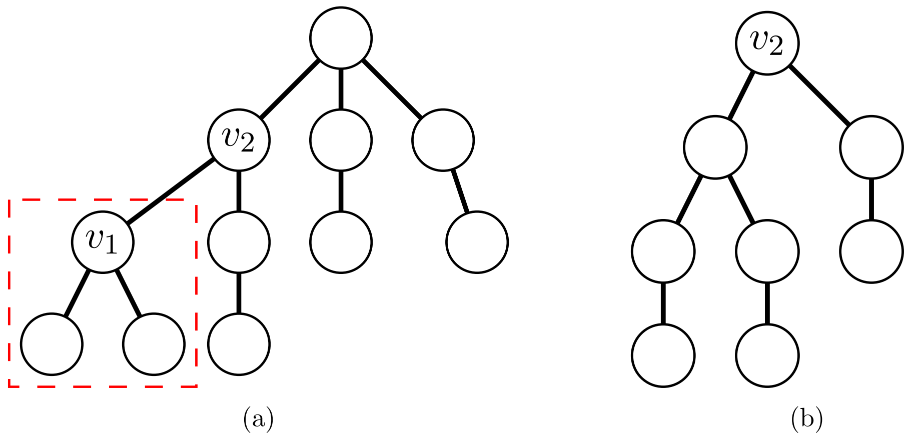


Figure 2.9: Illustration of subtree $T^{sub}(v_1, v_2)$. (a) A molecular tree T and $T^{sub}(v_1, v_2)$, which is surrounded by a red rectangle. (b) $T^{sub}(v_2, v_1)$.

Proposition 1. *For a node v that has the parent node v_p and a child node v_c in a center-rooted molecular tree T, $T^{sub}(v_p, v) \neq_m T^{sub}(v_c, v)$.*

Proof. The height of $T^{sub}(v_p, v)$ is larger than that of $T^{sub}(v_c, v)$ because T is center-rooted. Hence, $T^{sub}(v_p, v)$ is always different from $T^{sub}(v_c, v)$. $\square$

We define an equality $T_1 =_C T_2$ for two rooted carbon-position assigned trees T_1 and T_2 if $T_1 =_m T_2$, and $C_{v_1}^{T_1} = C_{v_2}^{T_2}$ for all benzene nodes $v_1 \in V(T_1)$, where $v_2 \in V(T_2)$ satisfies $index(v_1, T_1) = index(v_2, T_2)$, and C_v^T is a list of lists, called a carbon position list explained later, for a benzene node v in T. For convenience, we define another equality $T_1 =_{\underline{C}} T_2$ by removing the condition that $C_{r_1}^{T_1} = C_{r_2}^{T_2}$ for the roots r_1 and r_2 of T_1 and T_2, respectively, from the conditions of $T_1 =_C T_2$, if r_1 and r_2 are benzene nodes.

For a node v having the parent node v_p and a child node v_c, $T^{sub}(v_p, v) \neq_C T^{sub}(v_c, v)$ if $T^{sub}(v_p, v) \neq_m T^{sub}(v_c, v)$. Hence, only carbon position lists of descendant benzene nodes are needed to determine whether or not $T^{sub}(v_{c_1}, v) =_C T^{sub}(v_{c_2}, v)$ for child nodes v_{c_1} and v_{c_2} of v.

Definition 1. *An adjacent node list A_v^T of a benzene node v in a carbon position-assigned molecular tree T is defined as a list of lists of nodes adjacent to v using carbon position lists of descendant benzene nodes such that*

- $|A_v^T[i]| \leq |A_v^T[i+1]|$ *for all i,*
- $index(A_v^T[i][1]) < index(A_v^T[i+1][1])$ *if* $|A_v^T[i]| = |A_v^T[i+1]|$,
- $index(A_v^T[i][j]) < index(A_v^T[i][j+1])$ *for all i, j,*

- *$A_v^T[i] = (v')$ if (v, v') is a merge bond for some i,*
- *$v' \in A_v^T[i]$ if (v, v') is not a merge bond, and $T^{sub}(v', v) =_C T^{sub}(A_v^T[i][1], v)$.*

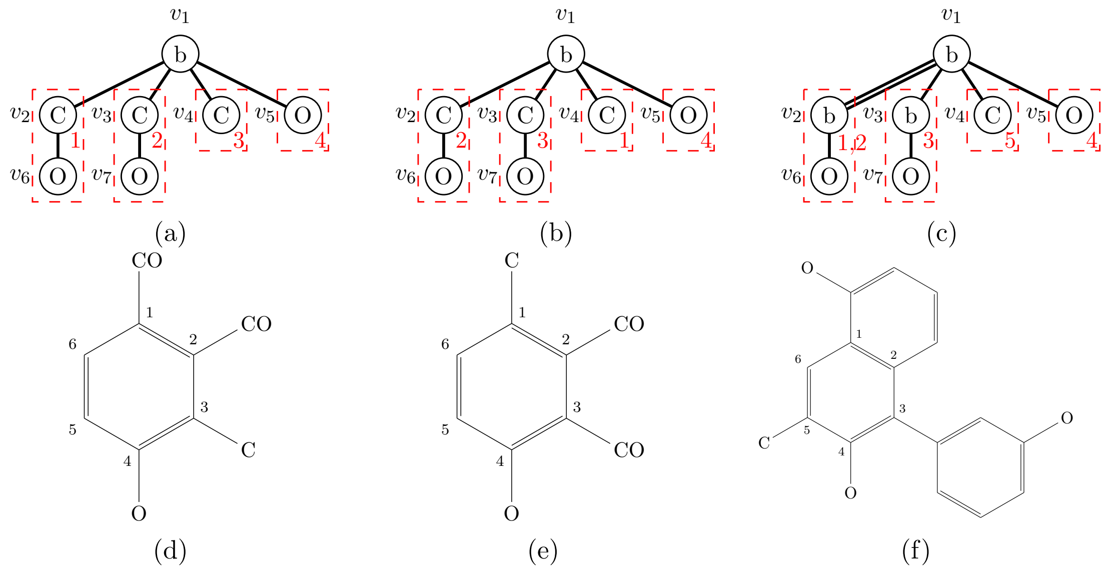


Figure 2.10: Examples of adjacent node lists and carbon position lists. (a) T_1. (b) T_2. (c) T_3. (d) Molecular graph of T_1. (e) Molecular graph of T_2. (f) Molecular graph of T_3. Red numbers represent carbon positions of node v_1.

Figure 2.10 shows examples of carbon position-assigned molecular trees, where benzene node v_1 in each tree has adjacent nodes v_2, v_3, v_4, and v_5. Then, $T_1^{sub}(v_2, v_1) =_C T_1^{sub}(v_3, v_1) \neq_C T_1^{sub}(v_4, v_1) \neq_C T_1^{sub}(v_5, v_1)$ and $index(v_4) < index(v_5)$, so we have $A_{v_1}^{T_1} = ((v_4), (v_5), (v_2, v_3))$. Also for T_2, $A_{v_1}^{T_2} = ((v_4), (v_5), (v_2, v_3))$. For T_3, $A_{v_1}^{T_3} = ((v_2), (v_3), (v_4), (v_5))$ because (v_2, v_1) is a merge bond. If (v_2, v_1) is not a merge bond and $C_{v_2}^{T_3} = C_{v_3}^{T_3}$, then $A_{v_1}^{T_3} = ((v_4), (v_5), (v_2, v_3))$.

Proposition 2. *For a benzene node v that has the parent node v_p in a center-rooted molecular tree T, $A_v^T[1] = (v_p)$.*

Proof. If v has no child, it is clear because the adjacent node of v is only v_p. We assume that v has a child v_c. From Proposition 1 and $index(v_p) < index(v_c)$, $A_v^T[1] = (v_p)$ always holds. □

A *carbon position list* C_v^T of a benzene node v in T is a list of lists, where $C_v^T[i]$ is a list of carbon positions of the nodes in $A_v^T[i]$. It is sufficient to enumerate $C_v^T[i]$ in ascending order because each node in $A_v^T[i]$ has the same subtree. If $(A_v^T[i][1], v)$ is a merge bond, $C_v^T[i]$ has two carbon positions instead of one as usual. It should be noted that $C_v^T[i] \subseteq \{1, \dots, 6\}$ and two carbon positions are assigned for a merge bond because a naphthalene ring shares two carbon atoms between two benzene rings. In the examples of Figure 2.10, $C_{v_1}^{T_1} = ((3),(4),(1,2))$ for $A_{v_1}^{T_1} = ((v_4),(v_5),(v_2,v_3))$, $C_{v_1}^{T_2} = ((1),(4),(2,3))$ for $A_{v_1}^{T_2} = ((v_4),(v_5),(v_2,v_3))$, $C_{v_1}^{T_3} = ((1,2),(3),(5),(4))$ for $A_{v_1}^{T_3} = ((v_2),(v_3),(v_4),(v_5))$.

Definition 2. *An adjacent node list $A_{(v_1,v_2)}^{T}$ for a naphthalene ring with two benzene nodes v_1, v_2, where (v_1, v_2) is a merge bond, is defined as a list of lists of nodes adjacent to v_1 or v_2 except v_1 and v_2 such that*

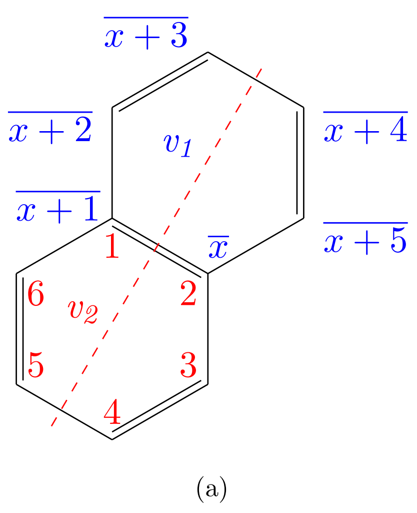
- $|A_{(v_1,v_2)}^{T}[i]| \leq |A_{(v_1,v_2)}^{T}[i+1]|$ *for all i,*
- *$index(A_{(v_1,v_2)}^{T}[i][1]) < index(A_{(v_1,v_2)}^{T}[i+1][1])$ if $|A_{(v_1,v_2)}^{T}[i]| = |A_{(v_1,v_2)}^{T}[i+1]|$,*
- *$index(A_{(v_1,v_2)}^{T}[i][j]) < index(A_{(v_1,v_2)}^{T}[i][j+1])$ for all i, j,*
- *$v' \in A_{(v_1,v_2)}^{T}[i]$ if $T^{sub}(v', bn(v')) =_C T^{sub}(A_{(v_1,v_2)}^{T}[i][1], bn(A_{(v_1,v_2)}^{T}[i][1]))$, where $bn(v)$ is v_1 or v_2 that is adjacent to v.*

For a benzene node v_2 that is connected by a merge bond with the parent node v_1, we suppose that the carbon atoms having positions 1,2 in v_2 are connected with the carbon atoms having positions $\overline{x+1}, \overline{x}$ in v_1, respectively, where x takes an integer between 1 and 6, and $\overline{x} = (x \mod 6){+}1$ (see Figure 2.11 a). Here, consider the case that v_1 has the parent node v_p. If T is in normal form (Definition 6), position 1 is assigned to the carbon atom connected with v_p (Proposition 5). Then, from Proposition 1, $T^{sub}(v_p, v_1) \neq_C T^{sub}(v_c, v_2)$ for any child node v_c of v_2, $T^{sub}(v_p, v_1) \neq_C T^{sub}(v_c, v_1)$ for any child node v_c of v_1 except v_2, and the naphthalene ring is not symmetric. Consider the case that v_1 does not have a parent node, that is, v_1 is the root. If $T^{sub}(v_1, v_2) \neq_{\underline{C}} T^{sub}(v_2, v_1)$, the naphthalene ring can be symmetric only with respect to the axis denoted by the dashed red line in Figure 2.11 a. Then, it is not needed to consider the other symmetry for the naphthalene ring.

Consider the case that $T^{sub}(v_1, v_2) =_{\underline{C}} T^{sub}(v_2, v_1)$. We can prove that $x = 1$ if T is in normal form (see Proposition 4). Then, a carbon position list $C_{(v_1,v_2)}^{T}$ of a naphthalene ring consisting of two benzene nodes v_1, v_2 is a list of lists determined from $C_{v_1}^{T}$ and $C_{v_2}^{T}$ according to the following rule, where $C_{(v_1,v_2)}^{T}[i]$ is a list of carbon positions of nodes in $A_{(v_1,v_2)}^{T}[i]$ in ascending order.

Definition 3. *Carbon positions in a naphthalene ring correspond to carbon positions in two benzene nodes v_1, v_2, where v_1 is the parent node of v_2, if $T^{sub}(v_1, v_2) =_{\underline{C}} T^{sub}(v_2, v_1)$, as follows (see Figure 2.11 b).*

- *For the benzene ring of v_1, positions $1, 2$ are assigned to carbons of the merge bond in $C_{v_1}^{T}$. Position i ($i = 3, \ldots, 6$) in $C_{v_1}^{T}$ corresponds to $i - 2$ in $C_{(v_1,v_2)}^{T}$.*
- *For the benzene ring of v_2, positions $1, 2$ are assigned to carbons of the merge bond in $C_{v_2}^{T}$. Position i ($i = 3, \ldots, 6$) in $C_{v_2}^{T}$ corresponds to $i + 2$ in $C_{(v_1,v_2)}^{T}$.*



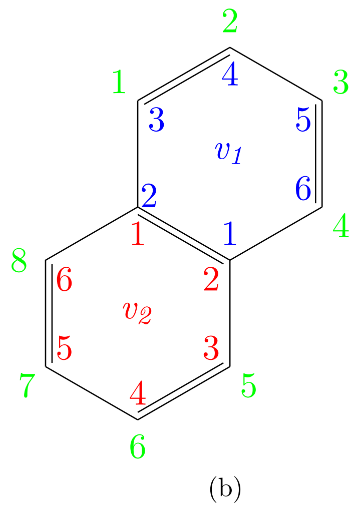


Figure 2.11: Correspondence between carbon positions in a naphthalene ring. (a) Correspondence between carbon positions involved with a merge bond in two benzene rings. (b) Correspondence between carbon positions of a naphthalene ring and two benzene rings in the case of $T^{sub}(v_1, v_2) =_{\underline{C}} T^{sub}(v_2, v_1)$. The upper benzene ring v_1 is the parent of the lower benzene ring v_2. $\overline{x}$ denotes $(x \mod 6) + 1$. Blue, red, and green numbers are positions of $C^T_{v_1}$, $C^T_{v_2}$, and $C^T_{(v_1,v_2)}$, respectively. The dashed red line denotes the symmetric axis of ϕ_{ref}.

Figure 2.12 shows examples of carbon position lists for a naphthalene ring, where T_4' is T_4 with $C^{T_4'}_{v_1} = ((1,2),(4),(3))$ and $C^{T_4'}_{v_2} = ((1,2),(4),(5))$, T_4'' is T_4 with $C^{T_4''}_{v_1} = ((1,2),(4),(5))$ and $C^{T_4''}_{v_2} = ((1,2),(4),(3))$. Then, $A^{T_4'}_{(v_1,v_2)} = A^{T_4''}_{(v_1,v_2)} = ((v_3,v_5),(v_4,v_6))$, $C^{T_4'}_{(v_1,v_2)} = ((2,6),(1,7))$, and $C^{T_4''}_{(v_1,v_2)} = ((2,6),(3,5))$.

Definition 4. *For carbon position lists $C^{T_1}_v$, $C^{T_2}_v$, where $A^{T_1}_v = A^{T_2}_v$, it is defined that $C^{T_1}_v < C^{T_2}_v$ if there exist two integers i and j such that*

- $C^{T_1}_v[i'][j'] = C^{T_2}_v[i'][j']$ *for all* $i' < i$ *and all* $j' = 1, \ldots, |C^{T_1}_v[i']|$,
- $C^{T_1}_v[i][j'] = C^{T_2}_v[i][j']$ *for all* $j' < j$,
- $C^{T_1}_v[i][j] < C^{T_2}_v[i][j]$.

This definition is applied to comparison of $C^{T_1}_{(v_1,v_2)}$ and $C^{T_2}_{(v_1,v_2)}$ for a naphthalene ring with v_1 and v_2 in the same way.

In the example of Figure 2.10, T_1 and T_2 have the same tree structure, and $C^{T_2}_{v_1} = ((1),(4),(2,3)) < ((3),(4),(1,2)) = C^{T_1}_{v_1}$ because $C^{T_2}_{v_1}[1][1] = 1 < 3 = C^{T_1}_{v_1}[1][1]$.

Let Aut_b and Aut_n be the automorphism groups of a benzene ring and a naphthalene ring, respectively (see Figure 2.5). We suppose that a list $\phi(C^T_v[i])$ of carbon positions for a map ϕ and $i = 1, \ldots, |C^T_v|$ is in ascending order by sorting elements of the list because all nodes in $A^T_v[i]$ have the same subtree. For example, $\phi_b(C^{T_1}_{v_1}) = ((6),(5),(1,2))$ for

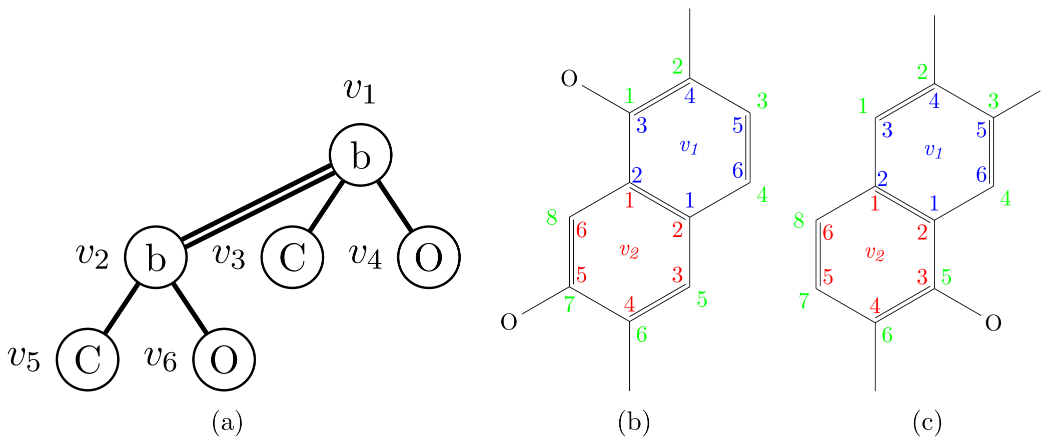


Figure 2.12: Example of carbon position lists for a naphthalene ring. (a) T_4. (b) Molecular graph of T_4', which is T_4 with $C_{v_1}^{T_4'} = ((1,2),(4),(3))$, $C_{v_2}^{T_4'} = ((1,2),(4),(5))$. (c) Molecular graph of T_4'', which is T_4 with $C_{v_1}^{T_4''} = ((1,2),(4),(5))$, and $C_{v_2}^{T_4''} = ((1,2),(4),(3))$.

$C_{v_1}^{T_1} = ((3),(4),(1,2))$ and the reflection map ϕ_b by the perpendicular bisector between carbon atoms of 1 and 2.

2.3.4 Normal form of a carbon position-assigned molecular tree

In order to prevent generating redundant molecular trees in enumeration, we define a normal form of a carbon position-assigned molecular tree. During the enumeration step, we generate only molecular trees satisfying this normal form rule and discard the rest.

Definition 5. *Let P be a path in T consisting of n nodes $(v_1, v_2, ..., v_n)$ $(n \geq 2)$. P is called a symmetric path if the following conditions are satisfied.*

- $T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor + 1}) =_m T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor + 1}, v_{n-\lfloor \frac{n}{2} \rfloor})$,
- $index(v_i, T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor + 1})) = index(v_{n-i+1}, T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor + 1}, v_{n-\lfloor \frac{n}{2} \rfloor}))$ *for all* $i = 1, \cdots, \lfloor \frac{n}{2} \rfloor$, *where* $\lfloor x \rfloor$ *is the largest integer less than or equal to* x,
- $C_v^T = C_{v'}^T$ *for all benzene nodes* $v \in V(T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor + 1})) \backslash V(T^{sub}(v_1, v_2))$, *where* $v' \in V(T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor + 1}, v_{n-\lfloor \frac{n}{2} \rfloor}))$ *satisfies* $index(v', T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor + 1}, v_{n-\lfloor \frac{n}{2} \rfloor})) = index(v, T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor + 1}))$, *and* $v \in V_1 \backslash V_2$ *means that* $v \in V_1$ *and* $v \notin V_2$.

Proposition 3. *For a center-rooted molecular tree, either of* $v_{\frac{n}{2}}$ *and* $v_{\frac{n}{2}+1}$ *is the root if the length of a symmetric path* $(v_1, \cdots, v_n)$ *is even. Otherwise, the depth of* $v_{\frac{n+1}{2}}$ *is less than that of any node in the path.*

Proof. For a path $(v_1, \cdots, v_n)$, v_{i+1} and v_{n-i} must be the parent nodes of v_i and v_{n-i+1}, respectively, for $i = 1, \cdots, \frac{n-1}{2}$ if n is odd and for $i = 1, \cdots, \frac{n}{2} - 1$ if n is even due to the center rooted property. Therefore, if the length of path is odd, $v_{\frac{n+1}{2}}$ is the parent node of both $v_{\frac{n+1}{2}-1}$ and $v_{\frac{n+1}{2}+1}$, which means that the depth of $v_{\frac{n+1}{2}}$ is less than that of any node in the path.

In the case that n is even, either $v_{\frac{n}{2}}$ or $v_{\frac{n}{2}+1}$ has the least depth among all nodes in the path and another node is the child node of that node. Assume that between these two nodes the parent node is v_a and the child node is v_b. v_a cannot have a parent node because the height of $T^{sub}(v_p, v_a)$, where v_p is the parent node of v_a, cannot be equal to the height of $T^{sub}(v_c, v_b)$ for any nodes v_c that are adjacent to v_b due the center-rooted condition, which means that $T^{sub}(v_a, v_b) =_m T^{sub}(v_b, v_a)$ cannot be hold and the first condition of symmetric path is violated. In other words, v_a, which is either $v_{\frac{n}{2}}$ or $v_{\frac{n}{2}+1}$, is the root node of the tree if n is even. □

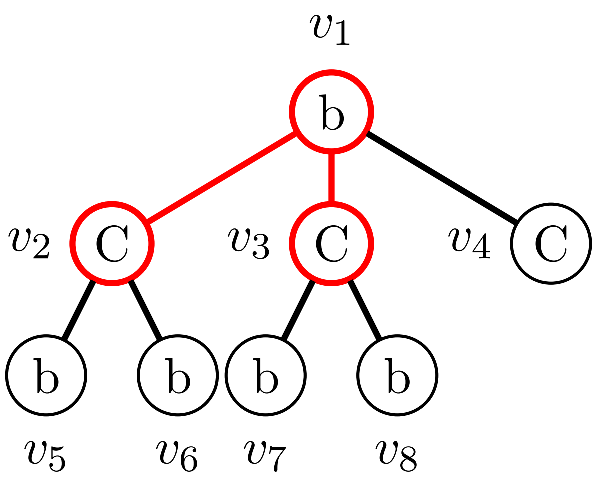
We say that v_1 is *left* of v_n for a symmetric path $(v_1, \ldots, v_n)$ when $v_{n-\lfloor\frac{n}{2}\rfloor+1}$ is the root, or $index(v_1) < index(v_n)$.

Figure 2.13 shows examples of symmetric paths, (v_2, v_1, v_3) in T_5 and (v_5, v_2, v_1, v_3) in T_6, where $T_5^{sub}(v_2, v_1) =_m T_5^{sub}(v_3, v_1)$, $T_6^{sub}(v_2, v_1) =_m T_6^{sub}(v_1, v_2)$, and $C_{v_4}^{T_6} = C_{v_6}^{T_6}$.

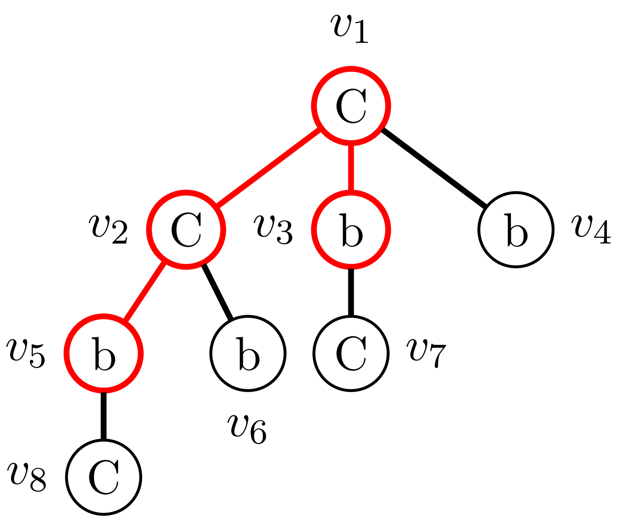
We define an inequality $T_1 >_C T_2$ for carbon position-assigned molecular trees T_1 and T_2 if $T_1 >_m T_2$, or $T_1 =_m T_2$, and there exists an integer i such that v_i is a benzene node, $C_{v_i}^{T_1} > C_{v'_i}^{T_2}$, and $C_{v_j}^{T_1} = C_{v'_j}^{T_2}$ for all benzene nodes v_j with $j > i$, where $index(v_k, T_1) = index(v'_k, T_2)$ for all $k = 1, \ldots, |V(T_1)|$.

