Novel Methods for Chemical Compound Inference Based on Machine Learning and Mixed Integer Linear Programming

Jianshen Zhu

Novel Methods for Chemical Compound Inference Based on Machine Learning and Mixed Integer Linear Programming

Jianshen Zhu

Department of Applied Mathematics and Physics
Graduate School of Informatics
Kyoto University
Kyoto, Japan

August 6, 2023

Doctoral dissertation
submitted to the
Graduate School of Informatics, Kyoto University
in partial fulfillment of
the requirement for the degree of
DOCTOR OF INFORMATICS

Preface

Analysis of the activities and properties of chemical compounds is one of the important applications of intelligent computing. The problem of computational prediction of chemical activities from their structural data has been studied for several decades under the name of quantitative structure activity relationship (QSAR). Also, inference of chemical structures with desired chemical activities under some constraints is an important task not only for chemical science but also for biological science because of its potential applications to drug design. This problem has been studied under the name of inverse quantitative structure activity relationship (inverse QSAR). However, most of the existing approaches for this problem including many Artificial Neural Network (ANN)-based approaches do not guarantee exact or optimal solutions due to inherent difficulties.

This research work aims to design a novel inverse QSAR method that can (i) predict chemical activities and properties of chemical compounds from their structural data (especially from their graph structure) with a good performance, (ii) infer new chemical structures with desired chemical activities or properties and guarantee the exact or optimal solutions or show that such a solution does not exist by using mixed integer linear programming (MILP) to formulate the problem, and (iii) generate many candidate solutions in a relatively short time.

We first extend one of the existing frameworks which was designed for the restricted case such that the chemical compounds are represented by trees to the case of "rank-2 chemical compounds", where the graph rank is defined to be the minimum number of edges whose removal makes the graph acyclic. We manage to treat three kinds of different rank-2 polymer topologies together in one MILP formulation. We show that the inverse problem can be solved efficiently with up to 30 non-hydrogen atoms.

We then extend the framework to the case of any arbitrary graph topologies with at least one cycle when the abstract graph topology is given. Also, a flexible way of specifying the topological structure of target chemical graphs is introduced to allow the possibility to include domain knowledge in the way of specifying graph structures. This entirely new mechanism of inferring chemical graphs together with a new mechanism to enumerate chemical graphs conserving certain constraints can generate chemical graphs with up to 50 non-hydrogen atoms in around 20 minutes, and later improved to around 20 times faster with

a more sophisticated MILP formulation.

Finally, to improve the learning performance of QSAR phase of the model, we include not only ANN but also other machine learning methods like linear regression, a newly-proposed method called adjustive linear regression, and multiple linear regression by using quadratic terms with feature reduction. The inverse problems of all those methods can be formulated as an MILP formulation and thus be integrated into the framework of the two-layered model, an extended MILP-based inverse QSAR model that can treat both the case of graph topologies of at least one cycle and the case of a tree. By applying these methods, we improved the learning performance of QSAR drastically for some chemical property datasets while the capable size of inferred chemical compounds and the time efficiency of solving MILP formulation was preserved.

We believe that the work described in the thesis will help develop the related fields in both theoretical and application aspects.

Kyoto, June 2023
Jianshen Zhu

Acknowledgement

This thesis would not have been possible without the help, support, and guidance of many others.

First of all, I would like to express my sincere and heartfelt gratitude to Professor Hiroshi Nagamochi for accepting me into the Discrete Mathematics Laboratory. Throughout the three and a half years as my supervisor, his unwavering support, dedication to academic research, problem-solving approach, and unique perspective on overcoming challenges have greatly influenced and shaped my growth as a researcher. Our thorough discussions and his continuous encouragement have played an instrumental role in my development. I am truly honored and privileged to have had the opportunity to learn from him, and I look forward to continuing to benefit from his wisdom throughout my life.

I would also like to extend my sincere thanks to Associate Professor Kazuya Haraguchi, who became my supervisor following Professor Hiroshi Nagamochi's retirement. His precise feedback on my thesis and invaluable guidance during the preparation phase has been immensely valuable. I hope to continue benefiting from his expertise in the following years.

I am deeply grateful to Professor Nobuo Yamashita and Professor Tatsuya Akutsu for being part of my dissertation review committee. Their insightful comments and suggestions have significantly contributed to improving the quality of my work and its presentation in various aspects.

Special appreciation goes to former Assistant Professor Aleksandar Shurbevski for his academic support and guidance during my initial days in the laboratory. His extensive experience in programming skills proved invaluable, providing significant assistance throughout my research process.

I extend my utmost gratitude to the Japan Society for the Promotion of Science (JSPS) for their generous fellowship awards and research grants. These funding opportunities were particularly crucial during the challenging period of the COVID-19 pandemic, enabling me to carry out my research effectively.

I would also like to express my sincere thanks to all the other members of the laboratory during my four-year tenure, especially Dr. Naveed Ahmed Azam. Their advice and assistance have contributed to making my research experience more enriching and fulfilling.

Lastly, I would like to convey my deepest appreciation to my family in China.

Their unwavering support and encouragement have been the foundation of my academic journey. Without them, I would not have had the opportunity to pursue my studies and research at Kyoto University.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Background In recent years, extensive studies have been done on the design of novel molecules using various machine learning techniques [31, 50]. Computational molecular design has also a long history in the field of chemo-informatics, and has been studied under the name of *quantitative structure activity relationship* (QSAR) [10, 44], and *inverse quantitative structure activity relationship* (inverse QSAR) [24, 33, 40]. Analysis of the activities and properties of chemical compounds is important not only for chemical science but also for biological science because chemical compounds play important roles in metabolic and many other pathways.

The purpose of QSAR is to predict chemical activities from given chemical structures [10]. In most of the existing QSAR studies, a chemical structure is given as an undirected graph called *chemical graphs*, and represented as a vector of real numbers called *descriptors* and correspond to *feature vectors* in machine learning. A prediction function is usually obtained from existing structure-activity relation data. To this end, various regression-based methods and machine learning-based methods, including artificial neural network (ANN)-based methods, have been utilized recently [31, 50].

Conversely, the purpose of inverse QSAR is to predict chemical structures from given chemical activities [24, 33, 40], where additional constraints may often be imposed to effectively restrict the possible structures. Inference of chemical structures with desired chemical activities under some constraints is also important because of its potential applications to drug design. Traditionally, a feature vector is firstly computed by applying some optimization or sampling method to the prediction method obtained by usual QSAR and then reconstruct the chemical structures from the feature vector. However, the reconstruction itself is not an easy task because the number of possible chemical graphs is huge [9]. For example, the number of chemical graphs with up to 30 atoms (vertices) `C`, `N`, `O`, and `S` may exceed 10^{60} [9]. Indeed, the problem to infer a chemical graph from a given feature vector is known as a computationally difficult problem (precisely, NP-hard) except for some simple cases [3]. Due to this inherent difficulty, most existing methods employ heuristic methods for reconstructions of chemical structures and thus do not guarantee the optimality or exactness of the solutions.

On the other hand, Artificial Neural Networks (ANN) and deep learning technologies have recently been applied to inverse QSAR because generative models are available [16]. Furthermore, graph-structured data can be directly handled by using graph convolutional networks [28]. Therefore, it is reasonable to try to apply ANNs to inverse QSAR [55]. Indeed, various ANN-based models have been proposed, which includes variational autoencoders [17], recurrent neural networks [41, 57], grammar variational autoencoders [29], generative adversarial networks [12], and invertible flow models [32, 42]. In these approaches, new chemical graphs are generated by solving a kind of inverse problems on neural networks, where neural networks are trained using known chemical compound property datasets. However, two important expected properties of the solutions, optimality, the quality of the solution for the inverse problem of the learning method, or exactness, whether the solution admits a valid chemical graph, is not yet guaranteed by these methods.

For example, recently Takeda et al. [47] proposed an approach for inverse molecular design. They encoded every chemical compound into a feature vector based on the frequencies of a variety of subgraphs, namely substructures, in the corresponding chemical graph. They constructed one ANN for the QSAR phase (namely encoder), and the inverse QSAR phase (namely decoder), respectively. The output of the decoder is a feature vector, and they thus combinatorically generated chemical graphs with such a feature vector based on a branching algorithm. We note that, in this approach, neither the optimality of the solution, i.e. whether the feature vector will obtain the desired property value when served as the input for the encoder, nor the exactness, i.e. whether there exists a valid chemical graph corresponding to the output feature vector from the decoder, is guaranteed. Also, as mentioned above, the construction from a feature vector to a chemical graph itself is a difficult problem. The computational experimental results showed that it took an average of six minutes to generate one chemical graph with around 10 non-hydrogen atoms, which was not very efficient.

There are many studies that do not utilize feature vectors but that deal with the graph more directly. For example, Shi et al. [42] proposed GraphAF, a flow-based autoregressive model to generate a chemical graph. In this approach, the input of the encoder and output of the decoder are directly a chemical graph and thus it is unnecessary to design a feature vector for a chemical compound. It iteratively samples random variables to map them to atom/bond features during the procedure of generating chemical graphs. However, the exactness of the solution, here the validity of the output graph as a chemical graph, is still not guaranteed theoretically. According to the experimental results shown by the

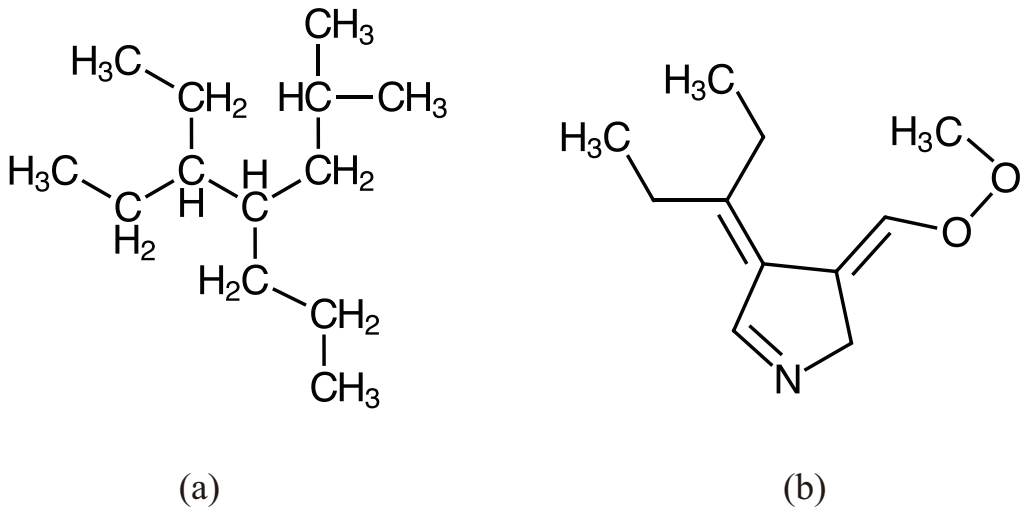


Figure 1.1. (a) An acyclic chemical graph; (b) A monocyclic chemical graph.

authors, only around 70% of the generated graphs are valid when extra checks for validity are not utilized. The optimality of the solutions is not guaranteed either. Moreover, as a method that uses random sampling during the process, the same chemical graph can be output for different trials of generating. Finally, because of the complexity of this proposed model, it requires an expensive computation environment and a huge running cost.

In order to guarantee the optimality for the inverse problem of ANNs, a novel approach has been proposed by Akutsu and Nagamochi [1] for ANNs with ReLU activation functions and sigmoid activation functions. They manage to formulate this inverse problem as a mixed integer linear programming (MILP) to guarantee the optimality of the solution theoretically, by simulating the process of ANN in the terms of linear constraints of some real and integer variables. In their approach, activation functions on neurons are efficiently encoded as piece-wise linear functions so as to represent ReLU functions exactly and sigmoid functions approximately. Based on this approach, several methods have been proposed [4, 11, 59, 60] based on an MILP formulation designed especially for acyclic chemical compounds (see Figure 1.1(a) as an example). Afterward, Ito et al. [25] designed a method of inferring monocyclic chemical graphs (chemical graphs with cycle index or rank 1, see Figure 1.1(b) as an example) by formulating a new MILP and using an efficient algorithm for enumerating monocyclic chemical graphs [46].

Contributions Although several methods have been proposed based on this framework for the case of acyclic chemical compounds [4, 11, 59, 60] and monocyclic chemical graphs [25], it is still far from being complete. The ratio of acyclic

and monocyclic chemical graphs in the chemical database PubChem [27] is only 2.91% and 16.26%, respectively. Motivated by the existing methods and applications, we apply this framework to a broader range of chemical compounds, which can cover most (nearly 97%) of the chemical graphs in PubChem.

The learning performance of the QSAR phase plays an important role in the quality of the inferred chemical compounds, but ANN sometimes cannot give a satisfactory performance because of its intrinsic complexity for finding a good local optimum. The issue of overfitting is also a serious problem when using ANN as the learning method for the QSAR [16], and one reason for this is that the dataset size may be too small to apply an ANN for some chemical properties. Considering this issue, we include the usage of several machine learning methods other than ANN inside this framework to manage to improve the learning performance for various chemical property datasets.

The rest chapters are organized as follows:

- In Chapter 2, we define some basic notations and terminologies for graphs, chemical graphs, and machine learning;
- In Chapter 3, we briefly review the framework for inferring chemical compounds proposed in [4]. This framework is based on ANNs and MILPs, and the optimality and exactness of the solutions are guaranteed.
- From Chapters 4 to 5, we extended the MILP-based approach in [4] for broader classes of chemical graphs:
  - Chapter 4: The class of "rank-2 chemical compounds" (chemical graphs with cycle index or rank 2);
  - Chapter 5: The class of arbitrary cyclic chemical compounds;

  We note that the MILP formulations in Chapter 5 can be even improved based on a characterization of a chemical acyclic graph [62].
- From Chapters 6 to 8, we apply machine learning methods other than ANN into the recently proposed two-layered model [43], which is an extended version of this MILP-based framework that can treat both the case of graph topologies of at least one cycle and the case of a tree:
  - Chapter 6: Lasso linear regression;
  - Chapter 7: Adjustive linear regression, a newly-proposed machine learning method;

  - Chapter 8: Multiple linear regression with quadratic descriptors where a heuristic for feature reduction is applied, where we also utilize a method called grid neighbor search [6] to systematically find more solutions of the MILP formulation;

  We note that some other machine learning methods like decision tree [49] can also be applied into the two-layered model.

- In Chapter 9, we give some conclusion remarks.

While this study mainly takes the small chemical compounds called monomers as the targets, we note that it is also possible to apply the framework to the kind of molecules called polymers [23], which also have a wide range of applications in both medical science and material science.

We put the detailed MILP formulations for Chapter 4 in Appendix A. Some details for Chapter 7 can be found in Appendix B.

Most of the contents in Chapters 4 to 8 have appeared as [1, 2, 5, 6, 7] in List of Author's Work, respectively.

2 Preliminaries

In this chapter, we give some notions and terminology that will be used throughout Chapters 3 to 8.

Let $\mathbb{R}$, $\mathbb{R}_+$, $\mathbb{Z}$ and $\mathbb{Z}_+$ denote the sets of reals, non-negative reals, integers and non-negative integers, respectively. For two integers a and b, let $[a, b]$ denote the set of integers i with $a \leq i \leq b$. For a vector $x \in \mathbb{R}^p$, the j-th entry of x is denoted by $x(j), j \in [1, p]$.

2.1 Graphs

Graphs When using the term *graph* G, we will always assume G as a connected simple graph. Given a graph G, let $V(G)$ and $E(G)$ denote the sets of vertices and edges, respectively. For a subset $V' \subseteq V(G)$ (resp., $E' \subseteq E(G)$) of a graph G, let $G - V'$ (resp., $G - E'$) denote the graph obtained from G by removing the vertices in V' (resp., the edges in E'), where we remove all edges incident to a vertex in V' in $G - V'$. An edge subset $E' \subseteq E(G)$ in a connected graph G is called *separating* (resp., *non-separating*) if $G - E'$ becomes disconnected (resp., $G - E'$ remains connected). The *rank* or *cycle index* $\mathrm{r}(G)$ of a connected graph G is defined to be the minimum $|F|$ of an edge subset $F \subseteq E(G)$ such that $G - F$ contains no cycle, where $\mathrm{r}(G) = |E(G)| - |V(G)| + 1$. Observe that $\mathrm{r}(G - E') = \mathrm{r}(G) - |E'|$ holds for any non-separating edge subset $E' \subseteq E(G)$. An edge $e = u_1u_2 \in E(G)$ in a connected graph G is called a *bridge* if $\{e\}$ is separating, i.e., $G - e$ consists of two connected graphs G_i containing vertex u_i, $i = 1, 2$. For a vertex $v \in V(G)$, the set of neighbors of v in G is denoted by $N_G(v)$, and the *degree* $\deg_G(v)$ of v is defined to be the number of times an edge in $E(G)$ is incident to v; i.e., $\deg_G(v) = |N_G(v)|$.

A vertex designated in a graph G is called a *root*. In this thesis, we designate at most two vertices as roots, and denote by $\mathrm{Rt}(G)$ the set of roots of G. We call a graph G *rooted* (resp., *bi-rooted*) if $|\mathrm{Rt}(G)| = 1$ (resp., $|\mathrm{Rt}(G)| = 2$), where we call G *unrooted* if $\mathrm{Rt}(G) = \emptyset$. For a graph G possibly with roots, a *leaf* is defined to be a non-root vertex $v \in V(G) \setminus \mathrm{Rt}(G)$ with degree 1.

A *rooted tree* is defined to be a tree where a vertex is designated as the root. The *height* $\mathrm{ht}(v)$ of a vertex v in a rooted tree T is defined to be the maximum

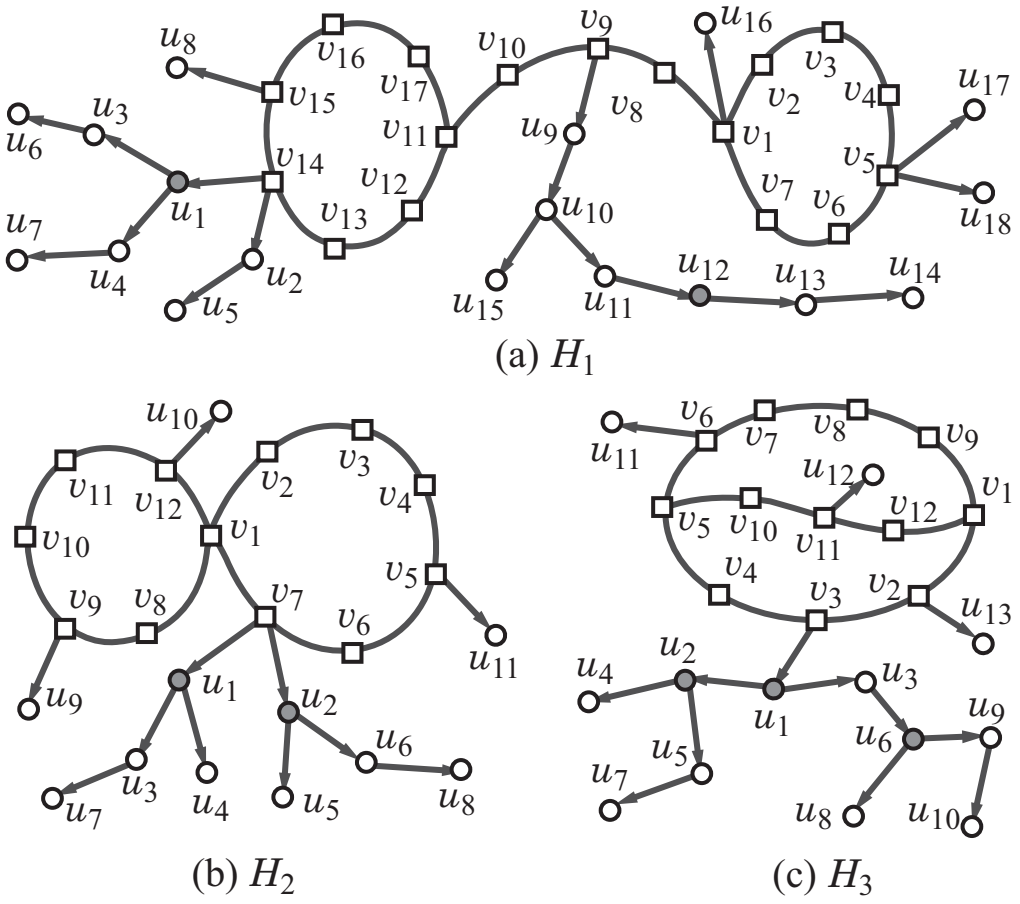


Figure 2.1. An illustration of rank-2 graphs H_i, $i = 1, 2, 3$. Core vertices are illustrated with white squares, non-core vertices with circles, where gray circles depict 2-branches.

length of a path from v to a leaf u, and the height $\mathrm{ht}(T)$ of T is defined to be the height $\mathrm{ht}(r)$ of the root r. For positive integers a, b and c with $b \geq 2$, let $\mathrm{T}(a, b, c)$ denote the rooted tree such that the number of children of the root is a, the number of children of each non-root internal vertex is b, and the distance from the root to each leaf is c. In the rooted tree $\mathrm{T}(a, b, c)$, we denote the vertices by $v_1, v_2, \ldots, v_n$ ($n = a(b^c - 1)/(b-1) + 1$) with a breadth-first-search order, and denote the edge between a vertex v_i with $i \in [2, n]$ and its parent by e_i. For each non-leaf vertex v_i in $\mathrm{T}(a, b, c)$, let $\mathrm{Cld}(i)$ denote the set of indices j such that v_j is a child of v_i when $i \in [1, a(b^{c-1} - 1)/(b-1) + 1]$, and for each non-root vertex v_i in $\mathrm{T}(a, b, c)$, let $\mathrm{prt}(i)$ denote the index j such that v_j is the parent of v_i when $i \in [2, n]$.

Core in Cyclic Graphs Let H be a connected simple graph with rank at least 1. The *core* $\mathrm{Cr}(H)$ of H is defined to be an induced subgraph $\mathrm{Cr}(H) = (V^{\mathrm{co}} = V_1' \cup V_2', E^{\mathrm{co}})$ such that V_1' is the set of vertices in a cycle of H , V_2' is the set of vertices each of which is in a path between two vertices $u, v \in V_1'$, and E^{co} is the set of edges between vertices in V^{co}. A vertex (resp., an edge) in H is called a *core-vertex* (resp., *core-edge*) if it is contained in the core $\mathrm{Cr}(H)$ and is called a *non-core-vertex* (resp., *non-core-edge*) otherwise. The *core size* $\mathrm{cs}(H)$ is defined to be the number of core-vertices in the core of H.

Let $H - E^{\mathrm{co}}$ denote the graph obtained from H by removing all core-edges. We call a connected component T with at least one edge in $H - E^{\mathrm{co}}$ an *exterior-tree*, which contains exactly one core-vertex v in $\mathrm{Cr}(H)$, where T is regarded as a rooted tree rooted at the core-vertex v. The *core height* $\mathrm{ch}(H)$ is defined to be the maximum height $\mathrm{ht}(T)$ of an exterior-tree T of H. Fig. 2.1 illustrates three examples of rank-2 graphs H_i, $i = 1, 2, 3$, where $\mathrm{cs}(H_1) = 17$, $\mathrm{ch}(H_1) = 6$, $\mathrm{cs}(H_2) = 12$, $\mathrm{ch}(H_2) = 3$, $\mathrm{cs}(H_3) = 12$ and $\mathrm{ch}(H_3) = 5$.

Branch-parameter Azam et al. [7] introduced "branch-parameter," a positive integer ρ to measure the "agglomeration degree" of trees.

A non-core vertex v is called a *ρ-internal vertex* (resp., a *ρ-external vertex*) if $\mathrm{ht}(v) \geq \rho$ (resp., $\mathrm{ht}(v) < \rho$). A non-core-edge e is called a *ρ-internal edge* if e is not incident to any ρ-external vertex, and called a *ρ-external edge* otherwise. A ρ-internal vertex v is called a *ρ-branch* if the number of ρ-internal edges incident to v is not 2, and a ρ-branch v is called a *leaf ρ-branch* if $\mathrm{ht}(v) = \rho$ (i.e., the number of ρ-internal edges incident to v is 1).

A *ρ-fringe-tree* is defined to be a maximal subtree T' of an exterior-tree T such that the edge set of T' consists of ρ-external edges. The *ρ-branch-leaf-number* $\mathrm{bl}_\rho(H)$ of H is defined to be the number of leaf ρ-branches in H.

We call a graph H *ρ-lean* if the set of ρ-internal edges in each exterior-tree T forms a single path from its root to a leaf ρ-branch. For $\rho = 2$, nearly 97% of the cyclic chemical compounds with up to 100 non-hydrogen atoms in PubChem [27] are 2-lean. For the graph H_1 in Fig. 2.1(a), u_1 and u_{12} are the leaf 2-branches, and H_1 is 2-lean. For the graph H_2 in Fig. 2.1(b), u_1 and u_2 are the two leaf 2-branches of the exterior tree rooted at vertex v_7. There are two paths from the root vertex v_7 to a leaf 2-branch, v_7 to u_1, and v_7 to u_2, and therefore H_2 is not 2-lean. For the graph H_3 in Fig. 2.1(c), u_1 is a 2-branch that is not a 2-branch leaf. Again, in the exterior tree rooted at vertex v_3 there are two paths from the root to a leaf 2-branch vertex, that are v_3 to u_2 and v_3 to u_6, and we see that H_3 is not 2-lean.