Definition 6. *Let ϕ_{ref} be the reflection map with the symmetric axis shown in Figure 2.11A. A carbon position-assigned molecular tree T that contains a carbon position list C_v^T for each benzene node v is in normal form if the following conditions are satisfied.*

1. *T is center-rooted and left-heavy.*
2. *$T(v) \geq_m T^{sub}(r, v)$ if the center of the longest path in T with the root r is the edge (r, v).*
3. *Positions in each sublist of C_v^T for each benzene node v are in ascending order.*
4. *$C_v^T \leq \phi_b(C_v^T)$ for all benzene nodes v that is not connected by a merge bond with the parent node and all $\phi_b \in Aut_b$.*
5. *For benzene nodes v_1, v_2 connected by a merge bond such that v_1 is the root of T,*
   - *(a) $C_{(v_1,v_2)}^T \leq \phi_n(C_{(v_1,v_2)}^T)$ for all $\phi_n \in Aut_n$ if $T^{sub}(v_1, v_2) =_{\underline{C}} T^{sub}(v_2, v_1)$, where $C_{(v_1,v_2)}^T$ is related with $C_{v_1}^T$ and $C_{v_2}^T$ by Definition 3.*
   - *(b) $C_{v_2}^T \leq \phi_{ref}(C_{v_2}^T)$ if $T^{sub}(v_1, v_2) \neq_{\underline{C}} T^{sub}(v_2, v_1)$ and $C_{v_1}^T = \phi_{ref}(C_{v_1}^T)$.*
6. *$T^{sub}(v_1, v_2) \geq_C T^{sub}(v_n, v_{n-1})$ for all pairs v_1,v_n of nodes such that the path $(v_1, \ldots, v_n)$ is a symmetric path, v_1 and $v_n (= v_2)$ are not connected by a merge bond, and v_1 is left of v_n.*



(a)



(b)

Figure 2.13: Examples of symmetric paths. The red lines denote symmetric paths. (a) T_5, where (v_2, v_1, v_3) is a symmetric path, and $T_5^{sub}(v_2, v_1) =_m T_5^{sub}(v_3, v_1)$. (b) T_6, where (v_5, v_2, v_1, v_3) is a symmetric path, $T_6^{sub}(v_2, v_1) =_m T_6^{sub}(v_1, v_2)$ and $C_{v_4}^{T_6} = C_{v_6}^{T_6}$.

We call a tree in normal form a *normal tree.*

Figure 2.12 also shows molecular trees in normal form and not in normal form. For condition 4 of the definition, $C^{T'_4}_{v_1} = ((1,2),(4),(3)) \leq \phi_b(C^{T'_4}_{v_1})$, $C^{T''_4}_{v_1} = ((1,2),(4),(5)) \leq \phi_b(C^{T''_4}_{v_1})$. T'_4 and T''_4 satisfy conditions 1,2, 3, and 4. For condition 5, $C^{T'_4}_{(v_1,v_2)} = ((2,6),(1,7)) \leq \phi_n(C^{T'_4}_{(v_1,v_2)})$, whereas $C^{T''_4}_{(v_1,v_2)} = ((2,6),(3,5)) > ((2,6),(1,7)) = \phi_{rot}(C^{T''_4}_{(v_1,v_2)})$ for rotation ϕ_{rot} of π radians, and T''_4 violates the condition. It is noted that T''_4 is rotated by π radians from T'_4. For condition 6, v_1 and v_2 are connected by a merge bond. Thus, T'_4 is a normal tree, and T''_4 is not a normal tree.

Proposition 4. *For a normal tree T with a benzene node v_1 that is connected by a merge bond with its child node v_2 and satisfies $T^{sub}(v_1,v_2) =_{\underline{C}} T^{sub}(v_2,v_1)$, positions 1,2 are assigned to the merge bond in the benzene ring of v_1. Furthermore, if $C^T_{(v_1,v_2)} \leq \phi_n(C^T_{(v_1,v_2)})$ for all $\phi_n \in Aut_n$, then $C^T_{v_1} \leq \phi_b(C^T_{v_1})$ for all $\phi_b \in Aut_b$.*

Proof. We assume that there exists a node v_l as a left sibling of v_2, and v_l is the leftmost child of v_1. Since T is left-heavy, $T(v_l) \geq_m T(v_2)$, and $l(v_l) = l(v_2) =$'b' is needed. However, $T(v_l) =_C T(v_c)$, where v_c is the leftmost child of v_2, because $T^{sub}(v_1,v_2) =_{\underline{C}} T^{sub}(v_2,v_1) =_C T(v_2)$. Hence, $T(v_l) <_m T(v_2)$. It contradicts the assumption, and v_2 is the leftmost child of v_1. Therefore, $A^T_{v_1}[1] = (v_2)$. From condition 4 of Definition 6, $C^T_{v_1}[1] = (1,2)$, and positions 1,2 are assigned to the merge bond, that is $x = 1$ in Figure 2.11 a.

For a map $\phi_b \in Aut_b$ other than the identity and reflection map ϕ_{ref} for a benzene ring, $C^T_{v_1} < \phi_b(C^T_{v_1})$ because each of $\phi_b(1)$ and $\phi_b(2)$ is at least 2. From $C^T_{(v_1,v_2)} \leq \phi_{ref}(C^T_{(v_1,v_2)})$ and the correspondence between $C^T_{v_1}$ and $C^T_{(v_1,v_2)}$, $C^T_{v_1} \leq \phi_{ref}(C^T_{v_1})$. Therefore, $C^T_{v_1} \leq \phi_b(C^T_{v_1})$ for all $\phi_b \in Aut_b$. □

Proposition 5. *For a benzene node v of a normal tree T, $C^T_v[1][1]$ is always equal to 1.*

Proof. If v is not connected by a merge bond with the parent node, from condition 4, C^T_v must be the least possible carbon position list. Hence, $C^T_v[1][1] = 1$. Otherwise, from Definition 3, $C^T_v[1][1] = 1$. □

Proposition 6. *For a normal tree T with a path $(v_1,\ldots,v_n)$, G' is the molecular graph obtained from the tree T' by removing $T^{sub}(v_1,v_2)$ and $T^{sub}(v_n,v_{n-1})$ except v_1 and v_n from T, where v_1 is left of v_n. If there is a non-identity map ϕ of the automorphism group of G' satisfying $\phi(v_i) = v_{n-i+1}$ for all $i = 1,\ldots,n$, then $T^{sub}(v_1,v_2) \geq_C T^{sub}(v_n,v_{n-1})$, where ϕ in G' is naturally extended to T.*

Proof. If $T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor+1}) >_m T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor+1}, v_{n-\lfloor \frac{n}{2} \rfloor})$, then $T^{sub}(v_1,v_2) >_m T^{sub}(v_n,v_{n-1})$, and $T^{sub}(v_1,v_2) >_C T^{sub}(v_n,v_{n-1})$. We assume $T^{sub}(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor+1}) =_m T^{sub}(v_{n-\lfloor \frac{n}{2} \rfloor+1}, v_{n-\lfloor \frac{n}{2} \rfloor})$. If the path $(v_1,\ldots,v_n)$ is a symmetric path, $T^{sub}(v_1,v_2) \geq_C T^{sub}(v_n,v_{n-1})$ from condition

6. We assume that $(v_{i+1},\dots,v_{n-i})$ is a symmetric path for some i, and $index(v_i, T^{sub}(v_{i+1}, v_{i+2})) > index(v_{n-i+1}, T^{sub}(v_{n-i}, v_{n-i-1}))$ (see Figure 2.14). Then,

$$T^{sub}(v_{i+1}, v_{i+2}) =_m T^{sub}(v_{n-i}, v_{n-i-1}), \tag{2.1}$$
$$T^{sub}(v_{i+1}, v_{i+2}) \geq_C T^{sub}(v_{n-i}, v_{n-i-1}). \tag{2.2}$$

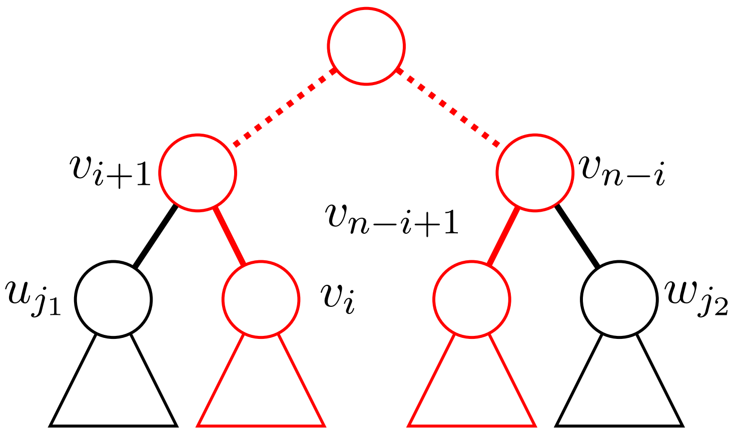


Figure 2.14: Illustration of an automorphism ϕ in the proof. The red path indicates $(v_1,\dots,v_n)$, where $\phi(v_i) = v_{n-i+1}$ for all $i = 1,\dots,n$.

Let u_j and w_j be child nodes of v_{i+1} and v_{n-i}, respectively. Then, $v_i = u_{j_2}$ and $v_{n-i+1} = w_{j_1}$, where $j_1 = index(v_{n-i+1}, T^{sub}(v_{n-i}, v_{n-i-1}))$ and $j_2 = index(v_i, T^{sub}(v_{i+1}, v_{i+2}))$. If v_{i+1} and v_{n-i} are benzene nodes, $T(u_{j_1}) =_C T(v_i)$, $T(v_{n-i+1}) =_C T(w_{j_2})$, and $T(v_i) =_C T(v_{n-i+1})$ because $C^T_{v_{i+1}} = C^T_{v_{n-i}}$ and $\phi(v_i) = v_{n-i+1}$.

We assume that v_{i+1} and v_{n-i} are not benzene nodes. For child nodes u_j of v_{i+1}, $T(u_j) \geq_C T(u_{j+1})$ because (u_j, v_{i+1}, u_{j+1}) is a symmetric path. Also for child nodes w_j of v_{n-i}, $T(w_j) \geq_C T(w_{j+1})$. From the definition of ϕ, $T(u_j) =_C T(\phi(u_j))$ for all $u_j \neq v_i$. If $index(\phi(u_{j+l})) < index(\phi(u_j))$ for $u_j, u_{j+l} \neq v_i$ and $l > 0$, $T(u_j) \geq_C T(u_{j+l}) =_C T(\phi(u_{j+l})) \geq_C T(\phi(u_j)) =_C T(u_j)$. It means $T(u_j) =_C T(u_{j+l})$. We assume that $index(\phi(u_j)) < index(\phi(u_{j+l}))$ for all $u_j \neq v_i$, that is, $\phi(u_j) = w_{j+1}$ for all $j = j_1,\dots,j_2 - 1$. Then,

$$T(u_j) =_C T(w_{j+1}) \leq_C T(w_j), \text{ and } T(v_i) \leq_C T(u_{j_2-1}) =_C T(w_{j_2}). \tag{2.3}$$

If $T^{sub}(v_{i+1}, v_{i+2}) >_C T^{sub}(v_{n-i}, v_{n-i-1})$, then there is an integer j ($j_1 \leq j \leq j_2$) such that $T(u_j) >_C T(w_j)$, and it contradicts Eq. (2.3). Therefore, $T^{sub}(v_{i+1}, v_{i+2}) =_C T^{sub}(v_{n-i}, v_{n-i-1})$, and $T(v_i) =_C T(v_{n-i+1})$. Also for the case that $(v_{i+1},\dots,v_{n-i})$ is a symmetric path for some i and $index(v_i, T^{sub}(v_{i+1}, v_{i+2})) < index(v_{n-i+1}, T^{sub}(v_{n-i}, v_{n-i-1}))$, then $T(v_i) =_C T(v_{n-i+1})$.

Thus, $T^{sub}(v_1, v_2) \geq_C T^{sub}(v_n, v_{n-1})$. □

2.4 Proof

Lemma 1. *Given a molecular graph G without cyclic structures except benzene rings and naphthalene rings, G can be represented by a normal tree.*

Proof. We can assign numbers to carbons in benzene rings and naphthalene rings of G such that the conditions of Definition 6 are satisfied. $\square$

Lemma 2. *Given two different molecular graphs G_1 and G_2, they cannot be represented by the same normal tree.*

Proof. We can unambiguously obtain a molecular graph from a normal tree by replacing all benzene nodes with benzene rings according to its carbon position lists. $\square$

Lemma 3. *Given two different normal trees T_1 and T_2, T_1 does not represent the same molecular graph as T_2.*

Proof. We assume that T_1 represents the same molecular graph as T_2. Let G_1 and G_2 be molecular graphs transformed from T_1 and T_2, respectively, where each carbon in benzene rings and naphthalene rings is connected with adjacent atoms according to carbon position lists of T_1 and T_2. From the assumption, there is an isomorphism ψ from G_1 to G_2. It means that $l(v_1) = l(\psi(v_1))$ for all $v_1 \in V(G_1)$, $(\psi(v_1), \psi(v_2)) \in E(G_2)$ if and only if $(v_1, v_2) \in E(G_1)$, and $mul(\psi(v_1), \psi(v_2)) = mul(v_1, v_2)$.

Consider the case that the automorphism group $Aut(G_1)$ of G_1 has only elements ϕ such that $\phi(v_1) \neq v_2$ for v_1 and v_2 belonging to distinct benzene rings. Let $T(G)$ be the molecular tree without carbon position lists, obtained from G by contracting benzene rings and naphthalene rings to benzene nodes, and satisfying conditions 1,2 of Definition 6. We suppose that maps ψ and ϕ in G_1 are naturally extended to $T(G_1)$. Since T_1 is different from T_2, there is a benzene node $v_1 \in V(T_1)$ such that

$$C_{v_1}^{T_1} \neq C_{\psi(v_1)}^{T_2}. \tag{2.4}$$

If v_1 is not connected by a merge bond with the parent node, there is a non-identity map $\phi_b \in Aut_b$ such that $C_{v_1}^{T_1} = \phi_b(C_{\psi(v_1)}^{T_2})$ because T_1 and T_2 represent the same molecular graph. It contradicts condition 4 of Definition 6. Suppose that v_1 is connected by a merge bond with the parent node v_p and $C_{v_p}^{T_1} = C_{\psi(v_p)}^{T_2}$. If $T^{sub}(v_p, v_1) =_{\underline{C}} T^{sub}(v_1, v_p)$, then v_p is the root, and there is a non-identity map $\phi_n \in Aut_n$ such that $C_{(v_p,v_1)}^{T_1} = \phi_n(C_{(\psi(v_p),\psi(v_1))}^{T_2})$ because T_1 and T_2 represent the same molecular graph. It contradicts condition 5a. Otherwise, $T^{sub}(v_p, v_1) \neq_{\underline{C}} T^{sub}(v_1, v_p)$. If v_p is not the root, then T_1 does not represent the same molecular graph as T_2 because $T^{sub}(v_a, v_p)$, where v_a is the parent of v_p, is different from other subtrees connected to the naphthalene ring. It contradicts the assumption. If v_p is the root, $C_{v_p}^{T_1} = \phi_{ref}(C_{v_p}^{T_1})$ and $C_{v_1}^{T_1} = \phi_{ref}(C_{\psi(v_1)}^{T_2})$ because T_1 and T_2 represent the same molecular graph. It contradicts condition 5b.

Consider the case that there is an element $\phi \in Aut(G_1)$ such that $\phi(v_1) = v_2$ for v_1 and v_2 belonging to distinct benzene rings. Since T_1 is different from T_2, there is a benzene node $v_1 \in V(T_1)$ such that

$$C^{T_1}_{v_1} \neq C^{T_2}_{\psi(v_1)}. \tag{2.5}$$

Here, we suppose that conditions 3, 4, 5 are satisfied for all benzene nodes in T_1 and T_2. Then, there is a path from v_1 to $\phi(v_1) = v_n$, $(v_1, \dots, v_n)$, in T_1. Since T_1 and T_2 represent the same molecular graph,

$$T^{sub}_1(v_1, v_2) =_C T^{sub}_2(\psi(v_n), \psi(v_{n-1})) \text{ and } T^{sub}_1(v_n, v_{n-1}) =_C T^{sub}_2(\psi(v_1), \psi(v_2)). \tag{2.6}$$

Here, we can assume that v_1 is left of v_n and $\psi(v_1)$ is left of $\psi(v_n)$ without loss of generality. Then, from Proposition 6, for paths of $(v_1, \dots, v_n)$ and $(\psi(v_1), \dots, \psi(v_n))$,

$$T^{sub}_1(v_1, v_2) \geq_C T^{sub}_1(v_n, v_{n-1}) \text{ and } T^{sub}_2(\psi(v_1), \psi(v_2)) \geq_C T^{sub}_2(\psi(v_n), \psi(v_{n-1})) \tag{2.7}$$

because T_1 and T_2 are normal trees. There is no carbon position lists that satisfy Eq. (2.5), (2.6), and (2.7).

Therefore, T_1 does not represent the same molecular graph as T_2. □

2.5 Methods

We propose an algorithm BfsBenNaphEnum for enumerating chemical compounds containing benzene rings and naphthalene rings as cyclic structures. BfsBenNaphEnum utilizes our previously developed algorithms BfsSimEnum, BfsMulEnum [95], and assigns carbon position lists.

2.5.1 Calculation of the number of benzene rings and naphthalene rings

Before enumerating the molecular trees, the number of benzene rings and naphthalene rings in the trees must be defined first. Since a naphthalene ring is represented by two benzene nodes bonding together with a merge bond, the number of merge bonds is calculated to determine the number of naphthalene rings. To calculate the number if benzene nodes and merge bonds, degree of unsaturation (DoU) of a chemical formula and the number of carbon atoms are used. DoU is equal to the number of double bonds and the number of cyclics plus twice the number of triple bonds in that chemical structure, which can be calculated directly from the chemical formula by the Equation 2.8 [75].

$$DoU = 1 + \frac{\sum_{l_i} num(l_i)(val(l_i) - 2)}{2} \tag{2.8}$$

, where $num(l_i)$ is the number of atom type l_i in that chemical formula and $val(l_i)$

is the valence of atom type l_i. Let $\Sigma = \{l_1, l_2, ..., l_m\}$ be a set of m atom types and $S = \{n_b, n_{l_1}, n_{l_2}, ..., n_{l_m}\}$ be a vector representation of a chemical formula, where n_{l_i} is the number of atom type l_i, and n_b is the number of benzene nodes in a chemical structure, which is initialized to zero. Let $num(S, l_i)$ be the number of atom with label l_i in S, and $DoU(S)$ be degree of unsaturation of chemical formula S calculated from Equation 2.8. Algorithm 1 takes a chemical formula as the input and computes a set of all possible pairs of chemical formula with the number of benzene nodes and the corresponding number of merge bonds.

2.5.2 Modification of BfsSimEnum and BfsMulEnum

Suppose that the numbers n_{l_i} of atoms with label l_i for all $l_i \in \Sigma$, the numbers n_b, n_n of benzene rings and naphthalene rings are given. BfsBenNaphEnum introduces a special label 'b' representing a benzene node to Σ with $b > l_i \in \Sigma$ and $val(b) = 6$, and executes BfsSimEnum to generate all non-redundant molecular trees T such that $num(T, l_i) = n_{l_i}$ for $l_i \in \Sigma$ except $l_i = b, C$ and $num(T, b) = n_b + 2n_n$, $num(T, C) = n_C - 6n_b - 10n_n$. At this time, all edges of enumerated trees are single because BfsSimEnum generates only simple trees. Then, we modify BfsMulEnum to assign n_n merge bonds to edges between benzene nodes in each tree enumerated by BfsSimEnum in addition to adding $1 + \sum_{l_i \in \Sigma, l_i \neq b} num(T, l_i)(val(l_i) - 2)/2$ bonds to edges between usual nodes. It should be noted that multiple bonds cannot be assigned to edges connected to benzene nodes since a carbon atom in benzene rings and naphthalene rings is connected with another adjacent atom by a single bond.

2.5.3 Assignment of carbon positions for molecular trees

In this algorithm, we traverse along the tree T from the rightmost deepest benzene node to the root in reverse BFS order because an adjacent node list depends on carbon position lists of descendant nodes. For each benzene node v we found, we assign a carbon position list not to violate the conditions of normal form.

Table 2.1: Carbon position lists for A_v^T, where v is the root, and $|A_v^T[1]| \geq 3$

$\lvert A_v^T[1]\rvert$	$\lvert A_v^T[2]\rvert$	C_v^T
3	0	((1,2,3)), ((1,2,4)), ((1,3,5))
3	3	((1,2,3),(4,5,6)), ((1,2,4),(3,5,6)), ((1,3,5),(2,4,6))
4	0	((1,2,3,4)), ((1,2,3,5)), ((1,2,4,5))
5	0	((1,2,3,4,5))
6	0	((1,2,3,4,5,6))

The pseudocode of assignment part in BfsBenNaphEnum is given in Algorithms 2 and 3. We always assign carbon position 1 to the first node in A_v^T (line 27 in ASSIGN function) due to Proposition 5, which is the parent node of v if v is not the root (Proposition 2). If

Algorithm 1 Calculation algorithm of the number of benzene nodes and merge bonds

```
1: function CALCULATE_NUM_BENZENE($S$)
2:     $Result := \emptyset$
3:     $Result.push((S, 0))$
4:     $unfinish\_list := Result$
5:     while $|unfinish\_list| > 0$ do
6:         $(S_1, num\_mergebond) := unfinish\_list.pop()$
7:         $S_2 := S_1$
8:         if $DoU(S_2) \geq 4$ and $num(S_2, carbon) \geq 6$ and $num\_mergebond = 0$ then
9:             $n_b := num(S_2, benzene) + 1$
10:            $set(S_2, benzene, n_b)$
11:            $n_C := num(S_2, carbon) - 6$
12:            $set(S_2, carbon, n_C)$
13:            $Result.push((S_2, num\_mergebond))$
14:            $unfinish\_list.push((S_2, num\_mergebond))$
15:        end if
16:        if $DoU(S_1) \geq 3$ and $num(S_1, carbon) \geq 4$ then
17:            if $num(S_1, benzene) - (2 \cdot num\_mergebond) \geq 1$ then
18:                $n_b := num(S_1, benzene) + 1$
19:                $set(S_1, benzene, n_b)$
20:                $n_C := num(S_1, carbon) - 4$
21:                $set(S_1, carbon, n_C)$
22:                $num\_mergebond := num\_mergebond + 1$
23:                $Result.push((S_1, num\_mergebond))$
24:                $unfinish\_list.push((S_1, num\_mergebond))$
25:            end if
26:        end if
27:    end while
28:    return Result
29: end function
```

Algorithm 2 Assignment algorithm of carbon positions for a molecular tree T

```
1: function ASSIGN_CARBON_POSITIONS($T$)
2:     $v$ := the last benzene node of $T$ in BFS order
3:     ASSIGN($T, v$)
4: end function
```

```
1: function ASSIGN($T, v$)
2:     if $v$ is null then
3:         $\mathcal{P}$ := the set of all pairs of nodes $(v_1, \ldots, v_n)$ such that $v_1$ is left of $v_n$, the path
       from $v_1$ to $v_n$ is a symmetric path, and $v_1$ and $v_n$ are not connected by a merge bond
4:         if $T^{sub}(v_1, v_2) \geq_C T^{sub}(v_n, v_{n-1})$ for all $(v_1, v_n) \in \mathcal{P}$ then
5:             output $T$
6:         end if
7:         return
8:     end if
9:     if the next benzene node of $v$ in reverse BFS order exists then
10:        $v'$ := the next benzene node of $v$ in reverse BFS order
11:    else
12:        $v'$ := null
13:    end if
14:    if $|A_v^T| = 0$ then
15:        ASSIGN($T, v'$)
16:        return
17:    end if
18:    if $v$ is the root of $T$ then
19:        if $|A_v^T[1]| \geq 3$ then
20:            for each valid carbon position list $p$ in Table 2.1 do
21:                $C_v^T := p$
22:                ASSIGN($T, v'$)
23:            end for
24:            return
25:        end if
26:    end if
27:    $C_v^T[1][1] := 1$
28:    if $(v, A_v^T[1][1])$ is a merge bond then
29:        $C_v^T[1][2] := 2$
30:    end if
31:    ASSIGN_CHILD($T, v, A_v^T[1][1], v'$)
32: end function
```

Algorithm 3 Assignment algorithm for adjacent nodes of a benzene node v

```
function ASSIGN_CHILD($T, v, w, v'$)
    if $w$ is $null$ then
        $flag := true$
        if $v$ is not connected by a merge bond with the parent node then
            if $\phi_b \in Aut_b$ such that $C_v^T > \phi_b(C_v^T)$ exists then
                $flag := false$
            end if
        end if
        if $v$ is the root of $T$ then
            if a benzene node connected by a merge bond with $v$ exists then
                $v_c :=$ the benzene node connected by a merge bond with $v$
                if $T^{sub}(v, v_c) =_C T^{sub}(v_c, v)$ then
                    if $\phi_n \in Aut_n$ such that $C^T_{(v,v_c)} > \phi_n(C^T_{(v,v_c)})$ exists then
                        $flag := false$
                    end if
                else
                    if $C_v^T = \phi_{ref}(C_v^T)$ and $C_{v_c}^T > \phi_{ref}(C_{v_c}^T)$ then
                        $flag := false$
                    end if
                end if
            end if
        end if
        if $flag$ then
            ASSIGN($T, v'$)
        end if
        return
    end if
    if the next node of $w$ in $A_v^T$ exists then
        $w' :=$ the next node of $w$ in $A_v^T$
    else
        $w' := null$
    end if
    if $w$ has been already assigned then
        ASSIGN_CHILD($T, v, w', v'$)
        return
    end if
    Let $i$ and $j$ be two integers such that $A_v^T[i][j] = w$
    for $p = 2, \ldots, 6$ do
        if $p$ has not been assigned and $p > \max_{j'<j} C_v^T[i][j']$ then
            $C_v^T[i][j] := p$
            if $(v, A_v^T[i][j])$ is a merge bond then
                $C_v^T[i][j+1] := p + 1$
            end if
            ASSIGN_CHILD($T, v, w', v'$)
        end if
    end for
end function
```

v is the root and $|A_v^T[1]| \geq 3$, we assign carbon position lists in Table 2.1 to v immediately for the sake of efficiency. Carbon position lists in Table 2.1 satisfy condition 4 of the normal form, and all the cases are included in the table. For other carbon positions from 2 to 6, we use ASSIGN_CHILD to assign such positions to the remaining adjacent nodes. For example, let T_1 in Figure 2.10 be output without any carbon position list by BfsMulEnum. T_1 has a benzene node v_1, and $A_{v_1}^{T_1} = ((v_4), (v_5), (v_2, v_3))$. First, carbon position 1 is assigned to $A_{v_1}^{T_1}[1][1] = v_4$, that is, $C_{v_1}^{T_1}[1][1] = 1$. Since v_1 is the root and $|A_{v_1}^{T_1}[1]| = 1 < 3$, Table 2.1 is not used, and the other nodes v_5, v_2, v_3 are assigned by ASSIGN_CHILD. For v_5, each carbon position from 2 to 6 is examined (line 38 in ASSIGN_CHILD). For v_2, each position from 2 to 6 except the position assigned to v_5 is examined (line 39). For v_3, each position from 2 to 6 that is more than the position assigned to v_2 except the position assigned to v_5 is examined (line 39) because v_2 and v_3 have the same subtree and condition 3 must be satisfied. Thus, $C_{v_1}^{T_1} = ((1), (2), (3, 4)), ((1), (2), (3, 5)), ((1), (2), (3, 6)), \ldots,$ $((1), (3), (2, 4)), ((1), (3), (2, 5)), \ldots, ((1), (6), (4, 5))$ are examined, where $((1), (6), (2, 3))$, $((1), (6), (2, 4)), ((1), (5), (2, 3))$ and so on are discarded in the next step.

The illustration of assigning position lists process is given in Figure 2.15, where the molecular tree in the rightmost leaf of family tree is discarded because its carbon position list is $((1), (4), (3, 6))$ and there is a mapping function ϕ such that $((1), (4), (3, 6)) > \phi((1), (4), (3, 6)) = ((1), (4), (2, 5))$ which violates the normal form rule.

Since an input of this part, that is, an output of BfsMulEnum, satisfies conditions 1, 2 of the normal form, BfsBenNaphEnum always outputs normal trees. In ASSIGN_CHILD, a distinct carbon position list is always assigned, and all patterns are assigned (line 40). For each benzene node v, after assignment of a carbon position list to A_v^T, whether or not C_v^T violates conditions 4, 5 of the normal form is confirmed (lines 5, 13, 17 in ASSIGN_CHILD). After carbon position lists are assigned to all benzene nodes, condition 6 is confirmed (line 4 in ASSIGN). Hence, BfsBenNaphEnum outputs all distinct normal trees corresponding to the input chemical formula.

Theorem 1. *BfsBenNaphEnum outputs all non-redundant molecular graphs that are solutions of Problem 1.*

Figure 2.16 shows another example T_7 of molecular trees. T_7 includes four benzene nodes v_5, v_4, v_3, v_2 in reverse BFS order, and edges (v_2, v_4), (v_3, v_5) are merge bonds. To generate normal trees from T_7, first, our algorithm assigns carbon position lists for $A_{v_5}^{T_7} = ((v_3), (v_7))$ as $C_{v_5}^{T_7} = ((1, 2), (3)), ((1, 2), (4)), ((1, 2), (5)), ((1, 2), (6))$. In a similar way, for $A_{v_4}^{T_7} = ((v_2), (v_6))$, $C_{v_4}^{T_7} = ((1, 2), (3)), ((1, 2), (4)), ((1, 2), (5)), ((1, 2), (6))$. For $A_{v_3}^{T_7} = ((v_1), (v_5))$, $C_{v_3}^{T_7} = ((1), (2, 3)), ((1), (3, 4)), ((1), (4, 5))$,
$((1), (5, 6))$ are examined. In line 5 of ASSIGN_CHILD, $((1), (4, 5))$ and $((1), (5, 6))$ are discarded because $\phi_b(((1), (4, 5))) = ((1), (3, 4))$, $\phi_b(((1), (5, 6))) = ((1), (2, 3))$ for the reflection map ϕ_b with respect to the axis through positions 1 and 4, and these violate condition 4. In a similar way, for $A_{v_2}^{T_7} = ((v_1), (v_4))$, $C_{v_2}^{T_7} = ((1), (2, 3)), ((1), (3, 4))$ are assigned.

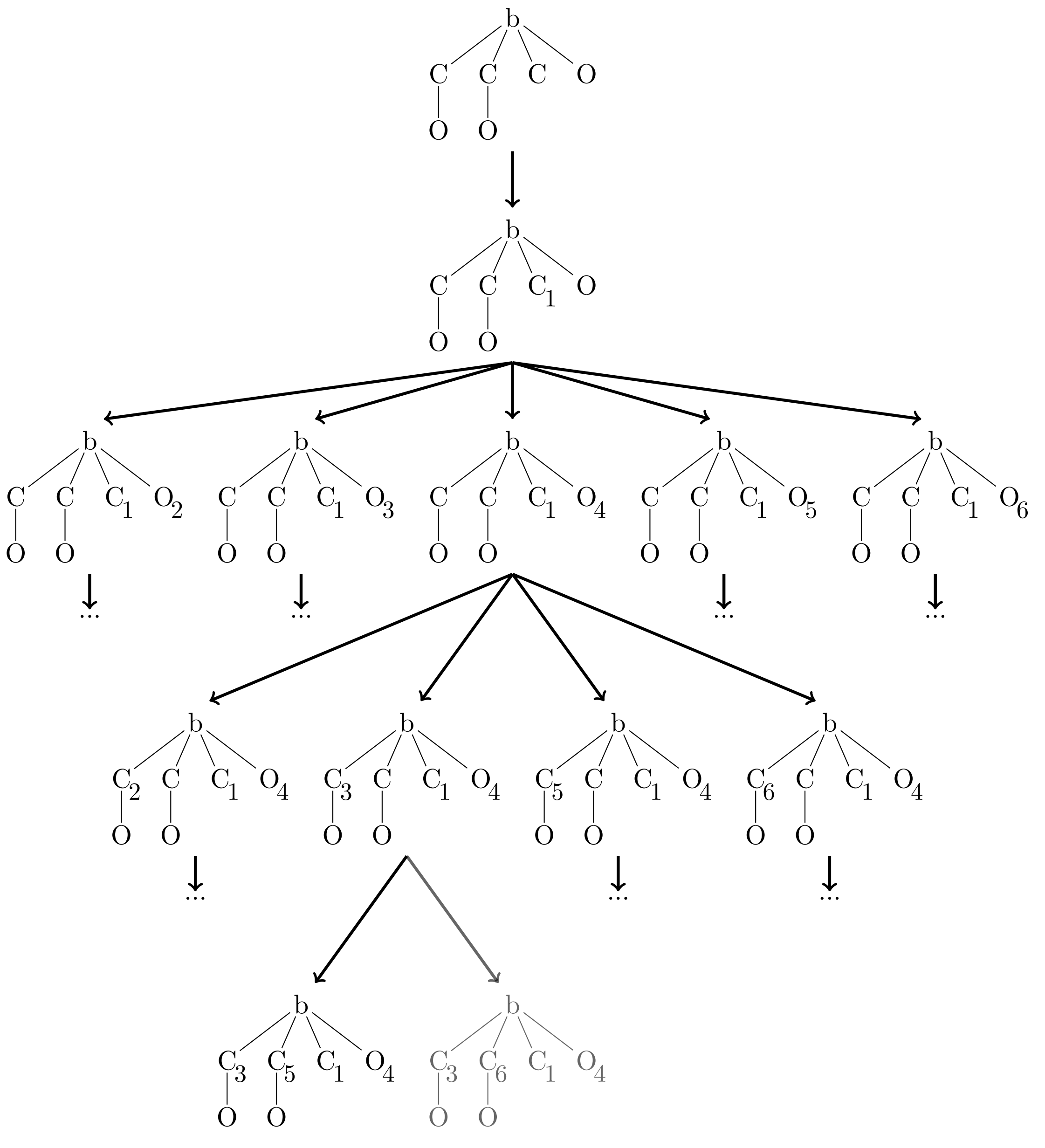


Figure 2.15: Family tree showing process of assigning carbon position lists to a benzene node, where *b* is a label of benzene node. Molecular graphs with low opacity contain invalid position lists.

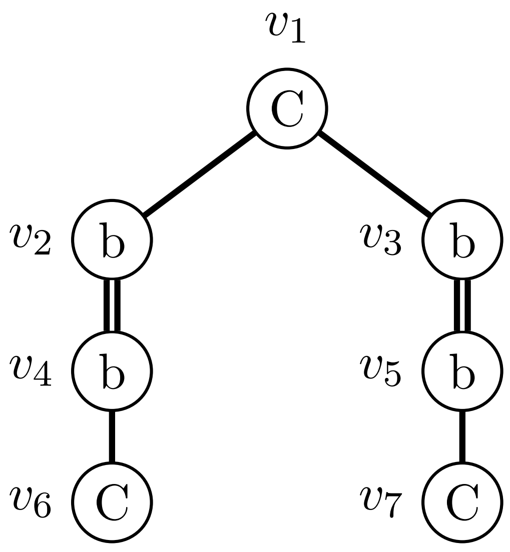


Figure 2.16: Example of a molecular tree T_7.

After carbon position lists are assigned to all benzene nodes, condition 6 is confirmed in line 4 of ASSIGN. If $C_{v_2}^{T_7} \neq C_{v_3}^{T_7}$, then there is one symmetric path, $\mathcal{P} = \{(v_2, v_3)\}$, and $T_7(v_2) \geq_C T_7(v_3)$ must be satisfied. It means that $C_{v_4}^{T_7} = C_{v_5}^{T_7} = ((1,2),(3)),((1,2),(4)),((1,2),(5)),((1,2),(6))$ and $C_{v_2}^{T_7} = ((1),(3,4)) > C_{v_3}^{T_7} = ((1),(2,3))$, or $C_{v_4}^{T_7} > C_{v_5}^{T_7}$ and $C_{v_2}^{T_7} \neq C_{v_3}^{T_7}$. Hence, there are $4 + \binom{4}{2} \cdot 2 = 16$ structures. On the other hands, if $C_{v_2}^{T_7} = C_{v_3}^{T_7} = ((1),(2,3))$ (or $C_{v_2}^{T_7} = C_{v_3}^{T_7} = ((1),(3,4))$), then $\mathcal{P} = \{(v_2, v_3), (v_4, v_5)\}$, and both of $T_7(v_2) \geq_C T_7(v_3)$ and $T_7(v_4) \geq_C T_7(v_5)$, that is, $C_{v_4}^{T_7} \geq C_{v_5}^{T_7}$, must be satisfied. Hence, there are $4 + 3 + 2 + 1 = 10$ structures. In total, $16 + 10 \cdot 2 = 36$ structures are generated by BfsBenNaphEnum for T_7.

2.5.4 Complexity analysis

Let n and n_C be the total number of input atoms and carbon atoms from users, respectively. First, we analyze the time complexity of the calculation of the number of benzene rings and naphthalene rings (Algorithm 1). In this algorithm, we perform a constant number of operations for one iteration, decrease the number of carbon atoms by at least four, and add at most two elements to the unfinish list. The first time (line 14) is adding one benzene ring to the chemical formula and another time (line 24) is adding one benzene ring and one merge bond to the chemical formula. Therefore, the time complexity for the first iteration is

$$T_1(n_C) \leq O(1) + 2 \cdot T_1(n_C - 4). \tag{2.9}$$

For each chemical formula added in the previous iteration, we perform the same set of operations as the previous one. Therefore, the time complexity for one chemical formula in the second iteration is

$$T_1(n_C - 4) \leq O(1) + 2 \cdot T_1(n_C - 8). \tag{2.10}$$

Next, we substitute Equation 2.10 into Equation 2.9, which results in

$$\begin{aligned}
T_1(n_C) &\leq O(1) + 2\,(O(1) + 2 \cdot T_1(n_C - 8)) && (2.11)\\
&\leq (1+2) \cdot O(1) + 2^2 \cdot T_1(n_C - 4 \cdot 2) && (2.12)\\
&\leq \left[\sum_{i=0}^{k-1} 2^i\right] \cdot O(1) + 2^k \cdot T_1(n_C - 4 \cdot k) && (2.13)\\
&\leq (2^k - 1) \cdot O(1) + 2^k \cdot T_1(n_C - 4 \cdot k), && (2.14)
\end{aligned}$$

where k is the number of iterations. In the worst case, we iterate until the number of carbon atoms equals to zero ($n_C - 4 \cdot k = 0$) so the maximum number of iterations, k, we need is $\frac{n_C}{4}$. Moreover, the last iteration performs a constant number of operations so $T_1(0) = O(1)$. Substitute $k = \frac{n_C}{4}$ and $T_1(0) = O(1)$ into Equation 2.14, we obtain

$$\begin{aligned}
T_1(n_C) &\leq (2^{\frac{n_C}{4}} - 1) \cdot O(1) + 2^{\frac{n_C}{4}} \cdot T_1(n_C - 4 \cdot \frac{n_C}{4}) && (2.15)\\
&\leq (2^{\frac{n_C}{4}} - 1) \cdot O(1) + 2^{\frac{n_C}{4}} \cdot T_1(0) && (2.16)\\
&\leq (2^{\frac{n_C}{4}} - 1) \cdot O(1) + 2^{\frac{n_C}{4}} \cdot O(1) = O(2^{\frac{n_C}{4}}). && (2.17)
\end{aligned}$$

Since $2^{1/4} \leq 1.1893$ and $n_C \leq n$, the time complexity of Algorithm 1 is $O(1.1893^n)$.

For the enumeration of molecular trees step, we utilize our previous work BfsSimEnum and BfsMulEnum [95], whose time complexity is exponential.

Algorithm 2, which is the algorithm for assigning carbon position lists to benzene nodes in a molecular tree, begins with function ASSIGN_CARBON_POSITIONS, which initializes a variable v and calls function ASSIGN. ASSIGN_CARBON_POSITIONS function requires the same time complexity as ASSIGN function for the same tree.

In ASSIGN function, we examine at most $\binom{n}{2} = O(n^2)$ paths to obtain the set P in line 3. For each path, we check whether it is a symmetric path or not by looking at label and a carbon position list of each node, which can be done in a constant time so the time complexity of line 3 is $O(n^2)$. However, we access line 3 in not all iterations but only when v is null (line 2) or we reach the root node of the tree, which is the last iteration of ASSIGN function.

Then, there are three ways to recursively call ASSIGN function itself (line 15, line 22, and line 31 via ASSIGN_CHILD function) depending on the condition of v. Line 15 and line 31 is called at most one time per iteration, while line 22 is called at most three times. However, line 22 is called only when v is the root node and thus we can ignore this factor. Because calling via ASSIGN_CHILD function (line 31) requires higher time complexity than line 15, we analyze the time complexity only by looking at Algorithm 3.

ASSIGN_CHILD recursively calls itself at most 5^6 times per a benzene ring because there are at most 5^6 ways to assign a carbon position list. Each time the number of atoms, n, decreases by six because one benzene node represents six carbon atoms. Moreover, there are a constant number of operations in one iteration of ASSIGN function so the time complexity of ASSIGN function is

$$\begin{aligned} T_2(n) &\leq O(1) + 5^6 T_2(n-6) && (2.18)\\ &\leq O(1) + \left(5^6\right)^k \cdot T_2(n - 6 \cdot k), && (2.19)\end{aligned}$$

where k is the number of iterations. In the worst case, ASSIGN function is repeated until $n - 6 \cdot k = 0$ or $k = n/6$.

$$T_2(n) \leq O(1) + (5^6)^{n/6} \cdot T_2(0) \tag{2.20}$$

Because the time complexity of the last iteration is $O(n^2)$, $T_2(0) = O(n^2)$, due to line 3, we obtain

$$T_2(n) \leq O(1) + 5^n \cdot O(n^2) = O(5^n \cdot n^2). \tag{2.21}$$

It is to be noted that if K chemical structures are outputted, the computation time per structure is $O(n^2 \cdot 5^n/K)$ for Algorithm 2 and Algorithm 3. Therefore, these two algorithms are not an output polynomial-time one. However, the factor 5^n is the worst-case estimate and it is highly expected that the algorithm works much faster in practice because of the pruning step, which discards the intermediate trees violating normal form rule immediately.

For space complexity, we store only one molecular graph at a time and do not store the information of previously enumerated compounds. Because we keep the information of a molecular tree in an array of C++ structures, where one structure represents one node in a molecular tree, this work uses a polynomial space complexity of degree one excluding the output file of the enumerated compounds.

2.6 Results

In this section, we show that our proposed method can enumerate chemical compounds with benzene rings and naphthalene rings correctly and efficiently. The implementation of BfsBenNaphEnum is available on our supplementary web site, `http://sunflower.kuicr.kyoto-u.ac.jp/jira/bfsenum/`. For the evaluation, although MOLGEN 3.5 is more suitable than MOLGEN 5.0 to enumerate tree-like compounds because MOLGEN 3.5 offered the possibility to define substructures like benzene or naphthalene as macro atoms but MOLGEN 3.5 cannot handle all the cases provided in Table 2.2, we compared proposed algorithm with MOLGEN 5.0. Thereby, we implemented the proposed algorithm and a well-known general purpose structure generator, MOLGEN 5.0, on a computer with 3.47 GHz intel Xeon CPU and 23.5 GiB memory and compared their computational time and enumerated structures together.

Since MOLGEN can enumerate chemical compounds without restriction on the structure, we must specify a benzene ring and a naphthalene ring as a substructure as well as input the number of cycles and aromatic bonds so that the enumerated structures contain only benzene rings and naphthalene rings as cyclic structures.

Table 2.2: Results on execution time (sec), the number of enumerated structures by BfsBenNaphEnum and MOLGEN for several instances.

Chemical	#atoms						#enumerated	Computational time (sec)	
formula	n	b	C	N	O	H	structures	BfsBenNaphEnum	MOLGEN
$C_7O_2H_8$	0	1	1	0	2	8	19	0.001	0.053
$C_8O_3H_{10}$	0	1	2	0	3	10	307	0.002	0.124
$C_9O_4H_{10}$	0	1	3	0	4	10	6,406	0.010	1.699
$C_{10}N_2O_4H_{10}$	0	1	4	2	4	10	8,333,991	12.260	957.53
	1	0	0	2	4	10	7,980	0.031	69.51
$C_{11}N_2H_{10}$	0	1	5	2	0	10	9,012	0.021	630.44
	1	0	1	2	0	10	56	0.005	24.061
$C_{12}N_1O_1H_{11}$	0	1	6	1	1	11	80,883	0.155	2,611.57
	0	2	0	1	1	11	33	0.001	98.99
	1	0	2	1	1	11	888	0.009	560.98
$C_{13}O_2H_{12}$	0	1	7	0	2	12	162,122	0.289	6,497.55
	0	2	1	0	2	12	190	0.002	2,069.3
	1	0	3	0	2	12	2,458	0.013	1,731.92
$C_{14}O_4H_{12}$	0	1	8	0	4	12	19,514,480	35.655	197,264.54
	0	2	2	0	4	12	15,581	0.021	107,509.42
	1	0	4	0	4	12	337,178	1.061	97,326.71

As can be seen from Table 2.2, where ‘n’ and ‘b’ denote a naphthalene ring and a benzene ring, respectively, BfsBenNaphEnum enumerated chemical compounds much faster than MOLGEN while giving the same number of enumerated structures. BfsBenNaphEnum was from 50 times to 5,000,000 times faster than MOLGEN for instances with 7 to 14 carbon atoms. In this study, we examined chemical formulas including up to two benzene rings and one naphthalene ring because MOLGEN was not able to output results in practical time for chemical formulas including more benzene rings and naphthalene rings.

We plotted the relation between the number of enumerated structures and the computational time for BfsBenNaphEnum and MOLGEN 5.0 in Figure 2.17, where both x-axis and y-axis are in a log scale. It is seen from the figure that the execution time of BfsBenNaphEnum is much smaller than that of MOLGEN. The period of time which MOLGEN 5.0 used to enumerate approximate 10 compounds is similar to the period of time which BfsBen-NaphEnum used to enumerate approximate 10,000 compounds, which is 1000 times more than MOLGEN.

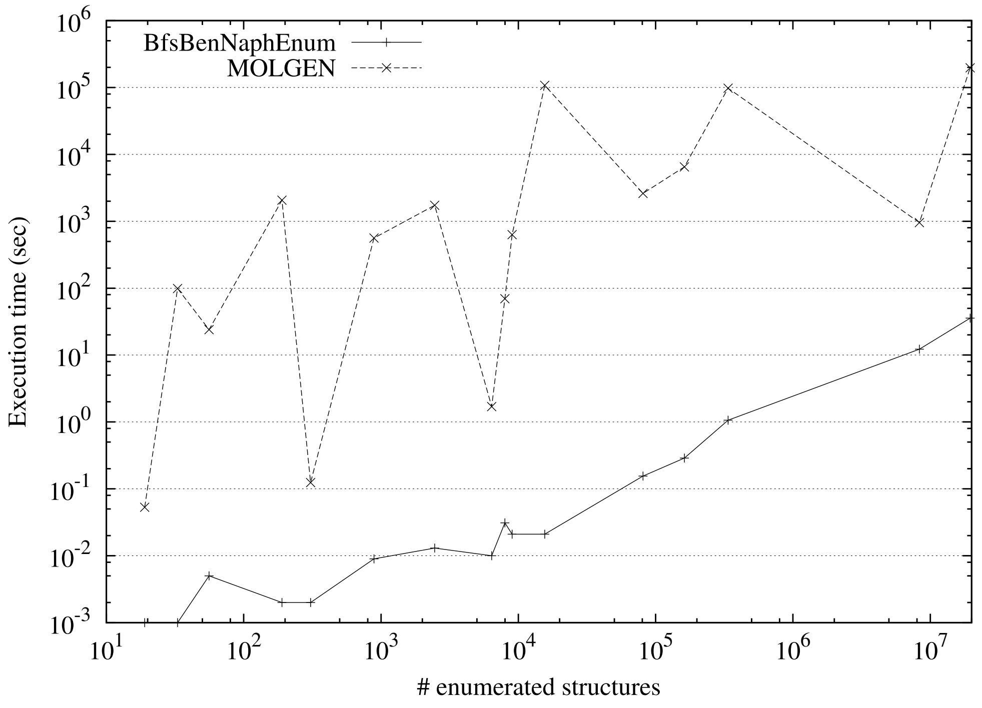


Figure 2.17: Relation between the number of enumerated structures and the computational time (sec).

The relation between the number of heavy atoms in a chemical formula and the computational time used to enumerate that chemical formula is also illustrated in Figure 2.18, where the y-axis is in a log scale. It is shown that BfsBenNaphEnum is significantly faster than MOLGEN. The enumeration of a chemical formula containing 13 heavy atoms by BfsBenNaphEnum is faster than the enumeration of a chemical formula containing nine heavy atoms by MOLGEN.

Table 2.3 also compares the number of discovered compounds in PubChem, which are not limited to tree-like chemical compounds, with the number of compounds enumerated by the proposed algorithm for several chemical formulas. When the number of carbon atoms grows, the number of discovered compounds is much less than that of enumerated compounds. This implies that there are a numerous number of unknown large compounds to be discovered, which possibly include important compounds with desired properties.

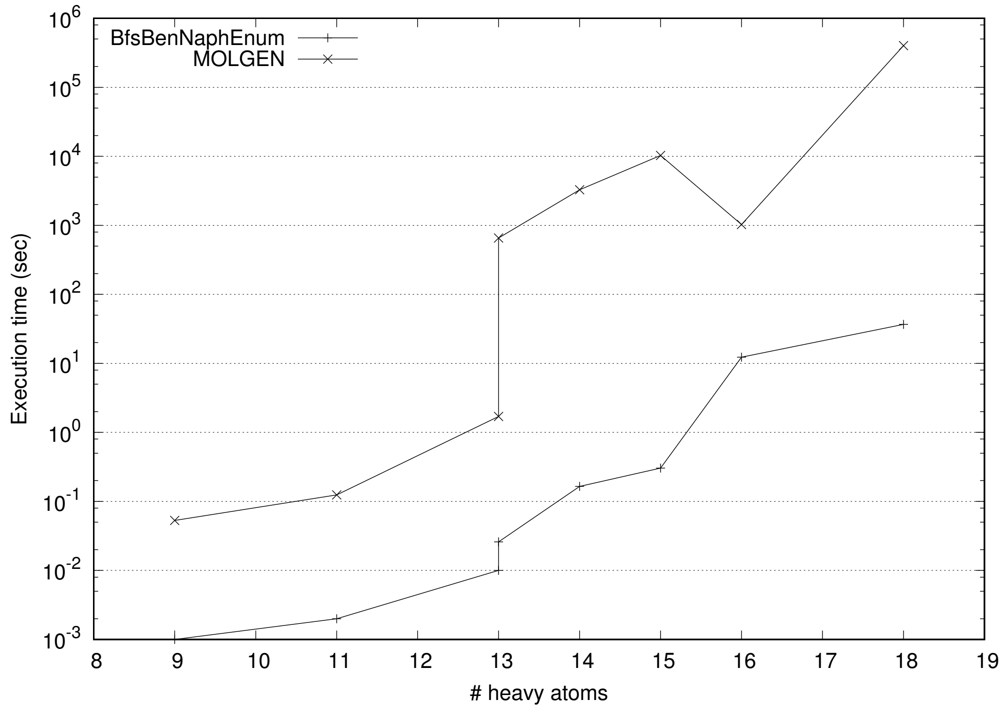


Figure 2.18: Relation between the number of heavy atoms in the chemical formula and the computational time (sec).

2.7 Discussion

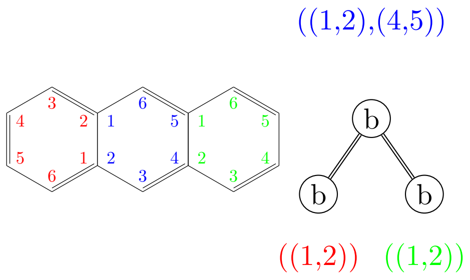
We proposed a way to represent a benzene ring in a molecular tree by regarding it as a new defined atom called a benzene node (nodes with label *b* in Figure 2.20) with valence six and introducing a new attribute named a carbon position list (a list next to benzene nodes in Figure 2.20) to benzene rings. Carbon position of an atom specifies which carbon in a benzene ring that the corresponding atom bonds with.

We also proposed a novel representation of a naphthalene ring in a tree structure. This representation considers a naphthalene ring as two fused benzene rings and denotes it as two benzene nodes bonding with a newly defined type of bond called a merge bond (a double edge between two benzene nodes in Figure 2.20). With a merge bond, a molecular tree can represent a structure containing naphthalene rings without defining new kind of atom. Moreover, the concept of this merge bond can be generalized to illustrate several polycyclic aromatic compounds, which decreases the limitation of the enumerated structures. For example, anthracene and phenanthrene can be represented by three benzene nodes bonding with two merge bonds with different carbon position lists (see Figure 2.19).