2.2 Modeling of Chemical Compounds

Chemical Graphs We introduce a set of chemical elements such as $\mathtt{H}$ (hydrogen), $\mathtt{C}$ (carbon), $\mathtt{O}$ (oxygen), $\mathtt{N}$ (nitrogen) and so on to represent a chemical compound. To distinguish a chemical element $\mathtt{a}$ with multiple valences such as $\mathtt{S}$ (sulfur), we denote a chemical element $\mathtt{a}$ with a valence i by $\mathtt{a}_{(i)}$, where we do not use such a suffix (i) for a chemical element $\mathtt{a}$ with a unique valence. Let Λ be a set of chemical elements $\mathtt{a}_{(i)}$. For example, $\Lambda = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{P}, \mathtt{S}_{(2)}, \mathtt{S}_{(4)}, \mathtt{S}_{(6)}\}$. Let $\mathrm{val} : \Lambda \to [1, 6]$ be a valence function. For example, $\mathrm{val}(\mathtt{H}) = 1$, $\mathrm{val}(\mathtt{C}) = 4$, $\mathrm{val}(\mathtt{O}) = 2$, $\mathrm{val}(\mathtt{P}) = 5$, $\mathrm{val}(\mathtt{S}_{(2)}) = 2$, $\mathrm{val}(\mathtt{S}_{(4)}) = 4$ and $\mathrm{val}(\mathtt{S}_{(6)}) = 6$. For each chemical element $\mathtt{a} \in \Lambda$, let $\mathrm{mass}(\mathtt{a})$ denote the mass of $\mathtt{a}$.

To represent a chemical compound, a *chemical graph* is defined to be a tuple $\mathbb{C} = (H, \alpha, \beta)$ of a simple, connected undirected graph H and functions $\alpha : V(H) \to \Lambda$ and $\beta : E(H) \to [1, 3]$. The chemical element assigned to a vertex $v \in V(H)$ is represented by $\alpha(v)$ and the bond-multiplicity between two adjacent vertices $u, v \in V(H)$ is represented by $\beta(e)$ of the edge $e = uv \in E(H)$. We say that two tuples $(H_i, \alpha_i, \beta_i), i = 1, 2$ are *isomorphic* if they admit an isomorphism ϕ, i.e., a bijection $\phi : V(H_1) \to V(H_2)$ such that $uv \in E(H_1), \alpha_1(u) = \mathtt{a}, \alpha_1(v) = \mathtt{b}, \beta_1(uv) = m \leftrightarrow \phi(u)\phi(v) \in E(H_2), \alpha_2(\phi(u)) = \mathtt{a}, \alpha_2(\phi(v)) = \mathtt{b}, \beta_2(\phi(u)\phi(v)) = m$. When H_i is rooted at a vertex $r_i, i = 1, 2$, $(H_i, \alpha_i, \beta_i), i = 1, 2$ are *rooted-isomorphic* (r-isomorphic) if they admit an isomorphism ϕ such that $\phi(r_1) = r_2$.

For a notational convenience, we use a function $\beta_{\mathbb{C}} : V(H) \to [0, 12]$ for a chemical graph $\mathbb{C} = (H, \alpha, \beta)$ such that $\beta_{\mathbb{C}}(u)$ means the sum of bond-multiplicities of edges incident to a vertex u; i.e.,

$$\beta_{\mathbb{C}}(u) \triangleq \sum_{uv \in E(H)} \beta(uv)$$

for each vertex $u \in V(H)$. For each vertex $u \in V(H)$, define the *electron-degree* $\mathrm{eledeg}_{\mathbb{C}}(u)$ to be

$$\mathrm{eledeg}_{\mathbb{C}}(u) \triangleq \beta_{\mathbb{C}}(u) - \mathrm{val}(\alpha(u)).$$

For each vertex $u \in V(H)$, let $\deg_{\mathbb{C}}(u)$ denote the number of vertices adjacent to the vertex u in $\mathbb{C}$.

For a chemical graph $\mathbb{C} = (H, \alpha, \beta)$, let $V_{\mathtt{a}}(\mathbb{C}), \mathtt{a} \in \Lambda$ denote the set of vertices $v \in V(H)$ such that $\alpha(v) = \mathtt{a}$ in $\mathbb{C}$ and define the *hydrogen-suppressed chemical graph* $\langle\mathbb{C}\rangle$ to be the graph obtained from H by removing all the vertices $v \in V_{\mathtt{H}}(\mathbb{C})$. We use a hydrogen-suppressed model because hydrogen atoms can be added at the final stage.

We introduce a total order $<$ over the elements in Λ and we write $\mathtt{a} < \mathtt{b}$ for chemical elements $\mathtt{a}, \mathtt{b} \in \Lambda$. Choose a set $\Gamma_<$ of tuples $\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Lambda \times \Lambda \times [1, 3]$

such that $\mathtt{a} < \mathtt{b}$. For a tuple $\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Lambda \times \Lambda \times [1,3]$, let $\overline{\gamma}$ denote the tuple $(\mathtt{b}, \mathtt{a}, k)$. Set $\Gamma_> = \{\overline{\gamma} \mid \gamma \in \Gamma_<\}$, $\Gamma_= = \{(\mathtt{a}, \mathtt{a}, k) \mid \mathtt{a} \in \Lambda, k \in [1,3]\}$ and $\Gamma = \Gamma_< \cup \Gamma_=$. A pair of two atoms $\mathtt{a}$ and $\mathtt{b}$ joined with a bond of multiplicity k is denoted by a tuple $\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Gamma$, called the *adjacency-configuration* of the atom pair.

2.3 Machine Learning

In most of existing QSAR studies, the problem of QSAR is usually formulated as a machine learning problem, particularly, a *regression problem* between a set of vectors of real numbers called *descriptors* or *features* that represents the chemical compounds and the corresponding real property values [16].

For an integer $K \geq 1$, define a feature space $\mathbb{R}^K$. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ be a set of feature vectors $x_i \in \mathbb{R}^K$ and let $a_i \in \mathbb{R}$ be a real assigned to a feature vector x_i. Let $A = \{a_i \mid i \in [1,m]\}$. A function $\eta : \mathbb{R}^K \to \mathbb{R}$ is called a *prediction function*.

A *regression problem* is asked to find a prediction function $\eta : \mathbb{R}^K \to \mathbb{R}$ based on a subset of $\{x_1, x_2, \ldots, x_m\}$ so that $\eta(x_i)$ is closed to the value a_i for many indices $i \in [1,m]$.

Typically, for a prediction function $\eta : \mathbb{R}^K \to \mathbb{R}$, we define an error function:

$$\mathrm{Err}(\eta; \mathcal{X}) \triangleq \sum_{i \in [1,m]} (a_i - \eta(x_i))^2$$

and define the *coefficient of determination*:

$$\mathrm{R}^2(\eta, \mathcal{X}) \triangleq 1 - \frac{\mathrm{Err}(\eta; \mathcal{X})}{\sum_{i \in [1,m]} (a_i - \widetilde{a})^2} = 1 - \frac{\sum_{i \in [1,m]} (a_i - \eta(f(x_i)))^2}{\sum_{i \in [1,m]} (a_i - \widetilde{a})^2}, \text{ for } \widetilde{a} = \frac{1}{m} \sum_{i \in [1,m]} a_i.$$

The coefficient of determination $\mathrm{R}^2(\eta, \mathcal{X})$ is usually used to evaluate the performance of the prediction function η on the dataset $\mathcal{X}$.

Artificial Neural Networks A digraph $G = (V, E)$ is called *layered* if it does not contain any directed cycle and the length of any path from a source $s \in V_{\mathrm{in}}$ to a sink $t \in V_{\mathrm{out}}$ is a constant, say k, where V is partitioned into $k+1$ disjoints subsets $V_0 (= V_{\mathrm{in}}), V_1, V_2, \ldots, V_k (= V_{\mathrm{out}})$ so that each edge (u, v) satisfies $u \in V_i$ and $v \in V_{i+1}$ for some $i \in [0, k-1]$. For each vertex $v \in V$, a vertex $u \in V$ with $(u, v) \in E$ is called an *in-neighbor* of v, and we let $N^-(v)$ denote the set of in-neighbors of v.

Artificial neural network (ANN) is one of the most frequently used tools in machine learning. Following [1], we consider an ANN $\mathcal{N}$ to be a weight system

(G, w, F) such that G is a layered digraph, where a function $f_v \in F$ for each vertex $v \in V \backslash (V_{\text{in}} \cup V_{\text{out}})$ is called an *activation function*. One example of the activation function is the *ReLU function*, defined to be $f(x) := \max(x, 0)$. The weight function w is a function $w : V \cup E \to \mathbb{R}$ on the digraph G, where we call $w(uv)$ the *weight* on directed edge $(u, v) \in E$ and $w(v)$ the *weight* on vertex $v \in V$.

Given a vector $(y_s)_{s \in V_{\text{in}}}$ of reals, we calculate the values y_v for $v \in V \backslash V_{\text{in}}$ as follows:

$$y_v = f_v\Big(\sum_{u \in N^-(v)} w(uv) y_u + w(v)\Big).$$

The vector $(y_t)_{t \in V_{\text{out}}}$ of reals is the output of this ANN.

3 A Framework for the Inverse QSAR

This chapter reviews the framework that solves the inverse QSAR by using MILPs [4]. We will follow this framework in Chapters 4 and 5, and make a slight necessary modification to include other machine learning methods in Chapters 6, 7 and 8.

3.1 A Framework for the Inverse QSAR

For a specified chemical property π such as boiling point, we denote by $a(G)$ the observed value of the property π for a chemical compound G.

3.1.1 Phase 1

As the first phase, we solve (I) Prediction Problem with the following three stages.

Stage 1: Let DB be a set of chemical graphs. For a specified chemical property π, choose a class $\mathcal{G}$ of graphs such as acyclic graphs or monocyclic graphs. Prepare a data set $D_\pi = \{G_i \mid i = 1, 2, \ldots, m\} \subseteq \mathcal{G} \cap \mathrm{DB}$ such that the value $a(G_i)$ of each chemical graph G_i, $i = 1, 2, \ldots, m$ is available. See Figure 3.1 for an illustration of Stage 1.

Stage 2: Introduce a feature function $f : \mathcal{G} \to \mathbb{R}^k$ for a positive integer k. We call $f(G)$ the *feature vector* of $G \in \mathcal{G}$, and call each entry of a vector $f(G)$ a *descriptor* of G. A vector $x \in \mathbb{R}^k$ is called *admissible* if there is a graph $G \in \mathcal{G}$ such that $f(G) = x$ [4]. Let $\mathcal{A}$ denote the set of admissible vectors $x \in \mathbb{R}^k$. We use the range-based method to define an applicability domain (AD) [35] to our inverse QSAR method. Set $\underline{x_j}$ and $\overline{x_j}$ to be the minimum and maximum values of the j-th descriptor x_j in $f(G_i)$ over all graphs G_i, $i = 1, 2, \ldots, m$. Define our AD $\mathcal{D}$ to be the set of vectors $x \in \mathbb{R}^k$ such that $\underline{x_j} \leq x_j \leq \overline{x_j}$ for the variable x_j of each j-th descriptor, $j = 1, 2, \ldots, k$. See Figure 3.2 for an illustration of Stage 2.

Stage 3: Construct a prediction function $\psi_{\mathcal{N}}$ with an ANN $\mathcal{N}$ that, given a vector in $\mathbb{R}^k$, returns a real so that $\psi_{\mathcal{N}}(f(G))$ takes a value nearly equal to $a(G)$

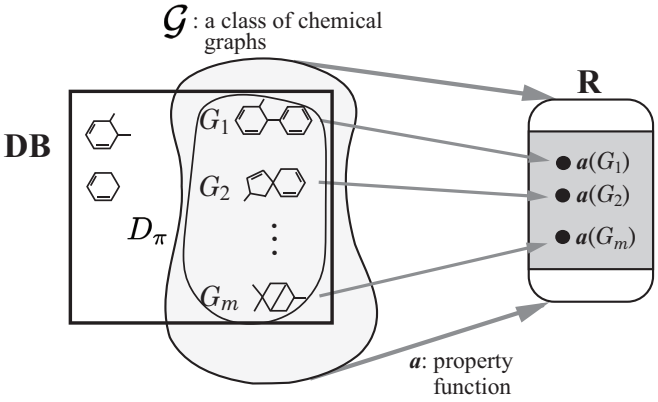


Figure 3.1. An illustration of Stage 1: A data set D_π of chemical graphs G_i, $i = 1, 2, \ldots, m$ in a class $\mathcal{G}$ of graphs whose values $a(G_i)$ of a chemical property π are available.

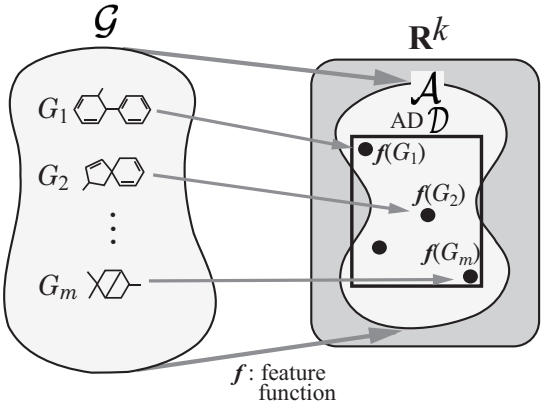


Figure 3.2. An illustration of Stage 2: Each chemical graph $G \in \mathcal{G}$ is mapped to a vector $f(G)$ in a feature vector space $\mathbb{R}^k$ for some positive integer k.

for many chemical graphs in D. See Figure 3.3 for an illustration of Stage 3.

3.1.2 Phase 2

As the second phase, we solve (II) INVERSE PROBLEM for the inverse QSAR by treating the following inference problems.

(II-a) Inference of Vectors
Input: A real $y^* \in [\underline{a}, \overline{a}]$.
Output: Vectors $x^* \in \mathcal{A} \cap \mathcal{D}$ and $g^* \in \mathbb{R}^h$ such that $\psi_{\mathcal{N}}(x^*) = y^*$ and g^* forms

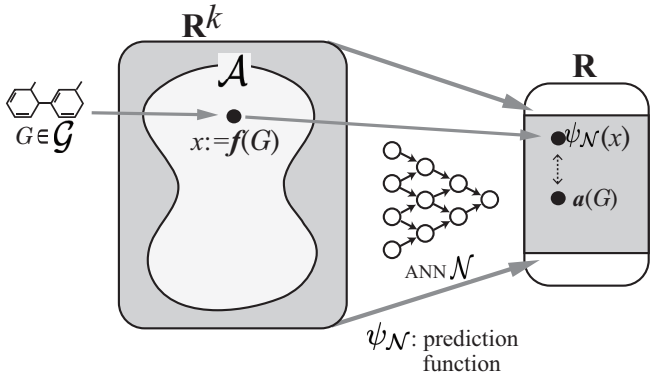


Figure 3.3. An illustration of Stage 3: A prediction function $\psi_{\mathcal{N}}$ from the feature vector space $\mathbb{R}^k$ to $\mathbb{R}$ is constructed based on an ANN $\mathcal{N}$.

a chemical graph $G^* \in \mathcal{G}$ with $f(G^*) = x^*$.

(II-b) Inference of Graphs
Input: A vector $x^* \in \mathcal{A} \cap \mathcal{D}$.
Output: All graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$.

To treat Problem (II-a), we use MILPs for inferring vectors in ANNs [1]. In MILPs, we can easily impose additional linear constraints or fix some variables to specified constants. We include into the MILP a linear constraint such that $x \in \mathcal{D}$ to obtain the next result.

Theorem 3.1 ([1]). *Let $\mathcal{N}$ be an ANN with a piecewise-linear activation function for an input vector $x \in \mathbb{R}^k$, n_A denote the number of nodes in the architecture and n_B denote the total number of break-points over all activation functions. Then there is an MILP $\mathcal{M}(x, y; \mathcal{C}_1)$ that consists of variable vectors $x \in \mathcal{D}$ ($\subseteq \mathbb{R}^k$), $y \in \mathbb{R}$, and an auxiliary variable vector $z \in \mathbb{R}^p$ for some integer $p = O(n_A + n_B)$ and a set $\mathcal{C}_1$ of $O(n_A + n_B)$ constraints on these variables such that: $\psi_{\mathcal{N}}(x^*) = y^*$ if and only if there is a vector (x^*, y^*) feasible to $\mathcal{M}(x, y; \mathcal{C}_1)$.*

To attain the admissibility of inferred vector x^*, we also introduce a variable vector $g \in \mathbb{R}^q$ for some integer q and a set $\mathcal{C}_2$ of constraints on x and g such that $x^* \in \mathcal{A}$ holds in the following sense: (x^*, g^*) is feasible to the MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ if and only if g^* forms a chemical graph $G^* \in \mathcal{G}$ with $f(G^*) = x^*$.

The second phase consists of the next two stages.
Stage 4: Formulate Problem (II-a) as the above MILP $\mathcal{M}(x, y, g; \mathcal{C}_1, \mathcal{C}_2)$ based on $\mathcal{G}$ and $\mathcal{N}$. Find a set F^* of vectors $x^* \in \mathcal{A} \cap \mathcal{D}$ such that $(1-\varepsilon)y^* \leq \psi_{\mathcal{N}}(x^*) \leq (1+\varepsilon)y^*$ for a tolerance ε set to be a small positive real. See Figure 3.4 for an illustration of Stage 4.

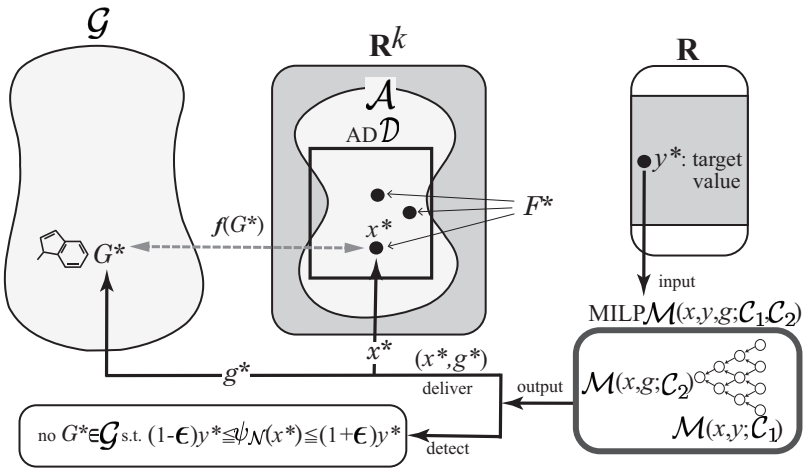


Figure 3.4. An illustration of Stage 4: Given a target value $y^* \in \mathbb{R}$, solving MILP $\mathcal{M}(x, y, g; \mathcal{C}_1, \mathcal{C}_2)$ either delivers a set F^* of vectors $x^* \in \mathcal{A} \cap \mathcal{D}$ such that $(1 - \varepsilon)y^* \leq \psi_{\mathcal{N}}(x^*) \leq (1 + \varepsilon)y^*$ or detects that no such vector x exists.

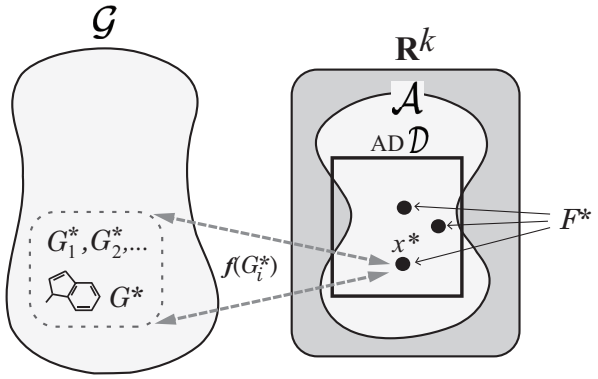


Figure 3.5. An illustration of Stage 5: For each vector $x^* \in F^*$, all chemical graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$ are generated.

Stage 5: To solve Problem (II-b), enumerate all graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$ for each vector $x^* \in F^*$. See Figure 3.5 for an illustration of Stage 5.

4 An Inverse QSAR Method for Rank-2 Chemical Compounds

4.1 Introduction

Recently, a new inverse QSAR framework has been proposed [4, 11, 59, 60] by combining two previous approaches; efficient enumeration of tree-like graphs [15], and MILP-based formulation of the inverse problem on ANNs [1]. Their methods were applicable only to acyclic chemical graphs (i.e., tree-structured chemical graphs), where the ratio of acyclic chemical graphs in a major chemical database, PubChem [27], is only 2.91%. Afterward, Ito et al. [25] designed a method of inferring monocyclic chemical graphs (chemical graphs with rank 1) by formulating a new MILP and using an efficient algorithm for enumerating monocyclic chemical graphs [46]. This still leaves a big limitation because the ratio of acyclic and monocyclic chemical graphs in PubChem [27] is only 16.26%.

To break this limitation, we significantly extend the MILP-based approach for inverse QSAR so that "rank-2 chemical compounds" can be efficiently handled, where the ratio of chemical graphs with rank at most 2 in the database PubChem is 44.5%. Note that there are three different topological structures, called *polymer-topologies* over all rank-2 chemical compounds. In particular, we propose a novel MILP formulation for (II-a) (see Section 3.1.2) along with a new set of descriptors. One big advantage of this new formulation is that an MILP instance has a solution if and only if there exists a rank-2 chemical graph satisfying given constraints, which is useful to significantly reduce redundant search in (II-b). We conducted computational experiments to infer rank-2 chemical compounds on several chemical properties.

The rest of this chapter is organized as follows. Section 4.2 introduces some notions on graphs, a modeling of chemical compounds, and a choice of descriptors. Section 4.3 introduces a method of modeling rank-2 chemical graphs with different cyclic structures in a unified way and proposes an MILP formulation that represents a rank-2 chemical graph G of n vertices, where our MILP requires only $O(n)$ variables and constraints when the maximum height of subtrees in G is constant. Section 4.4 reports the results on some computational experiments conducted for chemical properties such as octanol/water partition coefficient,

melting point, and boiling point. Section 4.5 makes some concluding remarks.

4.2 Preliminary

This section introduces some notions and terminology on graphs, a modeling of chemical compounds, and our choice of descriptors.

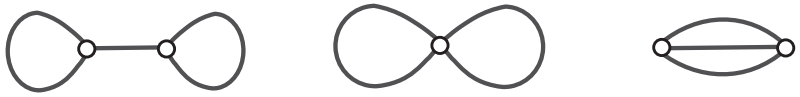
4.2.1 Multigraphs and Graphs

Multigraphs A *multigraph* is defined to be a pair (V, E) of a vertex set V and an edge set E such that each edge $e \in E$ joins two vertices $u, v \in V$ (possibly $u = v$) and the vertices u and v are called the *end-vertices* of the edge e, and let $V(e)$ denote the set of the end-vertices of an edge $e \in E$, where an edge e with $|V(e)| = 1$ is called a *loop*. We denote the vertex and edge sets of a multigraph M by $V(M)$ and $E(M)$, respectively. A path with end-vertices u and v is called a *u, v-path*, and the length of a path is defined to be the number of edges in the path.

Let M be a multigraph. An edge $e \in E(M)$ is called *multiple* (to an edge $e' \in E(M)$) if there is another edge $e' \in E(M)$ with $V(e) = V(e')$. For a vertex $v \in V(M)$, the *degree* $\deg_M(v)$ of v is defined to be the number of times an edge in $E(M)$ is incident to v; i.e., $\deg_M(v) = |\{e \in E(M) \mid v \in V(e), |V(e)| = 2\}| + 2|\{e \in E(M) \mid v \in V(e), |V(e)| = 1\}|$. A multigraph is called *simple* if it has no loop and there is at most one edge between any two vertices. We observe that the sum of the degrees over all vertices is twice the number of edges in any multigraph M; i.e.,

$$2|E(M)| = \sum_{v \in V(M)} \deg_M(v).$$

For a subset X of vertices in M, let $M - X$ denote the multigraph obtained from M by removing the vertices in X and any edge incident to a vertex in X. An operation of *subdividing* a non-loop edge (resp., loop) $e \in E(M)$ with $V(e) = \{v_1, v_2\}$ (resp., $V(e) = \{v_1 = v_2\}$) is to replace e with two new edges e_1 and e_2 such that each e_i is incident to v_i and a new vertex v_e. An operation of *contracting* a vertex u of degree 2 in M is to replace the two edges uv and uv' incident to u with a single edge vv' removing vertex u, where the resulting edge is a loop when $v = v'$. The *rank* $\mathrm{r}(M)$ of a multigraph M is defined to be the minimum number of edges to be removed to make the multigraph acyclic. We call a multigraph M with $\mathrm{r}(M) = k$ a *rank-k graph*. Let $V_{\deg,i}(M)$ denote the set of vertices of degree i in M. The *core* $\mathrm{Cr}(M)$ of M is defined to be an induced



(a) M_1 (b) M_2 (c) M_3

Figure 4.1. An illustration of the three rank-2 polymer topologies $M_1, M_2, M_3 \in \mathcal{PT}(2,4)$.

subgraph M^* that is obtained from $M' := M$ by setting $M' := M' - V_{\deg,1}(M')$ repeatedly until M^* contains at most two vertices or consists of vertices of degree at least 2. The core M^* of a connected multigraph M consists of a single vertex (resp., two vertices) if and only if M is a tree with an even (resp., odd) diameter. A vertex (resp., an edge) in M is called a *core vertex* (resp., *core edge*) if it is contained in the core of M and is called a *non-core vertex* (resp., *non-core edge*) otherwise. The *core size* $\mathrm{cs}(M)$ is defined to be the number of core vertices of M, and the *core height* $\mathrm{ch}(M)$ is defined to be the maximum length of a path between a vertex $v \in V(M^*)$ to a leaf of M without passing through any core edge. The set of non-core edges induces a collection of subtrees, each of which we call a *non-core component* of M, where each non-core component C contains exactly one core vertex v_C and we regard C as a tree rooted at v_C. Let C be a non-core component of M. The *height* $\mathrm{height}(v)$ of a vertex v in C is defined to be the maximum length of a path from v to a leaf u in the descendants of v.