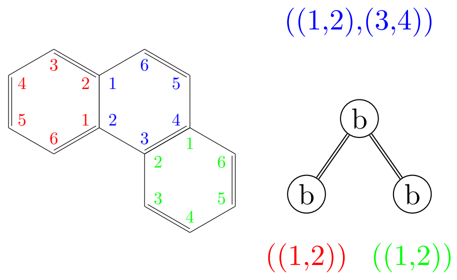
Moreover, since a benzene ring and a naphthalene ring are symmetric structures, we defined a rule to assign carbon position lists such that no redundant structures due to the symmetry of a benzene ring and a naphthalene ring are enumerated using the concept of

Table 2.3: The number of enumerated structures by BfsBenNaphEnum and the number of chemical compounds exist in PubChem database for several instances.

Chemical formula	#compounds in PubChem	#enumerated compounds
$C_7O_2H_8$	728	19
$C_8O_3H_{10}$	1,602	307
$C_9O_4H_{10}$	1,469	6,406
$C_{10}N_2O_4H_{10}$	1,592	8,341,971
$C_{11}N_2H_{10}$	790	9,068
$C_{12}N_1O_1H_{11}$	1,582	81,804
$C_{13}O_2H_{12}$	1,239	164,770
$C_{14}O_4H_{12}$	1,397	19,867,239



(a)



(b)

Figure 2.19: A representation of an anthracene ring (a) and a phenanthrene ring (b) by tree structures using the concept of merge bond

an automorphism group.

The algorithm of this work consists of two main steps. Given the number of benzene rings, the number of naphthalene rings as well as a chemical formula, BfsSimEnum and BfsMulEnum are applied such that they can enumerate non-redundant molecular trees with benzene nodes based on the center-rooted left-heavy condition. Next, the new extension *BfsBenNaphEnum* generates normal molecular trees by assigning all canonical carbon position lists to all benzene nodes. The canonical carbon position list is a list which is the smallest list among all lists representing the same structure. The example of a finalized center-rooted left-heavy molecular tree whose benzene nodes are assigned canonical carbon position lists is illustrated in Figure 2.20.

To show the performance of our algorithm, all non-redundant chemical structures were enumerated for several chemical formulas by BfsBenNaphEnum and MOLGEN 5.0, a well-known general purpose structure generator. It is shown that our algorithm is reliable

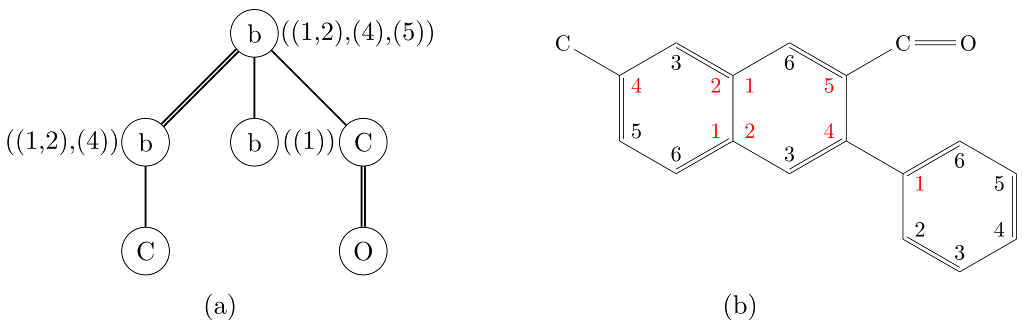


Figure 2.20: Illustration of a center-rooted left-heavy molecular tree (a) representing a chemical compound (b).

since it generated the same number of structures as MOLGEN, while expended much less computational time. BfsBenNaphEnum was from 50 times to 5,000,000 times faster than MOLGEN for instances with 7 to 14 carbon atoms in our experiments. This is mainly because the number of nodes decreases from six to one for each benzene ring and from ten to two for each naphthalene ring in a chemical structure. Another reason is that we enumerate chemical structures in the form of tree structures, which are easier to examine the redundancy than normal graphs.

However, our algorithm is limited to tree-like chemical structures without any cyclic structures except benzene rings and naphthalene rings while MOLGEN does not have such limitation. Therefore, in the future, we would like to extend the algorithm such that it can enumerate more complex cyclic structures, such as polycyclic aromatic compounds and nucleotides. Besides, in order to make enumeration tools practical, we need to rank enumerated structures because a large number of structures are usually enumerated. For that purpose, it might be useful to employ drug likeness filters such as Lipinski RO5, and QED score. Incorporation of such filters into our system is also important future work.

Chapter 3

Enumeration method for structural isomers containing user-defined structures based on breadth-first search approach

3.1 Background

Structure enumeration is a problem of generating all non-redundant chemical compounds based on given constraints. It can be applied to many problems in chemoinformatics and computational biology. For example, in structure elucidation and identification of unknown compounds, high-resolution mass spectrometry and nuclear magnetic resonance technique are widely used [5, 39, 41, 62]. The information obtained from these techniques is molecular fragments of a given compound, which requires a structure enumeration tool to generate candidate compounds before the exact structure is determined [26, 36, 54, 72]. Moreover, structure enumeration can also be applied in a kinetic modeling problem by generating reactant molecules from given analytic information [42] and to drug design by generating a list of compounds containing defined functional groups, active sites, or atom types [6, 71].

Structure enumeration has received much attention since the beginning of the DENDRAL project [47]. Since then, various software tools for structure enumeration have been developed, such as ASSEMBLE [4] and MOLGEN 5.0 [30], which aim to generate all theoretically possible compounds, and thus they may include unrealistic compounds. For this reason, OMG [63] was proposed using the dictionary of atom types provided in Chemistry Development Kit (CDK) [82, 83], a Java library for chemoinformatics, to eliminate the unrealistic compounds. These tools can generate all compounds but they consume significant resources in terms of time and computational power. On the other hand, alternative tools such as EnuMol [28, 35, 78], BfsSimEnum and BfsMulEnum [95], and BfsBenNaphEnum [38] have been developed in order to optimize the computation time but they have limitation in terms of the structure of compounds that can be enumerated. Because of this limitation, these alternative tools may not generate chemical compounds that satisfy users' requirements.

Moreover, the enumeration tool can be applied to several problems such as structure elucidation and drug design. Some of those problems have data of molecular fragments in the desired compounds. Accordingly, the objective of this work is to propose a novel enumeration tool, called BfsStructEnum, which uses a tree structure to represent a chemical compound for the efficiency of the computational time and allows users to input biconnected cyclic substructure of the desired compounds. The proposed tool, then, generates all non-redundant tree-like chemical compounds containing cyclic substructures defined by users corresponding to the input chemical formula. Allowing users to input cyclic substructures of the desired compounds has two main advantages over other enumeration tools consuming similarly low computation time. First, it ensures that all enumerated structures contain the required substructures. This can help them spending less time to find the target compound. Another one is that the scope of the structure of enumerated compounds is expanded from compounds containing a few kinds of cyclic substructure to acyclic compounds containing any cyclic substructures defined by users.

BfsStructEnum takes a chemical formula and biconnected chemical structures specified by users and returns all non-redundant chemical compounds containing no cyclic structures except for those provided by users. The chemical structures specified by users are called *substructures*, which must be biconnected structures. A cyclic structure is a structure containing at least one cycle. A biconnected structure with more than two nodes is a structure such that there exists no atom whose removal disconnects a structure. For example, Figure 3.1 (a) is a biconnected structure, while Figure 3.1 b is not because removing the red carbon atom disconnects the structure, where hydrogen atoms are omitted.

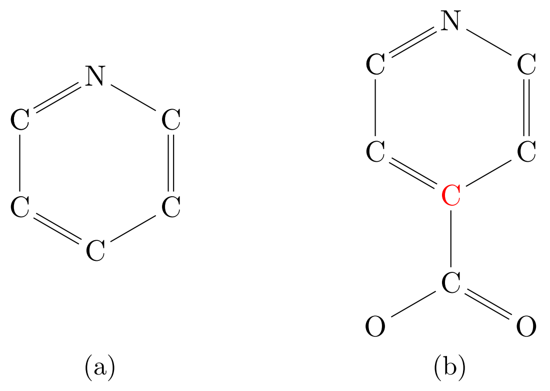


Figure 3.1: Example of a biconnected structure (a) and a non-biconnected structure (b)

In this work, a compound is represented by a molecular tree for the efficiency of the enumeration. In order for the molecular tree to keep the information of substructures, a substructure is compressed into a single node called *a substructure node*. A substructure node is labeled with special atom label indicating substructure that it represents. Besides, it has an attribute called *a position list* to keep the information of which atoms in the substructure bond with which adjacent nodes.

This work consists of four main steps as given in Figure 3.2 First, we analyze the structure of input substructures to determine their automorphism, which is necessary to detect the redundancy and calculate the number of structure nodes in the molecular graphs. After that, modified version of BfsSimEnum and BfsMulEnum [95] is used to generate molecular trees from the number of each atom types and structure nodes. Then, the position lists are assigned to all structure nodes in the tree such that no redundant structures are enumerated. The final result of this work is a collection of molecular graphs converted from molecular trees whose nodes and edges illustrate atoms and bonds of chemical compounds, respectively.

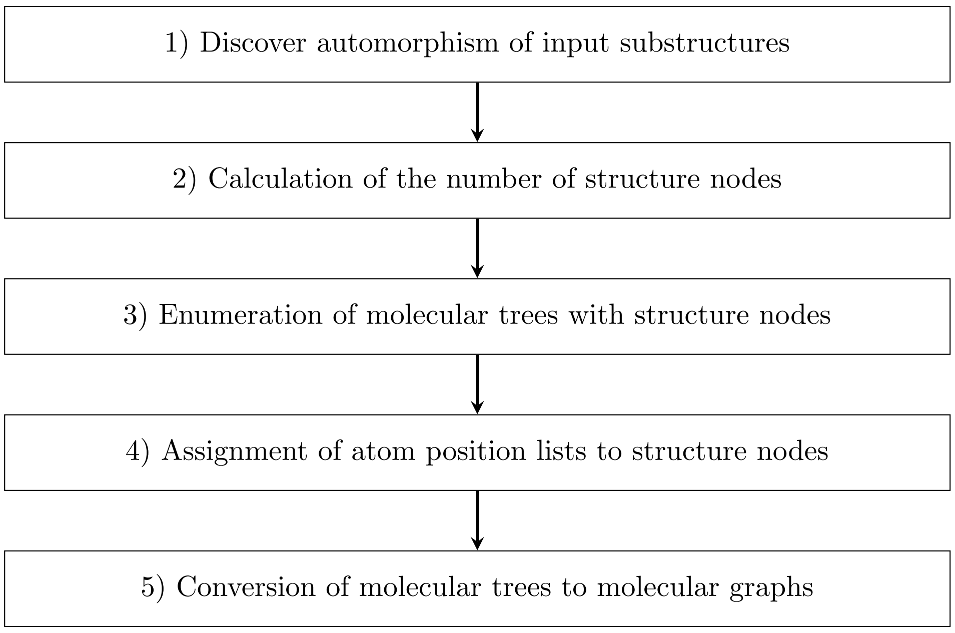

1) Discover automorphism of input substructures
2) Calculation of the number of structure nodes
3) Enumeration of molecular trees with structure nodes
4) Assignment of atom position lists to structure nodes
5) Conversion of molecular trees to molecular graphs

Figure 3.2: The flowchart concluding main processes of BfsBenNaphEnum

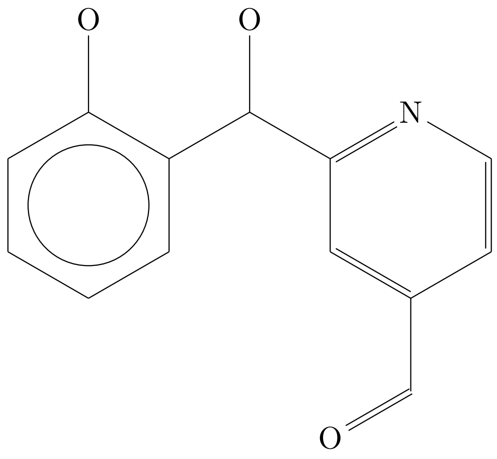
To assess the accuracy and efficiency of BfsStructEnum, we used it to enumerate chemical compounds from various chemical formulas and compared the results and computation time to those of MOLGEN 5.0. The comparison suggests that BfsStructEnum can enumerate chemical compounds correctly and efficiently.

3.2 Problem definition

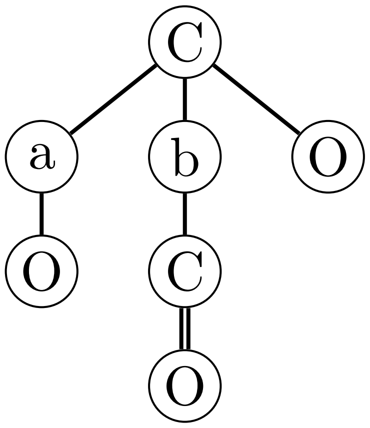
We first recall the definition of a set of atom labels (Σ), a molecular graph ($G(V, E)$), a label of a node $l(v)$, and a valence of an atom label $val(l_i)$ from Section 1.3.1, as well as the definition of a degree of a node $deg(v)$ and the number of nodes with label l_i in a graph G ($num(G, l_i)$) from Section 2.2. It must be noted that BfsStructEnum provides the most common valence for each chemical element. Users have to specify the valence manually if a chemical element has different valences. BfsStructEnum regards the same chemical element with different valences as different chemical elements. For example, nitrogen (N), which has valences of three and five, is generally considered to have a valence of three. If users would like to include nitrogen with valence five in the enumeration, they must specify nitrogen with valence five as another label, e.g. $N^{(5)}$, in Σ.

From the definition of isomorphism from Section 1.3.2, we define that two molecular graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are redundant with each other if there is an isomorphism between V_1 and V_2 because they correspond to the same chemical compounds. In this paper, the enumeration problem is defined as follows:

Problem 2. *Given the number n_{l_i} for each chemical element $l_i \in \Sigma$, a set of distinct cyclic and biconnected chemical structures with more than two nodes $S = \{s_1, s_2, ...\}$ such that s_a is not a subgraph of s_b, as well as the number n_{s_j} and valence of input structure s_j, $val(s_j)$, for each input structure $s_j \in S$, generate all non-redundant connected molecular graphs $G(V, E)$ containing cyclic structures $s_j \in S$ and no other cyclic structures such that $deg(v) = val(l(v))$ for all nodes $v \in V$ and $num(G, l_i) = n_{l_i}$.*



(a)



(b)

Figure 3.3: Example of a chemical compound (a) and its corresponding molecular tree (b), where a benzene ring and a pyridine ring are compressed into a node with label a and a node with label b, respectively, and hydrogen atoms are omitted.

During the enumeration step of this work, a chemical compound is represented by a tree structure, called a molecular tree, such that the specified substructure must be compressed into a single node, as shown in Figure 3.3. A molecular tree is defined as a rooted ordered tree $G(V, E)$, where V is a set of nodes labeled by chemical elements $l_i \in \Sigma$ or substructures $s_j \in S$ and E is a set of edges. A node whose label is a specified substructure is called *a structure node*. A molecular tree in which all structure nodes are labeled with the position lists is called *a position-assigned molecular tree*. A position-assigned molecular tree is converted to a molecular graph later by replacing structure nodes with cyclic structures and connecting atoms in cyclic structures with adjacent atoms based on the position lists of those structure nodes.

3.3 Preliminaries

3.3.1 A position list

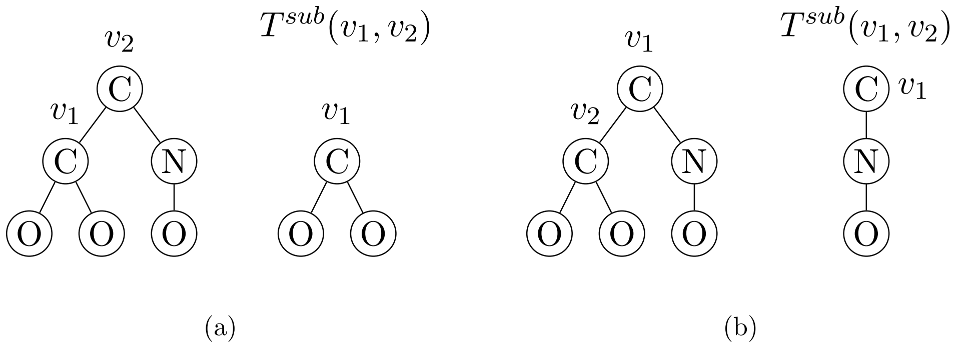


Figure 3.4: $T^{sub}(v_1, v_2)$, when v_1 is a child node of v_2 (a) and v_1 is the parent node of v_2 (b).

Given a molecular tree, T, and two nodes, v_1 and v_2, in T, $T^{sub}(v_1, v_2)$ is defined as an ordered tree rooted at v_1 obtained from T by eliminating the edge (v_1, v_2), as well as the connected component containing v_2 (see Figure 3.4). Let A be a list of lists of nodes, $A[i]$ is defined as the i^{th} list of A and $A[i][j]$ is defined as the j^{th} element of $A[i]$.

Definition 7. *An adjacent node list, $A^T(v)$, of a structure node v in a molecular tree T is a list of lists of nodes adjacent to v such that*

- *$u \in A^T(v)[i]$ if $T^{sub}(u, v) = T^{sub}(A^T(v)[i][1], s)$, where $T^{sub}(u, v)$ is defined in Section 2.3.3,*
- *$index(A^T(v)[i][j]) < index(A^T(v)[i][j+1])$ for all i, j,*
- *$index(A^T(v)[i][1]) < index(A^T(v)[i+1][1])$ for all i,*
- *all nodes w adjacent to v appear in $A^T(v)[i]$ for some i, and*

- *$A^T(v)[i]$ and $A^T(v)[j]$ are disjoint for all i and j such that $i \neq j$.*

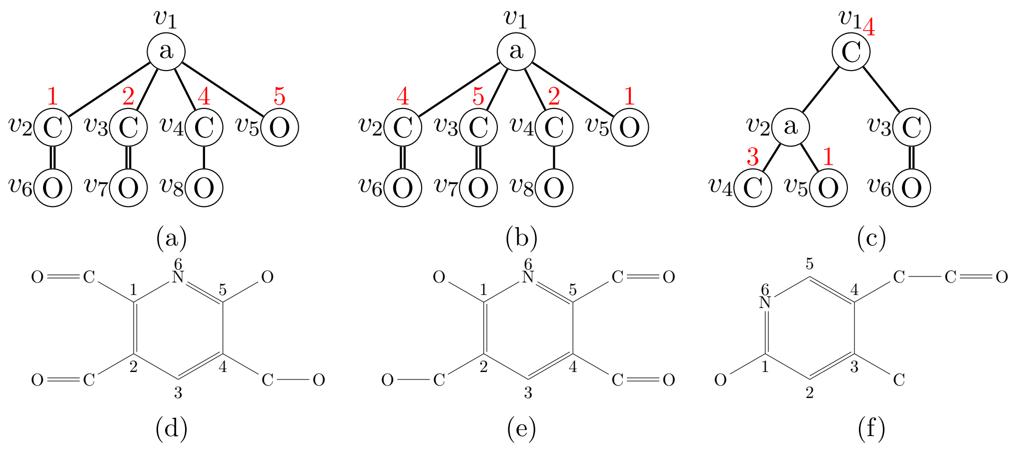


Figure 3.5: Examples of position-assigned molecular trees T_1 (a), T_2 (b), and T_3 (c) and corresponding molecular graphs of T_1 (d), T_2 (e), and T_3 (f), where a denotes a special atom type for a pyridine ring, and the red numbers represent the position of atoms in a pyridine ring that its adjacent nodes bond with.

From the examples of position-assigned molecular trees provided in Figure 3.5, we start by finding an adjacent node list of v_1 in T_1. Because $T_1^{sub}(v_2, v_1) = T_1^{sub}(v_3, v_1) \neq T_1^{sub}(v_4, v_1) \neq T_1^{sub}(v_5, v_1)$ and $T_1^{sub}(v_4, v_1) \neq T_1^{sub}(v_5, v_1)$, v_2 and v_1 are clustered together, whereas v_4 and v_5 are not. Accordingly, $A^{T_1}(v_1)$ is $((v_2, v_3), (v_4), (v_5))$. T_2 is isomorphic to T_1; therefore, $A^{T_2}(v_1)$ is the same as $A^{T_1}(v_1)$, which is $((v_2, v_3), (v_4), (v_5))$. However, $T_3^{sub}(v_1, v_2) \neq T_3^{sub}(v_4, v_2) \neq T_3^{sub}(v_5, v_2)$, so $A^{T_3}(v_1) = ((v_1), (v_4), (v_5))$.

Definition 8. *A position list, $P^T(v)$, of a structure node v in a position-assigned molecular tree T is defined as a list of lists with the same dimension as $A^T(v)$, such that $P^T(v)[i][j]$ is a position of the atom in v that node $A^T(v)[i][j]$ bonds with and $P^T(v)[i][j] < P^T(v)[i][j+1]$ holds for all i and j.*

It is worth noting that changing the order of the same set of positions in a position list of node v does not affect the overall structure of v because nodes in the same set correspond to the same substructure. As a result, the molecular graphs obtained from a molecular tree with different permutations of the same list of positions are isomorphic to each other. For example, the molecular graph obtained from a molecular tree with a position list $((1, 2), (3))$ is isomorphic to that with a position list $((2, 1), (3))$, which is denoted as $((1, 2), (3)) \cong ((2, 1), (3))$. Accordingly, we define that $P^T(v)[i][j] < P^T(v)[i][j + 1]$ for all i and j to remove the redundancy.

For example, given T_1 in Figure 3.5, because v_2, v_3, v_4, and v_5 bond with atoms at positions 1, 2, 4, and 5 of v_1, respectively (as shown in red numbers), and $A^{T_1}(v_1) = ((v_2, v_3), (v_4), (v_5))$, $P^{T_1}(v_1)$ is equal to $((1, 2), (4), (5))$. Similarly, $P^{T_2}(v_1)$ is $((4, 5), (2), (1))$ and $P^{T_3}(v_1)$ is $((5), (1), (2))$.

3.3.2 An automorphism group

If the specified substructure is symmetric, then different position-assigned molecular trees may represent the same structure, e.g. T_1 and T_2 in Figure 3.5. An automorphism group of the specified substructure is introduced to handle this kind of redundancy.

Given a graph $G(V, E)$, an automorphism of G is an isomorphism between nodes in G ($\theta : V \to V$) that preserves the adjacency [31], which means that

- $l(v) = l(\theta(v))$,
- $\{v_1, v_2\} \in E$ if and only if $\{\theta(v_1), \theta(v_2)\} \in E$, and
- $\forall v_1, v_2 \in V$, if $\{v_1, v_2\} \in E$, then $mul(v_1, v_2) = mul(\theta(v_1), \theta(v_2))$.

Definition 9. *An automorphism group of a structure node s, Aut(s), is defined as a set of all automorphisms of a molecular graph corresponding to s.*

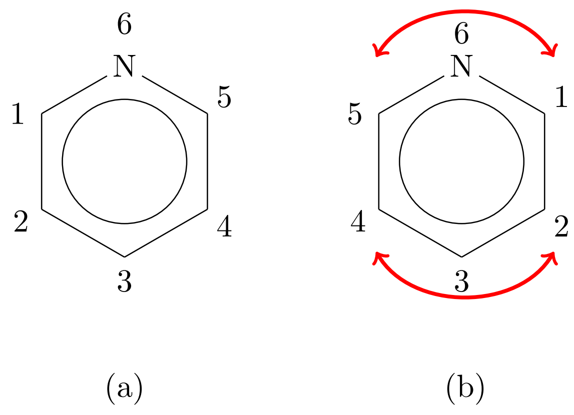


Figure 3.6: A molecular graph of a pyridine ring when nitrogen is at atom position 6 (a) and atom position 2 (c) and the automorphism of the first graph (b), where numbers indicate the index of corresponding nodes and red arrows indicate the mappings between two nodes.

For example, a pyridine ring has one automorphism, which is the following mapping function, where x is the position of an atom in a pyridine ring and y is the position of nitrogen in that pyridine ring. It is noted that the position of atoms in a substructure depends on the MOL file of substructure that users provide so y can be any number from one to six. For example, y is equal to 6 in Figure 3.6 (a) and (b), and is equal to 2 in Figure 3.6 (c). Let node p in Figure 3.6 (c) be the parent node of pyridine ring in a molecular tree. In this case, this pyridine ring contains no automorphism because the bijection of pyridine ring cannot map node p to another node and preserves the adjacency.

$$\theta(x) = \begin{cases} (2 \cdot y - x) \text{ modulo } 6, & \text{if } (2 \cdot y - x) \text{ modulo } 6 \neq 0 \\ 6, & \text{if } (2 \cdot y - x) \text{ modulo } 6 = 0 \end{cases} \tag{3.1}$$

Given a position list p and an automorphism function θ of p, $\theta(p)$ is a position list obtained by applying θ to each element in p before sorting elements in all sets in ascending order so that the condition $P^T(v)[i][j] < P^T(v)[i][j+1]$ holds.

Two different position lists, p_1 and p_2, of a structure node s are defined to be equivalent to each other, $p_1 \cong p_2$, if and only if there exists an automorphism θ in $Aut(s)$ such that $p_1 = \theta(p_2)$. Moreover, $p_1 \cong p_2$ means that p_1 and p_2 represent the same structure and are redundant with each other. For instance, from T_1 and T_2 in Figure 3.5, $P^{T_1}(v_1)$ and $P^{T_2}(v_1)$ are $((1,2),(4),(5))$ and $((4,5),(2),(1))$, respectively. Given a mapping function θ defined in Equation 3.1, we have $\theta(P^{T_2}(v_1)) = \theta(((4,5),(2),(1))) = ((2,1),(4),(5)) \cong ((1,2),(4),(5)) = P^{T_1}(v_1)$, which corresponds to the fact that the structure of a pyridine node in T_1 is isomorphic to that in T_2.

3.3.3 Normal form of a molecular tree

Because a molecular graph can be represented by several molecular trees depending on which node is the root node and how the sibling nodes are ordered, the center-rooted and left-heavy conditions, defined in Section 2.3.2, are used as a canonical way to select the root node and to sort the sibling nodes.

Then, we define that a position list p_1 is greater than another position list p_2 ($p_1 > p_2$) with the same dimension as p_1 if there exist two positive integers i and k such that (1) for all positive integers $j < i$, $p_1[j][l] = p_2[j][l]$ for all l, (2) for all positive integers $l < k$, $p_1[i][l] = p_2[i][l]$, and (3) $p_1[i][k] > p_2[i][k]$. From T_1 and T_2 in Figure 3.5, $P^{T_2}(v_1) = ((4,5),(2),(1)) > P^{T_1}(v_1) = ((1,2),(4),(5))$ because there exist two integers $i = 1$ and $k = 1$ that satisfy these three conditions.

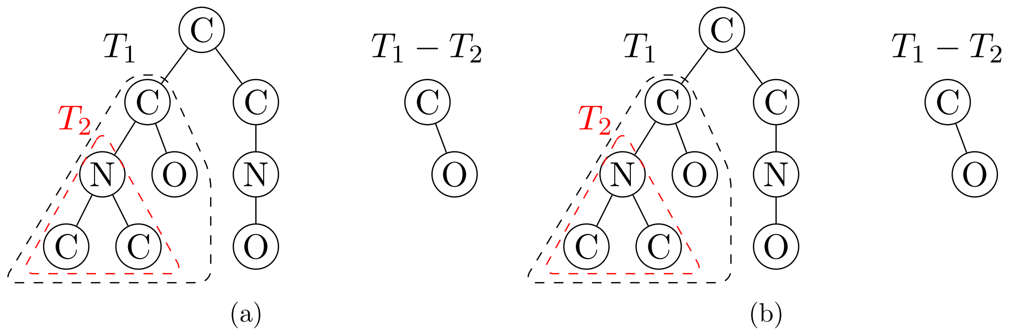


Figure 3.7: Examples of $T_1 - T_2$, where T_1 overlaps T_2 (a) and T_1 does not overlap T_2 (b).

Given a subtree T_1 in a left-heavy molecular tree T and a node v in T_1, $index(v, T_1)$ is defined as the order of v when traversing T_1 in the breadth-first search order. Moreover, given two subtrees $T_1(V_1, E_1)$ and $T_2(V_2, E_2)$ in T, $T_1 - T_2$ is defined as a subtree consisting of a set of nodes V_{1-2} and a set of edges E_{1-2}, where $V_{1-2} = \{v | v \in V_1, v \notin V_2\}$ and $E_{1-2} = \{\{v_a, v_b\} | v_a \in V_{1-2}, v_b \in V_{1-2}$ and $\{v_a, v_b\} \in E_1\}$ (see Figure 3.7).

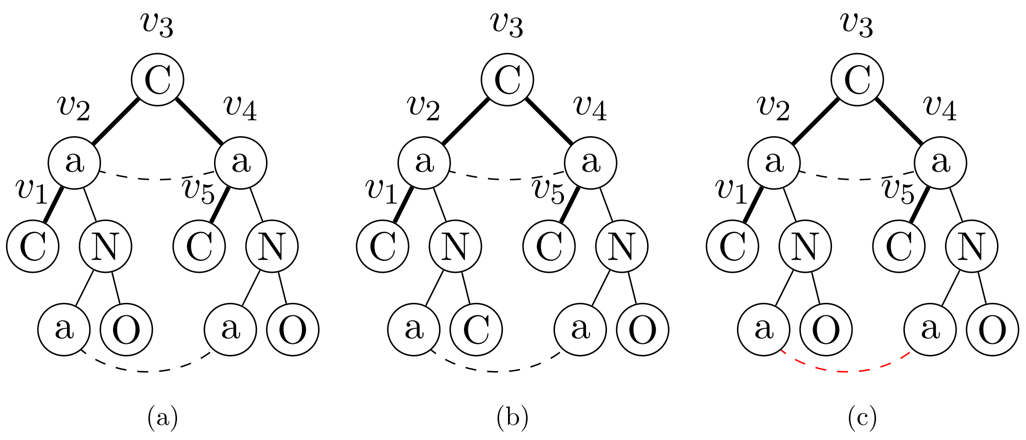


Figure 3.8: An example of a position-assigned molecular tree such that the path between v_1 and v_5 is a symmetric path (a) and the path between v_1 and v_5 is not a symmetric path (b,c), where "a" is a label of the structure node, and the black and red dash curves indicate the equality and inequality of position lists between two structure nodes, respectively.

For a path $q = (v_1, v_2, ..., v_n)$ from v_1 to v_n, we say that v_1 is *left* of v_n if $v_{n-\lfloor \frac{n}{2} \rfloor+1}$ is the root node or $index(v_1) < index(v_n)$. Given two nodes u and w in a position-assigned molecular tree T and two sequences of structure nodes, $(u_1, u_2, ..., u_a)$ and $(w_1, w_2, ..., w_b)$, obtained by traversing $T(u)$ and $T(w)$, respectively, in the breadth-first search order, we define that $T(u) >_p T(w)$ if

1. $T(u) >_m T(w)$, or
2. $T(u) =_m T(w)$ and there exists an integer i $(1 \leq i \leq n)$ such that $\forall j, i < j \leq n$, $P^T(u_j) = P^T(w_j)$ and $P^T(u_i) > P^T(w_i)$.

Then, we define that $T(u) =_p T(w)$ if and only if neither $T(u) >_p T(w)$ nor $T(w) >_p T(u)$.

Definition 10. *A path $q = (v_1, v_2, ..., v_n)$ in a position-assigned molecular tree T is a symmetric path if q holds all of the following three conditions:*

1. *if the length of q is odd and $v_{\frac{n+1}{2}}$ is a structure node, then there exists a non-identity mapping function $\theta \in Aut(l(v_{\frac{n+1}{2}}))$ such that $\theta(P^T(v_{\frac{n+1}{2}})) = P^T(v_{\frac{n+1}{2}})$,*
2. *$T^{sub}(v_i, v_{i+1}) =_m T^{sub}(v_{n-i+1}, v_{n-i})$ for all $i = 1$ to $\lfloor \frac{n}{2} \rfloor$, and*
3. *$T^{sub}(v_i, v_{i+1}) - T^{sub}(v_1, v_2) =_p T^{sub}(v_{n-i+1}, v_{n-i}) - T^{sub}(v_n, v_{n-1})$ for all $i = 2$ to $\lfloor \frac{n}{2} \rfloor$ (see Figure 3.8).*

Definition 11. *A position-assigned molecular tree T is a normal tree if it holds all of the following conditions:*

1. *T is a left-heavy tree.*

2. *T is a center-rooted tree. In the case that there are two center nodes, u and v, if $T^{sub}(u,v) \geq_p T^{sub}(v,u)$, v is the root node.*

3. *For each structure node v representing a chemical structure s in T, there exists no mapping function θ in $Aut(s)$ such that $P^T(v) > \theta(P^T(v))$.*

4. *For each symmetric path $P = (v_1, v_2, ..., v_n)$ between two structure nodes v_1 and v_n in T such that v_1 is left of v_n, $T^{sub}(v_1, v_2) \geq_p T^{sub}(v_n, v_{n-1})$.*

3.4 Proof

Lemma 4. *Given a molecular graph G, G can be represented by a normal tree.*

Proof. By contracting all cyclic structures in G into structure nodes, selecting the root node according to the center-rooted condition and sorting sibling nodes according to the left-heavy condition, G can be represented by a molecular tree satisfying the first and second conditions of the normal form. Then, the information about which atoms in those cyclic structures bond with which atoms is stored in position lists of structure nodes. For each structure node, we can find a position list that satisfies the third condition because there must exist the least position list among position lists obtained from all mapping functions in one automorphism group. Moreover, when a path between any two nodes v_1 and v_2 in a molecular tree T_1 is a symmetric path, swapping v_1 and v_2 does not structurally change the molecular tree. Consequently, the fourth condition eliminates only a molecular tree with $T^{sub}(v_1, v_2) <_p T^{sub}(v_n, v_{n-1})$ for a symmetric path $(v_1, ..., v_n)$ that represents the same molecular graph as a molecular tree with $T^{sub}(v_1, v_2) >_p T^{sub}(v_n, v_{n-1})$. Therefore, the fourth condition does not decrease the number of molecular graphs that normal trees can represent. In conclusion, any molecular graphs can be represented by normal trees. □

Lemma 5. *Given two different molecular graphs G_1 and G_2, they cannot be represented by the same normal tree.*

Proof. Because G_1 and G_2 have different structures, they must be represented by either different left-heavy center-rooted trees or the same left-heavy center-rooted tree containing at least one structure node with different position lists. Thus, we can uniquely generate two normal trees for G_1 and G_2. □

Proposition 7. *Given a normal tree T with a path $P = (v_1, ..., v_n)$, where v_1 is left of v_n, let $G'(V, E)$ be a molecular graph obtained from a molecular tree T', which is obtained by removing $T(v_1)$ and $T(v_n)$ except v_1 and v_n from T. If there is a non-identity mapping function $\phi : V \to V$ in G' such that $\phi(v_1) = v_n$ and $T(v_1) =_m T(v_n)$, then P is a symmetric path.*

Proof. First, we consider the case in which v_1 and v_n are connected to each other. Then, G' contains only P, which is a path consisting of two nodes (v_1, v_n), therefore there is a mapping function ϕ such that $\phi(v_1) = v_n$ in G'. Moreover, if $T(v_1) =_m T(v_n)$, then P is a symmetric path because $T^{sub}(v_1, v_n) =_m T^{sub}(v_n, v_1)$.

For the case in which v_1 and v_n are not connected to each other, v_{i+1} and v_{n-i} are the parent nodes of v_i and v_{n-i+1}, respectively, for $i = 1$ to $\lceil \frac{n}{2} \rceil - 1$ unless T violates the second condition of the normal form. Because ϕ maps v_1 to v_n, it also maps $T^{sub}(v_{a_1}, v_1)$ to $T^{sub}(v_{a_n}, v_n)$ for all pairs of nodes v_{a_1} and v_{a_n} adjacent to v_1 and v_n, respectively. Moreover, ϕ must map the parent node of v_i, which is v_{i+1}, to the parent node of v_{n-i+1}, which is v_{n-i}, for $i = 1$ to $\lceil \frac{n}{2} \rceil - 1$ because for all nodes v in a center-rooted tree, the height of $T^{sub}(v_p, v)$, where v_p is a parent node of v, is greater than the height of $T^{sub}(v_c, v)$ for any nodes v_c that are child nodes of v. Therefore, $T^{sub}(v_i, v_{i+1}) = T^{sub}(v_{n-i+1}, v_{n-i})$ holds for $i = 1$ to $\lceil \frac{n}{2} \rceil - 1$.

Moreover, in order for ϕ to be able to map atoms in a specified substructure to those in another specified substructure after converting T' into G', for all pairs of structure nodes v_{s_1} and v_{s_2} in T' such that $\phi(v_{s_1}) = v_{s_2}$, $P^T(v_{s_1}) = P^T(v_{s_2})$ must hold, which means that $T^{sub}(v_i, v_{i+1}) =_p T^{sub}(v_{n-i+1}, v_{n-i})$.

In conclusion, if there is a non-identity mapping function that maps v_1 to v_n in G' and $T(v_1) =_m T(v_n)$, then the path between v_1 and v_n satisfies all conditions of a symmetric path.

□

Lemma 6. *Given two different normal trees T_1 and T_2, T_1 does not represent the same molecular graph as T_2.*

Proof. Let T_1 and T_2 be two different normal trees representing two molecular graphs, G_1 and G_2, respectively. If T_1 and T_2 are different left-heavy center-rooted trees, then there cannot be an isomorphism θ such that $\theta(T_1) = T_2$, which implies that there exists no isomorphism θ that maps G_1 to G_2. Accordingly, T_1 and T_2 do not represent the same molecular graph in this case. If T_1 and T_2 are the same left-heavy center-rooted tree, there exists a mapping function θ that maps nodes in T_1 to nodes in T_2 and preserves the adjacency of nodes. Nevertheless, T_1 and T_2 are different normal trees, so there is at least one structure node v_1 in T_1 such that

$$P^{T_1}(v_1) \neq P^{T_2}(\theta(v_1)). \tag{3.2}$$

There are two possible cases, which are the case in which v_1 and $\theta(v_1)$ are the same node $(\theta(v_1) = v_1)$ and the case in which v_1 and $\theta(v_1)$ are two distinct nodes $(\theta(v_1) = v_2)$. Consider the case in which $\theta(v_1) = v_1$, if both T_1 and T_2 correspond to the same molecular graph, there exist two non-identity mapping functions σ_1 and σ_2 in $Aut(l(v_1))$ such that

$$P^{T_1}(v_1) = \sigma_1(P^{T_2}(v_1)) \text{ and } P^{T_2}(v_1) = \sigma_2(P^{T_1}(v_1)). \tag{3.3}$$

From Equations 3.2 and 3.3, either $P^{T_1}(v_1) > P^{T_2}(v_1) = \sigma_2(P^{T_1}(v_1))$ or $P^{T_2}(v_1) > P^{T_1}(v_1) = \sigma_1(P^{T_2}(v_1))$ must hold, which violates the third condition of the normal form. Accordingly, T_1 and T_2 cannot represent the same molecular graph in this case.

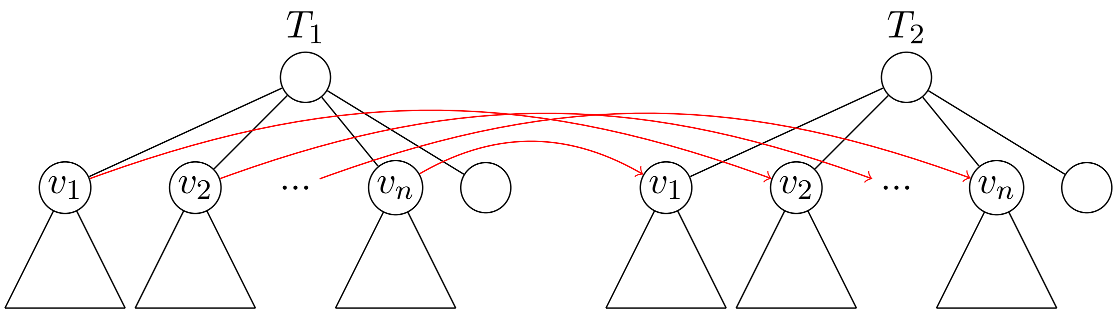


Figure 3.9: Two different normal trees T_1 and T_2, where the red dashed lines denote a mapping function θ that maps nodes in T_1 to nodes in T_2.

Next, we consider the case in which $\theta(v_1) = v_2$, where v_1 and v_2 are two distinct nodes. We assume that $\theta(v_i) = v_{i+1}$ $(1 \leq i \leq n-1)$ and $\theta(v_n) = v_1$, as shown in Figure 3.9. According to Proposition 7, because $\theta(v_i) = v_{i+1}$ and $\theta(v_n) = v_1$, the paths between v_i and v_{i+1} $(1 \leq i \leq n-1)$, as well as that between v_1 and v_n, are symmetric paths. Consequently, the following equation holds:

$$T_1(v_1) =_m T_1(v_2) =_m ... =_m T_1(v_n) =_m T_2(v_1) =_m ... =_m T_2(v_n) \tag{3.4}$$

Then, we assume that v_i is left of v_{i+1} in T_1 for all i. This also holds in T_2 because T_1 and T_2 are the same left-heavy center-rooted tree. From the fourth condition of the normal form,

$$T_1(v_i) \geq_p T_1(v_{i+1}) \text{ and } T_2(v_i) \geq_p T_2(v_{i+1}) \ (1 \leq i \leq n-1). \tag{3.5}$$

From Equations 3.4 and 3.5, it can be concluded that

$$P^{T_1}(v_1) \geq P^{T_1}(v_{i+1}) \text{ and } P^{T_2}(v_i) \geq P^{T_2}(v_{i+1}) \ (1 \leq i \leq n-1). \tag{3.6}$$

In order for T_1 and T_2 to represent the same molecular graph,

$$P^{T_1}(v_i) = P^{T_2}(\theta(v_i)) = P^{T_2}(v_{i+1}) \ (1 \leq i \leq n-1) \text{ and } P^{T_1}(v_n) = P^{T_2}(\theta(v_n)) = P^{T_2}(v_1) \tag{3.7}$$

must hold.

To satisfy Equations 3.6 and 3.7,

$$P^{T_1}(v_1) = P^{T_1}(v_2) = ... = P^{T_1}(v_n) = P^{T_2}(v_1) = ... = P^{T_2}(v_n) \tag{3.8}$$

must hold, which contradicts Equation 3.2.

Therefore, T_1 cannot represent the same molecular graph as T_2 unless $T_1 = T_2$. □

3.5 Methods

3.5.1 Finding an automorphism group

An automorphism group of the specified substructure is determined by Algorithm 4 in order to be able to check for the third and fourth properties of a normal tree. Given a molecular graph of the substructure $G(V, E)$, where V and E are a set of atoms and a set of bonds, respectively, we generate an adjacency matrix *adj* of the atoms in the substructure (line 2), where $adj[i][j]$ is the multiplicity of a bond between atom i and atom j and $adj[i][j] = 0$ if there is no bond between atom i and atom j. Then, we initialize an empty array denoting a permutation of atoms in the substructure called *permu_array*. *permu_array* corresponds to a mapping function in an automorphism group, where the i^{th} element in the group is the index of a node that is mapped to the i^{th} node in V. For example, a permutation array $\{5, 4, 3, 2, 1, 6\}$ is a mapping function that maps atom 1 to 5, atom 2 to 4, atom 3 to 3, atom 4 to 2, atom 5 to 1, and atom 6 to 6 because the first element is 5, the second is 4, the third is 3, the fourth is 2, the fifth is 1, and the sixth is 6.

After an empty *permu_array* is initialized, we find an atom with the same label as the first element in V and store its index in *permu_array* (line 6). After that, we search for a node that has the same label as the second element in V (line 12) and bonds with other nodes in *permu_array* in the same way that the second element in V bonds with other nodes in V (line 15). If such nodes exist, we copy that many *permu_array* and add their index as the second element of each copied *permu_array* (line 18) and continue to the next element in V until the size of *permu_array* becomes the same as the size of V.

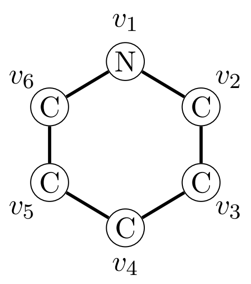


Figure 3.10: Example of a substructure to be calculated an automorphism group

Algorithm 4 Algorithm for calculating an automorphism group of a substructure

```
function FIND_AUTOMORPHISM($G(V, E)$)
    $adj$ := adjacency matrix of $G(V, E)$
    $permu\_array := \emptyset$
    for $i$ from 1 to $|V|$ do
        if $l(V[i]) = l(V[1])$ then
            $permu\_array.push(i)$
            FILL_ARRAY($permu\_array$, $adj$, 2, $V$)
            $permu\_array.pop()$
        end if
    end for
    return
end function
function FILL_ARRAY($permu\_array, adj, index, V$)
    $Result := \emptyset$
    for $i$ from 1 to $|V|$ do
        if $l(V[i]) = l(V[index])$ then
            $valid := true$
            for $j$ from 1 to $index - 1$ do
                if $adj[index][j] \neq adj[i][permu\_array[j]]$ then
                    $valid := false$
                end if
            end for
            if $valid$ then
                $permu\_array.push(i)$
                if $index = n$ then
                    $Result.push(permu\_array)$
                else
                    FILL_ARRAY($permu\_array$, $adj$, $index + 1$, $V$)
                end if
                $permu\_array.pop()$
            end if
        end if
    end for
    return $Result$
end function
```

For example, the process of calculating an automorphism group of a substructure in Figure 3.10, where $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, is as follows. Starting from an empty array $\{\ \}$, we find nodes with the same label as v_1, generate that many arrays, and add one of them to each array. Because only one node, v_1, in a substructure has a label N, we have one array $\{v_1\}$ as the output of this step. Then, we find nodes, which have the same label as the next node, v_2, and are bonded with the first element in the same way that v_1 bonded with v_2. There are five nodes with the same label as v_2, which are v_2, v_3, v_4, v_5, and v_6. However, v_2 has a single bond with v_1 and there are only two nodes out of five (v_2 and v_6) that have a single bond with the first element in the current array so we duplicate that one array and add v_2 and v_6 as the second element of each array. As a result, we have two permutation arrays, $\{v_1, v_2\}$ and $\{v_1, v_6\}$. Because the next node is v_3, we find nodes, which have the same label as v_3 and are bonded with all previous elements in the same way as v_3 bonded with v_1 and v_2. Nodes satisfying these conditions for $\{v_1, v_2\}$ is v_3 and those for $\{v_1, v_6\}$ is v_5 so we add v_3 and v_5 to their corresponding arrays, which results in two arrays, $\{v_1, v_2, v_3\}$ and $\{v_1, v_6, v_5\}$. We continuously move to the next node in V until we reach the end of V. The final automorphism group for this structure contains two permutation arrays, which are $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ (an identity mapping) and $\{v_1, v_6, v_5, v_4, v_3, v_2\}$.

3.5.2 Calculating the number of specified substructures

In this part, we calculate all possible combinations of the number of specified substructures that the given chemical formula can have to determine the number of structure nodes in the enumerated molecular trees. To calculate the number of specified substructures, the degree of unsaturation and the number of atoms of each atom type are used. The degree of unsaturation (DoU) is the summation of the number of double bonds, twice the number of triple bonds, and the number of rings in a structure, which can be calculated from the chemical formula by the following equation:

$$DoU = 1 + \frac{\sum_{l_i} n_{l_i}(v_{l_i} - 2)}{2}, \tag{3.9}$$

where n_{l_i} and v_{l_i} are the number and valence of chemical element l_i in a chemical formula, respectively. We look at the number of chemical elements and DoU of the given chemical formula. If they are greater than or equal to those of the substructure, one substructure is added to the chemical formula. Then, we subtract the number of atoms and DoU of the substructure from that chemical formula and the calculated DoU, respectively. Afterwards, we continue verifying the adequacy of the new chemical formula for one more substructure until either the number of atoms is less than that of the substructure or the DoU is less than that of the substructure.

For example, given a chemical formula $C_{14}N_2H_{14}$ and a benzene ring (C_6H_6, DoU = 4) as a specified substructure, first, an output set is initialized by the input chemical formula (output = $\{C_{14}N_2H_{14}\}$) and DoU is calculated from Equation 3.9 as shown in Equation 3.10.

$$\begin{aligned}\text{DoU} &= 1 + \frac{n_C(v_C - 2) + n_N(v_N - 2) + n_H(v_H - 2)}{2} \\ &= 1 + \frac{14(4-2) + 2(3-2) + 14(1-2)}{2} \\ &= 1 + \frac{28 + 2 - 14}{2} \\ &= 9 \end{aligned} \tag{3.10}$$

Then, we check whether a benzene ring can be added or not. Since both the number of carbon atoms and the number of hydrogen atoms in the chemical formula are greater than six and DoU is greater than four, we add one benzene ring, subtract six carbon atoms and six hydrogen atoms from the chemical formula and subtract four from DoU, which results in a new chemical formula $C_8N_2H_8b_1$ with DoU five. The new chemical formula is added to the output set (output = $\{C_{14}N_2H_{14},\ C_8N_2H_8b_1\}$). After that, the new chemical formula is checked to see if it can contain a benzene ring or not. Because the number of carbon atoms, the number of hydrogen atoms, and DoU of the new chemical formula are greater than those of one benzene ring, another benzene ring is added to the chemical formula. At the same time, six carbon atoms and six hydrogen atoms are subtracted from the chemical formula and DoU is decreased by four. Then, the retrieved chemical formula is included in the output set (output = $\{\{C_{14}N_2H_{14},\ C_8N_2H_8b_1,\ C_2N_2H_2b_2\}\}$). This process is repeated until the number of carbon atoms or hydrogen atoms becomes less than six or DoU becomes less than four.

3.5.3 Enumerating molecular trees

Molecular trees are enumerated from the input chemical formula and the previously calculated number of specified substructures in this step. First, we generate one special chemical element for one specified substructure. After that, we enumerate left-heavy center-rooted trees under the condition that a structure node cannot have multiple bonds with other nodes.

The enumeration of molecular trees starts from an empty tree. Nodes labeled with available chemical elements are added to the tree one by one until there is no available chemical element left. We define that a chemical element is available if the number of nodes labeled with that chemical element in the molecular tree is less than the number of that chemical element in the chemical formula. To prevent redundancy, a node is added to the molecular tree T as a child node of the last internal node u in T by breadth-first search traversal or any leaf node v such that $index(v) > index(u)$ (see Figure 3.11).

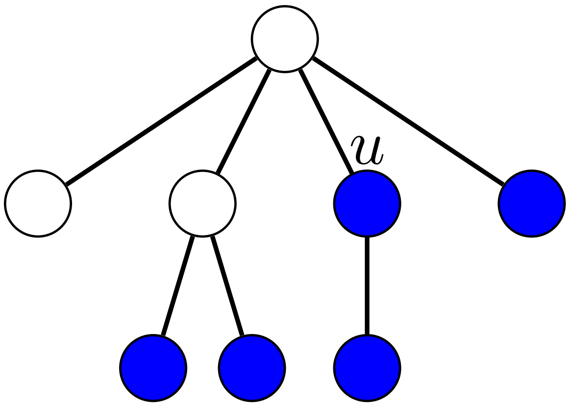


Figure 3.11: Example of a molecular tree, where the blue nodes are nodes that can be added a child node in this work and node u is the last internal node in breadth-first search order

After adding the node, the left-heavy condition and the center-rooted condition are verified and the tree is discarded if it fails either of two conditions according to method provided in [95]. If no available chemical element remains and the tree has not been discarded, then it is guaranteed that this tree satisfies the first two conditions of the normal form.