A multigraph is called a *polymer topology* if it is connected and the degree of every vertex is at least 3. Tezuka and Oike [51] pointed out that a classification of polymer topologies will lay a foundation for elucidation of structural relationships between different macro-chemical molecules and their synthetic pathways. For integers $r \geq 0$ and $d \geq 3$, let $\mathcal{PT}(r,d)$ denote the set of all rank-r polymer topologies with maximum degree at most d. Figure 4.1 illustrates the three rank-2 polymer topologies in $\mathcal{PT}(2,4)$.

For a polymer topology M, the *least simple graph* $S(M)$ of M is defined to be a simple graph obtained from M by subdividing each loop in M with two new vertices of degree 2 and subdividing all multiple edges (except for one) between every two adjacent vertices in M.

The *polymer topology* $\mathrm{Pt}(M)$ of a multigraph M with $\mathrm{r}(M) \geq 2$ is defined to be a multigraph M' of degree at least 3 that is obtained from the core $\mathrm{Cr}(M)$ by contracting all vertices of degree 2. Note that $\mathrm{r}(\mathrm{Pt}(M)) = \mathrm{r}(M)$. Figures 4.2(a) to (c) illustrate the least simple graph $S(M)$ of each polymer topology $M \in \mathcal{PT}(2,4)$ in Figure 4.1, where Figure 4.2(d) illustrates a graph that contains all least simple

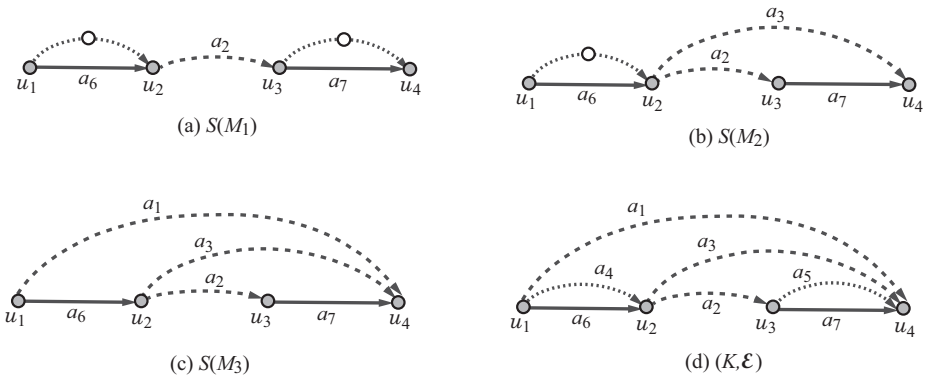


Figure 4.2. An illustration of the least simple graphs of the rank-2 polymer topologies $M_1, M_2, M_3 \in \mathcal{PT}(2,4)$ in Figure 4.1 and a scheme graph $(K, \mathcal{E})$: (**a**) $S(M_1)$; (**b**) $S(M_2)$; (**c**) $S(M_3)$; (**d**) a scheme graph $(K = (\{u_1, u_2, u_3, u_4\}, E), \mathcal{E} = (E_1, E_2, E_3))$ where each edge $u_i u_j$ is directed from one end-vertex u_i to the other end-vertex u_j with $i < j$, and $E_1 = \{a_1 = (u_1, u_4), a_2 = (u_2, u_3), a_3 = (u_2, u_4)\}$, $E_2 = \{a_4 = (u_1, u_2), a_5 = (u_3, u_4)\}$ and $E_3 = \{a_6 = (u_1, u_2), a_7 = (u_3, u_4)\}$, and the edges in E_1 (resp., E_2 and E_3) are depicted with dashed (resp., dotted and solid) lines.

graphs.

Graphs Let $H = (V, E)$ be a graph with a set V of vertices and a set E of edges. Define the *1-path connectivity* $\kappa_1(H)$ of H to be $\sum_{uv \in E} 1/\sqrt{\deg_H(u)\deg_H(v)}$.

Let H be a rank-2 connected graph such that the maximum degree is at most 4. We see that H contains two vertices v_a and v_b such that either there are three disjoint paths between v_a and v_b or H contains two edge disjoint cycles C and C', which are joined with a path between v_a and v_b (possibly $v_a = v_b$). We introduce the *topological parameter* $\theta(H)$ of rank-2 connected graph H as follows. When H has three disjoint paths between v_a and v_b, define $\theta(H)$ to be the minimum number of edges along a path between v_a and v_b. When H contains two edge disjoint cycles C and C', which are joined with a path P between v_a and v_b (possibly $v_a = v_b$), define $\theta(H)$ to be $-|E(P)|$.

4.2.2 Modeling of Chemical Compounds

Chemical Graphs In this chapter, a *chemical graph* over a set of chemical elements Λ and a set of adjacency-configurations Γ is defined to be a tuple $G = (H, \alpha, \beta)$ of a graph $H = (V, E)$, a function $\alpha : V \to \Lambda$ and a function $\beta : E \to [1, 3]$ such that

(i) H is connected;

(ii) $\sum_{uv\in E}\beta(uv) \leq \mathrm{val}(\alpha(u))$ for each vertex $u \in V$; and

(iii) $(\alpha(u), \alpha(v), \beta(uv)) \in \Gamma$ for each edge $uv \in E$.

Let $\mathcal{G}(\Lambda, \Gamma)$ denote the set of chemical graphs over Λ and Γ.

Descriptors In our method, we use only graph-theoretical descriptors for defining a feature vector, which facilitates our designing an algorithm for constructing graphs. Given a chemical graph $G = (H, \alpha, \beta)$, we define a *feature vector* $f(G)$ that consists of the following 14 kinds of descriptors:

- $n(G)$: the number of vertices in G;
- $\mathrm{cs}(G)$: the core size of G;
- $\mathrm{ch}(G)$: the core height of G;
- $\kappa_1(G)$: the 1-path connectivity of G;
- $\mathrm{dg}_i(G)$ ($i \in [1, 4]$): the number of vertices of degree i in G;
- $\mathrm{ce}^{\mathrm{co}}_{\mathtt{a}}(G)$ ($\mathtt{a} \in \Lambda$): the number of core vertices with chemical element $\mathtt{a} \in \Lambda$;
- $\mathrm{ce}^{\mathrm{nc}}_{\mathtt{a}}(G)$ ($\mathtt{a} \in \Lambda$): the number of non-core vertices with chemical element $\mathtt{a} \in \Lambda$;
- $\overline{\mathrm{ms}}(G)$: the average of mass* of atoms in G;
- $b^{\mathrm{co}}_k(G)$ ($k \in [2, 3]$): the number of double and triple bonds in core edges;
- $b^{\mathrm{nc}}_k(G)$ ($k \in [2, 3]$): the number of double and triple bonds in non-core edges;
- $\mathrm{ac}^{\mathrm{co}}_{\gamma}(G)$ ($\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Gamma$): the number of adjacency-configurations $(\mathtt{a}, \mathtt{b}, k)$ of core edges;
- $\mathrm{ac}^{\mathrm{nc}}_{\gamma}(G)$ ($\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Gamma$): the number of adjacency-configurations $(\mathtt{a}, \mathtt{b}, k)$ of non-core edges;
- $\theta(H)$: the topological parameter of H; and
- $n_{\mathtt{H}}(G)$: the number of hydrogen atoms to be included in G; i.e.,
  $n_{\mathtt{H}}(G) \triangleq \sum_{\mathtt{a}\in\Lambda} \mathrm{val}(\mathtt{a}) n_{\mathtt{a}}(G)$
  $= \sum_{\mathtt{a}\in\Lambda} \mathrm{val}(\mathtt{a}) n_{\mathtt{a}}(G) - 2(n(G) + 1 + b^{\mathrm{co}}_2(G) + b^{\mathrm{nc}}_2(G) + 2b^{\mathrm{co}}_3(G) + 2b^{\mathrm{nc}}_3(G))$.

The number k of descriptors in our feature vector $x = f(G)$ is $k = 2|\Lambda| + 2|\Gamma| + 13$.

4.3 Representing Rank-2 Chemical Graphs

This section introduces a method of modeling rank-2 chemical graphs with different cyclic structures in a unified way and proposes an MILP formulation that represents a rank-2 chemical graph G of n vertices.

Scheme Graphs and Tree-Extensions

Given positive integers n^* and p, a graph with n^* vertices and p edges can be represented as a subgraph of a complete graph K_{n^*} with $n^*(n^*-1)/2$ edges. However, formulating this as an MILP may require to prepare $\Omega((n^*)^2)$ variables and constraints. To reduce the number of variables and constraints in an MILP that represents a rank-2 graph, we decompose a rank-2 graph G into the core and non-core of G so that the core is represented by one of the three rank-2 polymer topologies and the non-core is a collection of trees in which the height is bounded by the core height of G. We do not specify how many subtrees will be attached to each edge in the polymer topology in advance, since otherwise we would need a different MILP for a distinct combination of such assignments of subtrees. Instead we allow each edge in a polymer topology to collect a necessary number of subtrees in our MILP (see the next section for more detail). In this section, we introduce a "scheme graph" to represent three possible rank-2 polymer topologies, an "extension" of the scheme graph to represent the core of a rank-2 graph and a "tree-extension" to represent a combination of the core and non-core of a rank-2 graph, so that any of the three kinds of rank-2 polymer topologies can be selected in a single MILP formulation.

Scheme Graphs Formally, we define the *scheme graph* for rank 2 to be a pair $(K, \mathcal{E})$ of a multigraph K and an ordered partition $\mathcal{E} = (E_1, E_2, E_3)$ of the edge set $E(K)$. Figure 4.2(d) illustrates the scheme graph $(K = (\{u_1, u_2, u_3, u_4\}, E), \mathcal{E} = (E_1, E_2, E_3))$. An edge in E_1 is called a *semi-edge*, an edge in E_2 is called a *virtual edge* and an edge in E_3 is called a *real edge*.

Extensions of Scheme Graphs Based on the scheme graph $(K, \mathcal{E})$, we construct the core of a rank-2 graph H as an "extension," which is defined as follows. An *extension* of the scheme graph $(K, \mathcal{E})$ is defined to be a simple graph obtained from K by using each real edge $e = uv \in E_3$, by eliminating or replacing each virtual edge $e = uv \in E_2$ (resp., semi-edge $e = uv \in E_1$) with a u, v-path of length at least two (resp., 1) in the core of H, where a u, v-path of length 1 means an edge uv. Figure 4.4(a) illustrates an extension H_{core} of the scheme graph $(K, \mathcal{E})$ which is obtained by removing virtual edges $a_4, a_5 \in E_2$ and by replacing semi-edge $a_1 \in E_1$ with a path $(u_{1,1}, v_{1,1}, v_{2,1}, u_{4,1})$, semi-edge $a_2 \in E_1$

with a path $(u_{2,1}, v_{3,1}, v_{4,1}, v_{5,1}, u_{3,1})$ and by using semi-edge $a_3 \in E_1$ and real edges $a_6, a_7 \in E_3$. The extension $H_{\rm core}$ in Figure 4.4(a) is isomorphic to the core of the rank-2 graph H in Figure 4.4(b). Observe that each of the least simple graphs $S(M_i)$, $i = 1, 2, 3$ in Figure 4.2 is obtained as an extension of the scheme graph $(K, \mathcal{E})$ in Figure 4.2(d).

Tree-extensions Let $s^* = |V(K)| = 4$ denote the number of vertices in the scheme graph. For non-negative integers a, b and c, we consider a rank-2 graph H such that $\mathrm{cs}(H) = s^* + a = 4 + a$, $\mathrm{ch}(H) = b$ and the maximum degree of a core vertex is at most c. We define an "(a, b, c)-tree-extension" as a minimal supergraph of all such rank-2 graphs H. Formally, the *(a, b, c)-tree-extension* (or a *tree-extension*) is defined to be the graph obtained by augmenting the graph K as follows:

(i) For each vertex $u_s \in V(K)$, $s \in [1, s^*]$, create a copy S_s of the rooted tree $T(c-2, c-1, b)$. For each $s \in [1, s^*]$, let the root of rooted tree S_s be equal to the vertex u_s and denote by $u_{s,i}$ the copy of the i-th vertex of $T(c-2, c-1, b)$ in S_s (see Figure 4.3(a)).

(ii) Create a new path $(v_{1,1}, v_{2,1}, \ldots, v_{a,1})$ with a vertices, where the edge between $v_{t,1}$ and $v_{t+1,1}$ is denoted by e_{t+1} (see Figure 4.3(c)). For each $t \in [1, a]$, create a copy T_t of the rooted tree $T(c-2, c-1, b)$, let the root of rooted tree T_t be equal to the vertex $v_{1,1}$ and denote by $v_{t,i}$ the copy of the i-th vertex of $T(c-2, c-1, b)$ in T_t (see Figure 4.3(b)).

(iii) For every pair (s, t) with $s \in [1, s^*]$ and $t \in [1, a]$, join vertices $u_{s,1}$ and $v_{t,1}$ with an edge $u_{s,1}v_{t,1}$ (see Figure 4.3(c)).

Figure 4.3 illustrates the $(3, 2, 4)$-tree-extension of the scheme graph. We show how a rank-2 graph can be constructed as a subgraph of a tree-extension with some example. Figure 4.4(b) illustrates a rank-2 graph H with $n(H) = 21$, $\mathrm{cs}(H) = 9$, $\mathrm{ch}(H) = 2$ and $\theta(H) = 1$, where the maximum degree of a non-core vertex is 3. To prepare a tree-extension so that the graph H can be a subgraph of the tree-extension, we set $\mathrm{cs}^* := \mathrm{cs}(H)$, $a := t^* := \mathrm{cs}^* - s^* = 5$, $b := \mathrm{ch}^* := \mathrm{ch}(H) = 2$ and $c := d_{\max} := 3$. Figure 4.4(c) illustrates a subgraph H' of the $(t^* = 5, \mathrm{ch}^* = 2, d_{\max} = 3)$-tree-extension such that H' is isomorphic to the rank-2 graph H.

MILPs for Rank-2 Chemical Graphs

We present an outline of our MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ in Stage 4 of the framework. For integers $d_{\max}, n^*, \mathrm{cs}^*, \mathrm{ch}^*, \theta^* \in \mathbb{Z}$, let $\mathcal{H}(d_{\max}, n^*, \mathrm{cs}^*, \mathrm{ch}^*, \theta^*)$ denote the set of

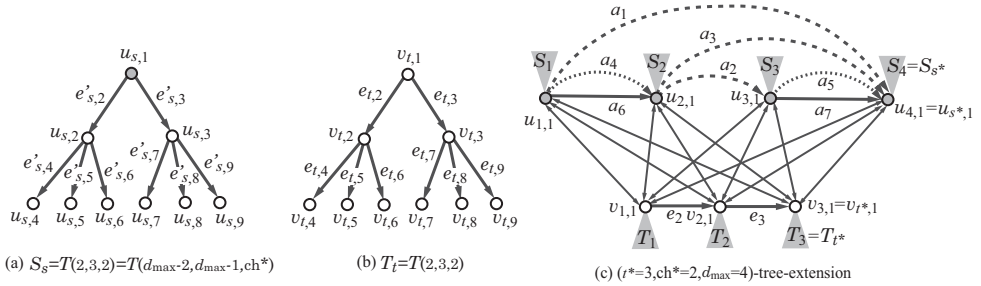


Figure 4.3. An illustration of a tree-extension, where the vertices in $V(K)$ are depicted with gray circles: (**a**) The structure of the rooted tree S_s rooted at a vertex $u_{s,1}$; (**b**) the structure of the rooted tree T_t rooted at a vertex $v_{t,1}$; (**c**) the (a, b, c)-tree-extension of the scheme graph in Figure 4.2(d) for $a = t^* = 3$, $b = \mathrm{ch}^* = 2$ and $c = d_{\max} = 4$.

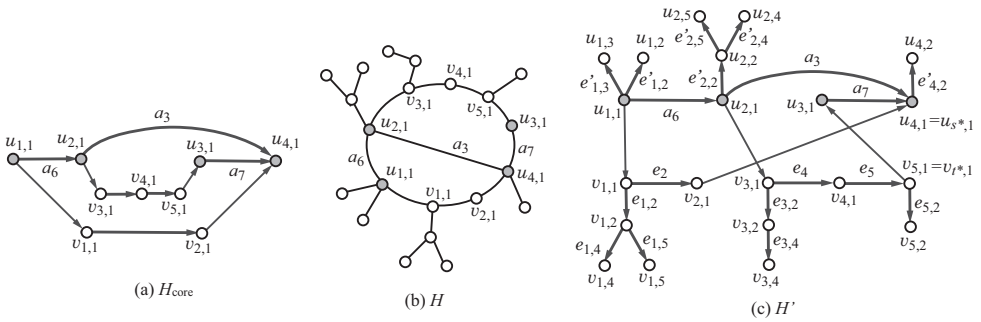


Figure 4.4. (**a**) An example of an extension of the scheme graph; (**b**) an example of a rank-2 graph H with $n(H) = 21$, $\mathrm{cs}(H) = 9$, $\mathrm{ch}(H) = 2$ and $\theta(H) = 1$, where the labels of some vertices and edges indicate the corresponding vertices and edges in the $(t^*, \mathrm{ch}^*, d_{\max})$-tree-extension for $\mathrm{cs}^* = \mathrm{cs}(H)$, $\mathrm{ch}^* = \mathrm{ch}(H)$, $s^* = 4$, $t^* = \mathrm{cs}^* - s^*$ and $d_{\max} = 3$; (**c**) a subgraph H' of $(t^* = 5, \mathrm{ch}^* = 2, d_{\max} = 3)$-tree-extension isomorphic to the rank-2 graph H in (**b**).

rank-2 graphs H such that the degree of each core vertex is at most 4, the degree of each non-core vertex is at most $d_{\max}$, $n(H) = n^*$, $\mathrm{cs}(H) = \mathrm{cs}^*$, $\mathrm{ch}(H) = \mathrm{ch}^*$ and $\theta(H) = \theta^*$. In this chapter, we obtain the following result.

Theorem 4.2. *Let Λ be a set of chemical elements, Γ be a set of adjacency-configurations, where $|\Lambda| \leq |\Gamma|$, and $k = 2|\Lambda| + 2|\Gamma| + 13$. Given integers $d_{\max} \in \{3, 4\}$, $n^* \geq 3$, $\mathrm{cs}^* \geq 3$ $\mathrm{ch}^* \geq 0$ and θ^*, there is an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ that consists of variable vectors $x \in \mathbb{R}^k$ and $g \in \mathbb{R}^q$ for some integer $q = O(|\Gamma| \cdot \mathrm{cs}^* \cdot (d_{\max}-1)^{\mathrm{ch}^*})$ and a set $\mathcal{C}_2$ of $O(|\Gamma| + \mathrm{cs}^* \cdot (d_{\max}-1)^{\mathrm{ch}^*})$ constraints on these variables such that: (x^*, g^*) is feasible to $\mathcal{M}(x, g; \mathcal{C}_2)$ if and only if g^* forms a rank-2 chemical graph*

$G^* = (H, \alpha, \beta) \in \mathcal{G}(\Lambda, \Gamma)$ *such that* $H \in \mathcal{H}(d_{\max}, n^*, \mathrm{cs}^*, \mathrm{ch}^*, \theta^*)$ *and* $f(G^*) = x^*$.

Note that our MILP requires only $O(n^*)$ variables and constraints when the maximum core height of a subtree in the non-core of G^* and $|\Gamma|$ are constant. We formulate an MILP in Theorem 4.2 so that such a graph H is selected as a subgraph of the scheme graph.

We explain the basic idea of our MILP. Define

$t^* \triangleq \mathrm{cs}^* - s^*$,

$c^* \triangleq |E_1 \cup E_2|$ for $(K, \mathcal{E} = (E_1, E_2, E_3))$,

$n_{\mathrm{tree}} \triangleq 1+2((d_{\max}-1)^{\mathrm{ch}^*}-1)/(d_{\max}-2)$ and $n_{\mathrm{in}} \triangleq 1+2((d_{\max}-1)^{\mathrm{ch}^*-1}-1)/(d_{\max}-2)$,

where n_{tree} and n_{in} are the numbers of vertices and non-leaf vertices in the rooted tree $T(d_{\max}-2, d_{\max}-1, \mathrm{ch}^*)$, respectively. The MILP mainly consists of the following three types of constraints.

1. Constraints for selecting a rank-2 graph H as a subgraph of the $(t^*, \mathrm{ch}^*, d_{\max})$-tree-extension of the scheme graph $(K, \mathcal{E})$;
2. Constraints for assigning chemical elements to vertices and multiplicity to edges to determine a chemical graph $G = (H, \alpha, \beta)$;
3. Constraints for computing descriptors from the selected rank-2 chemical graph G; and
4. Constraints for reducing the number of rank-2 chemical graphs that are isomorphic to each other but can be represented by the above constraints.

In the constraints of 1, we treat each edge in the tree-extension as a directed edge because describing some condition for H to belong to $\mathcal{H}(d_{\max}, n^*, \mathrm{cs}^*, \mathrm{ch}^*, \theta^*)$ becomes slightly easier than the case of undirected graphs. More formally we prepare the following.

(i) In the scheme graph $(K, \mathcal{E})$, denote the edges in $E_1 \cup E_2 \cup E_3$ by $E_1 = \{a_1, a_2, \ldots, a_{|E_1|}\}$, $E_2 = \{a_{|E_1|+1}, \ldots, a_{c^*}\}$ and $E_3 = \{a_{c^*+1}, \ldots, a_m\}$ (where $c^* = |E_1 \cup E_2|$), and regard each edge $a_i = u_{s,1}u_{s',1} \in E_1 \cup E_2 \cup E_3$ as a directed edge from one end-vertex $u_{s,1}$ to the other end-vertex $u_{s',1}$ with $s < s'$. Let $a(i)$ be a binary variable for each edge a_i, $i \in [1, m]$.

(ii) In each tree S_s (resp., T_t) in the tree-extension, we regard each edge $e'_{s,i}$, $i \geq 2$ in the rooted tree S_s, $s \in [1, s^*]$ (resp., $e_{t,i}$, $i \geq 2$ in the rooted tree T_t, $t \in [1, t^*]$) as a directed edge from vertex $u_{s,\mathrm{prt}(i)}$ to vertex $u_{s,i}$ (resp., from vertex $v_{t,\mathrm{prt}(i)}$ to vertex $v_{t,i}$). Let $u(s,i)$ (resp., $v(t,i)$) be a binary variable for vertex $u_{s,i}$, $s \in [1, s^*]$ (resp., $t \in [1, t^*]$) and $i \in [1, n_{\mathrm{tree}}]$;

(iii) In the path P_{t^*} consisting of the roots of trees T_t, $[t \in 1, t^*]$, we regard each edge e_t, $t \in [2, t^*]$ as a directed edge from vertex $v_{t-1,1}$ to vertex $v_{t,1}$; and

(iv) We regard each edge $u_{s,1}v_{t,1}$ for $s \in [1, s^*]$ and $t \in [1, t^*]$ as two directed edges, one directed from vertex $u_{s,1}$ to vertex $v_{t,1}$ and the other directed oppositely. Let $e(s,t)$ (resp., $e(t,s)$) be a binary variable of directed edge $(u_{s,1}, v_{t,1})$ (resp., $(v_{t,1}, u_{s,1})$).

Based on these, we include constraints with some more additional variables so that a selected subgraph H is a connected rank-2 graph. See constraints Equations (A.1.10) to (A.1.42) in Appendix A for the details.

In the constraints of 2, we prepare an integer variable $\widetilde{\alpha}(u)$ for each vertex u in the tree-extension that represents the chemical element $\alpha(u) \in \Lambda$ if u is in a selected graph H (or $\widetilde{\alpha}(u) = 0$ otherwise) and an integer variable $\widetilde{\beta}(e) \in [0,3]$ (resp., $\widehat{\beta}(e) \in [0,3]$) for each edge e (resp., $e = e(s,t)$ or $e(t,s)$, $s \in [1, s^*]$, $t \in [1, t^*]$) in the tree-extension that represents the multiplicity $\beta(e) \in [1,3]$ if e is in a selected graph H (or $\widetilde{\beta}(e)$ or $\widehat{\beta}(e)$ takes 0 otherwise). This determines a chemical graph $G = (H, \alpha, \beta)$. Also we include constraints for a selected chemical graph G to satisfy the valence condition $(\alpha(u), \alpha(v), \beta(uv)) \in \Gamma$ for each edge $uv \in E$. See constraints Equations (A.1.43) to (A.1.61) in Appendix A for the details.