3.5.4 Assigning position list to structure nodes

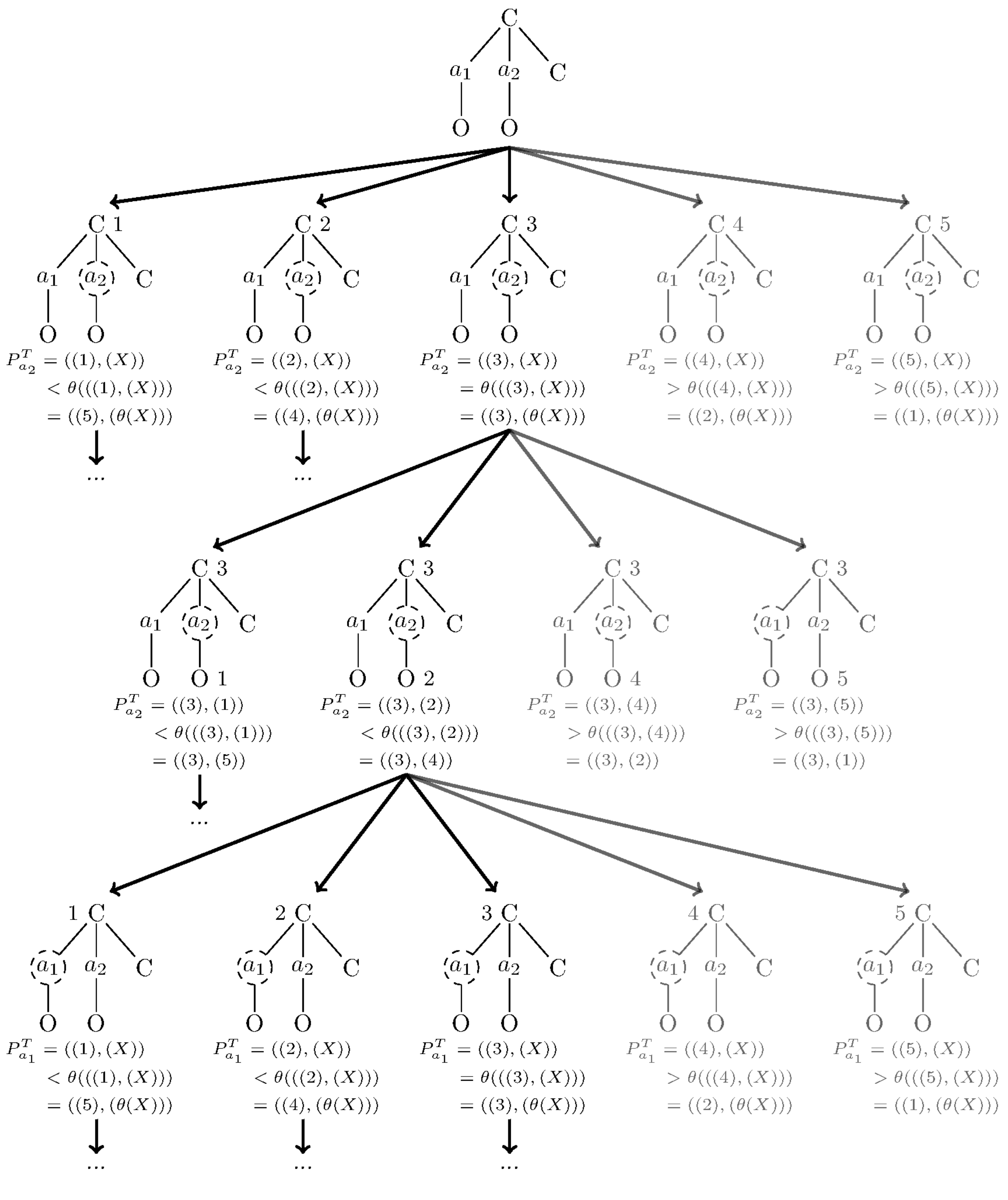
For each left-heavy center-rooted molecular tree T generated in the previous step, we traverse along T from the rightmost deepest node to the root node in the reverse order of the breadth-first search and assign position lists to all structure nodes such that the third and fourth conditions of the normal form are satisfied. The pseudocode of position list assignment is given in Algorithm 5. The third condition is verified immediately after a new element is added to the position list (line 25) to reduce the search space for the valid position lists. However, the fourth condition is determined when position lists of all structure nodes are known (line 7) because position lists of all structure nodes are necessary to determine the symmetry of a path.

The output of this algorithm is a collection of position-assigned molecular trees, which can be converted to molecular graphs by removing all bonds between structure nodes and their adjacent nodes, converting all structure nodes into corresponding cyclic structures and connecting atoms in cyclic structures to their adjacent nodes according to the position lists.

Details of how position lists are assigned to structure nodes are illustrated in Figure 3.12. The input of this process is the uppermost left-heavy center-rooted molecular tree, where both a_1 and a_2 are structure nodes representing pyridine rings whose automorphism group contains one mapping function θ defined in Equation 3.1. We traverse the tree in reverse order of the breadth-first search and assign position lists to the structure nodes

Algorithm 5 Algorithm for assigning position list to structure nodes in a molecular tree

```
 1: function ASSIGN_TREE($T$)
 2:     $v$ := the last structure node of $T$ in BFS order
 3:     ASSIGN_NODE($T$,$v$)
 4:     return
 5: end function
 6: function ASSIGN_NODE($T$, $v$)
 7:     if $v$ is null then
 8:         $P := \{(v_1, v_2, ..., v_n)|(v_1, ..., v_n)$ is a symmetric path and $v_1$ is left of $v_n\}$
 9:         if $\forall(v_1, ..., v_n) \in P$, $T^{sub}(v_1, v_2) >_p T^{sub}(v_n, v_{n-1})$ then
10:             output $T$
11:         end if
12:         return
13:     end if
14:     $A$ := an adjacent nodes list of $v$ from Definition 7
15:     ASSIGN_POSITION($T$, $v$, $A[1][1]$, $A$)
16:     return
17: end function
18: function ASSIGN_POSITION($T$, $v$, $w$, $A$)
19:     if $w$ is null then
20:         if the structure node next to $v$ in reverse BFS order exists then
21:             $v'$ := the structure node before $v$ in BFS order
22:         else
23:             $v'$ := null
24:         end if
25:         ASSIGN_NODE($T$, $v'$)
26:         return
27:     end if
28:     if the node next to $w$ in $A$ exists then
29:         $w'$ := the node next to $w$ in $A$
30:     else
31:         $w'$ := null
32:     end if
33:     for $i$ from 1 to the number of atoms in substructure $l(v)$ do
34:         Let $j$ and $k$ be two integers such that $w$ corresponds to the node $A[j][k]$
35:         for each atom $p$ in substructure $l(v)$ do
36:             if $p$ is unsaturated and $p > P_v^T[j][k']$ for all $k' < k$ then
37:                 $P_v^T[j][k] := p$
38:                 if $\theta \in Aut(l(v))$ such that $P_v^T > \theta(P_v^T)$ exists then
39:                     return
40:                 end if
41:                 ASSIGN_POSITION($T$, $v$, $w'$)
42:             end if
43:         end for
44:     end for
45:     return
46: end function
```



$P^T_{a_2} = ((1),(X)) < \theta(((1),(X))) = ((5),(\theta(X)))$
$P^T_{a_2} = ((2),(X)) < \theta(((2),(X))) = ((4),(\theta(X)))$
$P^T_{a_2} = ((3),(X)) = \theta(((3),(X))) = ((3),(\theta(X)))$
$P^T_{a_2} = ((4),(X)) > \theta(((4),(X))) = ((2),(\theta(X)))$
$P^T_{a_2} = ((5),(X)) > \theta(((5),(X))) = ((1),(\theta(X)))$
$P^T_{a_2} = ((3),(1)) < \theta(((3),(1))) = ((3),(5))$
$P^T_{a_2} = ((3),(2)) < \theta(((3),(2))) = ((3),(4))$
$P^T_{a_2} = ((3),(4)) > \theta(((3),(4))) = ((3),(2))$
$P^T_{a_2} = ((3),(5)) > \theta(((3),(5))) = ((3),(1))$
$P^T_{a_1} = ((1),(X)) < \theta(((1),(X))) = ((5),(\theta(X)))$
$P^T_{a_1} = ((2),(X)) < \theta(((2),(X))) = ((4),(\theta(X)))$
$P^T_{a_1} = ((3),(X)) = \theta(((3),(X))) = ((3),(\theta(X)))$
$P^T_{a_1} = ((4),(X)) > \theta(((4),(X))) = ((2),(\theta(X)))$
$P^T_{a_1} = ((5),(X)) > \theta(((5),(X))) = ((1),(\theta(X)))$

Figure 3.12: Process of assigning position lists to structure nodes, where a_1 and a_2 are structure nodes representing a pyridine ring whose automorphism group consists of one mapping function θ defined in Equation 3.1. Molecular graphs with low opacity contain invalid position lists.

found in this order. The structure node being assigned the position list is indicated by a dashed circle. The number next to a node is a position of an atom in the structure node with a dashed circle that the corresponding node bonds with. The current position list of a structure node with a dashed circle is written below the molecular tree, where X represents an unknown position for adjacent nodes that are not assigned a position yet. The current position list is compared with another position list obtained from θ. If the current position list is less than or equal to the one obtained from θ, it does not violate the third property of the normal form and we assign the position to the next adjacent node. Otherwise, the current position list violates the third property and is discarded, as indicated by the low opacity of the molecular tree.

Because the third part of the algorithm (Section 3.5.3) generates all left-heavy center-rooted molecular trees from a chemical formula, input trees of this part are guaranteed to satisfy the first two conditions of the normal form. Moreover, all possible position lists are generated (line 23) before checking for the third (line 25) and fourth conditions (line 7), so BfsStructEnum enumerates all distinct normal trees.

Theorem 2. *Algorithm 5 enumerates all non-redundant molecular graphs that are the solution of Problem 2.*

3.5.5 Conversion of molecular trees to molecular graphs

After all structure nodes in a molecular tree is assigned a position list, we can convert that molecular tree to the corresponding molecular graphs, which represent a chemical compound, by three main steps. The first step is removing the edges connecting a structure node with its adjacent node (see Figure 3.13 B). Second, we replace a structure node with a molecular graph of its corresponding substructure, where nodes of the extracted molecular graph are labeled according to the order of atoms in the input Mol file (see Figure 3.13 C). Finally, we connect the adjacent nodes with nodes of extracted molecular graph according to the position in the position list. In this step, the adjacent node, whose corresponding position is i, is bonded with nodes, whose label is i (see Figure 3.13 D).

3.5.6 Complexity analysis

First, we analyze the time complexity of Algorithm 4, which calculates an automorphism group of one input substructure. Let that substructure contain n_s atoms and G be the graph representing the substructure. FIND_AUTOMORPHISM function contains a loop from one to n (line 4). Each loop checks the condition of the first node in G (line 5) and calls FILL_ARRAY function at most one time to check for the second node (line 7) if the condition is satisfied. Accordingly, the time complexity of FIND_AUTOMORPHISM, $T_1(n_s)$, is

$$T_1(n_s) \leq O(n_s) + n_s \cdot T_2(n_s - 1), \tag{3.11}$$

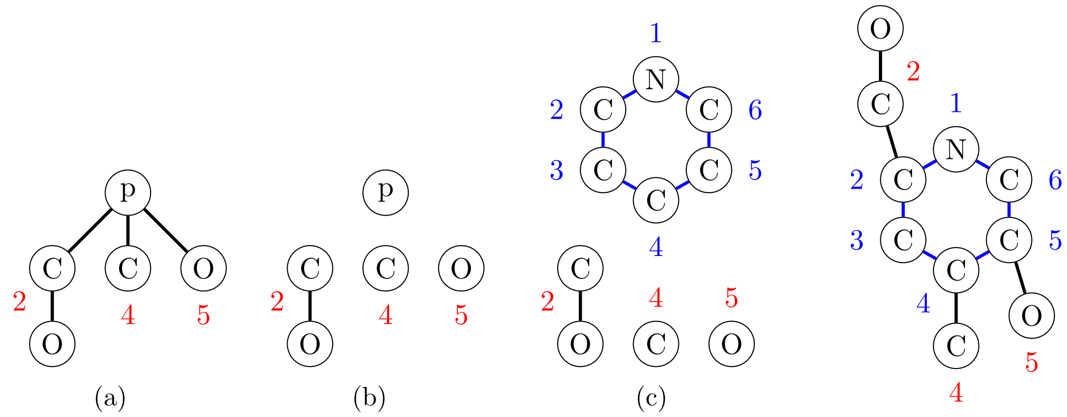


Figure 3.13: Process of conversion from a molecular tree to a molecular graph, where p is a label of a pyridine substructure, blue edges are aromatic bonds, blue numbers are the label of atoms in the substructure according to the input Mol file, and red numbers are the corresponding position in the position list

where $T_2(n_s - 1)$ is the time complexity of FILL_ARRAY function starting from the second node in G. In FILL_ARRAY function, there is a loop with n_s iterations (line 15). Each loop performs at most one loop, which contains a constant number of operations with n_s iterations (line 18), as well as a recursive call of itself to move to the next node (line 28) if the current node holds some conditions (line 16, line 23, and line 27). The time complexity of this function is

$$\begin{aligned} T_2(n_s - 1) &\leq n_s \cdot [(n_s \cdot O(1)) + T_2(n_s - 2)] && (3.12) \\ &\leq \left[\sum_{i=2}^{k+1} n_s^i\right] + n_s^k \cdot T_2(n_s - (k+1)) && (3.13) \\ &\leq \left[n_s^{k+2} - n - 1\right] + n_s^k \cdot T_2(n_s - k - 1), && (3.14) \end{aligned}$$

where k is the number of iterations of FILL_ARRAY function. Because we traverse from the second node to the last node in G by moving to the next node via the FILL_ARRAY function, the FILL_ARRAY function is called $n_s - 1$ times, or $k = n_s - 1$. Moreover, the last iteration performs at most two loops (line 15 and line 18). Both loops iterate n_s times and one loop is inside another loop so the time complexity of the last iteration is $O(n_s^2)$. Then, the time complexity of Algorithm 4 can be written as

$$\begin{aligned}
T_2(n_s-1) &\leq (n_s^{n_s+1}-n_s-1)+n_s^{n_s-1}\cdot T_2(n_s-(n_s-1)-1) && (3.15)\\
&\leq (n_s^{n_s+1}-n_s-1)+n_s^{n_s-1}\cdot T_2(0) && (3.16)\\
&\leq (n_s^{n_s+1}-n_s-1)+n_s^{n_s-1}\cdot O(n_s^2) && (3.17)\\
&\leq O(n_s^{n_s+1})+O(n_s^{n_s+1} && (3.18)\\
&\leq O(2\cdot n_s^{n_s+1}). && (3.19)
\end{aligned}$$

Then, we substitute Equation 3.19 into Equation 3.11 and have

$$T_1(n_s) \leq n_s \cdot O(2\cdot n_s^{n_s+1}) = O(2\cdot n_s^{n_s+2}). \quad (3.20)$$

It must be noted that n_s is the number of atoms in the input substructure, which is expected to be significantly less than the number of input atoms.

Next, we analyze the time complexity of Algorithm 5. This algorithm, first, calls function ASSIGN_TREE. This function initializes a variable v and calls another function ASSIGN_NODE. Therefore, the time complexity of ASSIGN_TREE is the same as that of ASSIGN_NODE for the same tree.

Let n be the number of nodes in a tree being assigned an atom position list. If we reach is the root node of the tree, v will be null and we examine at most $\binom{n}{2} = O(n^2)$ paths to obtain the set P in line 8. For each path, we check the label and the atom position list of each node, which can be done in a constant time so the time complexity of line 8 is $O(n^2)$. Therefore, the time complexity of the last iteration of ASSIGN_NODE, where we reach the root node, is $O(n^2)$ and the time complexity of other iterations is a constant of n or $O(1)$.

The ASSIGN_NODE recursively calls itself via a function ASSIGN_POSITION (line 25). In ASSIGN_POSITION function, it calls itself at most $n_s^{n_s}$ times, where n_s is the number of atoms in the substructure representing by the current node, because there are at most $n_s^{n_s}$ possible atom position lists for the substructure. When ASSIGN_NODE is called, a constant number of operations, except ASSIGN_POSITION, are performed and the number of atoms in the tree decreases by n_s instead of 1. Thus, the time complexity of ASSIGN_NODE function is

$$\begin{aligned}
T_3(n) &\leq O(1)+n_s^{n_s}\cdot T_3(n-n_s) && (3.21)\\
&\leq O(1)+({n_s}^{n_s})^k\cdot T_3(n-n_s\cdot k), && (3.22)
\end{aligned}$$

where k is the number of iterations. The ASSIGN_NODE calls itself until all structure nodes are assigned an atom position list, which is at most n/n_s times ($k = n/n_s$). Then, we have

$$\begin{aligned} T_3(n) &\leq O(1) + ({n_s}^{n_s})^{n/n_s} \cdot T_3(n - n_s \cdot \frac{n}{n_s}) && (3.23) \\ &\leq O(1) + n_s^n \cdot T_3(0). && (3.24) \end{aligned}$$

Because $T_3(0)$ is the time complexity of the last iteration of ASSIGN_NODE, we substitute $T_3(0) = O(n^2)$ in 3.24 to obtain the time complexity of ASSIGN_NODE.

$$\begin{aligned} T_3(n) &\leq O(1) + n_s^n \cdot O(n^2) && (3.25) \\ &= O(n_s^n \cdot n^2) && (3.26) \end{aligned}$$

For a memory complexity, we use an array of C++ structures to keep the information of a molecular tree, where the number of elements in an array is the same as the number of atoms excluding hydrogen atoms in a molecular tree and keep the information of a molecular tree being enumerated only. Therefore, this work requires a polynomial space complexity of degree one.

3.6 Results

We compared the performance of the proposed algorithm, BfsStructEnum, with another commercial well-known universal structure generator, MOLGEN (version 5.0), by enumerating various chemical formulas with two specified substructures, a benzene ring and a pyridine ring, on a computer with a 3.47 GHz intel Xeon CPU and 23.5 GiB of memory. Since MOLGEN does not limit the structure of the enumerated compounds, we specified the number of substructures in the execution of MOLGEN to obtain only desired results. The results are shown in Table 3.1, where "b" and "p" denote a benzene ring and a pyridine ring, respectively. We did not compare our proposed method to OMG because it has been shown that OMG is slower than MOLGEN [63].

Table 3.1 points out the accuracy of our method by showing that the number of enumerated structures is the same as that of MOLGEN for all input chemical formulas. Then, we compared the computation time to show that our method is much more efficient than MOLGEN. With MOLGEN, some executions extended beyond one week or 604,800 seconds, denoted by "-" in the table.

We also plotted the relationship between the number of enumerated structures and the execution time of the proposed method and MOLGEN in Figure 3.14, where both the X and Y axes are in log-scale. We can see that the computation time of our method was significantly less than that of MOLGEN.

Chemical formula	#nodes						#enumerated structures	Computation time (sec)	
	b	p	C	N	O	H		BfsStructEnum	MOLGEN
$C_7O_2H_8$	1	0	1	0	2	8	19	0.003	0.053
$C_8N_2H_{10}$	1	0	2	2	0	10	130	0.005	0.459
	0	1	3	1	0	10	293	0.018	0.507
$C_9N_1O_1H_9$	1	0	3	1	1	9	884	0.009	2.566
	0	1	4	0	1	9	949	0.006	2.687
$C_{10}O_2H_8$	1	0	4	0	2	8	742	0.008	12.133
$C_{11}N_2H_{10}$	1	1	0	1	0	10	22	0.004	7.375
	1	0	5	2	0	10	9,012	0.044	630.44
	0	2	1	0	0	10	36	0.004	7.315
	0	1	6	1	0	10	14,725	0.021	631.75
$C_{12}N_1O_1H_{11}$	2	0	0	1	1	11	33	0.004	98.99
	1	1	1	0	1	11	205	0.004	98.58
	1	0	6	1	1	11	80,883	0.342	2,611.57
	0	1	7	0	1	11	52,948	0.066	2,559.38
$C_{13}O_2H_{12}$	2	0	1	0	2	12	190	0.005	2,069.3
	1	0	7	0	2	12	162,122	0.686	6,497.55
$C_{14}N_2H_{14}$	2	0	2	2	0	14	1,351	0.014	149,552.32
	1	1	3	1	0	14	5,965	0.021	154,156.83
	1	0	8	2	0	14	1,742,330	7.940	279,698.37
	0	2	4	0	0	14	2,531	0.008	149,630.26
	0	1	9	1	0	14	1,698,545	1.882	276,061.83
$C_{16}N_3H_{13}$	2	0	4	3	0	13	135,450	0.728	-
	1	2	0	1	0	13	511	0.003	84,698.32
	1	1	5	2	0	13	1,753,458	7.107	-
	1	0	10	3	0	13	100,773,971	513.35	-
	0	3	1	0	0	13	538	0.003	84,738.84
	0	2	6	1	0	13	1,583,598	1.871	-
	0	1	11	2	0	13	143,255,509	170.29	-

Table 3.1: The number of enumerated structures and computation time for given chemical formulas by BfsStructEnum and MOLGEN 5.0, where the specified substructures are a benzene ring (b) and a pyridine ring (p), and “-” denotes that the computation time is more than one week.

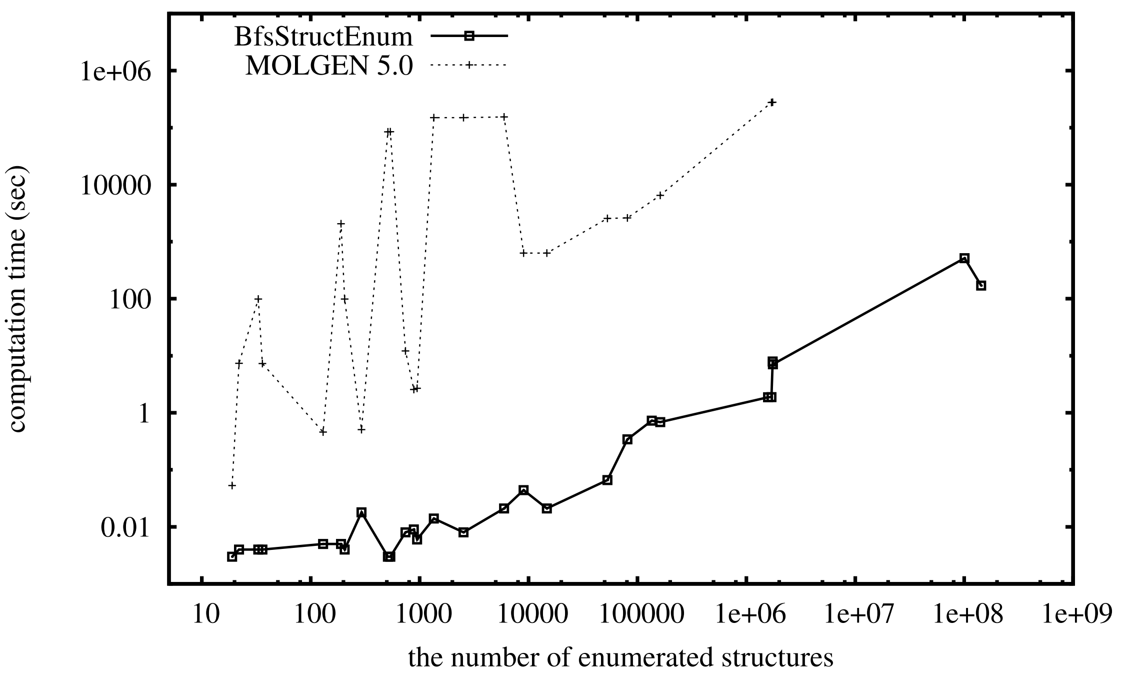


Figure 3.14: Relationship between the number of enumerated structures and computation time of BfsStructEnum (solid line) and MOLGEN (dashed line).

3.7 Discussion

We propose a novel and efficient enumeration algorithm, BfsStructEnum. This algorithm takes a chemical formula and additional biconnected cyclic chemical substructures from users as the input and enumerates all non-redundant molecular graphs without cycles except for the biconnected cyclic substructures specified by users. The substructure option provided in this tool can decrease the time users spend to find the target compounds by using the substructure as another constraint to filter the enumerated compounds. Moreover, this option allows the tool to be able to enumerate tree-like chemical compounds containing any kinds of cyclic substructures given by users.

This work introduced a method to represent a chemical compound containing user-defined cyclic substructures by a tree structure to optimize the computational cost. It compresses one cyclic structure into a single node called a structure node. To expand the tree structure with structure nodes back to a chemical compound, we introduced a new attribute for the structure node called an atom position list, which stores the way that atoms in a substructure node bonding with atoms adjacent to the structure node.

Using the first two conditions of the normal form, this algorithm first enumerates non-redundant molecular trees from a chemical formula, where a specified substructure is compressed into a single node so a node represents either a chemical atom or a substructure and an edge represents a chemical bond. Then, position lists are assigned to all structure nodes in molecular trees in order to convert them to molecular graphs.The third and fourth conditions of the normal form are defined to remove redundancy from the position list. Figure 3.15 illustrates the example of a normal molecular tree, which is the output of this work, where a pyridine ring and a naphthalene ring are the input substructures with label "p" and "n", respectively.

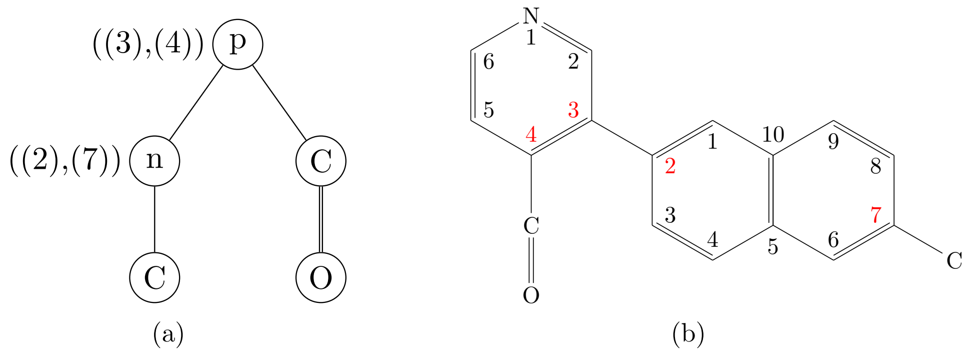


Figure 3.15: Illustration of a center-rooted left-heavy molecular tree (a) representing a chemical compound (b).

In order to evaluate the effectiveness of BfsStructEnum, we enumerated chemical structures from a set of chemical formulas by BfsStructEnum and MOLGEN 5.0, a well-known general purpose structure generator. In addition to producing the same number of enumerated structures, BfsStructEnum could enumerate 17 to approximately 7,000,000 times faster than MOLGEN, although MOLGEN can enumerate all chemical structures without limitation while BfsStructEnum restricts to compounds containing no cycles except for those specified by users. The main reason of this efficiency is that we enumerate molecular trees, which is less complicated than enumerating molecular graphs. Another reason is the decrease of the number of nodes during the enumeration by treating a cyclic structure as a single node.

Drug discovery, synthesis of new chemical compounds, and structure identification problems use an enumeration tool as a part of their processes, making it a fundamental problem in the field of chemoinformatics. Because the enumeration tool usually handles a large number of compounds in practice, the memory usage and computation time are two important factors to be considered. The proposed algorithm aims to optimize memory usage by enumerating one structure at a time without storing the information of previous enumerated structures. The speed of enumeration is maximized by opting to enumerate tree structures, instead of graphs, because enumerating trees is much more efficient than enumerating graphs. These tree structures represent the chemical compounds.

However, since some applications of enumeration tool, e.g. drug discovery, require the enumerated structure to contain desired substructures, it allows users to specify biconnected chemical compounds as substructures of the enumerated structures and uses a single node to represent each substructure. Enumerating in such a way limits the enumerated compounds to be compounds containing no cyclic substructures except for biconnected substructures provided by users only therefore we aims to extend this limitation in the future work. Besides, ranking and filtering the enumerated structures based on their properties, e.g. druglikeness, hydrophobicity, are important options for the enumeration tool. Including these options to the tool is also our future objective.

Chapter 4

Host-pathogen protein interaction prediction based on local topology structures of a protein interaction network

4.1 Background

Infectious diseases cause a significant loss in terms of illness and mortality across the world. Pathogens, the agent of the infectious disease, cause the infection by adhering to the host cell, invading the host cell, disturbing the function of its host, and protecting itself from the host immune system. To adhere the host cell, pathogens uses their specialized organelle called *adhesins* to attach with the receptors on the host cell's membrane. This attachment triggers the internalization mechanism that allows pathogens to enter the host cell. After that, pathogens generate a toxic protein to disrupting and damage the host cell. These processes involve the interaction between their proteins and the host's proteins [27]. By recognizing which pathogen's proteins interact with which host's proteins, we are one step closer to understanding the underlying infectious mechanisms, and providing the opportunity to discover a novel efficient therapeutics.

The simplest way to obtain such information is examining the interaction of all protein pairs. However, the number of proteins in hosts and pathogens is so large that conducting a reliable experiment to test all protein pairs is time and resource consuming and practically infeasible. Due to this limitation, several bioinformatics tools have been introduced to address this problem by generating the host-pathogen protein interaction prediction models from available data. These models are implemented to discover protein pairs that have high probability to interact with each other. Accordingly, we can reduce the number of protein pairs to be experimentally tested and the development of drugs becomes faster and more efficient.

The major challenge of a host-pathogen protein interaction prediction modeling is the scarcity of the verified host-pathogen protein interaction data [61], which is necessary for training the model and benchmarking its prediction performance. Thus, extracting the information from limited available data in the form of suitable features is one important way to overcome this problem. Several prediction methods have been proposed, where their features are based on heterogeneous information, such as the homology of proteins [45], the structure of proteins [20], the biological pathway that the host proteins are involved in [46], the protein domain and motif [22, 24]. Another method uses the information of proteins that probably interact with each other but the experimental evidence is not enough to confirm such interaction [66]. It has also been shown that pathogens tend to target human proteins that interact with many other proteins or human proteins that lie on many shortest paths in the protein-protein interaction network (PIN) [23]. As a result, betweenness centrality and clustering coefficient of the proteins in the PIN are firstly utilized as features in [85] and demonstrate their usefulness.

Later, it was found that the HIV virus does not always target the proteins that interact with many other proteins but tends to target the proteins in the cancer-related pathways [16]. This suggests that the proteins of different pathogens have a high tendency to interact with proteins involved in the same biological pathway and have the similar function. Moreover, it has been shown that the proteins with a similar local topology in the PIN have similar gene ontology and function [19]. Accordingly, we hypothesize that pathogens tend to target the proteins with a similar local topology in the PIN.

The objective of this work is to test this hypothesis by proposing a high-accuracy computational method for solving host-pathogen protein interaction prediction problem using a local topology of a PIN as a feature. The local topology is represented by a graphlet degree vector (GDV) of a human protein in a human PIN, which has never been used in the host-pathogen protein interaction prediction problem before. Then, we compared the performance of the proposed method with the existing method. Our results showed that the GDV significantly improves the prediction accuracy. This suggests that this useful feature contains the essential information for host-pathogen protein interaction prediction.

4.2 Preliminaries

4.2.1 Protein-protein interaction network

In order to perform its function, a protein usually interacts with other proteins and forms a protein complex. A lot of publications dedicated to the protein-protein interaction result in an increasing number of experimentally verified interactions. This interaction data is usually represented as the protein-protein interaction network (PIN), which is a graph representing interacting proteins, where a node denotes a protein and an edge connecting two interacting proteins. By using the PIN, it is more convenient and easier to analyze the protein-protein interactions because proteins forming a protein complex are adjacent to each other in the PIN and graph-related algorithm can be applied to the PIN. PIN is a scale-free network and the distribution of its edges also follows the power law. In other words, PIN contains hub nodes, which are more highly connected to other nodes than average [18].

4.2.2 Graphlet degree vector

A graphlet degree vector (GDV) of a node v, denoted as GDV(v), is a vector of the number of orbits in the graphlets that v touches, where the i^{th} element $(GDV(v)[i])$ corresponds to the orbit i, which is the node with label i in Figure 4.1. It is proposed by [65] to analyze the protein-protein interaction network of yeast and fruitfly.

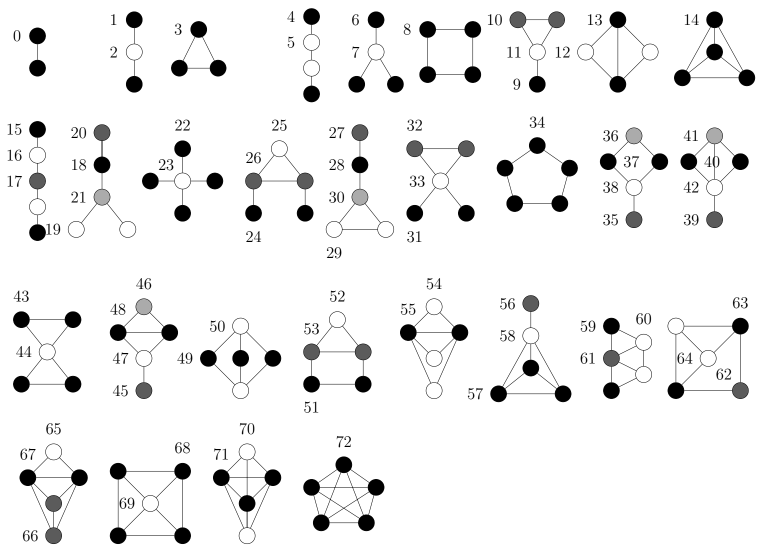
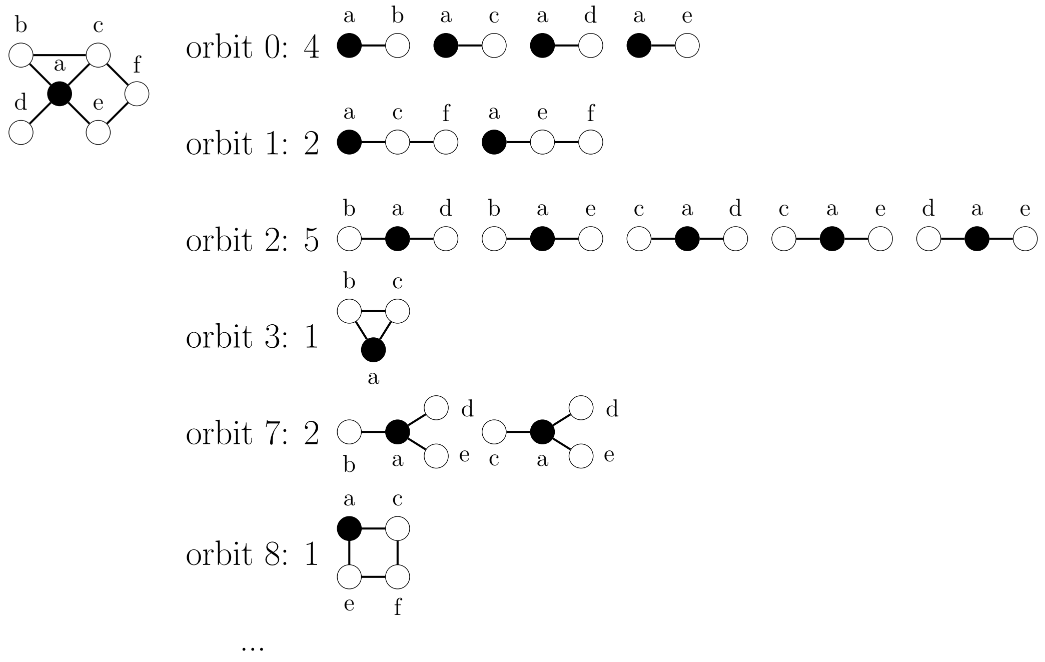


Figure 4.1: 30 graphlets from two to five nodes where nodes are labeled with the corresponding orbit and nodes with the same color in one graphlet have the same topology and belong to the same orbit

GDV4 and GDV5 denote a GDV of graphlets containing up to four nodes and five nodes, respectively. A GDV represents the pattern that a node connects with their adjacent nodes for all possible connecting subgraphs, which captures the local topological structure of that node in a network. By finding the GDV of a protein in the PIN, we can approximately determine how many proteins it interacts with, the pattern of those interactions, and the number and size of protein complexes harboring that protein, which reflects how important it is.

An example of the calculation of the first eight elements of the GDV of a node with label a, GDV(a), is illustrated in Figure 4.2, where the elements, whose values are zero, are omitted.



orbit	0	1	2	3	4	5	6	7	8	...
GDV(a)	4	2	5	1	0	0	0	2	1	...

Figure 4.2: Calculation of the first eight elements of GDV(a), where the elements with value zero are omitted

4.3 Methods

4.3.1 Prediction model

Given a host protein $P_{h(i)}$ and a pathogen protein $P_{p(i)}$, the host-pathogen protein interaction prediction is solved as a binary classification problem, where the input x is a feature vector of a pair of host-pathogen proteins $< P_{h(i)}, P_{p(i)} >$, and the class label $y_i \in \{+1, -1\}$ represents interacting and non-interacting protein pairs, respectively. Two weight-optimization methods are used for the classification model in this work. The first one is a stochastic gradient descent (SGD) algorithm with a logistic loss function shown in Equation 4.1, where $\mathbf{w}$ is a weight vector, $\mathbf{x}$ is a training feature vector and y is a training label.

$$L(\mathbf{w}, \mathbf{x}, y) = \frac{\ln(1 + e^{(\mathbf{w}^T\mathbf{x})y})}{\ln 2} \tag{4.1}$$

SGD is an optimization method that divides a training data set into several groups and updates the weight vector based on the gradient of the cost function using training data of each group at one time. The idea of stochastic optimization was firstly introduced by Robbins and Monroe [70].

Another learning method implemented in this work is a soft confidence-weighted (SCW) learning method [89], which is a second-order online learning method that allows non-separable data and conserves an adaptive margin property. SCW assumes that the weight vector follows the Gaussian distribution and updates the weight using one training data at a time such that the change of the weight vector and the number of misclassification, the first term and the second term in Formula 4.2, respectively, are minimized.

$$\frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + C \cdot max(0, 1 - y_t(\mathbf{w} \cdot \mathbf{x}_t)), \tag{4.2}$$

where $\mathbf{w}$ is the updated weight vector, $\mathbf{w}_t$ is the current weight vector, $\mathbf{x}_t$ and y_t are a training feature vector and the corresponding label of this round, respectively. Compared with other online learning methods, SCW has been shown to outperform or generate comparable accuracy at the low computational cost [89].

From the experiment, we found that overall performance of SCW is better than that of SGD except for *Y. pestis*. Thus, we combine these two algorithms together by assigning label +1 to a sample if at least one of these two method assign label +1 and assigning label -1 otherwise as shown in the following equation.

$$y = \begin{cases} +1 \text{ if label of SGD or label of SCW is } +1 \\ -1 \text{ otherwise} \end{cases} \tag{4.3}$$

4.3.2 Feature set

To optimize the computational time of the model, we select five features to represent a pair of host-pathogen proteins $< P_h, P_p >$, which include 1) a conjoint triad of P_h and P_p; 2) degree; 3) betweenness centrality; 4) clustering coefficient, and 5) a GDV of P_h in the human PIN. The first feature proposed by Shen et al. [77] is the number of all subsequences of 3-mer amino acids in the amino acid sequence of that protein. However, Shen et al. further classified 20 amino acids into seven groups according to the electrostaticity and hydrophobicity as shown in Table 4.1 and counted the compositions of the groups of three consecutive amino acids.

Thus, a conjoint triad consists of $7^3 = 343$-dimensional feature values instead of $20^3 = 8000$ values. The rest of the features are graph-based features that are calculated from the human PIN, where degree is the number of edges connecting to that node, betweenness centrality is the number of the shortest paths passing that node divided by the number of all shortest paths in the graph, clustering coefficient is the number of edges connecting any two of its adjacent nodes divided by the number of all possible edges that its adjacent nodes can have, and GDV is defined in the Section 4.2.2. We used a combinatorial method proposed by Hočevar and Demsar [34] to generate the GDV features of human proteins.

group number	amino acids
1	Ala, Gly, Val
2	Ile, Leu, Phe, Pro
3	Tyr, Met, Thr, Ser
4	His, Asn, Gln, Tpr
5	Arg, Lys
6	Asp, Glu
7	Cys

Table 4.1: Classification of 20 amino acids into seven groups

4.4 Results

4.4.1 Data set

The data of host-pathogen interacting protein pairs used in this work was downloaded from the PHISTO database [86]. Since we cannot confirm which host protein definitely does not interact with which pathogen protein, negative samples were generated by randomly selecting a host protein and a pathogen protein that have not been experimentally verified to interact with each other in the positive samples. The number of generated negative samples was 100 times of the number of positive samples according to the previous studies [46, 85] because they suggested one interacting pair for every 100 random host-pathogen protein pairs. The amino acid sequences used to calculate the conjoint triad feature in this work were obtained from the UniProt database [87] and the human PIN was generated from the interaction data provided in the Human Protein Reference Database (HPRD) [64]. For a pair of protein $< P_h, P_p >$ whose GDV of P_h cannot be found, we handled missing values in three ways: 1) eliminating those samples; 2) replacing missing values with zero, and 3) replacing missing values with the mean, and compare the accuracy of those methods to find the best one.

4.4.2 Experiments and results

We implemented the proposed method to predict the protein-protein interactions between human and four pathogens, *B. anthracis*, *F. tularensis*, *S. typhi*, and *Y. pestis*. 10-fold cross validation was used to evaluate the performance of this method in terms of the F-score (Equation 4.4) due to the imbalance between the numbers of positive and negative data.

$$\begin{aligned}
F - score &= \frac{2 \cdot precision \cdot recall}{precision + recall}, \text{ where} \\
precision &= \frac{\#true\ positive}{\#true\ positive + \#false\ positive}, \text{ and} \\
recall &= \frac{\#true\ positive}{\#true\ positive + \#false\ negative}
\end{aligned} \tag{4.4}$$

First, we compared the performance of the prediction model using three missing-data handling methods based on the SCW optimization technique. The numbers of all positive samples and those without missing features for all species are shown in Table 4.2.

Pathogens	#positive samples	#positive samples without missing features
B. anthracis	3097	1208
F. tularensis	1332	513
S. typhi	103	49
Y. pestis	4099	1615

Table 4.2: The numbers of all positive samples and positive samples without missing features

The F-score results of all prediction models illustrated in Figure 4.3 shows that imputing with zero led to the highest average accuracy followed by imputing with the mean and removing samples with missing features, respectively.

Because imputing missing values with zero led to the best and most consistent performance for all species, the result gives rise to the question of whether the GDV contains necessary information for the prediction or not. To further examine the significance of the GDV to host-pathogen PPI prediction, we implemented the prediction using three sets of features: 1) all features except GDV; 2) all features with GDV4, and 3) all features with GDV5, denoted as noGDV, GDV4, and GDV5, respectively. Missing values were imputed with zero for GDV4 and GDV5. The number of features in these three features sets is shown in Table 4.3.

feature sets	#features
noGDV	689
GDV4	704
GDV5	762

Table 4.3: The number of features in noGDV, GDV4, and GDV5 features sets

The prediction performance using three sets of features shown in Figure 4.4 confirms that although missing GDV was imputed with zero, GDV4 and GDV5 increased the pre-

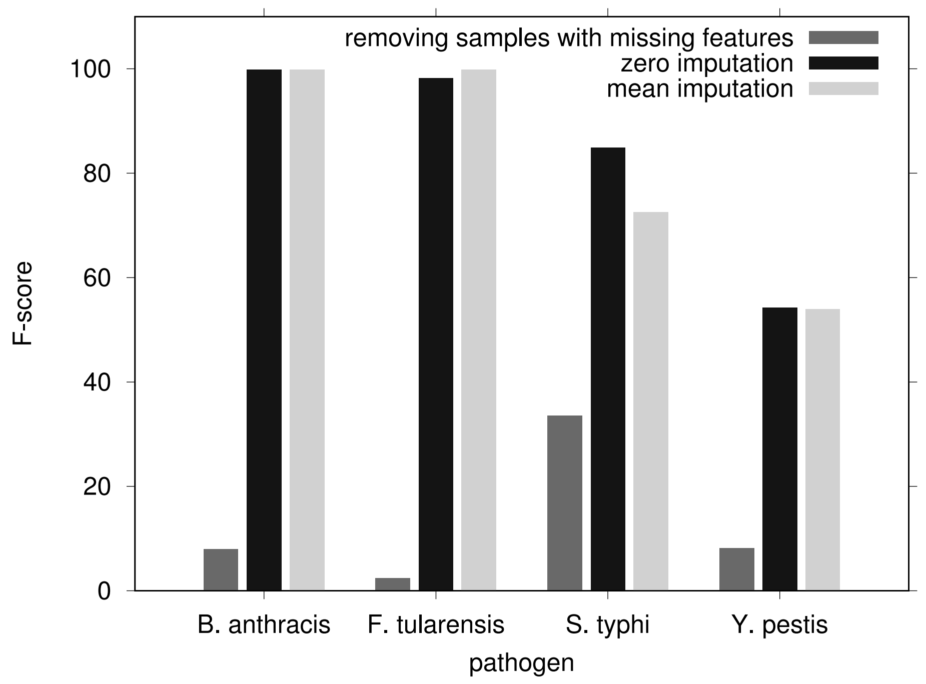


Figure 4.3: F-score of prediction using three missing-data handling methods

diction accuracy of host-pathogen PPI pairs for all pathogens especially *B. anthracis* and *F. tularensis*.

Then, we implemented the combination of two weight-optimization methods, which are SGD and SCW, where missing features were imputed by zero. The performance of the proposed method was compared with that of the existing prediction model using the multitask learning technique [46]. In the multitask learning, the prediction of a host-pathogen protein interaction pair is considered as a task and the information of several tasks is integrated together to predict the interaction between the human host and a pathogen via the proposed feature called *pathway vector*. A pathway vector is a vector representing the biological pathway where the host proteins in positive samples are involved in. Apart from the pathway vector, features used in this multitask learning also consist of: 1) n-mer of amino acid sequence ($n = 2, 3, 4, 5$); 2) degree; 3) betweenness centrality, and 4) clustering coefficient of human proteins in the PIN; 5) gene ontology similarity between the host protein and the pathogen protein, and 6) gene expression of the host protein. The number of features that method is variant among different pathogens, where the minimum number of features is 349,155 for *S. typhi* and the maximum is 886,480 for *Y. pestis* as shown in Table 4.4.

The first feature is similar to the conjoint triad of this work except it includes the information of two, four, and five consecutive amino acids. The second, third, and fourth features are the same as previous work, while we use GDV5 of host proteins instead of the fifth and sixth features. For the data set of their work, the data of *B. anthracis F.*

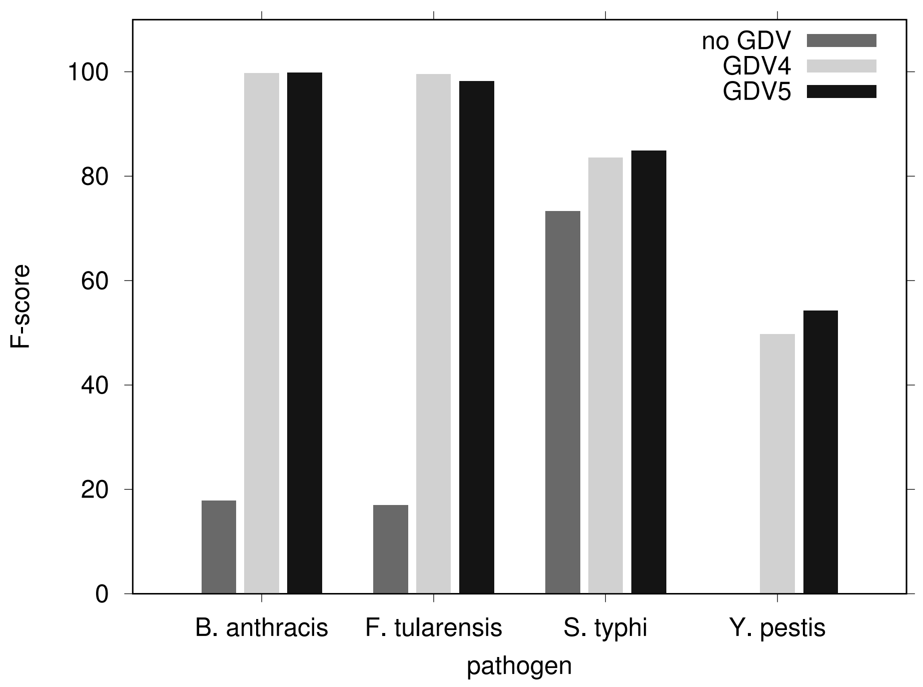


Figure 4.4: F-score of prediction using three sets of features (noGDV, GDV4, and GDV5)

B. anthracis	*F. tularensis*	*S. typhi*	*Y. pestis*
694,715	468,955	349,155	886,480

Table 4.4: The number of features used in the multitask learning method

tularensis and *Y. pestis* were downloaded from PHISTO, which is the same database as this work, but the data of *S. typhi* was downloaded from Salmonella-host interactome [73]. The average F-score of this work and multitask learning are shown in Figure 4.5.

For the three species, *B. anthracis F. tularensis* and *Y. pestis*, whose data are extracted from the same source, this work significantly outperformed the multitask learning method. For *S. typhi*, the proposed method is slightly better than the multitask learning method. The results suggest that by using GDV5 as a feature, the accuracy of the prediction model outperforms another model using the same feature except GDV5 along with two additional features, namely, gene ontology similarity and gene expression. This implies that GDV5 is an outstanding feature of the host protein to predict whether it interacts with a pathogen protein or not.

4.5 Discussion

Because of the scarcity of the available host-pathogen protein interaction data, especially for human-bacteria data, extracting useful information from the limited data is one important task to improve the prediction accuracy. This work introduced a new feature to

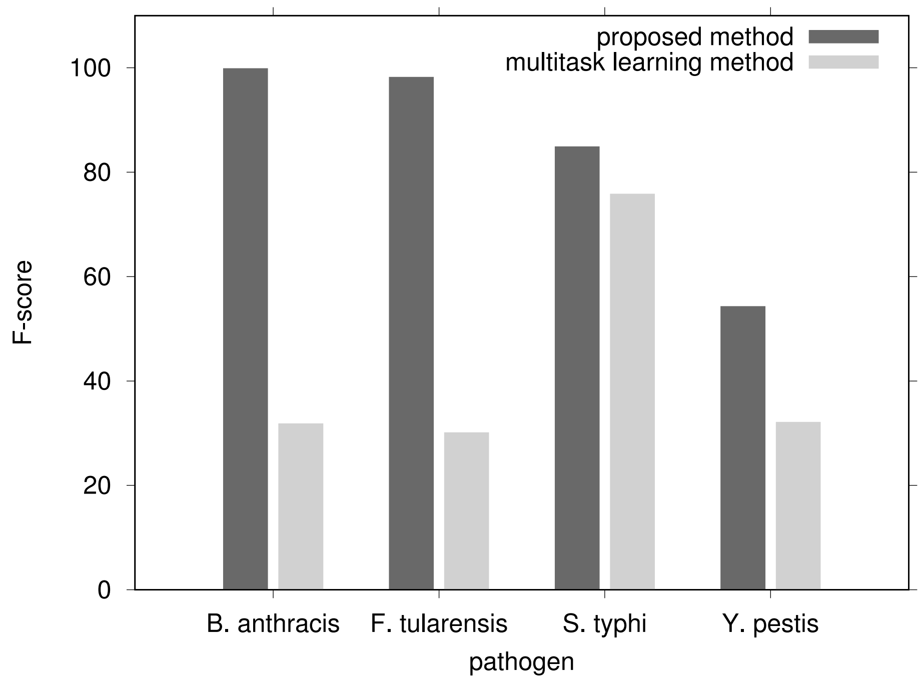


Figure 4.5: F-score of the host-pathogen protein interaction prediction by the proposed method and the existing algorithm (multitask learning)

extract more information from a host-pathogen interacting protein pairs. The introduced feature is a graphlet degree vector, which is a graph-based feature representing a local topology of a protein-protein interaction network. A graphlet degree vector has been used in several problems related with the topology of network but has never been used in this problem before.

The rationale behind the use of this feature is originated from two previous studies. The first one suggests that different pathogens tend to target a host protein with similar function and another one is a study revealing the relationship between the protein function and its local topological structure in the PIN. Accordingly, in this work, we hypothesize that pathogens can also target host proteins with a similar local topology in the PIN and examine it by integrating the GDV of host proteins as a feature during the prediction.