In the constraints of 3, we introduce a variable for each descriptor and constraints with some more variables to compute the value of each descriptor in $f(G)$ for a selected chemical graph G. See constraints Equations (A.1.62) to (A.1.113) in Appendix A for the details.

With constraints 1 to 3, our MILP formulation already represents a rank-2 chemical graph G and a feature vector $x \in \mathbb{R}^k$ so that $x = f(G)$ holds. In the constraints of 4, we include some additional constraints so that the search space required for an MILP solver to solve an instance of our MILP problem is reduced. For this, we consider a graph-isomorphism of rooted subtrees of each tree S_s or T_s and define a canonical form among subtrees that are isomorphic to each other. We try to eliminate a chemical graph G that has a subtree in S_s or T_s that is not a canonical form. See constraints Equations (A.1.114) to (A.1.119) in Appendix A for the details.

4.4 Experimental Results

We implemented our method of Stages 1 to 5 for inferring rank-2 chemical graphs and conducted experiments to evaluate the computational efficiency for three

chemical properties π: octanol/water partition coefficient (Kow), melting point (Mp), and boiling point (Bp). We executed the experiments on a PC with Intel Core i5 1.6 GHz CPU and 8GB of RAM running under the Mac OS operating system version 10.14.6. We show 2D drawings of some of the inferred chemical graphs, where ChemDoodle version 10.2.0 is used for constructing the drawings.

Results on Phase 1.

Stage 1. We set a graph class $\mathcal{G}$ to be the set of all rank-2 chemical graphs. For each property $\pi \in \{\text{Kow}, \text{Mp}, \text{Bp}\}$, we select a set Λ of chemical elements and collected a data set D_π on rank-2 chemical graphs over Λ provided by HSDB from PubChem. To construct the data set, we eliminated chemical compounds that have at most three carbon atoms or contain a charged element such as $\mathtt{N}^+$ or an element $\mathtt{a} \in \Lambda$ in which the valence is different from our setting of valence function val.

Table 4.1 shows the size and range of data sets that we prepared for each chemical property in Stage 1, where we denote the following:

- π: one of the chemical properties Kow, Mp and Bp;
- $|D_\pi|$: the size of data set D_π for property π;
- Λ: the set of chemical elements over data set D_π (hydrogen atoms are added at the final stage); Λ is one of the following 2 datasets:
  $\Lambda_1 = \{\mathtt{C}, \mathtt{N}, \mathtt{O}\}$, $\Lambda_2 = \{\mathtt{C}, \mathtt{N}, \mathtt{O}, \mathtt{S}, \mathtt{P}, \mathtt{Cl}\}$.
- $|\Gamma|$: the number of tuples in Γ;
- $[\underline{n}, \overline{n}]$: the minimum and maximum number $n(G)$ of non-hydrogen atoms over data set D_π;
- $[\underline{\mathrm{cs}}, \overline{\mathrm{cs}}]$, $[\underline{\mathrm{ch}}, \overline{\mathrm{ch}}]$: the minimum and maximum core size and core height over chemical compounds in D_π, respectively;
- $[\underline{\theta}, \overline{\theta}]$: the minimum and maximum values of the topological parameter $\theta(G)$ over data set D_π; and
- $[\underline{a}, \overline{a}]$: the minimum and maximum values of $a(G)$ in π over data set D_π.

Stage 2. We used a feature function f that consists of the descriptors defined in Section 4.2.

Stage 3. We used `scikit-learn` version 0.21.6 with Python 3.7.4 to construct ANNs $\mathcal{N}$ where the tool and activation function are set to be MLPRegressor and ReLU, respectively. We tested several different architectures of ANNs for

Table 4.1. The results of Stage 1 in Phase 1.

π	$\lvert D_\pi \rvert$	Λ	$\lvert\Gamma\rvert$	$[\underline{n}, \overline{n}]$	$[\underline{\mathbf{cs}}, \overline{\mathbf{cs}}]$	$[\underline{\mathbf{ch}}, \overline{\mathbf{ch}}]$	$[\underline{\theta}, \overline{\theta}]$	$[\underline{a}, \overline{a}]$
Kow	93	Λ_1	9	[9, 31]	[7, 16]	[0, 13]	[−5, 3]	[−3.7, 12.2]
MP	63	Λ_1	7	[9, 31]	[7, 17]	[0, 4]	[−6, 3]	[−80, 300]
BP	45	Λ_2	9	[9, 25]	[7, 15]	[0, 7]	[−4, 3]	[155, 420]

Table 4.2. The results of Stages 2 and 3 in Phase 1.

π	k	Activation	Architecture	L-Time	Test R^2 (ave.)	(Best)
Kow	37	relu	(37,10,1)	3.92	0.866	0.964
MP	33	relu	(33,10,1)	21.68	0.805	0.916
BP	43	relu	(43,10,1)	11.88	0.802	0.947

each chemical property. To evaluate the performance of the resulting prediction function $\psi_{\mathcal{N}}$ with cross-validation, we partition a given data set D_π into five subsets $D_\pi^{(i)}$, $i \in [1, 5]$ randomly, where $D_\pi \setminus D_\pi^{(i)}$ is used for a training set and $D_\pi^{(i)}$ is used for a test set in five trials $i \in [1, 5]$. Table 4.2 shows the results on Stages 2 and 3, where

- k: the number of descriptors for the chemical compounds in data set D_π for property π;
- Activation: the choice of activation function;
- Architecture: $(a, b, 1)$ consists of an input layer with a nodes, a hidden layer with b nodes, and an output layer with a single node, where a is equal to the number of descriptors;
- L-time: the average time (sec.) to construct ANNs for each trial;
- test R^2 (ave.): the average of coefficient of determination over the five test sets; and
- test R^2 (best): the largest value of coefficient of determination over the five test sets.

For each chemical property π, we selected the ANN $\mathcal{N}$ that attained the best test R^2 score among the five ANNs to formulate an MILP $\mathcal{M}(x, y, z; \mathcal{C}_1)$ in the second phase.

Results on Phase 2. We implemented Stages 4 and 5 in Phase 2 as follows.

Stage 4. In this stage, we solve the MILP $\mathcal{M}(x, y, g; \mathcal{C}_1, \mathcal{C}_2)$ formulated based on the ANN $\mathcal{N}$ obtained in Phase 1. To solve an MILP in Stage 4, we use `CPLEX` version 12.10. In our experiment, we choose a target value $y^* \in [\underline{a}, \overline{a}]$ and fix or bound some descriptors in our feature vector as follows:

- Fix variable θ that represents the polymer parameter $\theta(H)$ to be each integer in $\{-2, 0, 2\}$;
- Set $d_{\max}$ to be each of 3 and 4;
- Fix n^* to be some four integers in $\{15, 19, 20, 25, 30\}$ for $\theta \in \{-2, 0\}$ and $\{15, 19, 20, 22, 25\}$ for $\theta = 2$;
- Choose three integers from $[7, 16]$ and fix cs^* to be each of the three integers;
- Fix ch^* to be each of the four integers in $[2, 5]$.

Based on the above setting, we generated 12 instances for each n^*. We set $\varepsilon = 0.02$ in Stage 4.

Tables 4.3–4.8 show the results of Stage 4 for $d_{\max} = 3$ and 4, respectively, where we denote the following:

- y^*_π: a target value in $[\underline{a}, \overline{a}]$ for a property π;
- n^*: a specified number of vertices in $[\underline{n}, \overline{n}]$;
- $|F^*|/\#\mathrm{I}$: #I means the number of MILP instances in Stage 4 (where #I=12), and $|F^*|$ means the size of set F^* of vectors x^* generated from all feasible instances among the #I MILP instances in Stage 4;
- IP-time: the average time (sec.) to solve one of the #I MILP instances to find a set F^* of vectors x^*.

Figure 4.5(a) to (c) illustrate some rank-2 chemical graphs G^* with $\theta(G^*) = -2$ constructed from the vector g^* obtained by solving the MILP in Stage 4.

Figure 4.6(a) to (c) illustrate some rank-2 chemical graphs G^* with $\theta(G^*) = 0$ constructed from the vector g^* obtained by solving the MILP in Stage 4.

Figures 4.7(a) to (c) illustrate some rank-2 chemical graphs G^* with $\theta(G^*) = 2$ constructed from the vector g^* obtained by solving the MILP in Stage 4.

Stage 5. In this stage, we modified the algorithms proposed by Tamura et al. [48] and Yamashita et al. [56] to enumerate all rank-2 graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$ for each $x^* \in F^*$. We stop the execution when either the total number of graphs inferred over all vectors $x^* \in F^*$ exceeds 100 or the execution time exceeds one hour.

Tables 4.3–4.8 show the results on Stage 5 for $d_{\max} = 3$ and 4, respectively,

Table 4.3. Results of Stages 4 and 5 with $d_{\max} = 3$ and $\theta = -2$.

π	y^*_π	n^*	$\|F^*\|/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	12/12	9.96	100	2236.0
Kow	5	20	12/12	30.38	12	>1 h
Kow	5	25	12/12	47.57	12	>1 h
Kow	5	30	12/12	69.38	12	>1 h
MP	150	15	12/12	9.52	100	2069.0
MP	150	20	12/12	22.79	12	>1 h
MP	150	25	12/12	47.20	12	>1 h
MP	150	30	12/12	66.90	12	>1 h
BP	250	15	11/12	9.50	100	103.5
BP	250	19	12/12	19.08	12	>1 h
BP	250	22	12/12	25.78	12	>1 h
BP	250	25	12/12	67.64	12	>1 h

Table 4.4. Results of Stages 4 and 5 with $d_{\max} = 4$ and $\theta = -2$.

π	y^*_π	n^*	$\|F^*\|/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	11/12	31.84	100	413.8
Kow	5	20	12/12	69.65	12	>1 h
Kow	5	25	12/12	144.20	11	>1 h
Kow	5	30	12/12	352.01	12	>1 h
MP	150	15	9/12	20.68	100	947.4
MP	150	20	11/12	73.73	11	>1 h
MP	150	25	9/12	140.09	9	>1 h
MP	150	30	12/12	304.04	12	>1 h
BP	250	15	7/12	28.51	100	232.7
BP	250	19	11/12	82.01	11	>1 h
BP	250	22	12/12	150.55	12	>1 h
BP	250	25	12/12	239.84	12	>1 h

- $\#G^*$: the number of all (or up to 100) rank-2 chemical graphs G^* that are computed under 1 h time limit in Stage 5, where $f(G^*) = x^*$ for some $x^* \in F^*$. (Note that $|F^*|$ such graphs G^* have been found in Stage 4, and Figures 4.5–4.7 illustrate some of such graphs G^*.);

- G-time: the running time (sec.) to execute Stage 5, where ">1 h" means

Table 4.5. Results of Stages 4 and 5 with $d_{\max} = 3$ and $\theta = 0$.

π	y^*_π	n^*	$\lvert F^*\rvert/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	12/12	11.00	100	121.1
Kow	5	20	12/12	25.64	12	>1 h
Kow	5	25	12/12	38.79	12	>1 h
Kow	5	30	12/12	49.65	12	>1 h
Mp	150	15	12/12	8.45	100	373.4
Mp	150	20	12/12	18.94	12	>1 h
Mp	150	25	12/12	37.13	12	>1 h
Mp	150	30	12/12	44.745	4	>1 h
Bp	250	15	9/12	8.450	100	74.2
Bp	250	19	11/12	16.31	11	>1 h
Bp	250	22	12/12	21.71	12	>1 h
Bp	250	25	12/12	45.80	12	>1 h

Table 4.6. Results of Stages 4 and 5 with $d_{\max} = 4$ and $\theta = 0$.

π	y^*_π	n^*	$\lvert F^*\rvert/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	9/12	36.33	100	23.2
Kow	5	20	12/12	82.01	12	>1 h
Kow	5	25	12/12	138.96	12	>1 h
Kow	5	30	12/12	292.79	12	>1 h
Mp	150	15	9/12	19.89	100	557.6
Mp	150	20	11/12	63.62	11	>1 h
Mp	150	25	12/12	112.49	12	>1 h
Mp	150	30	12/12	171.11	12	>1 h
Bp	250	15	3/12	34.60	100	11.2
Bp	250	19	6/12	203.65	6	>1 h
Bp	250	22	9/12	218.07	9	>1 h
Bp	250	25	11/12	783.80	11	>1 h

that the execution time exceeds the limit.

We also conducted some additional experiments to demonstrate that our MILP-based method is flexible to control conditions on the inference of chemical graphs. In Stage 3, we constructed an ANN $\mathcal{N}_\pi$ for each of the three chemical properties $\pi \in \{\mathrm{Kow}, \mathrm{Mp}, \mathrm{Bp}\}$, and formulated the inverse problem of each ANN

Table 4.7. Results of Stages 4 and 5 with $d_{\max} = 3$ and $\theta = 2$.

π	y^*_π	n^*	$\|F^*\|/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	12/12	11.64	100	1386.7
Kow	5	20	12/12	23.84	12	>1 h
Kow	5	25	12/12	33.71	12	>1 h
Kow	5	30	12/12	61.85	12	>1 h
MP	150	15	12/12	9.80	100	1614.3
MP	150	20	12/12	20.15	12	>1 h
MP	150	25	12/12	36.42	12	>1 h
MP	150	30	12/12	40.58	12	>1 h
BP	250	15	11/12	10.25	100	1756.1
BP	250	19	12/12	16.02	12	>1 h
BP	250	22	12/12	23.63	12	>1 h
BP	250	25	12/12	63.84	12	>1 h

Table 4.8. Results of Stages 4 and 5 with $d_{\max} = 4$ and $\theta = 2$.

π	y^*_π	n^*	$\|F^*\|/\#\mathrm{I}$	IP-Time	$\#G^*$	G-Time
Kow	5	15	11/12	28.15	100	20.3
Kow	5	20	12/12	71.90	12	>1 h
Kow	5	25	12/12	112.71	12	>1 h
Kow	5	30	12/12	267.21	12	>1 h
MP	150	15	9/12	22.53	100	2748.1
MP	150	20	11/12	53.44	11	>1 h
MP	150	25	12/12	143.33	12	>1 h
MP	150	30	12/12	220.63	12	>1 h
BP	250	15	6/12	27.33	100	254.2
BP	250	19	9/12	75.50	9	>1 h
BP	250	22	11/12	133.01	11	>1 h
BP	250	25	12/12	228.75	12	>1 h

$\mathcal{N}_\pi$ as an MILP $\mathcal{M}_\pi$. Since the set of descriptors is common to all three properties Kow, MP, and BP, it is possible to infer a rank-2 chemical graph G^* that satisfies a target value y^*_π for each of the three properties at the same time (if one exists). We specify the size of graph so that $n := 22$, core size:=14, core height:= 3, $\theta := -2$ and $d_{\max} := 3$, and set target values with $y^*_{\mathrm{Kow}} := 5$, $y^*_{\mathrm{Mp}} := 150$ and $y^*_{\mathrm{Bp}} := 250$ in an MILP that consists of the three MILPs $\mathcal{M}_{\mathrm{Kow}}$, $\mathcal{M}_{\mathrm{Mp}}$ and $\mathcal{M}_{\mathrm{Bp}}$.

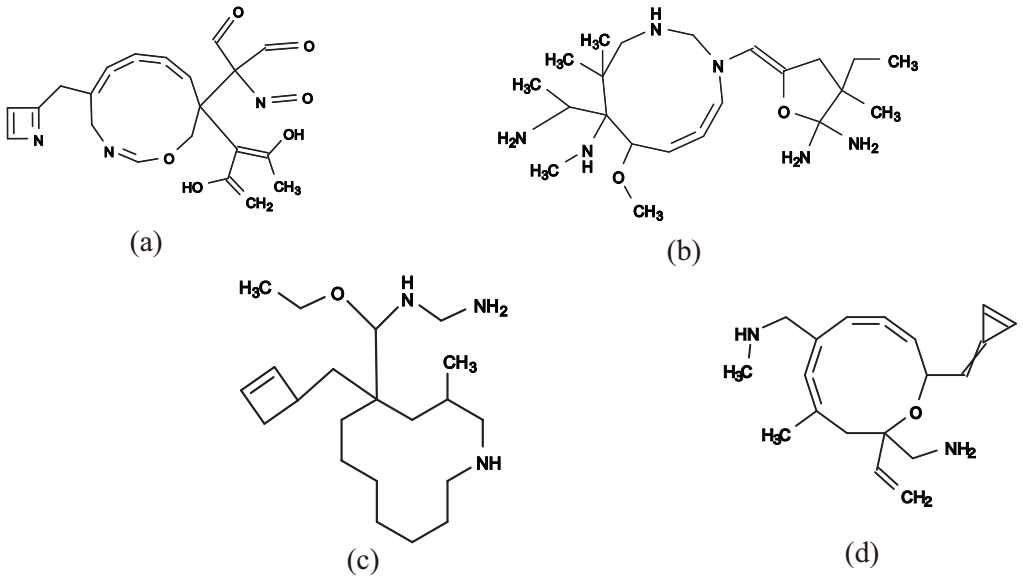


Figure 4.5. An illustration of inferred rank-2 chemical graphs G^* with $\theta = -2$: (**a**) $y^*_{\mathrm{Kow}} = 5$, $\theta = -2$, $n = 30$, core size = 16, core height = 3, $d_{\max} = 4$; (**b**) $y^*_{\mathrm{Mp}} = 150$, $\theta = -2$, $n = 30$, core size = 16, core height = 2, $d_{\max} = 3$; (**c**) $y^*_{\mathrm{Bp}} = 250$, $\theta = -2$, $n = 25$, core size = 17, core height = 4, $d_{\max} = 3$; (**d**) $y^*_{\mathrm{Kow}} = 5$, $y^*_{\mathrm{Mp}} = 150$, $y^*_{\mathrm{Bp}} = 250$, $\theta = -2$, $n = 22$, core size = 14, core height = 3, $d_{\max} = 3$.

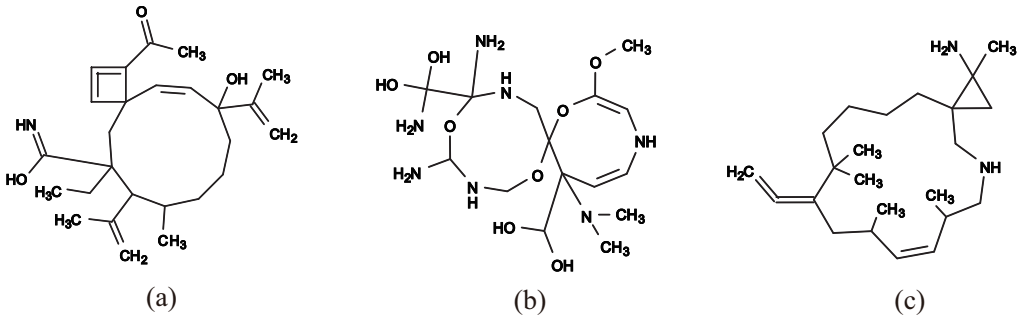


Figure 4.6. An illustration of inferred rank-2 chemical graphs G^*: (**a**) $y^*_{\mathrm{Kow}} = 5$, $\theta = 0$, $n = 30$, core size = 14, core height = 2, $d_{\max} = 3$; (**b**) $y^*_{\mathrm{Mp}} = 150$, $\theta = 0$, $n = 30$, core size = 16, core height = 2, $d_{\max} = 4$; (**c**) $y^*_{\mathrm{Bp}} = 250$, $\theta = 0$, $n = 25$, core size =17, core heigh t= 2, $d_{\max} = 3$.

The MILP was solved in 268.11 (sec) and we obtained a rank-2 chemical graph G^* illustrated in Figure 4.5(d).

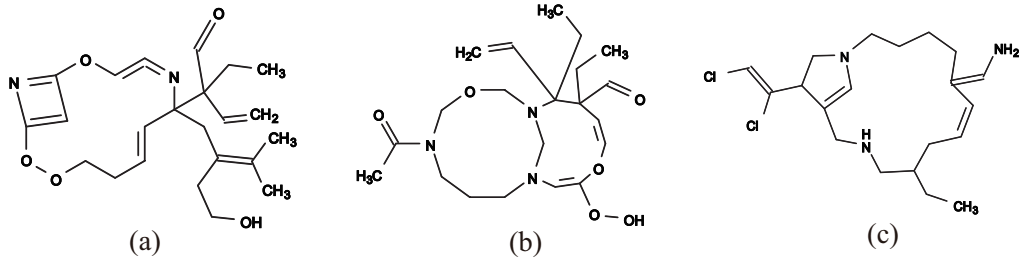


Figure 4.7. An illustration of inferred rank-2 chemical graphs G^*: (**a**) $y^*_{\mathrm{Kow}} = 5$, $\theta = 2$, $n = 30$, core size = 15, core height = 5, $d_{\max} = 4$; (**b**) $y^*_{\mathrm{Mp}} = 150$, $\theta = 2$, $n = 30$, core size = 17, core height = 2, $d_{\max} = 3$; (**c**) $y^*_{\mathrm{Bp}} = 250$, $\theta = 2$, $n = 25$, core size = 17, core height = 3, $d_{\max} = 3$.

4.5 Concluding Remarks

In this chapter, we proposed a new method for the inverse QSAR to rank-2 chemical graphs by significantly enhancing the framework for acyclic chemical graphs due to Azam et al. [4] and Zhang et al. [59], and the framework for rank-1 chemical graphs due Ito et al. [25], and implemented it for inferring rank-2 chemical graphs using the algorithms for enumerating rank-2 chemical graphs due to Tamura et al. [48] and Yamashita et al. [56]. From the results on some computational experiments, we observe that the proposed method runs efficiently for an instance with $n^* \leq 30$ non-hydrogen atoms up to Stage 4 and an instance with $n^* \leq 15$ non-hydrogen atoms up to Stage 5. Due to this development, the ratio of chemical compounds covered in the PubChem database increased from 16.26% to 44.5%. It is left as future work to apply our new method for the inverse QSAR to a wider class of graphs. The ratio of the number of chemical graphs with rank at most 3 (resp., 4) to the number of all chemical graphs in database PubChem is 68.8% (resp., 84.7%). Among rank-4 chemical compounds, Remdesivir $\mathtt{C_{27}H_{35}N_6O_8P}$, an antiviral medication, which is being studied as a possible post-infection treatment for COVID-19, has a chemical graph G with $\mathrm{r}(G) = 4$, $n(G) = 42$, $\mathrm{cs}(G) = 24$, and $\mathrm{ch}(G) = 8$. The number of polymer topologies with rank 3 (resp., 4) such that the maximum degree is at most 4 is 12 (resp., 73). Our MILP formulation can be easily extended to the case of rank 3 or 4 by replacing the current set of constraints for the scheme graph with a set of those for a new scheme graph that is designed for rank-3 or -4 polymer topologies.

5 An Inverse Method for Arbitrary Cyclic Chemical Compounds[1]

5.1 Introduction

In many of methods for inverse QSAR, inference or enumeration of graph structures from a given set of descriptors is a crucial subtask, and thus various methods have been developed [15, 20, 30, 38]. However, enumeration in itself is a challenging task, since the number of molecules (i.e., chemical graphs) with up to 30 atoms (vertices) `C`, `N`, `O`, and `S`, may exceed 10^{60} [9]. Furthermore, enumeration methods tended to be based on branching algorithms, where individual atoms or substructures are iteratively appended to extend a certain structure. Such a computation process quickly comes into a combinatorial explosion. Even for moderately-sized target chemical graphs with around 20 non-hydrogen atoms, the computation process might not produce even a single output after several hours.

Following the novel framework proposed by [4, 11, 59], Ito et al. [25] and Zhu et al. [61] designed a method of inferring chemical graphs with rank 1 and 2, respectively, where we presented the content of [61] (i.e. rank 2) in Chapter 4. Each of them formulated a new MILP, and used an efficient algorithm for enumerating chemical graphs with rank 1 [46] and rank 2 [48, 56]. The computational results conducted with the above methods [25, 61] on instances with n non-hydrogen atoms show that a feature vector x^* can be inferred for up to around $n = 40$ whereas graphs G^* can be enumerated for up to around $n = 15$. Note that for each different graph class, an entirely new MILP formulation and graph enumeration algorithm needed to be designed, which are themselves daunting tasks.

Recently Azam et al. [7] introduced a new characterization of acyclic graph structure, called "branch-height" to define a class of acyclic graphs with a re-

[1] ©2021 IEEE. Reprinted, with permission, from J. Zhu, N. A. Azam, F. Zhang, A. Shurbevski, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A novel method for inferring chemical compounds with prescribed topological substructures based on integer programming. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6): 3233–3245, 2021.

stricted structure that still covers most (99.46%) of the acyclic chemical compounds in the PubChem database [27]. They also employed the dynamic programming method to design a new algorithm for generating chemical acyclic graphs which now works for instances with size up to $n(G^*) = 50$.

The above framework has been applied so far to the case of chemical compounds with a rather abstract topological structure, such as acyclic or monocyclic graphs, and graphs with a specified polymer topology with rank up to 2. When there is a more specific requirement on some part of the graph structure and the assignment of chemical elements in a chemical graph to be inferred, none of the above-mentioned methods can be used directly. The main reason is that generating chemical graphs from a given feature vector is a considerably hard problem. In particular, an efficient algorithm needed to be newly designed for each different class of graphs.

This chapter builds upon and extends the recently proposed framework for inferring acyclic chemical graphs [4, 11, 59], and later extended to the cases of mono-cyclic [25] and chemical graphs with cycle rank 2 [61]. We note that, to the best of our knowledge, this framework is unique. First and foremost, it offers an end-to-end system for chemical graph design that includes pattern recognition from data and mathematical exactness.

A similar approach for inverse molecular design has been recently proposed by Takeda et al. [47], however, the two approaches have significant differences. Notably, the method of defining descriptors in a feature vector, reported in [47], employs general substructure-based indices, whereas the work in this chapter uses graph-theoretical descriptors, which preserve explainability. Then, for solving the problem of reconstructing a feature vector once a regression function has been constructed, Takeda et al. [47] report to use custom-implemented gradient search method, which does not necessarily guarantee optimality of the obtained solution. On the other hand, as a foundation in this chapter, we use an MILP formulation, which comes with a definite guarantee for the optimality, or the level of deviation of an obtained solution. Next, the method in [47] for combinatorially generating chemical graphs is based on a branching algorithm, which necessitates computing a canonical labeling of a graph, a step with exponential computational complexity, and not suited for compounds with more than around 20 atoms. Furthermore, the report of Takeda et al. [47] is of a commercial implementation that is not available for experimental evaluation by the community.

Another report that resembles the framework that we build on is due to Sumita et al. [45]. However, it includes a Monte-Carlo based search that takes on the order of several days of computation time to uncover some chemical com-

pounds with desired properties. As shall be seen from our computational experience in Section 5.6, an end-to-end application of our framework took on the order of minutes, and at most few hours of computation time, to obtain a wealth of chemical compounds, all within mathematically proven range of set desired target properties as predicted by the regression function constructed in the framework.

In this chapter, we propose an entirely new mechanism of generating chemical graphs. In particular, the contributions include:

- Our mechanism can be readily applied to any graph topology that contains at least one cycle. Thereby, we dispense with the necessity of designing a new algorithm for different graph classes. In particular, we no longer need to design an MILP formulation anew for different graph classes, nor a specific-purpose combinatorial graph generation algorithm.

- Introduce a flexible way of specifying the topological structure of target chemical graphs. As such, certain substructures, such as benzene rings can be included, while at the same time constraints on the global topological structure of target graphs can be imposed. This allows for the possibility to include domain knowledge in the way of specifying graph structures.

- We introduce a combinatorial graph construction algorithm based on the dynamic programming paradigm, that can efficiently explore a huge space [9] of possible chemical graphs with a given feature vector.

- We open the source code of our implementation of the proposed method/system, available at GitHub
  `https://github.com/ku-dml/mol-infer/tree/master/Cyclic`.
  (Accessed: May 24, 2023.)

The rest of this chapter is organized as follows. Section 5.2 introduces some notions on graphs, a convention for modeling chemical compounds as graphs, and a choice of descriptors. Section 5.3 introduces a method of specifying topological substructures of target chemical graphs to be inferred. Section 5.4 presents an idea of a formulation of an MILP that can infer a chemical graph under a given specification of the target chemical graphs. Section 5.5 describes a new idea of generating chemical graphs G^* that have the same feature vector as a given chemical graph $G^\dagger$. Section 5.6 reports the results on some computational experiments conducted for some chemical properties. Section 5.7 makes some concluding remarks.

5.2 Preliminary

This section introduces some notation and terminology on a modeling of chemical compounds, and our choice of descriptors.

5.2.1 Modeling of Chemical Compounds

We represent the graph structure of a chemical compound as a graph H with labels on vertices and multiplicity on edges in a hydrogen-suppressed model. In a cyclic graph H, we regard each non-core-edge $uv \in E$ as a directed edge (u, v) from a vertex u to a child v of u in an exterior-tree of H in order to define a descriptor that exploits the direction of non-core-edges.

We introduce "edge-configuration", a refined notion of adjacency-configuration. We call a pair $(\mathtt{a}, i)$ of a chemical element $\mathtt{a}$ and degree i a *chemical symbol*, written as $\mathtt{a}i$. Let $\Lambda_{\rm dg}$ denote the set of all chemical symbols. We call a tuple $(\mathtt{a}i, \mathtt{b}j, m)$ with $\mathtt{a}i, \mathtt{b}j \in \Lambda_{\rm dg}$ and $m \in [1, 3]$ an *edge-configuration*. We choose a branch-parameter $\rho \in \mathbb{Z}_+$, two sets, $\Lambda_{\rm dg}^{\rm co}$ and $\Lambda_{\rm dg}^{\rm nc}$, of chemical symbols, and three sets, $\Gamma^{\rm co}$, $\Gamma^{\rm in}$, and $\Gamma^{\rm ex}$, of edge-configurations.

Let $e = uv$ be an edge in a chemical graph G such that $\mathtt{a}, \mathtt{b} \in \Lambda$ are assigned to the vertices u and v, the degrees of u and v are i and j, respectively, and the bond-multiplicity between them is m. When uv is a core-edge, the edge-configuration $\tau(e)$ of edge e is defined to be $(\mathtt{a}i, \mathtt{b}j, m)$ if $\mathtt{a}i \leq \mathtt{b}j$ in a total order over $\Lambda_{\rm dg}$ (or $(\mathtt{b}j, \mathtt{a}i, m)$ otherwise). When uv is a non-core-edge which is regarded as a directed edge (u, v) where u is the parent of v in some exterior-tree, the edge-configuration $\tau(e)$ of a ρ-internal (resp., ρ-external) edge e is defined to be $(\mathtt{a}i, \mathtt{b}j, m) \in \Gamma^{\rm in}$ (resp., $(\mathtt{a}i, \mathtt{b}j, m) \in \Gamma^{\rm ex}$).

A *chemical cyclic graph* is defined to be a tuple $G = (H, \alpha, \beta)$ of a cyclic graph $H = (V, E)$ and functions $\alpha : V \to \Lambda$ and $\beta : E \to [1, 3]$, such that
(i) H is connected;
(ii) $\sum_{uv \in E} \beta(uv) \leq \mathrm{val}(\alpha(u))$ for each vertex $u \in V$; and
(iii) $\tau(e) \in \Gamma^{\rm co}$, $\tau(e) \in \Gamma^{\rm in}$ and $\tau(e) \in \Gamma^{\rm ex}$ for each core-edge $e \in E$, ρ-internal edge $e \in E$ and ρ-external edge $e \in E$, respectively.

We represent the graph structure of a chemical compound as a graph with labels on vertices and multiplicity on edges in a hydrogen-suppressed model.

In our method, we use only graph-theoretical descriptors for defining a feature vector. A *feature vector* $f(G)$ of a chemical cyclic graph $G = (H = (V, E), \alpha, \beta)$ consists of the following 16 kinds of descriptors.

- $n(G)$: the number $|V|$ of vertices;
  $\mathrm{cs}(G)$: the core size of G;

$\mathrm{ch}(G)$: the core height of G;
$\mathrm{bl}_\rho(G)$: the ρ-branch-leaf-number of G;

- $\overline{\mathrm{ms}}(G)$: the average mass of atoms in G;
  $\mathrm{ns}_{\mathtt{H}}(G)$: the number of hydrogen atoms suppressed in G;

- $\mathrm{dg}_i^{\mathrm{co}}(G)$, $\mathrm{dg}_i^{\mathrm{nc}}(G)$: the numbers of core-vertices and non-core-vertices of degree $i \in [1,4]$ in G;

- $\mathrm{bd}_m^{\mathrm{co}}(G)$, $\mathrm{bd}_m^{\mathrm{in}}(G)$, $\mathrm{bd}_m^{\mathrm{ex}}(G)$: the numbers of core-edges, ρ-internal edges and ρ-external edges with bond multiplicity $m \in [1,3]$ in G;

- $\mathrm{ns}_\mu^{\mathrm{co}}(G)$, $\mu \in \Lambda_{\mathrm{dg}}^{\mathrm{co}}$, $\mathrm{ns}_\mu^{\mathrm{nc}}(G)$, $\mu \in \Lambda_{\mathrm{dg}}^{\mathrm{nc}}$: the numbers of core-vertices and non-core-vertices v with $\alpha(v) = \mathtt{a}$ and degree i for $\mu = \mathtt{a}i$; and

- $\mathrm{ec}_\gamma^{\mathrm{co}}(G)$, $\mathrm{ec}_\gamma^{\mathrm{in}}(G)$, $\mathrm{ec}_\gamma^{\mathrm{ex}}(G)$: the numbers of core-edges $e \in E$ such that $\tau(e) = \gamma \in \Gamma^{\mathrm{co}}$, ρ-internal edges $e \in E$ such that $\tau(e) = \gamma \in \Gamma^{\mathrm{in}}$, and ρ-external edges $e \in E$ such that $\tau(e) = \gamma \in \Gamma^{\mathrm{ex}}$ in G.

5.3 Specifying Target Chemical Graphs

This section presents a flexible way of specifying a topological structure of the core and assignments of chemical elements and bond-multiplicities of a target chemical graph. We define a *target specification* $(G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$ with a multigraph G_{C} and sets σ_{co}, σ_{nc}, and $\sigma_{\alpha\beta}$, of lower and upper bounds on several descriptors. The choice of descriptors in the target specification is aimed to allow flexibility in our model to specify the structure of a chemical graph based on some intuition of the graph structure. Below we describe in more detail the structure of a target specification.

Seed Graphs A *seed graph* $G_{\mathrm{C}} = (V_{\mathrm{C}}, E_{\mathrm{C}})$ is defined to be a multigraph with no self-loops such that the edge set E_{C} consists of four sets $E_{(\geq 2)}$, $E_{(\geq 1)}$, $E_{(0/1)}$, and $E_{(=1)}$. Fig. 5.1(a) illustrates an example of a seed graph. From a seed graph G_{C}, the core of a cyclic graph will be constructed in the following way: Each edge $e = uv \in E_{(\geq 2)}$ will be replaced with a u, v-path P_e of length at least 2; Each edge $e = uv \in E_{(\geq 1)}$ will be replaced with a u, v-path P_e of length at least 1; Each edge $e \in E_{(0/1)}$ is either used or discarded; and Each edge $e \in E_{(=1)}$ is always used directly.

Core Specification The core of a target chemical graph is constructed from a seed graph G_{C} by a *core specification* σ_{co} that consists of the following:

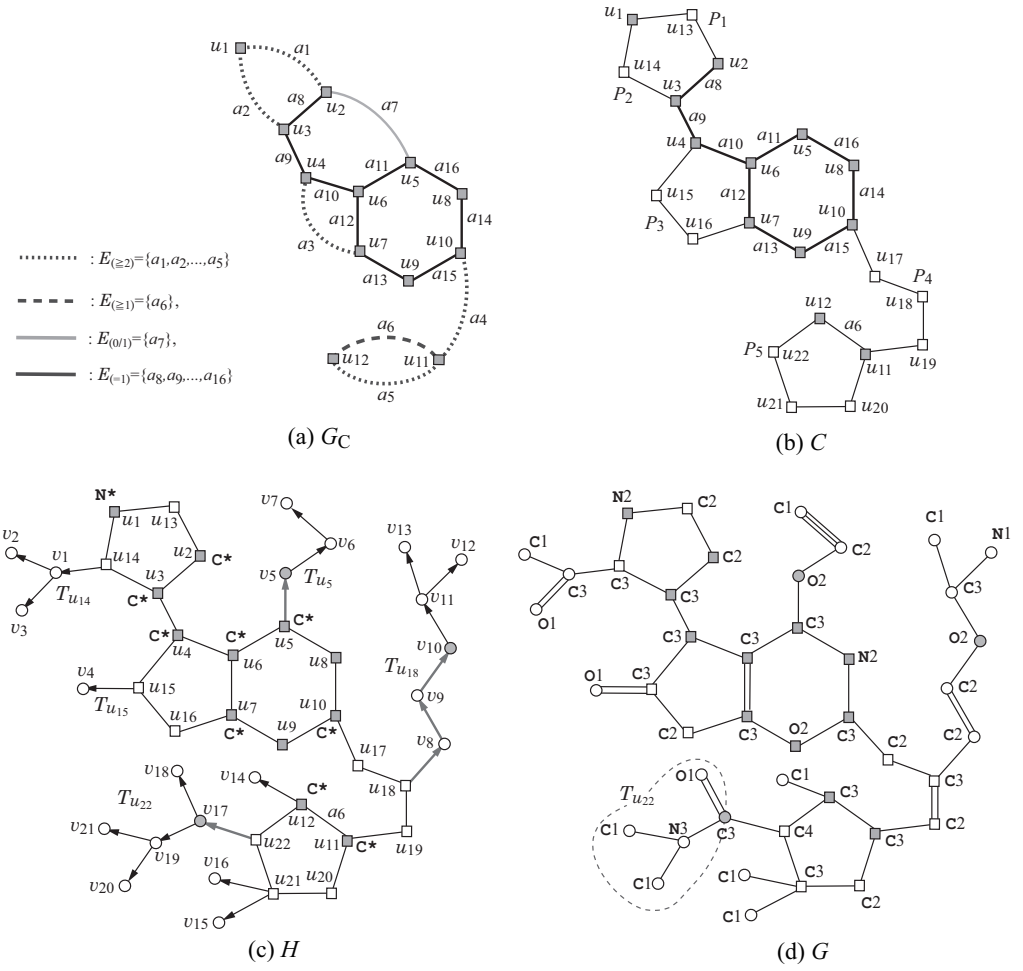


Figure 5.1. (a) A seed graph G_{C}; (b) A σ_{co}-extension C with $\mathrm{cs}(C) = 22$; (c) A $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H with $\mathrm{Cr}(H) = C$, $n(H) = 43$, $\mathrm{ch}(H) = 5$, and $\mathrm{bl}_2(H) = 3$; (d) A $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$-extension G of G_{C} in Fig. 5.1(a). ©2021 IEEE.

- Lower and upper bound functions $\ell_{\mathrm{LB}}, \ell_{\mathrm{UB}} : E_{(\geq 2)} \cup E_{(\geq 1)} \to \mathbb{Z}_+$; For notational convenience, set $\ell_{\mathrm{LB}}(e) := 0$, $\ell_{\mathrm{UB}}(e) := 1$, $e \in E_{(0/1)}$ and $\ell_{\mathrm{LB}}(e) := 1$, $\ell_{\mathrm{UB}}(e) := 1$, $e \in E_{(=1)}$.

- Lower and upper bounds $\mathrm{cs}_{\mathrm{LB}}, \mathrm{cs}_{\mathrm{UB}} \in \mathbb{Z}_+$ on the core size, where we assume $\mathrm{cs}_{\mathrm{LB}} \geq |V_{\mathrm{C}}| + \sum_{e \in E_{(\geq 2)} \cup E_{(\geq 1)}} (\ell_{\mathrm{LB}}(e) - 1)$.

Example 1 of σ_{co}. A core specification σ_{co} to G_{C} in Fig. 5.1(a) is given as follows: $\mathrm{cs}_{\mathrm{LB}} = 20$, $\mathrm{cs}_{\mathrm{UB}} = 28$ and a sequence of $(\ell_{\mathrm{LB}}(a_i), \ell_{\mathrm{LB}}(a_i)), i \in [1, 6]$ is given by $[(2, 3), (2, 4), (2, 3), (3, 5), (2, 4), (1, 4)]$.

A σ_{co}-*extension* of a seed graph G_{C} is defined to be a graph C with $|V(C)| \in [\mathrm{cs}_{\mathrm{LB}}, \mathrm{cs}_{\mathrm{UB}}]$ obtained by replacing each edge $e = uv \in E_{(\geq 2)} \cup E_{(\geq 1)}$ with a u, v-

path P_e of length $\ell(P_e) \in [\ell_{\rm LB}(e), \ell_{\rm UB}(e)]$. Let C be a $\sigma_{\rm co}$-extension of $G_{\rm C}$, where each edge $e = uv \in E_{(\geq 2)} \cup E_{(\geq 1)}$ is replaced with a u, v-path P_e (where possibly P_e is equal to e). For each edge $e = uv \in E_{(\geq 2)} \cup E_{(\geq 1)}$, let $\mathcal{F}(P_e)$ denote the set of trees T_w rooted at internal vertices w of the u, v-path P_e (where $w \neq u, v$).

Fig. 5.1(b) illustrates a $\sigma_{\rm co}$-extension C of $G_{\rm C}$ in Fig. 5.1(a) with core specification $\sigma_{\rm co}$ in Example 1, where the edge $a_7 \in E_{(0/1)}$ is discarded.

Non-core Specification We next construct a ρ-lean[2] cyclic graph H obtained from a $\sigma_{\rm co}$-extension C of $G_{\rm C}$, by appending a tree T_v with at most one leaf ρ-branch at each vertex $v \in V(C)$, where possibly $E(T_v) = \emptyset$. We call the vertices in C *core-vertices* of H and the newly added vertices *non-core-vertices* of H.

We specify the structure of the non-core part of such a graph H by a *non-core specification* $\sigma_{\rm nc}$ that consists of the following:

- Lower and upper bounds $n_{\rm LB}, n^* \in \mathbb{Z}_+$ on the number of vertices, where ${\rm cs}_{\rm LB} \leq n_{\rm LB} \leq n^*$;
- An upper bound ${\rm dg}^{\rm nc}_{4,{\rm UB}} \in \mathbb{Z}_+$ on the number of non-core-vertices of degree 4;
- Lower and upper functions ${\rm ch}_{\rm LB}, {\rm ch}_{\rm UB} : V_{\rm C} \to \mathbb{Z}_+$ and ${\rm ch}_{\rm LB}, {\rm ch}_{\rm UB} : E_{(\geq 2)} \cup E_{(\geq 1)} \to \mathbb{Z}_+$ on the maximum height of trees rooted at a vertex $v \in V_{\rm C}$ or at an internal vertex of a path P_e with $e \in E_{(\geq 2)} \cup E_{(\geq 1)}$;
- A branch-parameter $\rho \in \mathbb{Z}_+$;
- Lower and upper functions ${\rm bl}_{\rm LB}, {\rm bl}_{\rm UB} : V_{\rm C} \to \{0, 1\}$ on the number of leaf ρ-branches in the tree rooted at a vertex $v \in V_{\rm C}$; and
- Lower and upper functions ${\rm bl}_{\rm LB}, {\rm bl}_{\rm UB} : E_{(\geq 2)} \cup E_{(\geq 1)} \to \mathbb{Z}_+$ on the number of leaf ρ-branches in the trees rooted at internal vertices in a path P_e constructed for an edge $e \in E_{(\geq 2)} \cup E_{(\geq 1)}$.

Example 2 of $\sigma_{\rm nc}$. A non-core specification $\sigma_{\rm nc}$ to $G_{\rm C}$ in Fig. 5.1(a) is given as follows:
$n_{\rm LB} = 30$, $n^* = 50$, $\rho = 2$,
a sequence of $({\rm ch}_{\rm LB}(u_i), {\rm ch}_{\rm LB}(u_i)), i \in [1, 12]$, is given by
$[(0,1), (0,0), (0,0), (0,0), (1,3), (0,0), (0,1), (0,1), (0,0), (0,1), (0,2), (0,4)]$,
a sequence of $({\rm ch}_{\rm LB}(a_i), {\rm ch}_{\rm LB}(a_i))$, $i \in [1, 6]$, is given by
$[(0,3), (1,3), (0,1), (4,6), (3,5), (0,2)]$,
a sequence of $({\rm bl}_{\rm LB}(u_i), {\rm bl}_{\rm LB}(u_i)), i \in [1, 12]$ is given by

[2] See the definition in Chapter 2.

$[(0,1),(0,1),(0,1),(0,1),(0,1),(0,0),(0,0),(0,0),(0,0),\ (0,0),(0,0),(0,0)]$
and a sequence of $(\mathrm{bl}_{\mathrm{LB}}(a_i), \mathrm{bl}_{\mathrm{LB}}(a_i)), i \in [1,6]$, is given by
$[(0,1),(0,1),(0,0),(1,2),(1,1),(0,0)]$.

We call the above ρ-lean cyclic graph H a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-*extension* of G_{C} if the following conditions hold:

- $n(H) \in [n_{\mathrm{LB}}, n^*]$;
- $\mathrm{dg}_4^{\mathrm{nc}}(H) \leq \mathrm{dg}_{4,\mathrm{UB}}^{\mathrm{nc}}$;
- For each vertex $v \in V_{\mathrm{C}}$, the tree T_v attached to v satisfies $\mathrm{ht}(T_v) \in [\mathrm{ch}_{\mathrm{LB}}(v), \mathrm{ch}_{\mathrm{UB}}(v)]$;
- For each edge $e \in E_{(\geq 2)} \cup E_{(\geq 1)}$, $\max\{\mathrm{ht}(T) \mid T \in \mathcal{F}(P_e)\} \in [\mathrm{ch}_{\mathrm{LB}}(e), \mathrm{ch}_{\mathrm{UB}}(e)]$;
- Each tree T_v, $v \in V(C)$, contains at most one leaf ρ-branch; and
- For each edge $e \in E_{(\geq 2)} \cup E_{(\geq 1)}$, $\sum\{\mathrm{bl}_\rho(T) \mid T \in \mathcal{F}(P_e)\} \in [\mathrm{bl}_{\mathrm{LB}}(e), \mathrm{bl}_{\mathrm{UB}}(e)]$.

Fig. 5.1(c) illustrates a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C} in Fig. 5.1(a) with the specification $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$ from Examples 1 and 2.

Chemical Specification To infer a chemical graph $G = (H, \alpha, \beta)$ from a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C}, we finally specify a way of assigning elements in Λ and bond-multiplicities by a *chemical specification* $\sigma_{\alpha\beta}$ that consists of the following:

- Sets $\Lambda^{\mathrm{co}}, \Lambda^{\mathrm{nc}}$ of chemical elements. For a chemical graph G, let $\mathrm{na}_{\mathtt{a}}(G)$ (resp., $\mathrm{na}_{\mathtt{a}}^{\mathrm{co}}(G)$ and $\mathrm{na}_{\mathtt{a}}^{\mathrm{nc}}(G)$) denote the number of vertices (resp., core-vertices and non-core-vertices) in G assigned chemical element $\mathtt{a} \in \Lambda$ (resp., $\mathtt{a} \in \Lambda^{\mathrm{co}}$ and $\mathtt{a} \in \Lambda^{\mathrm{nc}}$);
- Sets $\Lambda_{\mathrm{dg}}^{\mathrm{co}}, \Lambda_{\mathrm{dg}}^{\mathrm{nc}}$ of chemical symbols and Γ^{co}, $\Gamma^{\mathrm{in}}, \Gamma^{\mathrm{ex}}$ of edge-configurations;
- Fix the following sets of adjacency-configurations:
  $\Gamma_{\mathrm{ac}}^{\mathrm{co}} := \{\mathrm{ac}(\gamma) \mid \gamma \in \Gamma^{\mathrm{co}}\}$, $\Gamma_{\mathrm{ac}}^{\mathrm{in}} := \{\mathrm{ac}(\gamma) \mid \gamma \in \Gamma^{\mathrm{in}}\}$,
  $\Gamma_{\mathrm{ac}}^{\mathrm{ex}} := \{\mathrm{ac}(\gamma) \mid \gamma \in \Gamma^{\mathrm{ex}}\}$.
  Define the adjacency-configuration of a core-edge uv to be $(\mathtt{a}, \mathtt{b}, \beta(uv))$ with $\{\mathtt{a}, \mathtt{b}\} = \{\alpha(u), \alpha(v)\}$ and the adjacency-configuration of a directed non-core edge (u, v) to be $(\alpha(u), \alpha(v), \beta(uv))$. Let $\mathrm{ac}_\nu^{\mathrm{co}}(G)$ (resp., $\mathrm{ac}_\nu^{\mathrm{in}}(G)$ and $\mathrm{ac}_\nu^{\mathrm{ex}}(G)$) denote the number of core-edges (resp., directed ρ-internal edges and directed ρ-external edges) in G assigned adjacency-configuration $\nu \in \Gamma_{\mathrm{ac}}^{\mathrm{co}}$ (resp., $\nu \in \Gamma_{\mathrm{ac}}^{\mathrm{in}}$ and $\nu \in \Gamma_{\mathrm{ac}}^{\mathrm{ex}}$);
- Subsets $\Lambda^*(v)$, $v \in V_{\mathrm{C}}$, of elements that are allowed to be assigned to vertex $v \in V_{\mathrm{C}}$;

- Lower and upper bound functions $\mathrm{na}_{\mathrm{LB}}, \mathrm{na}_{\mathrm{UB}} : \Lambda \to [1, n^*]$ and $\mathrm{na}^{\mathrm{t}}_{\mathrm{LB}}, \mathrm{na}^{\mathrm{t}}_{\mathrm{UB}} : \Lambda^{\mathrm{t}} \to [1, n^*]$ (resp., $\mathrm{ns}_{\mathrm{LB}}, \mathrm{ns}_{\mathrm{UB}} : \Lambda_{\mathrm{dg}} \to [1, n^*]$ and $\mathrm{ns}^{\mathrm{t}}_{\mathrm{LB}}, \mathrm{ns}^{\mathrm{t}}_{\mathrm{UB}} : \Lambda^{\mathrm{t}}_{\mathrm{dg}} \to [1, n^*]$), $\mathrm{t} \in \{\mathrm{co}, \mathrm{nc}\}$ on the number of core-vertices and non-core-vertices, respectively, assigned chemical element $\mathtt{a}$ (resp., chemical symbol μ);

- Lower and upper bound functions $\mathrm{ac}^{\mathrm{t}}_{\mathrm{LB}}, \mathrm{ac}^{\mathrm{t}}_{\mathrm{UB}} : \Gamma^{\mathrm{t}}_{\mathrm{ac}} \to \mathbb{Z}_+$ (resp., $\mathrm{ec}^{\mathrm{t}}_{\mathrm{LB}}, \mathrm{ec}^{\mathrm{t}}_{\mathrm{UB}} : \Gamma^{\mathrm{t}} \to \mathbb{Z}_+$), $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$ on the number of core-edges, directed ρ-internal edges and directed ρ-external edges, respectively, assigned adjacency-configuration ν (resp., edge-configurations γ); and

- Lower and upper bound functions $\mathrm{bd}_{m,\mathrm{LB}}, \mathrm{bd}_{m,\mathrm{UB}} : E_{\mathrm{C}} \to \mathbb{Z}_+$, $m \in [2, 3]$, where $\mathrm{bd}_{2,\mathrm{LB}}(e) + \mathrm{bd}_{3,\mathrm{LB}}(e) \leq \ell_{\mathrm{UB}}(e)$, $e \in E_{\mathrm{C}}$.