We implemented a prediction model using SCW to optimize the weight using three methods to deal with such missing data. The best method was imputing with zero, while the second was imputing with mean and the last one was removing samples with missing features. By removing samples with missing features, less than half of the training data remain after the elimination as shown in Table 4.2, resulting in the decrease of the model's accuracy. Then, we compared the performance of the prediction models with and without the GDV as a feature. The model without the GDV as the feature had a decreased accuracy. The results suggest that proteins without the GDV feature may seldom interact with other proteins and overall so imputing with zero led to the best performance.

Then, we compared the performance of this work with the existing method based on a multitask learning technique. That existing method does not include GDV in the feature set but uses a gene ontology similarity between a host protein and a pathogen protein and a gene expression of a host protein instead. We found that most of the predictions using GDV outperformed the existing method. This highlights the significance of GDV as an excellent feature for improving the host-pathogen protein interaction prediction.

This work is at the early-stage of applying GDV to solve the host-pathogen protein interaction prediction problem. In order to improve the accuracy, more complicated weight optimization method and parameter-tuning algorithm should be applied to the model. Moreover, there may exist other unused features containing important information for this problem similar to GDV. Therefore, apart from developing weight optimization technique and data preprocessing, identifying new features is also of particular value to improve the accuracy of the prediction.

Chapter 5

Conclusion and future work

5.1 Conclusion

The infectious disease caused by pathogens is one main cause of human death and illness. Due to the pathogens' ability to mutate themselves, the discovery of new drugs is necessary to efficiently eliminate pathogens and decrease the loss from the infectious disease. This thesis provides three computational methods that can be applied to drug discovery research according to a graph structure.

The first work develops a novel efficient enumeration algorithm that generates all non-redundant tree-like chemical compounds containing benzene rings and naphthalene rings from a chemical formula.This algorithm can be applied to various chemoinformatics problems such as structure elucidation, chemical space analysis, and drug discovery. By enumerating compounds from a chemical formula, this algorithm can decrease the size of chemical compound space to be searched for the desired compound if the number of atoms in that compound is known. This work uses a tree-structure representation of chemical compound during the enumeration process and compresses a benzene ring and a naphthalene ring into a single node and two nodes, respectively. The concept of isomorphism is used to examine the redundancy of compounds. We utilized the proposed method and a general-purpose structure generator named *MOLGEN* to enumerate chemical compounds from several chemical formulas. The results showed that the proposed method is reliable and consumes significantly less computational time than the existing method. The number of the enumerated compounds is also compared with the number of discovered compounds in the PubChem database. The comparison suggests that there are a numerous number of compounds that have not been discovered yet. Therefore, this algorithm is one way to discover unknown compounds from a chemical formula.

Second, apart form the atom types, users sometimes have the information of substructures in the desired compounds, e.g. specific receptors that can interact with a target protein or spectral data from a nuclear magnetic resonance spectroscopy experiment of an unknown compound. Therefore, an efficient enumeration algorithm for tree-like chemical compounds containing cyclic substructures defined by users is proposed. Compared with our first work, the limitation of the cyclic substructure within the enumerated compounds decreases from benzene rings and naphthalene rings to any cyclic substructures defined by users. Allowing users to specify substructures can decrease the scope of enumerated compounds as well as the time users spend to find a perfect compound from all candidates. We compared the results of our proposed method with those of another state-of-art method to confirm the correctness and efficiency.

Finally, to design a drug for a specific pathogen, we have to search for the suitable target of a drug. One of the widely used drug target is a protein because most infection mechanisms are related to proteins. The host-pathogen protein interaction prediction method is proposed to inspect the interaction between host proteins and pathogen proteins and discover proteins involved in the infection mechanisms, which are good candidates for the drug target. The proposed method is the first one that uses a graphlet degree vector (GDV) of the host's protein-protein interaction network (PIN) as a feature for predicting the interaction between a host protein and a pathogen protein. This model considers a host-pathogen protein prediction problem as a binary classification with two labels, +1 and -1 denoting interacting and non-interacting protein pairs, respectively. Two learning methods, stochastic gradient descent and soft confidence-weighted, are combined together to classify a given pair of host-pathogen proteins. For the protein-protein interaction between human and four pathogens, the prediction accuracy of this proposed method is better than that of the multitask learning method using the same feature set as this work except GDV but using a gene ontology similarity and a gene expression profile instead. The results suggest that GDV contains information that is important for solving the host-pathogen protein interaction prediction problem. It also supports the previous study showing that proteins with similar function tend to have similar local topology in the PIN.

5.2 Future work

Three computational methods proposed in this thesis outperform existing methods in terms of either computational time or accuracy. However, further development is still necessary to decrease the remaining limitation or improve the performance of these methods.

In the first work, we regarded a naphthalene ring as two benzene nodes bonding together with a special kind of bond, named *merge bond*, instead of another special label of node. The concept of merge bond can be extended to represent more complicated polycyclic aromatic compounds, such as anthracene and phenanthrene, by treating those substructures as a chain of benzene nodes bonded by merge bonds. Being able to enumerate polycyclic aromatic compound decreases the limitation on the structure of the enumerated compounds.

In the second work, the number of the enumerated structures in practical use is usually large. Integrating chemistry with the proposed algorithm to rank the enumerated structures based on their chemical property is also an important future work because it helps the users spend less time to find their target compounds. In the drug discovery problem, the chemical property used to rank the compounds can be druglikeness property such as water solubility, rule of five proposed by Lipinski, and QED score.

The last work in this thesis introduces a graphlet degree vector (GDV) as a feature for solving a host-pathogen protein interaction prediction problem with two simple learning methods. This is the early-stage of using GDV in this problem and the performance can be improved by implementing more complex learning methods to train the model. Although GDV has never been used before in this problem, the results of this work show that GDV is a desirable feature. Accordingly, there might be several unknown yet important features regarding this problem that could be further investigated.

Bibliography

[1] `http://sunflower.kuicr.kyoto-u.ac.jp/tools/enumol/`.

[2] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell. 4th Edition.* Garland Science, 2002.

[3] A. D. Andricopulo, R. V. Guido, and G. Oliva. Virtual screening and its integration with modern drug design technologies. *Current Medicinal Chemistry*, 15(1):37–46, 2008.

[4] M. Badertscher, A. Korytko, K. P. Schulz, M. Madison, M. E. Munk, P. Portmann, M. Junghans, P. Fontana, and E. Pretsch. Assemble 2.0: a structure generator. *Chemometrics and Intelligent Laboratory Systems*, 51(1):73–79, 2000.

[5] K. Balasubramanian. Computer-assisted enumeration of NMR signals. *Journal of Magnetic Resonance (1969)*, 48(2):165–177, 1982.

[6] A. R. Beccari, C. Cavazzoni, C. Beato, and G. Costantino. LiGen: a high performance workflow for chemistry driven de novo design. *Journal of Chemical Information and Modeling*, 53(6):1518–1527, 2013.

[7] C. Benecke, T. Grüner, A. Kerber, R. Laue, and T. Wieland. MOLecular structure GENeration with MOLGEN, new features and future developments. *Fresenius' Journal of Analytical Chemistry*, 359(1):23–32, 1997.

[8] L. C. Blum and J. L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, 131(25):8732–8733, 2009.

[9] B. G. Buchanan and E. A. Feigenbaum. DENDRAL and Meta-DENDRAL: Their applications dimension. Technical report, Defense Technology Information Center, 1978.

[10] J. J. Bürgi, M. Awale, S. D. Boss, T. Schaer, F. Marger, J. M. Viveros Paredes, S. Bertrand, J. Gertsch, D. Bertrand, and J. L. Reymond. Discovery of potent positive allosteric modulators of the $\alpha 3\beta 2$ nicotinic acetylcholine receptor by a chemical space walk in ChEMBL. *ACS Chemical Neuroscience*, 5(5):346–359, 2014.

[11] L. Bytautas, D. J. Klein, and T. G. Schmalz. All acyclic hydrocarbons: Formula periodic table and property overlap plots via chemical combinatorics. *New Journal of Chemistry*, 24(5):329–336, 2000.

[12] R. E. Carhart, D. H. Smith, H. Brown, and C. Djerassi. Applications of artificial intelligence for chemical inference. XVII. Approach to computer-assisted elucidation of molecular structure. *Journal of the American Chemical Society*, 97(20):5755–5762, 1975.

[13] R. E. Carhart, D. H. Smith, N. A. B. Gray, J. G. Nourse, and C. Djerassi. GENOA: A computer program for structure elucidation utilizing overlapping and alternative substructures. *Journal of Organic Chemistry*, 46:1708–1718, 1981.

[14] A. Casadevall and L. Pirofski. Host-pathogen interactions: Basic concepts of microbial commensalism, colonization, infection, and disease. *Infection and Immunity*, 68(12):6511–6518, 2000.

[15] A. Cayley. On the analytical forms called trees, with application to the theory of chemical combinations. *Report of British Association for the Advancement of Science*, 45:257–305, 1875.

[16] K. Chen, T. Wang, and C. Chan. Associations between HIV and human pathways revealed by protein-protein interactions and correlated gene expression profiles. *PLOS ONE*, 7(3):e34240, 2012.

[17] C. J. Churchwell, M. D. Rintoul, S. Martin, D. P. Visco, A. Kotu, R. S. Larson, L. O. Sillerud, D. C. Brown, and J.-L. Faulon. The signature molecular descriptor: 3. inverse-quantitative structure–activity relationship of ICAM-1 inhibitory peptides. *Journal of Molecular Graphics and Modelling*, 22(4):263–273, 2004.

[18] M. M. Cox and G. N. Phillips. *Handbook of Proteins: Structure, Function and Methods*. Wiley, 2007.

[19] D. Davis, O. N. Yaveroğlu, N. M. Dognin, A. Stojmirovic, and N. Pržulj. Topology-function conservation in protein-protein interaction networks. *Bioinformatics*, 31(10):1632–1639, 2015.

[20] F. P. Davis, D. T. Barkan, N. Eswar, J. H. McKerrow, and A. Sali. Host–pathogen protein interactions predicted by comparative modeling. *Protein Science*, 16(12):2585–2596, 2007.

[21] J. Drews. Drug discovery: A historical perspective. *Science*, 287(5460):1960–1964, 2000.

[22] M. D. Dyer, T. Murali, and B. W. Sobral. Computational prediction of host-pathogen protein–protein interactions. *Bioinformatics*, 23(13):i159–i166, 2007.

[23] M. D. Dyer, T. Murali, and B. W. Sobral. The landscape of human proteins interacting with viruses and other pathogens. *PLOS Pathogens*, 4(2):e32, 2008.

[24] P. Evans, W. Dampier, L. Ungar, and A. Tozeren. Prediction of HIV-1 virus-host protein interactions using virus and host sequence motifs. *BMC Medical Genomics*, 2(1):1, 2009.

[25] J. Faulon, D. P. Visco, and D. Roe. Enumerating molecules. *Reviews in Computational Chemistry*, 21:209, 2005.

[26] J. L. Faulon. Stochastic generator of chemical structure. 1. Application to the structure elucidation of large molecules. *Journal of Chemical Information and Computer Sciences*, 34(5):1204–1218, 1994.

[27] B. B. Finlay and P. Cossart. Exploitation of mammalian host cell functions by bacterial pathogens. *Science*, 276(5313):718–725, 1997.

[28] H. Fujiwara, J. Wang, L. Zhao, H. Nagamochi, and T. Akutsu. Enumerating treelike chemical graphs with given path frequency. *Journal of Chemical Information and Modeling*, 48(7):1345–1357, 2008.

[29] K. Funatsu and S. Sasaki. Recent advances in the automated structure elucidation system, CHEMICS. utilization of two-dimensional NMR spectral information and development of peripheral functions for examination of candidates. *Journal of Chemical Information and Computer Sciences*, 36(2):190–204, 1996.

[30] R. Gugisch, A. Kerber, A. Kohnert, R. Laue, M. Meringer, C. Rücker, A. Wassermann, S. Basak, G. Restrepo, and J. Villavecess. MOLGEN 5.0, a molecular structure generator. *Advances in Mathematical Chemistry and Applications*, 1:113–138, 2014.

[31] F. Harary. *Graph Theory.* Addison-Wesley Publishing Company, 1969.

[32] S. Hardinger and University of California, Los Angeles Department of Chemistry and Biochemistry. *Chemistry 14D: Organic Reactions and Pharmaceuticals : Course Thinkbook, Lecture Supplements, Concept Focus Questions, OWLS Problems, Practice Problems.* Hayden-McNeil Publishing, Plymouth, MI 48170, 2008.

[33] F. He, A. Hanai, H. Nagamochi, and T. Akutsu. Enumerating naphthalene isomers of tree-like chemical graphs. *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies*, 3:258–265, 2016.

[34] T. Hočevar and J. Demšar. A combinatorial approach to graphlet counting. *Bioinformatics*, 30(4):559–565, 2014.

[35] Y. Ishida, Y. Kato, L. Zhao, H. Nagamochi, and T. Akutsu. 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.

[36] M. Jaspars. Computer assisted structure elucidation of natural products using two-dimensional NMR spectroscopy†. *Natural Product Reports*, 16(2):241–248, 1999.

[37] J. Jindalertudomdee, M. Hayashida, and T. Akutsu. Enumeration method for structural isomers containing user-defined structures based on breadth-first search approach. *Journal of Computational Biology*, 2016.

[38] J. Jindalertudomdee, M. Hayashida, Y. Zhao, and T. Akutsu. Enumeration method for tree-like chemical compounds with benzene rings and naphthalene rings by breadth-first search order. *BMC Bioinformatics*, 17(1), 2016.

[39] C.-L. Kee, X. Ge, M.-Y. Low, and H.-L. Koh. Structural elucidation of a new sildenafil analogue using high-resolution Orbitrap mass spectrometry. *Rapid Communications in Mass Spectrometry*, 27(12):1380–1384, 2013.

[40] A. Kerber, R. Laue, T. Gruner, and M. Meringer. MOLGEN 4.0. *MATCH Communications in Mathematical and in Computer Chemistry*, 37:205–208, 1998.

[41] T. Kind and O. Fiehn. Advances in structure elucidation of small molecules using mass spectrometry. *Bioanalytical Reviews*, 2(1-4):23–60, 2010.

[42] M. T. Klein, G. Hou, R. J. Quann, W. Wei, K. H. Liao, R. S. Yang, J. A. Campain, M. A. Mazurek, and L. J. Broadbelt. BioMOL: a computer-assisted biological modeling tool for complex chemical mixtures and biological processes at the molecular level. *Environmental Health Perspectives*, 110(Suppl 6):1025–1029, 2002.

[43] M. A. Koch, A. Schuffenhauer, M. Scheck, S. Wetzel, M. Casaulta, A. Odermatt, P. Ertl, and H. Waldmann. Charting biologically relevant chemical space: A structural classification of natural products (SCONP). *Proceedings of the National Academy of Sciences of the United States of America*, 102(48):17272–17277, 2005.

[44] S. Koichi, M. Arisaka, H. Koshino, A. Aoki, S. Iwata, T. Uno, and H. Satoh. Chemical structure elucidation from ^{13}C NMR chemical shifts: Efficient data processing using bipartite matching and maximal clique algorithms. *Journal of Chemical Information and Modeling*, 54(4):1027–1035, 2014.

[45] O. Krishnadev and N. Srinivasan. Prediction of protein–protein interactions between human host and a pathogen and its application to three pathogenic bacteria. *International Journal of Biological Macromolecules*, 48(4):613–619, 2011.

[46] M. Kshirsagar, J. Carbonell, and J. Klein-Seetharaman. Multitask learning for host–pathogen protein interactions. *Bioinformatics*, 29(13):i217–i226, 2013.

[47] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. Applications of artificial intelligence for organic chemistry: the DENDRAL project. *McGraw-Hill, Inc.*, 1980.

[48] J. Maitland Jones. *Organic Chemistry 3rd Edition.* W. W. Norton & Company, 2004.

[49] S. Martin. Lattice enumeration for inverse molecular design using the signature descriptor. *Journal of Chemical Information and Modeling*, 52(7):1787–1797, 2012.

[50] C. D. Mathers and D. Loncar. Projections of global mortality and burden of disease from 2002 to 2030. *PLOS Medicine*, 3(11):e442, 2006.

[51] H. Mauser and M. Stahl. Chemical fragment spaces for de novo design. *Journal of Chemical Information and Modeling*, 47(2):318–324, 2007.

[52] A. D. McNaught and A. Wilkinson. *Compendium of Chemical Terminology: IUPAC Recommendations.* Blackwell Scientific Publications, 1997.

[53] M. Meringer. *Handbook of Chemoinformatics Algorithms*, chapter 8 Structure Enumeration and Sampling, pages 233–267. CRC Press, Boca Raton, Florida, 2010.

[54] M. Meringer and E. L. Schymanski. Small molecule identification with MOLGEN and mass spectrometry. *Metabolites*, 3(2):440–462, 2013.

[55] K. Mishima, H. Kaneko, and K. Funatsu. Development of a new de novo design algorithm for exploring chemical space. *Molecular Informatics*, 33(11-12):779–789, 2014.

[56] M. S. Molchanova, V. V. Shcherbukhin, and N. S. Zefirov. Computer generation of molecular structures by the SMOG program. *Journal of Chemical Information and Computer Sciences*, 36(4):888–899, 1996.

[57] C. Montecucco, E. Papini, and G. Schiavo. Bacterial protein toxins penetrate cells via a four-step mechanism. *FEBS Letters*, 346(1):92–98, 1994.

[58] J. B. Moon and W. J. Howe. Computer design of bioactive molecules: A method for receptor-based de novo ligand design. *Proteins: Structure, Function, and Bioinformatics*, 11(4):314–328, 1991.

[59] M. E. Munk, C. A. Shelley, H. B. Woodruff, and M. O. Trulson. Computer-assisted structure elucidation. *Fresenius' Zeitschrift für Analytische Chemie*, 313(6):473–479, 1982.

[60] C. J. Murray and A. D. Lopez. Alternative projections of mortality and disability by cause 1990–2020: Global burden of disease study. *The Lancet*, 349(9064):1498–1504, 1997.

[61] E. Nourani, F. Khunjush, and S. Durmuş. Computational approaches for prediction of pathogen-host protein-protein interactions. *Frontiers in Microbiology*, 6:94–103, 2015.

[62] I. Ojanperä, M. Kolmonen, and A. Pelander. Current use of high-resolution mass spectrometry in drug screening relevant to clinical and forensic toxicology and doping control. *Analytical and Bioanalytical Chemistry*, 403(5):1203–1220, 2012.

[63] J. E. Peironcely, M. Rojas-Chertó, D. Fichera, T. Reijmers, L. Coulier, J.-L. Faulon, and T. Hankemeier. OMG: open molecule generator. *Journal of Cheminformatics*, 4(1), 2012.

[64] T. K. Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, et al. Human protein reference database—2009 update. *Nucleic Acids Research*, 37(suppl 1):D767–D772, 2009.

[65] N. Pržulj, D. G. Corneil, and I. Jurisica. Modeling interactome: Scale-free or geometric? *Bioinformatics*, 20(18):3508–3515, 2004.

[66] Y. Qi, O. Tastan, J. G. Carbonell, J. Klein-Seetharaman, and J. Weston. Semi-supervised multi-task learning for predicting interactions between HIV-1 and human proteins. *Bioinformatics*, 26(18):i645–i652, 2010.

[67] J.-L. Reymond, L. Ruddigkeit, L. Blum, and R. van Deursen. The enumeration of chemical space. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(5):717–733, 2012.

[68] J. L. Reymond, R. van Deursen, L. C. Blum, and L. Ruddigkeit. Chemical space as a source for new drugs. *MedChemComm*, 1(1):30–38, 2010.

[69] D. Ribet and P. Cossart. How bacterial pathogens colonize their hosts and invade deeper tissues. *Microbes and Infection*, 17(3):173–183, 2015.

[70] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[71] L. Ruddigkeit, R. Van Deursen, L. C. Blum, and J.-L. Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012.

[72] K. Scheubert, F. Hufsky, and S. Böcker. Computational mass spectrometry for small molecules. *Journal of Cheminformatics*, 5(1), 2013.

[73] S. Schleker, J. Sun, B. Raghavan, M. Srnec, N. Müller, M. Koepfinger, L. Murthy, Z. Zhao, and J. K. Seetharaman. The current Salmonella–host interactome. *Proteomics Clinical Applications*, 6:117–133, 2012.

[74] G. Schneider and U. Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649–663, 2005.

[75] N. Schore and P. Vollhardt. *Organic Chemistry: Structure and Function 6th Edition*. Freeman Publisher, 2011.

[76] A. Schüller, V. Hähnke, and G. Schneider. SmiLib v2.0: A java-based tool for rapid combinatorial library enumeration. *QSAR & Combinatorial Science*, 26(3):407–410, 2007.

[77] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li, and H. Jiang. Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences of the United States of America*, 104(11):4337–4341, 2007.

[78] M. Shimizu, H. Nagamochi, and T. Akutsu. Enumerating tree-like chemical graphs with given upper and lower bounds on path frequencies. *BMC Bioinformatics*, 12(14), 2011.

[79] D. H. Smith, N. A. Gray, J. G. Nourse, and C. W. Crandell. The DENDRAL project: Recent advances in computer-assisted structure elucidation. *Analytica Chimica Acta*, 133:471–497, 1981.

[80] T. G. Solomons, C. B. Fryhle, and S. A. Snyder. *Organic Chemistry 11th Edition*. Wiley, 2012.

[81] C. M. Song, P. H. Bernardo, C. L. Chai, and J. C. Tong. CLEVER: Pipeline for designing in silico chemical libraries. *Journal of Molecular Graphics and Modelling*, 27(5):578–583, 2009.

[82] C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E. Willighagen. The chemistry development kit (CDK): An open-source java library for chemo-and bioinformatics. *Journal of Chemical Information and Computer Sciences*, 43(2):493–500, 2003.

[83] C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, and E. L. Willighagen. Recent developments of the chemistry development kit (CDK)-an open-source java library for chemo-and bioinformatics. *Current Pharmaceutical Design*, 12(17):2111–2120, 2006.

[84] M. Suzuki, H. Nagamochi, and T. Akutsu. Efficient enumeration of monocyclic chemical graphs with given path frequencies. *Journal of Cheminformatics*, 6(1), 2014.

[85] O. Tastan, Y. Qi, J. G. Carbonell, and J. Klein-Seetharaman. Prediction of interactions between HIV-1 and human proteins by information integration. *Pacific Symposium on Biocomputing*, pages 516–527, 2009.

[86] S. D. Tekir, T. Çakır, E. Ardıç, A. S. Sayılırbaş, G. Konuk, M. Konuk, H. Sarıyer, A. Uğurlu, İ. Karadeniz, A. Özgür, et al. PHISTO: pathogen-host interaction search tool. *Bioinformatics*, 29(10):1357–1358, 2013.

[87] The UniProt Consortium. UniProt: a hub for protein information. *Nucleic Acids Research*, 43:D204–D212, 2015.

[88] N. Trinajstić. *Chemical Graph Theory*, chapter 11 Isomer Enumeration, pages 275–391. CRC press, Boca Raton, Florida, 2 edition, 1992.

[89] J. Wang, P. Zhao, and S. C. Hoi. Exact soft confidence-weighted learning. *arXiv preprint arXiv:1206.4612*, 2012.

[90] R. A. Ward and J. G. Kettle. Systematic enumeration of heteroaromatic ring systems as reagents for use in medicinal chemistry. *Journal of Medicinal Chemistry*, 54(13):4670–4677, 2011.

[91] W. A. Warr. Computer-assisted structure elucidation. *Analytical Chemistry*, 65(24):1087A–1095A, 1993.

[92] T. Wieland, A. Kerber, and R. Laue. Principles of the generation of constitutional and configurational isomers. *Journal of Chemical Information and Computer Sciences*, 36:413–419, 1996.

[93] S. G. Williamson. *Combinatorics for Computer Science.* Courier Corporation, 1985.

[94] R. J. Wilson. *Introduction to Graph Theory (5th Edition).* Pearson Education Limited, 2012.

[95] Y. Zhao, M. Hayashida, J. Jindalertudomdee, H. Nagamochi, and T. Akutsu. Breadth-first search approach to enumeration of tree-like chemical compounds. *Journal of Bioinformatics and Computational Biology*, 11(06), 2013.

List of Publications

Journal Papers

1. Jindalertudomdee, J., Hayashida, M., and Akutsu, T. (2016). Enumeration Method for Structural Isomers Containing User-defined Structures Based on Breadth-first Search Approach. Journal of Computational Biology, **(Chapter 3)**

2. Jindalertudomdee, J., Hayashida, M., Zhao, Y., and Akutsu, T. (2016). Enumeration method for tree-like chemical compounds with benzene rings and naphthalene rings by breadth-first search order. BMC bioinformatics, 17(1), 1. **(Chapter 2)**

3. Hayashida, M., Jindalertudomdee, J., Zhao, Y., and Akutsu, T. (2015). Parallelization of enumerating tree-like chemical compounds by breadth-first search order. BMC medical genomics, 8(Suppl 2), S15.

4. Zhao, Y., Hayashida, M., Jindalertudomdee, J., Nagamochi, H., and Akutsu, T. (2013). Breadth-first search approach to enumeration of tree-like chemical compounds. Journal of bioinformatics and computational biology, 11(06), 1343007.

Conference Paper

1. Jindalertudomdee, J., Hayashida, M., Song, J., and Akutsu, T. Host-pathogen Protein Interaction Prediction Based on Local Topology Structures of a Protein Interaction Network. Accepted for IEEE 16th International Conference on BioInformatics and BioEngineering, Taichung Taiwan, 31 October - 2 November 2016.

Conference Abstracts

1. Jindalertudomdee, J. Breadth-first Search Based Approach to Enumerating Chemical Compounds Containing Outerplanar Fused Benzen Ring Substructures. International Workshop on Bioinformatics and Systems Biology, Boston University, 19-22 July 2015.

2. Jindalertudomdee, J., Hayashida, M., Zhao, Y., and Akutsu, T. ナフタレン環を持つ木状化学構造の幅優先探索による列挙手法. 第37回情報化学討論会, Toyohashi Chamber of Commerce and Industry, 20 November 2014.

3. Jindalertudomdee, J., Hayashida, M., Zhao, Y., and Akutsu, T. ベンゼン環を持つ木状化学構造の幅優先探索による列挙手法. 第36回情報化学討論会, Tsukuba University, 31 October 2013.