Example 3 of $\sigma_{\alpha\beta}$. A chemical specification $\sigma_{\alpha\beta}$ to G_{C} in Fig. 5.1(a) is given by Table 5.1.

A $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$-*extension* of G_{C} is a chemical graph $G = (H, \alpha, \beta)$ for a graph $H \in \mathcal{H}(G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$ such that:

1. $\sum_{uv \in E} \beta(uv) \leq \mathrm{val}(\alpha(u))$ for each vertex $u \in V(H)$;

2. $\tau(e) \in \Gamma^{\mathrm{co}}$ (resp., $\tau(e) \in \Gamma^{\mathrm{in}}$ and $\tau(e) \in \Gamma^{\mathrm{ex}}$) for each core-edge (resp., ρ-internal edge and ρ-external edge) e;

3. $\alpha(v) \in \Lambda^*(v)$ for each vertex $v \in V_{\mathrm{C}}$; and

4. the specified lower and upper bounds are satisfied; i.e.,

   - $\mathrm{na}_{\mathtt{a}}(G) \in [\mathrm{na}_{\mathrm{LB}}(\mathtt{a}), \mathrm{na}_{\mathrm{UB}}(\mathtt{a})]$, $\mathtt{a} \in \Lambda$ and $\mathrm{na}^{\mathrm{t}}_{\mathtt{a}}(G) \in [\mathrm{na}^{\mathrm{t}}_{\mathrm{LB}}(\mathtt{a}), \mathrm{na}^{\mathrm{t}}_{\mathrm{UB}}(\mathtt{a})]$, $\mathtt{a} \in \Lambda^{\mathrm{t}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{nc}\}$;
   - $\mathrm{ns}_{\mu}(G) \in [\mathrm{na}_{\mathrm{LB}}(\mu), \mathrm{na}_{\mathrm{UB}}(\mu)]$, $\mu \in \Lambda_{\mathrm{dg}}$, $\mathrm{ns}^{\mathrm{t}}_{\mu}(G) \in [\mathrm{ns}^{\mathrm{t}}_{\mathrm{LB}}(\mu), \mathrm{ns}^{\mathrm{t}}_{\mathrm{UB}}(\mu)]$, $\mu \in \Lambda^{\mathrm{t}}_{\mathrm{dg}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{nc}\}$;
   - $\mathrm{ac}^{\mathrm{t}}_{\nu}(G) \in [\mathrm{ac}^{\mathrm{t}}_{\mathrm{LB}}(\nu), \mathrm{ac}^{\mathrm{t}}_{\mathrm{UB}}(\nu)]$, $\nu \in \Gamma^{\mathrm{t}}_{\mathrm{ac}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$;
   - $\mathrm{ec}^{\mathrm{t}}_{\gamma}(G) \in [\mathrm{ec}^{\mathrm{t}}_{\mathrm{LB}}(\gamma), \mathrm{ec}^{\mathrm{t}}_{\mathrm{UB}}(\gamma)]$, $\gamma \in \Gamma^{\mathrm{t}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$; and
   - $|\{e' \in E(P_e) \mid \beta(e') = m\}| \in [\mathrm{bd}_{m,\mathrm{LB}}(e), \mathrm{bd}_{m,\mathrm{UB}}(e)]$ for each edge $e \in E_{(\geq 2)} \cup E_{(\geq 1)}$.

Fig. 5.1(d) illustrates a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$-extension G of G_{C} in Fig. 5.1(a) with the specifications from Examples 1 to 3.

Table 5.1. Example 3 of a chemical specification $\sigma_{\alpha\beta}$.

$\Lambda = \{\mathtt{C}, \mathtt{N}, \mathtt{O}\}$	$\Lambda_{\mathrm{dg}}^{\mathrm{co}} = \{\mathtt{C2}, \mathtt{C3}, \mathtt{C4}, \mathtt{N2}, \mathtt{O2}\}$	$\Lambda_{\mathrm{dg}}^{\mathrm{nc}} = \{\mathtt{C1}, \mathtt{C2}, \mathtt{C3}, \mathtt{C4}, \mathtt{N1}, \mathtt{N3}, \mathtt{O1}, \mathtt{O2}\}$

$\Gamma_{\mathrm{ac}}^{\mathrm{co}}$	$\nu_1^{\mathrm{co}} = (\mathtt{C}, \mathtt{C}, 1), \nu_2^{\mathrm{co}} = (\mathtt{C}, \mathtt{C}, 2), \nu_3^{\mathrm{co}} = (\mathtt{C}, \mathtt{N}, 1), \nu_4^{\mathrm{co}} = (\mathtt{C}, \mathtt{O}, 1)$
$\Gamma_{\mathrm{ac}}^{\mathrm{in}}$	$\nu_1^{\mathrm{in}} = (\mathtt{C}, \mathtt{C}, 1), \nu_2^{\mathrm{in}} = (\mathtt{C}, \mathtt{C}, 2), \nu_3^{\mathrm{in}} = (\mathtt{C}, \mathtt{O}, 1)$
$\Gamma_{\mathrm{ac}}^{\mathrm{ex}}$	$\nu_1^{\mathrm{ex}} = (\mathtt{C}, \mathtt{C}, 1), \nu_2^{\mathrm{ex}} = (\mathtt{C}, \mathtt{C}, 3), \nu_3^{\mathrm{ex}} = (\mathtt{C}, \mathtt{N}, 1), \nu_4^{\mathrm{ex}} = (\mathtt{N}, \mathtt{C}, 1),$ $\nu_5^{\mathrm{ex}} = (\mathtt{C}, \mathtt{O}, 1), \nu_6^{\mathrm{ex}} = (\mathtt{C}, \mathtt{O}, 2), \nu_7^{\mathrm{ex}} = (\mathtt{O}, \mathtt{C}, 1)$

Γ^{co}	$\gamma_1^{\mathrm{co}} = (\mathtt{C2}, \mathtt{C2}, 1), \gamma_2^{\mathrm{co}} = (\mathtt{C2}, \mathtt{C3}, 1), \gamma_3^{\mathrm{co}} = (\mathtt{C2}, \mathtt{C3}, 2), \gamma_4^{\mathrm{co}} = (\mathtt{C2}, \mathtt{C4}, 1), \gamma_5^{\mathrm{co}} = (\mathtt{C3}, \mathtt{C3}, 1),$ $\gamma_6^{\mathrm{co}} = (\mathtt{C3}, \mathtt{C3}, 2), \gamma_7^{\mathrm{co}} = (\mathtt{C3}, \mathtt{C4}, 1), \gamma_8^{\mathrm{co}} = (\mathtt{C2}, \mathtt{N2}, 1), \gamma_9^{\mathrm{co}} = (\mathtt{C3}, \mathtt{N2}, 1), \gamma_{10}^{\mathrm{co}} = (\mathtt{C3}, \mathtt{O2}, 1)$
Γ^{in}	$\gamma_1^{\mathrm{in}} = (\mathtt{C2}, \mathtt{C2}, 2), \gamma_2^{\mathrm{in}} = (\mathtt{C3}, \mathtt{C2}, 1), \gamma_3^{\mathrm{in}} = (\mathtt{C3}, \mathtt{C3}, 1), \gamma_4^{\mathrm{in}} = (\mathtt{C2}, \mathtt{O2}, 1), \gamma_5^{\mathrm{in}} = (\mathtt{C3}, \mathtt{O2}, 1)$
Γ^{ex}	$\gamma_1^{\mathrm{ex}} = (\mathtt{C3}, \mathtt{C1}, 1), \gamma_2^{\mathrm{ex}} = (\mathtt{C2}, \mathtt{C1}, 3), \gamma_3^{\mathrm{ex}} = (\mathtt{C3}, \mathtt{C3}, 1), \gamma_4^{\mathrm{ex}} = (\mathtt{C4}, \mathtt{C1}, 1), \gamma_5^{\mathrm{ex}} = (\mathtt{C3}, \mathtt{N1}, 1),$ $\gamma_6^{\mathrm{ex}} = (\mathtt{C3}, \mathtt{N3}, 1), \gamma_7^{\mathrm{ex}} = (\mathtt{C3}, \mathtt{O1}, 2), \gamma_8^{\mathrm{ex}} = (\mathtt{O2}, \mathtt{C2}, 1), \gamma_9^{\mathrm{ex}} = (\mathtt{O2}, \mathtt{C3}, 1), \gamma_{10}^{\mathrm{ex}} = (\mathtt{N3}, \mathtt{C1}, 1)$

$\Lambda^*(u_1) = \{\mathtt{N}\}$, $\Lambda^*(u_8) = \{\mathtt{C}, \mathtt{N}\}$, $\Lambda^*(u_9) = \{\mathtt{C}, \mathtt{O}\}$, $\Lambda^*(u) = \{\mathtt{C}\}$, $u \in V_{\mathrm{C}} \setminus \{u_1, u_8, u_9\}$

	C	N	O
$\mathrm{na}_{\mathrm{LB}}(\lambda)$	27	1	1
$\mathrm{na}_{\mathrm{UB}}(\lambda)$	37	4	8

	C	N	O
$\mathrm{na}_{\mathrm{LB}}^{\mathrm{co}}(\lambda)$	9	1	0
$\mathrm{na}_{\mathrm{UB}}^{\mathrm{co}}(\lambda)$	23	4	5

	C	N	O
$\mathrm{na}_{\mathrm{LB}}^{\mathrm{nc}}(\lambda)$	9	1	2
$\mathrm{na}_{\mathrm{UB}}^{\mathrm{nc}}(\lambda)$	18	3	8

	C1	C2	C3	C4	N1	N2	N3	O1	O2
$\mathrm{ns}_{\mathrm{LB}}(\mu)$	6	7	12	0	0	0	0	0	0
$\mathrm{ns}_{\mathrm{UB}}(\mu)$	10	11	18	2	2	2	2	5	5

	C2	C3	C4	N2	N3	O2
$\mathrm{ns}_{\mathrm{LB}}^{\mathrm{co}}(\mu)$	3	5	0	0	0	0
$\mathrm{ns}_{\mathrm{UB}}^{\mathrm{co}}(\mu)$	8	15	2	2	3	5

	C1	C2	C3	C4	N1	N2	N3	O1	O2
$\mathrm{ns}_{\mathrm{LB}}^{\mathrm{nc}}(\mu)$	6	1	1	0	0	0	0	0	0
$\mathrm{ns}_{\mathrm{UB}}^{\mathrm{nc}}(\mu)$	10	5	5	2	2	2	2	5	5

i	1	2	3	4
$\mathrm{ac}_{\mathrm{LB}}^{\mathrm{co}}(\nu_i^{\mathrm{co}})$	0	0	0	0
$\mathrm{ac}_{\mathrm{UB}}^{\mathrm{co}}(\nu_i^{\mathrm{co}})$	30	10	10	10

i	1	2	3
$\mathrm{ac}_{\mathrm{LB}}^{\mathrm{in}}(\nu_i^{\mathrm{in}})$	0	0	0
$\mathrm{ac}_{\mathrm{UB}}^{\mathrm{in}}(\nu_i^{\mathrm{in}})$	5	5	5

i	1	2	3	4	5	6	7
$\mathrm{ac}_{\mathrm{LB}}^{\mathrm{ex}}(\nu_i^{\mathrm{ex}})$	0	0	0	0	0	0	0
$\mathrm{ac}_{\mathrm{UB}}^{\mathrm{ex}}(\nu_i^{\mathrm{ex}})$	10	10	10	10	10	10	10

i	1	2	3	4	5	6	7	8	9	10
$\mathrm{ec}_{\mathrm{LB}}^{\mathrm{co}}(\gamma_i^{\mathrm{co}})$	0	0	0	0	0	0	0	0	0	0
$\mathrm{ec}_{\mathrm{UB}}^{\mathrm{co}}(\gamma_i^{\mathrm{co}})$	4	15	4	4	10	5	4	4	6	4

i	1	2	3	4	5
$\mathrm{ec}_{\mathrm{LB}}^{\mathrm{in}}(\gamma_i^{\mathrm{in}})$	0	0	0	0	0
$\mathrm{ec}_{\mathrm{UB}}^{\mathrm{in}}(\gamma_i^{\mathrm{in}})$	3	3	3	3	3

i	1	2	3	4	5	6	7	8	9	10
$\mathrm{ec}_{\mathrm{LB}}^{\mathrm{ex}}(\gamma_i^{\mathrm{ex}})$	0	0	0	0	0	0	0	0	0	0
$\mathrm{ec}_{\mathrm{UB}}^{\mathrm{ex}}(\gamma_i^{\mathrm{ex}})$	8	4	4	4	4	4	6	4	4	4

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\mathrm{bd}_{2,\mathrm{LB}}(a_i)$	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
$\mathrm{bd}_{2,\mathrm{UB}}(a_i)$	1	1	0	2	2	0	0	0	0	0	0	1	0	0	0	0
$\mathrm{bd}_{3,\mathrm{LB}}(a_i)$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\mathrm{bd}_{3,\mathrm{UB}}(a_i)$	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

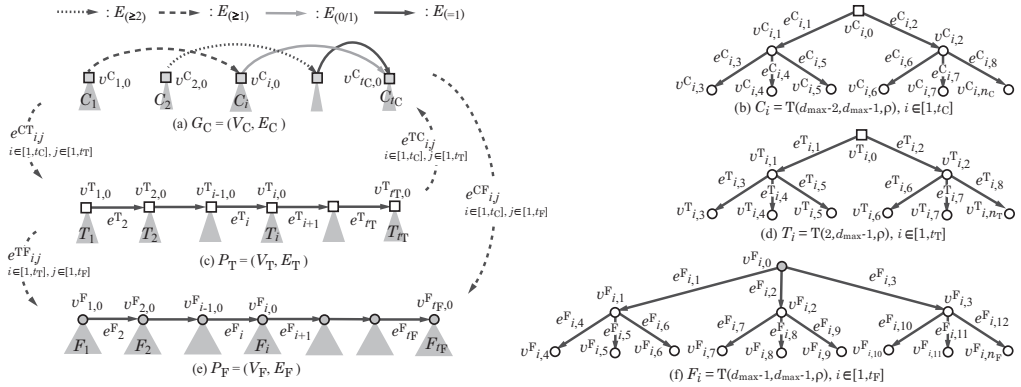


Figure 5.2. An illustration of a scheme graph SG: (a) A seed graph G_{C}; (b) A tree C_i, $i \in [1, t_{\mathrm{C}}]$ rooted at a core-vertex $v^{\mathrm{C}}{}_{i,0} \in V_{\mathrm{C}}$; (c) A path P_{T} of length $t_{\mathrm{T}} - 1$; (d) A tree T_i, $i \in [1, t_{\mathrm{T}}]$ rooted at a core-vertex $v^{\mathrm{T}}{}_{i,0} \in V_{\mathrm{T}}$; (e) A path P_{F} of length $t_{\mathrm{F}} - 1$; (f) A rooted tree F_i, $i \in [1, t_{\mathrm{F}}]$ rooted at a ρ-internal vertex $v^{\mathrm{F}}{}_{i,0} \in V_{\mathrm{F}}$. ©2021 IEEE.

5.4 An MILP Formulation for Stage 4

In this section, we show an outline of an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ in Stage 4 for inferring a chemical ρ-lean cyclic graph $G \in \mathcal{G}(G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$. The details of the MILP can be found in [2].

Scheme Graph Our method first regards a given seed graph G_{C} as a digraph and then adds some more vertices and edges to construct a digraph, called a *scheme graph* $\mathrm{SG} = (\mathcal{V}, \mathcal{E})$ so that any $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C} can be chosen as a subgraph of SG.

For a given specification $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$, define integers that determine the size of a scheme graph SG as follows:
$d_{\max} := 3$ if $\mathrm{dg}^{\mathrm{nc}}_{4,\mathrm{UB}} = 0$; $d_{\max} := 4$ if $\mathrm{dg}^{\mathrm{nc}}_{4,\mathrm{UB}} \geq 1$, $t_{\mathrm{C}} := |V_{\mathrm{C}}|$, $t_{\mathrm{T}} := \mathrm{cs}_{\mathrm{UB}} - |V_{\mathrm{C}}|$ and $t_{\mathrm{F}} := n^* - \mathrm{cs}_{\mathrm{LB}}$. Let n_{C}, n_{T} and n_{F} denote the number of "edges" in the rooted tree $\mathrm{T}(d_{\max} - 2, d_{\max} - 1, \rho)$, $\mathrm{T}(2, d_{\max} - 1, \rho)$ and $\mathrm{T}(d_{\max} - 1, d_{\max} - 1, \rho)$, respectively.

Formally, the scheme graph $\mathrm{SG} = (\mathcal{V}, \mathcal{E})$ is defined with a vertex set $\mathcal{V} = V_{\mathrm{C}} \cup V_{\mathrm{T}} \cup V_{\mathrm{F}} \cup V^{\mathrm{ex}}_{\mathrm{C}} \cup V^{\mathrm{ex}}_{\mathrm{T}} \cup V^{\mathrm{ex}}_{\mathrm{F}}$ and an edge set $\mathcal{E} = E_{\mathrm{C}} \cup E_{\mathrm{T}} \cup E_{\mathrm{F}} \cup E_{\mathrm{CT}} \cup E_{\mathrm{TC}} \cup E_{\mathrm{CF}} \cup E_{\mathrm{TF}} \cup E^{\mathrm{ex}}_{\mathrm{C}} \cup E^{\mathrm{ex}}_{\mathrm{T}} \cup E^{\mathrm{ex}}_{\mathrm{F}}$.

Construction of the core $\mathrm{Cr}(H)$ of a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C}: Denote the vertex set V_{C} and the edge set E_{C} in the seed graph G_{C} by $V_{\mathrm{C}} = \{v^{\mathrm{C}}{}_{i,0} \mid i \in [1, t_{\mathrm{C}}]\}$ and $E_{\mathrm{C}} = \{a_i \mid i \in [1, m_{\mathrm{C}}]\}$, respectively, where V_{C} is

always included in $\mathrm{Cr}(H)$. For including additional core-vertices in $\mathrm{Cr}(H)$, introduce a path $P_{\mathrm{T}} = (V_{\mathrm{T}} = \{v^{\mathrm{T}}{}_{1,0}, v^{\mathrm{T}}{}_{2,0}, \ldots, v^{\mathrm{T}}{}_{t_{\mathrm{T}},0}\}, E_{\mathrm{T}} = \{e^{\mathrm{T}}{}_2, e^{\mathrm{T}}{}_3, \ldots, e^{\mathrm{T}}{}_{t_{\mathrm{T}}}\})$ of length $t_{\mathrm{T}} - 1$ and a set E_{CT} (resp., E_{TC}) of directed edges $e^{\mathrm{CT}}{}_{i,j} = (v^{\mathrm{C}}{}_{i,0}, v^{\mathrm{T}}{}_{j,0})$ (resp., $e^{\mathrm{TC}}{}_{i,j} = (v^{\mathrm{T}}{}_{j,0}, v^{\mathrm{C}}{}_{i,0})$) $i \in [1, t_{\mathrm{C}}]$, $j \in [1, t_{\mathrm{T}}]$. In $\mathrm{Cr}(H)$, an edge $a_k = (v^{\mathrm{C}}{}_{i,0}, v^{\mathrm{C}}{}_{i',0}) \in E_{(\geq 2)} \cup E_{(\geq 1)}$ is allowed to be replaced with a path P_k from core-vertex $v^{\mathrm{C}}{}_{i,0}$ to core-vertex $v^{\mathrm{C}}{}_{i',0}$ that visits a set of consecutive vertices $v^{\mathrm{T}}{}_{j,0}, v^{\mathrm{T}}{}_{j+1,0}, \ldots, v^{\mathrm{T}}{}_{j+p,0} \in V_{\mathrm{T}}$ and edge $e^{\mathrm{CT}}{}_{i,j} = (v^{\mathrm{C}}{}_{i,0}, v^{\mathrm{T}}{}_{j,0}) \in E_{\mathrm{CT}}$, then edges $e^{\mathrm{T}}{}_{j+1}, e^{\mathrm{T}}{}_{j+2}, \ldots, e^{\mathrm{T}}{}_{j+p} \in E_{\mathrm{T}}$, and finally, edge $e^{\mathrm{TC}}{}_{i',j+p} = (v^{\mathrm{T}}{}_{j+p,0}, v^{\mathrm{C}}{}_{i',0}) \in E_{\mathrm{TC}}$. The vertices in V_{T} in the path will be core-vertices in $\mathrm{Cr}(H)$.

Construction of paths with ρ-internal edges in a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C}: Introduce a path $P_{\mathrm{F}} = (V_{\mathrm{F}} = \{v^{\mathrm{F}}{}_{1,0}, v^{\mathrm{F}}{}_{2,0}, \ldots, v^{\mathrm{F}}{}_{t_{\mathrm{F}},0}\}, E_{\mathrm{F}} = \{e^{\mathrm{F}}{}_2, e^{\mathrm{F}}{}_3, \ldots, e^{\mathrm{F}}{}_{t_{\mathrm{F}}}\})$ of length $t_{\mathrm{F}} - 1$, a set E_{CF} of directed edges $e^{\mathrm{CF}}{}_{i,j} = (v^{\mathrm{C}}{}_{i,0}, v^{\mathrm{F}}{}_{j,0})$, $i \in [1, t_{\mathrm{C}}]$, $j \in [1, t_{\mathrm{F}}]$, and a set E_{TF} of directed edges $e^{\mathrm{TF}}{}_{i,j} = (v^{\mathrm{T}}{}_{i,0}, v^{\mathrm{F}}{}_{j,0})$, $i \in [1, t_{\mathrm{T}}]$, $j \in [1, t_{\mathrm{F}}]$. In H, a path P with ρ-internal edges that starts from a core-vertex $v^{\mathrm{C}}{}_{i,0} \in V_{\mathrm{C}}$ (resp., $v^{\mathrm{T}}{}_{i,0} \in V_{\mathrm{T}}$) visits a set of consecutive vertices $v^{\mathrm{F}}{}_{j,0}, v^{\mathrm{F}}{}_{j+1,0}, \ldots, v^{\mathrm{F}}{}_{j+p,0} \in V_{\mathrm{F}}$ and edge $e^{\mathrm{CF}}{}_{i,h} = (v^{\mathrm{C}}{}_{i,0}, v^{\mathrm{F}}{}_{j,0}) \in E_{\mathrm{CF}}$ (resp., $e^{\mathrm{TF}}{}_{i,j} = (v^{\mathrm{T}}{}_{i,0}, v^{\mathrm{F}}{}_{j,0}) \in E_{\mathrm{TF}}$) and edges $e^{\mathrm{F}}{}_{j+1}, e^{\mathrm{F}}{}_{j+2}, \ldots, e^{\mathrm{F}}{}_{j+p} \in E_{\mathrm{F}}$. In H, the edges and the vertices (except for $v^{\mathrm{C}}{}_{i,0}$) in the path P are regarded as ρ-internal edges and ρ-internal vertices, respectively.

Construction of ρ-fringe-trees in a $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$-extension H of G_{C}: In H, the root of a ρ-fringe-tree can be any vertex in $V_{\mathrm{C}} \cup V_{\mathrm{T}} \cup V_{\mathrm{F}}$. Let $\mathrm{X} \in \{\mathrm{C}, \mathrm{T}, \mathrm{F}\}$. Introduce a rooted tree X_i, $i \in [1, t_{\mathrm{X}}]$ at each vertex $v^{\mathrm{X}}{}_{i,0}$, where each C_i is isomorphic to $\mathrm{T}(d_{\max} - 2, d_{\max} - 1, \rho)$, each T_i is isomorphic to $\mathrm{T}(2, d_{\max} - 1, \rho)$ and each F_i is isomorphic to $\mathrm{T}(d_{\max} - 1, d_{\max} - 1, \rho)$. The j-th vertex (resp., edge) in each rooted tree X_i is denoted by $v^{\mathrm{X}}{}_{i,j}$ (resp., $e^{\mathrm{X}}{}_{i,j}$). (See Fig. 5.2.) Let $V^{\mathrm{ex}}_{\mathrm{X}}$ and $E^{\mathrm{ex}}_{\mathrm{X}}$ denote the set of non-root vertices $v^{\mathrm{X}}{}_{i,j}$ and the set of edges $e^{\mathrm{X}}{}_{i,j}$ over all rooted trees X_i, $i \in [1, t_{\mathrm{X}}]$. In H, a ρ-fringe-tree is selected as a subtree of X_i, $i \in [1, t_{\mathrm{X}}]$ with root $v^{\mathrm{X}}{}_{i,0}$.

An MILP for Choosing a Chemical Graph from a Scheme Graph

Let K denote the dimension of a feature vector $x = f(G)$ used in constructing a prediction function ψ over a set of chemical graphs G. Based on the scheme graph SG, we obtain the following MILP formulation.

Theorem 5.3. *Let $(\sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$ be a target specification and $|\Gamma| = |\Lambda^{\mathrm{co}}_{\mathrm{dg}}| + |\Lambda^{\mathrm{nc}}_{\mathrm{dg}}| + |\Gamma^{\mathrm{co}}| + |\Gamma^{\mathrm{in}}| + |\Gamma^{\mathrm{ex}}|$ for sets of chemical symbols and edge-configuration in $\sigma_{\alpha\beta}$. Then there is an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ that consists of variable vectors $x \in \mathbb{R}^K$ and $g \in \mathbb{R}^q$ for an integer $q = O(\mathrm{cs}_{\mathrm{UB}}(|E_{\mathrm{C}}| + n^*) + (|E_{\mathrm{C}}| + |\mathcal{V}|)|\Gamma|)$*

and a set $\mathcal{C}_2$ *of* $O([\mathrm{cs}_{\mathrm{UB}}(|E_{\mathrm{C}}| + n^*) + |\mathcal{V}|]|\Gamma|)$ *constraints on* x *and* g *such that:* (x^*, g^*) *is feasible to* $\mathcal{M}(x, g; \mathcal{C}_2)$ *if and only if* g^* *forms a chemical* ρ*-lean graph* $G \in \mathcal{G}(G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$ *such that* $f(G) = x^*$.

Note that our MILP requires only $O(n^*)$ variables and constraints when the branch-parameter ρ, integers $|E_{\mathrm{C}}|$, $\mathrm{cs}_{\mathrm{UB}}$ and $|\Gamma|$ are constant. We explain the basic idea of our MILP in Theorem 5.3. The MILP mainly consists of the following three types of constraints.

C1. Constraints for selecting a ρ-lean graph $H \in \mathcal{H}(G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}})$ as a subgraph of the scheme graph SG;

C2. Constraints for assigning chemical elements to vertices and multiplicity to edges to determine a chemical graph $G = (H, \alpha, \beta)$; and

C3. Constraints for computing descriptors from the selected chemical graph G.

In the constraints of C1, more formally, we prepare the following.

Variables:

A binary variable $v^{\mathrm{X}}(i) \in \{0, 1\}$ for each vertex ${v^{\mathrm{X}}}_i \in V_{\mathrm{X}}$, $\mathrm{X} \in \{\mathrm{C}, \mathrm{T}, \mathrm{F}\}$ so that $v^{\mathrm{X}}(i) = 1 \Leftrightarrow$ vertex ${v^{\mathrm{X}}}_i$ is used in a graph H selected from SG;
a binary variable $e^{\mathrm{X}}(i) \in \{0, 1\}$ (resp., $e^{\mathrm{C}}(i) \in \{0, 1\}$) for each edge ${e^{\mathrm{X}}}_i \in E_{\mathrm{T}} \cup E_{\mathrm{F}}$ (resp., ${e^{\mathrm{C}}}_i = a_i \in E_{(\geq 2)} \cup E_{(\geq 1)} \cup E_{(0/1)}$) so that $e^{\mathrm{X}}(i) = 1 \Leftrightarrow$ edge ${e^{\mathrm{X}}}_i$ is used in a graph H selected from SG. To save the number of variables in our MILP formulation, we do not prepare a binary variable $e^{\mathrm{X}}(i, j) \in \{0, 1\}$ for any edge ${e^{\mathrm{X}}}_{i,j} \in E_{\mathrm{CT}} \cup E_{\mathrm{TC}} \cup E_{\mathrm{CF}} \cup E_{\mathrm{TC}}$, where we represent a choice of edges in these sets by a set of $O(n^*|E_{\mathrm{C}}|)$ variables (see [2] for the details);

Constraints:

Linear constraints so that each ρ-fringe-tree of a graph H from SG is selected as a subtree of some of the rooted trees C_i, $i \in [1, t_{\mathrm{C}}]$, T_i, $i \in [1, t_{\mathrm{T}}]$ and F_i, $i \in [1, t_{\mathrm{F}}]$;
linear constraints such that each edge ${e^{\mathrm{C}}}_i = a_i \in E_{(=1)}$ is always used as a core-edge in H and each edge ${e^{\mathrm{C}}}_i = a_i \in E_{(0/1)}$ is used as a core-edge in H if necessary;
linear constraints such that for each edge $a_k = ({v^{\mathrm{C}}}_i, {v^{\mathrm{C}}}_{i'}) \in E_{(\geq 2)}$, vertex ${v^{\mathrm{C}}}_i \in V_{\mathrm{C}}$ is connected to vertex ${v^{\mathrm{C}}}_{i'} \in V_{\mathrm{C}}$ in H by a path P_k that passes through some core-vertices in V_{T} and edges ${e^{\mathrm{CT}}}_{i,j}, {e^{\mathrm{T}}}_{j+1}, {e^{\mathrm{T}}}_{j+2}, \ldots, {e^{\mathrm{T}}}_{j+p}, {e^{\mathrm{TC}}}_{i',j+p}$ for some integers j and p;
linear constraints such that for each edge $a_k = ({v^{\mathrm{C}}}_i, {v^{\mathrm{C}}}_{i'}) \in E_{(\geq 1)}$, either the edge a_k is used as a core-edge in H or vertex ${v^{\mathrm{C}}}_i \in V_{\mathrm{C}}$ is connected to vertex ${v^{\mathrm{C}}}_{i'} \in V_{\mathrm{C}}$ in H by a path P_k as in the case of edges in $E_{(\geq 2)}$;
linear constraints for selecting a path P with ρ-internal edges ${e^{\mathrm{CF}}}_{i,j}$ (or ${e^{\mathrm{TF}}}_{i,j}$), ${e^{\mathrm{F}}}_{j+1}, {e^{\mathrm{F}}}_{j+2}, \ldots, {e^{\mathrm{F}}}_{j+p}$ for some integers j and p.

In the constraints of C2, we prepare an integer variable $\alpha^{\mathrm{X}}(i,j)$ for each vertex ${v^{\mathrm{X}}}_{i,j} \in \mathcal{V}$, $\mathrm{X} \in \{\mathrm{C},\mathrm{T},\mathrm{F}\}$ in the scheme graph that represents the chemical element $\alpha({v^{\mathrm{X}}}_{i,j}) \in \Lambda$ if ${v^{\mathrm{X}}}_{i,j}$ is in a selected graph H (or $\alpha({v^{\mathrm{X}}}_{i,j}) = 0$ otherwise); integer variables $\beta^{\mathrm{C}} : E_{\mathrm{C}} \to [0,3]$, $\beta^{\mathrm{T}} : E_{\mathrm{T}} \to [0,3]$ and $\beta^{\mathrm{F}} : E_{\mathrm{F}} \to [0,3]$ that represent the bond-multiplicity of edges in $E_{\mathrm{C}} \cup E_{\mathrm{T}} \cup E_{\mathrm{F}}$; and integer variables $\beta^{+}, \beta^{-} : E_{(\geq 2)} \cup E_{(\geq 1)} \to [0,3]$ and $\beta^{\mathrm{in}} : V_{\mathrm{C}} \cup V_{\mathrm{T}} \to [0,3]$ that represent the bond-multiplicity of edges in $E_{\mathrm{CT}} \cup E_{\mathrm{TC}} \cup E_{\mathrm{CF}} \cup E_{\mathrm{TF}}$. This determines a chemical graph $G = (H, \alpha, \beta)$. Also we include constraints for a selected chemical graph G to satisfy the valence condition at each vertex v with the edge-configurations $\tau(e)$ of the edges incident to v and the chemical specification $\sigma_{\alpha\beta}$.

In the constraints of C3, we introduce a variable for each descriptor and constraints with some more variables to compute the value of each descriptor in $f(G)$ for a selected chemical graph G.

5.5 A New Mechanism for Stage 5

This section describes the idea of a new method for Stage 5. Execution of Stage 5 (i.e., generating chemical graphs G^* that satisfy $f(G^*) = x^*$ for a given feature vector $x^* \in \mathbb{Z}_+^K$) is a challenging issue for a relatively large instance with size $n(G^*) \geq 20$. There have been proposed algorithms for Stage 5 for classes of graphs with rank 0 to 2 [15, 46, 48]. All of these are designed based on the branch-and-bound method where an enormous number of chemical graphs are constructed by repeatedly appending and removing a vertex one by one until a target chemical graph is constructed. These algorithms can generate a target chemical graph G^* with size $n(G^*) \leq 20$. To break this barrier, Azam et al. [7] recently employed the dynamic programming method for designing a new algorithm in Stage 5 based on "frequency vectors" (where the frequency vector consists of some descriptors of the feature vector and the rest of the descriptors can be derived from the frequency vector). They defined the frequency vector $f(G)$ of a chemical graph G to be the occurrence of each adjacency-configuration in G. Note that a single frequency vector can represent a large number of chemical graphs. The search space over frequency vectors is much more compact than that of over chemical graphs. They also observed that most of the acyclic chemical compounds in the PubChem database have at most three leaf 2-branches. Their algorithm constructs the frequency vectors of some subtrees of such a chemical acyclic graph G^* without directly building subtrees of G^*, until a required chemical acyclic graph G^* is constructed from a final set of frequency vectors. Given a vector x, their algorithm generates chemical acyclic graphs G^* with at most three leaf

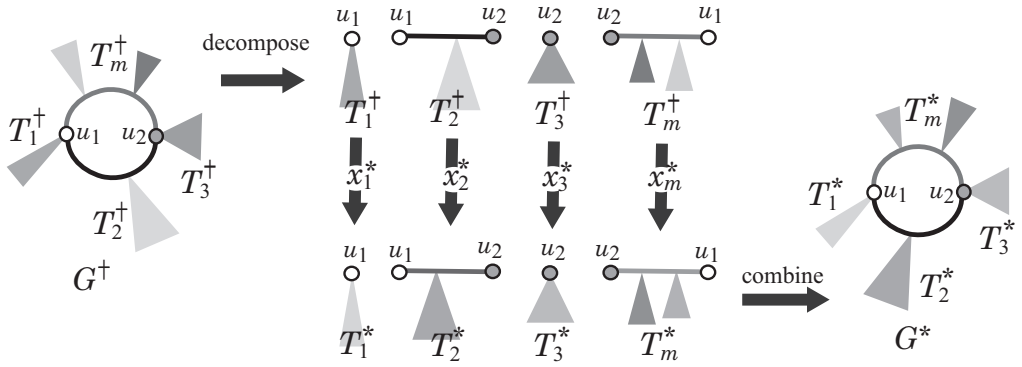


Figure 5.3. An illustration of a new mechanism to Stage 5, where a given chemical graph $G^\dagger$ is decomposed into chemical trees $T_i^\dagger$, $i = 1, 2, \ldots, m$ based on a set $V_B = \{u_1, u_2\}$ of core-vertices and a chemical tree T_i^* such that $f(T_i^*) = x_i^*$ is constructed for each vector $x_i^* = f(T_i^\dagger)$, before a new target graph G^* is obtained as a combination of $T_1^*, \ldots, T_m^*$. ©2021 IEEE.

2-branches such that $f(G^*) = x$ for $n(G^*) \leq 50$.

However, for a class of graphs with a different rank, we may need to design again a new algorithm by the dynamic programming method. Moreover, algorithms for higher ranks can be more complicated and do not run as fast as the algorithm for acyclic graphs due to Azam et al. [7].

In this chapter, as a new mechanism of Stage 5, we adopt an idea of utilizing the chemical graph $G^\dagger \in \mathcal{G}$ obtained as part of a feasible solution of an MILP in Stage 4. The frequency vector $f(G)$ is now set to be the occurrence of each edge-configuration in G. In other words, we modify the chemical graph $G^\dagger$ to generate other chemical graphs G^* that are "chemically isomorphic" to $G^\dagger$ in the sense that $f(G^*) = f(G^\dagger)$ holds. Informally speaking, we reduce the problem of finding such a graph G^* into the problem of generating chemical acyclic graphs with two 2-leaf branches, to which we have obtained an efficient dynamic programming algorithm [7]. We first decompose $G^\dagger$ into a collection of chemical trees $T_1^\dagger, T_2^\dagger, \ldots, T_m^\dagger$ such that for a subset V_B of the core-vertices of $G^\dagger$, any tree $T_i^\dagger$ contains at most two vertices in V_B, as illustrated in Fig. 5.3. Let x_i^* denote the feature vector $f(T_i^\dagger)$. For each index i, we generate chemical acyclic graphs T_i^* such that $f(T_i^*) = x_i^*$. Finally, we combine the generated chemical trees $T_1^*, T_2^*, \ldots, T_m^*$ to construct a chemical cyclic graph G^* such that $f(G^*) = \sum_{i \in [1,m]} x_i^* = f(G^\dagger)$. (See [2] for the details on the dynamic programming algorithm.)

5.6 Experimental Results

We implemented our method of Stages 1 to 5 for inferring chemical 2-lean graphs and conducted experiments to evaluate the computational efficiency. We executed the experiments on a PC with Processor: 3.0 GHz Core i7-9700, Memory: 16 GB RAM DDR4. We used ChemDoodle version 10.2.0 for constructing 2D drawings of chemical graphs.

To conduct experiments for Stages 1 to 5, we selected three chemical properties π: octanol/water partition coefficient (Kow), boiling point (BP) and melting point (MP).

Results on Phase 1.

We implemented Stages 1, 2, and 3, in Phase 1 as follows.

Stage 1. We set a graph class $\mathcal{G}$ to be the set of all chemical graphs with rank at least 1, and set a branch-parameter ρ to be 2. For each property $\pi \in \{$Kow, BP, MP$\}$, we first select a set Λ of chemical elements and then collect a data set D_π on chemical cyclic graphs over the set Λ of chemical elements provided by HSDB from PubChem [27].

Table 5.2 shows the size and range of data sets that we prepared for each chemical property in Stage 1, where we denote the following:

- Λ: the set of selected chemical elements (hydrogen atoms are added at the final stage); Λ is one of the following 2 datasets:
  $\Lambda_1 = \{\mathtt{C}, \mathtt{N}, \mathtt{O}\}$, $\Lambda_2 = \{\mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}, \mathtt{Cl}\}$
- $|D_\pi|$: the size of data set D_π over Λ for property π;
- $|\Gamma^{\mathrm{co}}|$, $|\Gamma^{\mathrm{in}}|$, $|\Gamma^{\mathrm{ex}}|$: the number of different edge-configurations of core-edges, 2-internal edges and 2-external edges over the compounds in D_π;
- $[\underline{n}, \overline{n}]$, $[\underline{\mathrm{cs}}, \overline{\mathrm{cs}}]$, $[\underline{\mathrm{ch}}, \overline{\mathrm{ch}}]$, $[\underline{\mathrm{bl}}, \overline{\mathrm{bl}}]$: the minimum and maximum values of $n(G)$, $\mathrm{cs}(G)$, $\mathrm{ch}(G)$ and $\mathrm{bl}_2(G)$ over the compounds G in D_π; and
- $[\underline{a}, \overline{a}]$: the minimum and maximum values of $a(G)$ in π over compounds G in D_π.

Stage 2. We used a feature function f that consists of the descriptors defined in Section 5.2.

Stage 3. We used `scikit-learn` version 0.23.2 with Python 3.8.5, MLPRegressor and ReLU activation function to construct ANNs $\mathcal{N}$. We evaluated the resulting prediction function $\psi_{\mathcal{N}}$ with cross-validation over five subsets $D_\pi^{(i)}$, $i \in [1, 5]$ of a given data set D_π.

Table 5.2. Data Sets for Stage 1 in Phase 1. ©2021 IEEE.

π	Λ	$\lvert D_\pi \rvert$	$\lvert\Gamma^{\rm co}\rvert$, $\lvert\Gamma^{\rm in}\rvert$, $\lvert\Gamma^{\rm ex}\rvert$	$[\underline{n}, \overline{n}]$	$[\underline{\rm cs}, \overline{\rm cs}]$	$[\underline{\rm ch}, \overline{\rm ch}]$	$[\underline{\rm bl}, \overline{\rm bl}]$	$[\underline{a}, \overline{a}]$
Kow	Λ_1	424	23, 19, 41	[5, 58]	[3, 43]	[0, 19]	[0, 2]	[-7.53, 13.45]
Kow	Λ_2	580	27, 24, 59	[5, 69]	[3, 43]	[0, 19]	[0, 5]	[-7.53, 13.45]
Bp	Λ_1	175	19, 13, 30	[5, 30]	[3, 24]	[0, 12]	[0, 2]	[31.5, 470.0]
Bp	Λ_2	219	20, 14, 39	[5, 30]	[3, 24]	[0, 12]	[0, 2]	[31.5, 470.0]
Mp	Λ_1	256	22, 15, 36	[4, 122]	[3, 87]	[0, 28]	[0, 3]	[-142.5, 300.0]
Mp	Λ_2	340	25, 19, 48	[4, 122]	[3, 87]	[0, 28]	[0, 3]	[-142.5, 300.0]

Table 5.3. Results of Stages 2 and 3 in Phase 1. ©2021 IEEE.

π	Λ	Architecture	L-Time	test R^2 ave.	best
Kow	Λ_1	$(118, 13, 1)$	0.55	0.952	0.961
Kow	Λ_2	$(149, 13, 1)$	0.64	0.932	0.948
Bp	Λ_1	$(97, 11, 1)$	4.90	0.814	0.893
Bp	Λ_2	$(112, 13, 13, 1)$	3.42	0.722	0.871
Mp	Λ_1	$(108, 42, 10, 1)$	0.63	0.740	0.878
Mp	Λ_2	$(130, 64, 1)$	1.73	0.800	0.859

Table 5.3 shows the results on Stages 2 and 3, where we denote the following: K: the number of descriptors for the chemical compounds in data set D_π for property π; Λ: the set of selected chemical elements (hydrogen atoms are added at the final stage); Architecture: $(K, a, 1)$ (resp., $(K, a_1, a_2, 1)$) consists of an input layer with K nodes, a hidden layer with a nodes (resp., two hidden layers with a_1 and a_2 nodes, respectively) and an output layer with a single node, where K is equal to the number of descriptors in the feature vector; L-time: the average time (sec.) to construct ANNs for each trial; and test R^2 (ave.), test R^2 (best): the average value and the largest value of coefficient of determination over the five tests.

From Table 5.3, we see that the execution of Stage 3 was considerably successful, where the best of test R^2 is around 0.86 to 0.96 for all three chemical properties.

Results on Phase 2.

We prepared the following instances (a)-(d) for conducting experiments of

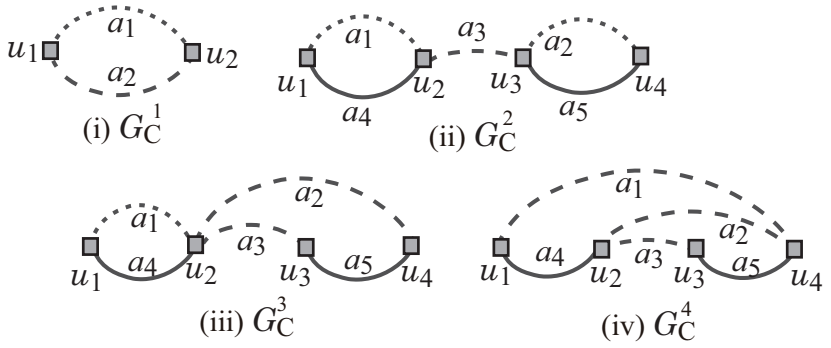


Figure 5.4. An illustration of seed graphs: (i) A monocyclic graph G_{C}^1; (ii) A rank-2 cyclic graph G_{C}^2 with two vertex-disjoint cycles; (iii) A rank-2 cyclic graph G_{C}^3 with two disjoint cycles sharing a vertex; (iv) A rank-2 cyclic graph G_{C}^4 with three cycles.

Stages 4 and 5 in Phase 2.

(a) $I_{\mathrm{a}} = (G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$: The instance used in Section 5.3 to explain the target specification.

(b) $I_{\mathrm{b}}^i = (G_{\mathrm{C}}^i, \sigma_{\mathrm{co}}^i, \sigma_{\mathrm{nc}}^i, \sigma_{\alpha\beta}^i)$, $i = 1, 2, 3, 4$: An instance for inferring chemical graphs with rank at most 2. Instance I_{b}^1 is given by the monocyclic seed graph G_{C}^1 in Fig. 5.4(i) and Instances I_{b}^i, $i = 2, 3, 4$ are given by the rank-2 seed graph G_{C}^i, $i = 2, 3, 4$ in Fig. 5.4(ii)-(iv). See [2] for the details of the specification $(\sigma_{\mathrm{co}}^i, \sigma_{\mathrm{nc}}^i, \sigma_{\alpha\beta}^i)$.

We define instances in (c) and (d) in order to find chemical graphs that have an intermediate structure of given two chemical 2-lean cyclic graphs $G_A = (H_A = (V_A, E_A), \alpha_A, \beta_A)$ and $G_B = (H_B = (V_B, E_B), \alpha_B, \beta_B)$. Let Λ_A^{co} and Λ_A^{nc} denote the sets of chemical elements of the core-vertices and the non-core-vertices in G_A and Γ_A^{co}, Γ_A^{in} and Γ_A^{ex} denote the sets of edge-configurations of the core-edges, the 2-internal edges and the 2-external edges in G_A, respectively. Analogously define sets Λ_B^{co}, Λ_B^{nc}, Γ_B^{co}, Γ_B^{in} and Γ_B^{ex} in G_B.

(c) $I_{\mathrm{c}} = (G_{\mathrm{C}}, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$: An instance aimed to infer a chemical graph $G^\dagger$ such that the core of $G^\dagger$ is equal to the core of G_A and the frequency of each edge-configuration in the non-core of $G^\dagger$ is equal to that of G_B. We use chemical compounds CID 24822711 and CID 59170444 in Fig. 5.5(a) and (b) for G_A and G_B, respectively. Set a seed graph $G_{\mathrm{C}} = (V_{\mathrm{C}}, E_{\mathrm{C}} = E_{(=1)})$ to be the core C_A of G_A; Set $n_{\mathrm{LB}} := n^* := \mathrm{cs}(G_A) + (n(G_B) - \mathrm{cs}(G_A))$, $\Lambda^{\mathrm{co}} := \Lambda_A^{\mathrm{co}}$, $\Lambda^{\mathrm{nc}} := \Lambda_B^{\mathrm{nc}}$ and $\Lambda^*(v) := \{\alpha_A(v)\}$, $v \in V_{\mathrm{C}}$; Set $\mathrm{ec}_{\mathrm{LB}}^{\mathrm{co}}(\gamma) = \mathrm{ec}_{\mathrm{UB}}^{\mathrm{co}}(\gamma)$ to be the

Table 5.4. Features of test instances, where $\Lambda = \{\mathtt{C}, \mathtt{O}, \mathtt{N}\}$ for all instances.

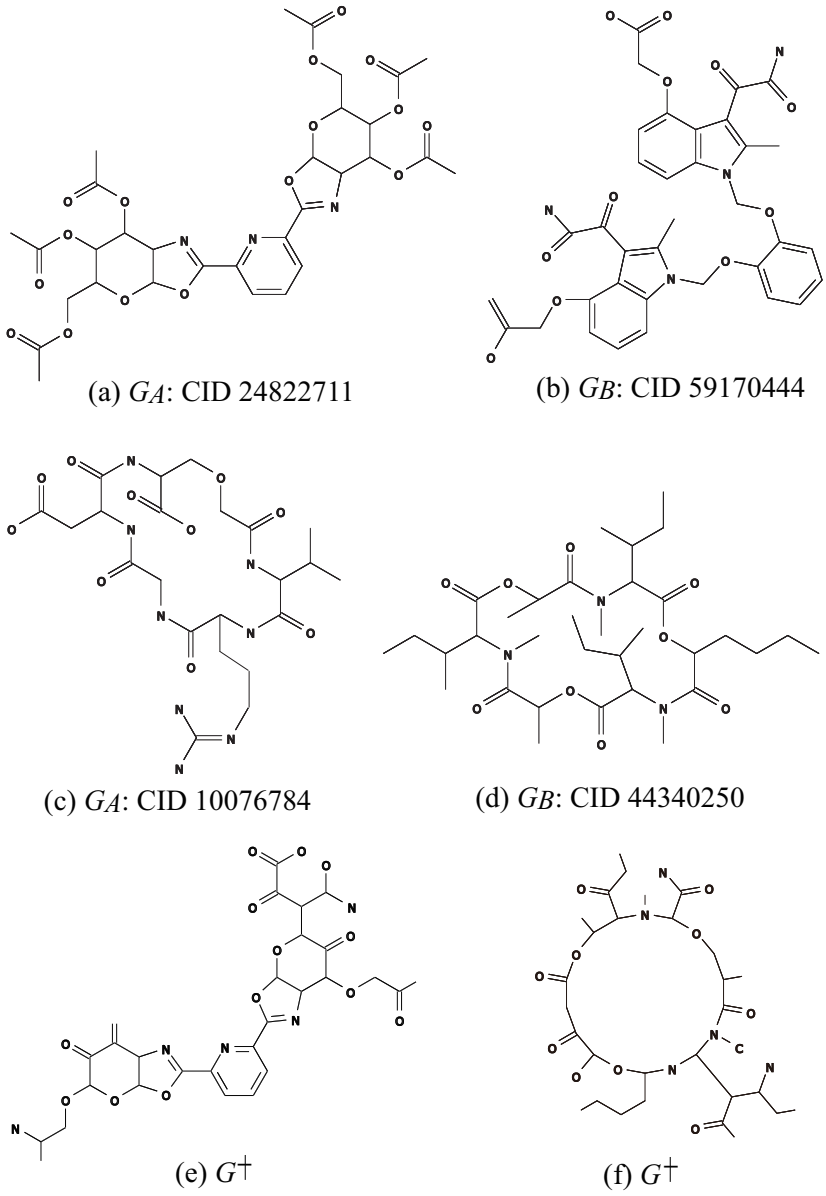
	$\Gamma^{\mathrm{co}}, \Gamma^{\mathrm{in}}, \Gamma^{\mathrm{ex}}$	n_{LB}, n^*	$\mathrm{cs}_{\mathrm{LB,UB}}$	$\mathrm{ch}_{\mathrm{LB,UB}}$	$\mathrm{bl}_{\mathrm{LB,UB}}$
I_{a}	10, 5, 10	30, 50	20, 28	4, 6	2, 10
I_{b}^1	28, 46, 74	38, 38	6, 6	1, 5	0, 22
I_{b}^2	28, 46, 74	50, 50	30, 30	1, 5	0, 34
I_{b}^3	28, 46, 74	50, 50	30, 30	1, 5	0, 34
I_{b}^4	28, 46, 74	50, 50	30, 30	1, 5	0, 34
I_{c}	8, 3, 7	46, 46	24, 24	0, 4	0, 24
I_{d}	7, 4, 11	40, 45	18, 18	0, 5	0, 32

number of core-edges with $\gamma \in \Gamma^{\mathrm{co}}$ in G_A and $\mathrm{ec}_{\mathrm{LB}}^{\mathrm{in}}(\gamma) = \mathrm{ec}_{\mathrm{UB}}^{\mathrm{in}}(\gamma)$ (resp., $\mathrm{ec}_{\mathrm{LB}}^{\mathrm{ex}} = \mathrm{ec}_{\mathrm{UB}}^{\mathrm{ex}}(\gamma)$) to be the number of 2-internal edges (resp., 2-external edges) in G_B with edge-configuration γ.

(d) $I_{\mathrm{d}} = (G_{\mathrm{C}}^1, \sigma_{\mathrm{co}}, \sigma_{\mathrm{nc}}, \sigma_{\alpha\beta})$: An instance aimed to infer a chemical monocyclic graph $G^\dagger$ such that the frequency vector of edge-configurations in $G^\dagger$ is a vector obtained by merging those of G_A and G_B. We use chemical monocyclic compounds CID 10076784 and CID 44340250 in Fig. 5.5(c) and (d) for G_A and G_B, respectively. Set a seed graph to be the monocyclic seed graph G_{C}^1 in Fig. 5.4(i); Set $n_{\mathrm{LB}} := \min\{n(G_A), n(G_B)\}$; $n^* := \max\{n(G_A), n(G_B)\}$, $\Lambda^{\mathrm{co}} := \Lambda_A^{\mathrm{co}} \cup \Lambda_B^{\mathrm{co}}$ and $\Lambda^{\mathrm{nc}} := \Lambda_A^{\mathrm{nc}} \cup \Lambda_B^{\mathrm{nc}}$; For each edge-configuration $\gamma \in \Gamma^{\mathrm{co}}$ (resp., $\gamma \in \Gamma^{\mathrm{in}}, \Gamma^{\mathrm{ex}}$), let $x_A^*(\gamma^{\mathrm{co}})$ (resp., $x_A^*(\gamma^{\mathrm{in}}), x_A^*(\gamma^{\mathrm{ex}})$) denote the number of core-edges (resp., 2-internal edges and 2-external edges) with γ; Analogously, define $x_B^*(\gamma^{\mathrm{t}})$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$; For each edge-configuration $\gamma^{\mathrm{t}} \in \Gamma^{\mathrm{t}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$, let $x_{\min}^*(\gamma^{\mathrm{t}}) := \min\{x_A^*(\gamma^{\mathrm{t}}), x_B^*(\gamma^{\mathrm{t}})\}$ and $x_{\max}^*(\gamma^{\mathrm{t}}) := \max\{x_A^*(\gamma^{\mathrm{t}}), x_B^*(\gamma^{\mathrm{t}})\}$; Set $\mathrm{ec}_{\mathrm{LB}}^{\mathrm{t}}(\gamma) := \lfloor (3/4)x_{\min}^*(\gamma^{\mathrm{t}}) + (1/4)x_{\max}^*(\gamma^{\mathrm{t}}) \rfloor$, $\mathrm{ec}_{\mathrm{UB}}^{\mathrm{t}}(\gamma) := \lceil (1/4)x_{\min}^*(\gamma^{\mathrm{t}}) + (3/4)x_{\max}^*(\gamma^{\mathrm{t}}) \rceil$, $\gamma^{\mathrm{t}} \in \Gamma^{\mathrm{t}}$, $\mathrm{t} \in \{\mathrm{co}, \mathrm{in}, \mathrm{ex}\}$.

Table 5.4 shows the features of the seven test instances, where we denote the following: Γ^{co}, Γ^{in}, Γ^{ex}: the number of different edge-configurations of core-edges, 2-external edges and 2-external edges for inferring a target graph; and n_{LB}, n^*, $\mathrm{cs}_{\mathrm{LB}}, \mathrm{cs}_{\mathrm{UB}}$, $\mathrm{ch}_{\mathrm{LB}}, \mathrm{ch}_{\mathrm{UB}}$, $\mathrm{bl}_{\mathrm{LB}}, \mathrm{bl}_{\mathrm{UB}}$: the lower and upper bounds on $n(G^\dagger)$, $\mathrm{cs}(G^\dagger)$, $\mathrm{ch}(G^\dagger)$ and $\mathrm{bl}_2(G^\dagger)$ for inferring a target graph $G^\dagger$.

To solve an MILP formulation in Stage 4, we used `CPLEX` version 12.10. Tables 5.5 to 5.7 show the results on Stages 4 and 5, where we denote the following:



(a) G_A: CID 24822711 (b) G_B: CID 59170444

(c) G_A: CID 10076784 (d) G_B: CID 44340250

(e) $G^\dagger$ (f) $G^\dagger$

Figure 5.5. An illustration of chemical compounds: (a) G_A: CID 24822711; (b) G_B: CID 59170444; (c) G_A: CID 10076784; (d) G_B: CID 44340250; (e) $G^\dagger$ inferred from $I_{\rm c}$ with $y^* = 0.82$ of $\mathrm{K_{ow}}$; (f) $G^\dagger$ inferred from $I_{\rm d}$ with $y^* = 220$ of BP.

y^*: a target value in $[\underline{a}, \overline{a}]$ for a property π; #v (resp., #c): the number of variables (resp., constraints) in the MILP $\mathcal{M}(x, y, g; \mathcal{C}_1, \mathcal{C}_2)$ in Stage 4; IP-time: the time (sec.) to solve the MILP in Stage 4; n, cs, ch, bl: $n(G^\dagger)$, $\mathrm{cs}(G^\dagger)$, $\mathrm{ch}(G^\dagger)$ and $\mathrm{bl}_2(G^\dagger)$ in the chemical 2-lean cyclic graph $G^\dagger$ inferred in Stage 4; DP-time: the

Table 5.5. Results of Stages 4 and 5 for Kow. ©2021 IEEE.

instance	y^*	#v	#c	IP-time	n	cs	ch	bl	DP-time	G-LB	#G
I_{a}	3.00	13801	11356	44.1	47	22	5	3	0.070	2	2
I_{b}^1	2.80	43176	11202	561.2	38	6	5	4	0.112	8	2
I_{b}^2	2.80	50565	16236	1523.4	50	30	2	0	0.127	3.2×10^6	100
I_{b}^3	2.80	50634	16249	1214.1	50	30	2	0	0.199	1.0×10^5	100
I_{b}^4	2.80	50703	16260	1143.9	50	30	2	0	1.940	7.6×10^7	100
I_{c}	0.82	10348	9746	19.8	46	24	4	3	0.129	7	6
I_{d}	0.50	13858	11259	345.3	41	18	4	3	0.179	1.8×10^8	100

Table 5.6. Results of Stages 4 and 5 for Bp. ©2021 IEEE.

instance	y^*	#v	#c	IP-time	n	cs	ch	bl	DP-time	G-LB	#G
I_{a}	682	13780	11293	27.1	43	23	4	3	0.069	8	4
I_{b}^1	220	43155	11139	648.4	38	6	4	1	0.109	108	7
I_{b}^2	220	50544	16173	12058.9	50	30	2	0	0.137	1296	48
I_{b}^3	220	50613	16186	7206.3	50	30	2	0	0.169	1.5×10^7	100
I_{b}^4	220	50682	16197	4981.0	50	30	4	1	0.008	6.0×10^4	100
I_{c}	630	10327	9683	2.39	46	24	4	4	0.067	6	2
I_{d}	220	13837	11196	121.8	45	18	4	3	0.551	5.4×10^8	100

running time (sec.) to execute the dynamic programming algorithm in Stage 5; G-LB: a lower bound on the number of all chemical isomers G^* of $G^\dagger$; and #G: the number of all (or up to 100) chemical isomers G^* of $G^\dagger$ generated in Stage 5.

Fig. 5.5(e) illustrates the chemical graph $G^\dagger$ inferred from instance I_{c} with $y^* = 0.82$ of Kow in Table 5.5.

Fig. 5.5(f) illustrates the chemical graph $G^\dagger$ inferred from instance I_{d} with $y^* = 220$ of Bp in Table 5.6.

Recall that I_{b}^1 (resp., I_{b}^i, $i = 2, 3, 4$) is an instance for inferring a chemical monocyclic graph (resp., a chemical rank-2 graph) $G^\dagger$ where the sizes $n(G)$, $\mathrm{cs}(G)$, and $\mathrm{ch}(G)$ are specified. Ito et al. [25] and Zhu et al. [61] conducted similar experiments for inferring chemical monocyclic and rank-2 chemical graphs $G^\dagger$, respectively, where their feature vector contains adjacency-configuration as a descriptor, which is simpler than edge-configuration used as a descriptor in our feature vector. They solved instances with up to $n(G^\dagger) = 30$. From Tables 5.5

Table 5.7. Results of Stages 4 and 5 for MP. ©2021 IEEE.

instance	y^*	#v	#c	IP-time	n	cs	ch	bl	DP-time	G-LB	#G
$I_{\rm a}$	284	13699	11119	10.7	49	23	4	4	0.176	12	10
$I_{\rm b}^1$	40	43074	10965	57.2	38	6	5	6	0.173	1200	40
$I_{\rm b}^2$	40	50463	15999	168.8	50	30	5	4	0.237	9.5×10^5	100
$I_{\rm b}^3$	40	50532	16012	149.2	50	30	3	2	0.349	2.5×10^8	100
$I_{\rm b}^4$	40	50601	16023	61.5	50	30	2	0	1.730	2.0×10^6	100
$I_{\rm c}$	270	10246	9509	1.71	46	24	4	3	0.065	10	6
$I_{\rm d}$	240	13756	11022	27.9	44	18	4	4	0.753	2.3×10^7	100

to 5.7, we observed that our MILP formulation successfully inferred monocyclic and rank-2 chemical graphs $G^\dagger$ with up to $n(G^\dagger) = 50$ even for our new feature vector containing the edge-configuration descriptor. Furthermore, both Ito et al. [25] and Zhu et al. [61] employed in Stage 5 a branching type algorithm for graph enumeration, which failed to obtain graph structures for instances with more than 15 non-hydrogen atoms. This is a foreseeable consequence of the combinatorial explosion that occurs with branching-type algorithms. On the other hand, the dynamic-programming based algorithm that we employed in this work could efficiently construct many graphs out of a huge possible number within well less than a second of computation time.

Table 5.8 shows the results on Stage 4 when we change the target value y^* for instance $I_{\rm a}$, where we denote the following:

- y^*: a target value for property MP, where we use values outside the range $[\underline{a}, \overline{a}] = [-142.5, 300.0]$ in Table 5.2 to observe the computational behavior of the MILP $\mathcal{M}(x, y, g; \mathcal{C}_1, \mathcal{C}_2)$;
- IP-time: the time (sec.) to solve an MILP in Stage 4;
- status: F means that the MILP was solved and a feasible solution $G^\dagger$ was obtained; inF means that the MILP was solved and no feasible solution existed; and
- n: the number of non-hydrogen atoms in $G^\dagger$ when the status is F;

From Table 5.8, we see that the instance I_a is infeasible for a target value y^* smaller than 0 or larger than 350, where the infeasibility indicated that I_a admits no chemical graph $G^\dagger$ such that $\psi_{\mathcal{N}}(f(G^\dagger)) = y^*$. We also observe that solving an MILP takes a larger computation time for infeasible instances.

Table 5.8. Results of Stage 4 for instance I_a and property Mp. ©2021 IEEE.

y^*	-150	-100	0	100	150	200	250	300	350	850	1000
IP-time	322.1	329.1	432.5	85.1	20.7	27.6	26.2	11.7	68.1	156.5	681.2
status	inF	inF	inF	F	F	F	F	F	F	inF	inF
n	-	-	-	41	38	48	48	49	49	-	-

5.7 Concluding Remarks

In this chapter, we employed the new mechanism of utilizing a target chemical graph $G^\dagger$ obtained in Stage 4 of the framework for inverse QSAR to generate a larger number of target graphs G^* in Stage 5. We showed that a family of graphs G^* that are chemically isomorphic to $G^\dagger$ can be obtained by the dynamic programming algorithm (see [2] for the details). Based on the new mechanism of Stage 5, we proposed a target specification on a seed graph as a flexible way of specifying a family of target chemical graphs. With this specification, we can realize requirements on partial topological substructure of the core of graphs and partial assignment of chemical elements and bond-multiplicity within the framework for inverse QSAR by ANNs and MILPs.

We have implemented the proposed method to construct a system for inferring chemical compounds with a prescribed topological substructure. The results of computational experiments using such chemical properties as octanol/water partition coefficient, boiling point, and melting point, suggest that the proposed system can infer chemical graphs with 50 non-hydrogen atoms.

The current topological specification proposed in this chapter does not allow to fix part of the non-core structure of a graph beyond the non-core specification outlined in Section 5.3. We remark that it is not technically difficult to extend the MILP formulation in Section 5.4 so that a more general specification for the non-core structure can be handled.

6 Two-Layered Model with Linear Regression

6.1 Introduction

After the model of Chapter 5 had been developed, Shi et al. [43] proposed a more sophisticated model to deal with an arbitrary graph in the framework called a *two-layered model* to represent the feature of a chemical graph. Also, the set of rules for describing a topological specification was refined so that a prescribed structure can be included in both of the acyclic and cyclic parts of a chemical graph $\mathbb{C}$. In this model, for a chemical graph $\mathbb{C}$ with an integer $\rho \geq 1$, we consider two parts, namely, the exterior and the interior of the hydrogen-suppressed graph $\langle \mathbb{C} \rangle$ that is obtained by removing hydrogens from $\mathbb{C}$. The exterior consists of maximal acyclic induced subgraphs with height at most ρ in $\langle \mathbb{C} \rangle$ and the interior is the connected subgraph of $\langle \mathbb{C} \rangle$ obtained by excluding the exterior. Shi et al. [43] also defined a feature vector $f(\mathbb{C})$ of a chemical graph $\mathbb{C}$ as a combination of the frequency of adjacent atom pairs in the interior and the frequency of chemical acyclic graphs among the set of chemical rooted trees T_u rooted at interior-vertices u. Recently, Tanaka et al. [49] extended the model in order to directly treat a chemical graph with hydrogens so that the feature of the exterior can be represented with more variety of chemical rooted trees.

The contribution of this chapter is consists of two parts. Firstly, we make a slight modification to a model of chemical graphs proposed by Tanaka et al. [49] so that we can treat a chemical element with multi-valence such as sulfur `S` and a chemical graph with cations and anions. Second, we employ linear regression as the learning method in Stage 3. One of the most important factors in the framework is the quality of a prediction function η constructed in Stage 3. Also, overfitting is pointed out as a major issue in ANN-based approaches for QSAR because many parameters need to be optimized for ANNs [16]. In this chapter, to construct a prediction function in Stage 3, we use linear regression instead of ANNs or decision trees. A learning algorithm for an ANN may not find a set of weights and biases that minimizes the error function since the algorithm simply iterates modification of the current weights and biases until it terminates

at a local minimum value, and linear regression is much simpler than ANNs and decision trees and thereby we regard the performance of a prediction function by linear regression as the basis for other more sophisticated machine learning methods. In this chapter, we derive an MILP formulation $\mathcal{M}(x, y; \mathcal{C}_1)$ in Stage 4 to simulate the computation process of a prediction function by linear regression. For an MILP formulation $\mathcal{M}(x, g; \mathcal{C}_2)$ that represents a feature function f and a specification σ in Stage 4, we can use the same formulation proposed by Tanaka et al. [49] with a slight modification. In Stage 5, we can also use the dynamic programming algorithm due to Azam et al. [5] with a slight modification to generate target chemical graphs $\mathbb{C}^*$ and the details are omitted.

We implemented the framework based on the refined two-layered model and a prediction function by linear regression. The results of our computational experiments reveal a set of chemical properties to which a prediction function constructed by linear regression on our feature function performs well, in comparison with ANNs in our previous work. We also observe that chemical graphs with up to 50 non-hydrogen atoms can be inferred by the proposed method.

The rest of this chapter is organized as follows. Section 6.2 reviews the two-layered model and describes our modification. Section 6.3 reviews the idea of linear regression and formulates an MILP $\mathcal{M}(x, y; \mathcal{C}_1)$ that simulates the computing process of a prediction function constructed by linear regression. Section 6.4 reports the results of some computational experiments conducted for 18 chemical properties such as vapor density and optical rotation. Section 6.5 gives conclusions with future work. We refer [63] for the complete MILP formulations.

6.2 Two-layered Model

This section reviews the idea of the two-layered model introduced by Shi et al. [43], and describes our modifications to the model.

Let $\mathbb{C} = (H, \alpha, \beta)$ be a chemical graph and $\rho \geq 1$ be an integer, which is called a *branch-parameter*.

A *two-layered model* of $\mathbb{C}$ introduced by Shi et al. [43] is a partition of the hydrogen-suppressed chemical graph $\langle\mathbb{C}\rangle$ into an "interior" and an "exterior" in the following way. We call a vertex $v \in V(\langle\mathbb{C}\rangle)$ (resp., an edge $e \in E(\langle\mathbb{C}\rangle)$) of G an *exterior-vertex* (resp., *exterior-edge*) if $\mathrm{ht}(v) < \rho$ (resp., e is incident to an exterior-vertex) and denote the sets of exterior-vertices and exterior-edges by $V^{\mathrm{ex}}(\mathbb{C})$ and $E^{\mathrm{ex}}(\mathbb{C})$, respectively and denote $V^{\mathrm{int}}(\mathbb{C}) = V(\langle\mathbb{C}\rangle) \setminus V^{\mathrm{ex}}(\mathbb{C})$ and $E^{\mathrm{int}}(\mathbb{C}) = E(\langle\mathbb{C}\rangle) \setminus E^{\mathrm{ex}}(\mathbb{C})$, respectively. We call a vertex in $V^{\mathrm{int}}(\mathbb{C})$ (resp., an edge in $E^{\mathrm{int}}(\mathbb{C})$) an *interior-vertex* (resp., *interior-edge*). The set $E^{\mathrm{ex}}(\mathbb{C})$ of

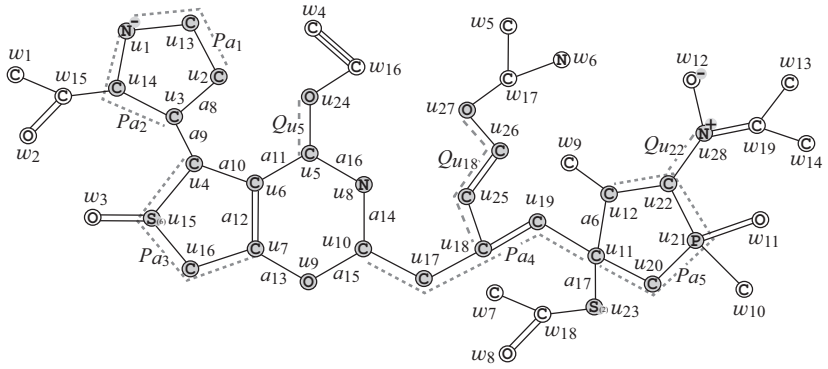


Figure 6.1. An illustration of a hydrogen-suppressed chemical graph $\langle \mathbb{C} \rangle$ obtained from a chemical graph $\mathbb{C}$ with $\mathrm{r}(\mathbb{C}) = 4$ by removing all the hydrogens, where for $\rho = 2$, $V^{\mathrm{ex}}(\mathbb{C}) = \{w_i \mid i \in [1, 19]\}$ and $V^{\mathrm{int}}(\mathbb{C}) = \{u_i \mid i \in [1, 28]\}$.

exterior-edges forms a collection of connected graphs each of which is regarded as a rooted tree T rooted at the vertex $v \in V(T)$ with the maximum $\mathrm{ht}(v)$. Let $\mathcal{T}^{\mathrm{ex}}(\langle \mathbb{C} \rangle)$ denote the set of these chemical rooted trees in $\langle \mathbb{C} \rangle$. The *interior* $\mathbb{C}^{\mathrm{int}}$ of $\mathbb{C}$ is defined to be the subgraph $(V^{\mathrm{int}}(\mathbb{C}), E^{\mathrm{int}}(\mathbb{C}))$ of $\langle \mathbb{C} \rangle$.

Figure 6.1 illustrates an example of a hydrogen-suppressed chemical graph $\langle \mathbb{C} \rangle$. For a branch-parameter $\rho = 2$, the interior of the chemical graph $\langle \mathbb{C} \rangle$ in Figure 6.1 is obtained by removing the set of vertices with degree 1 $\rho = 2$ times; i.e., first remove the set $V_1 = \{w_1, w_2, \ldots, w_{14}\}$ of vertices of degree 1 in $\langle \mathbb{C} \rangle$ and then remove the set $V_2 = \{w_{15}, w_{16}, \ldots, w_{19}\}$ of vertices of degree 1 in $\langle \mathbb{C} \rangle - V_1$, where the removed vertices become the exterior-vertices of $\langle \mathbb{C} \rangle$.

For each interior-vertex $u \in V^{\mathrm{int}}(\mathbb{C})$, let $T_u \in \mathcal{T}^{\mathrm{ex}}(\langle \mathbb{C} \rangle)$ denote the chemical tree rooted at u (where possibly T_u consists of vertex u) and define the *ρ-fringe-tree* $\mathbb{C}[u]$ to be the chemical rooted tree obtained from T_u by putting back the hydrogens originally attached to T_u in $\mathbb{C}$. Let $\mathcal{T}(\mathbb{C})$ denote the set of ρ-fringe-trees $\mathbb{C}[u], u \in V^{\mathrm{int}}(\mathbb{C})$. Figure 6.2 illustrates the set $\mathcal{T}(\mathbb{C}) = \{\mathbb{C}[u_i] \mid i \in [1, 28]\}$ of the 2-fringe-trees of the example $\mathbb{C}$ in Figure 6.1.

Feature Function We extend the feature function of a chemical graph $\mathbb{C}$ introduced by Tanaka et al. [49].

For an integer K, a feature vector $f(\mathbb{C})$ of a chemical graph $\mathbb{C}$ is defined by a *feature function* f that consists of K descriptors. We call $\mathbb{R}^K$ the *feature space*.

Tanaka et al. [49] defined a feature vector $f(\mathbb{C}) \in \mathbb{R}^K$ to be a combination of the frequency of edge-configurations of the interior-edges and the frequency of chemical rooted trees among the set of chemical rooted trees $\mathbb{C}[u]$ over all interior-vertices u. In this chapter, we introduce the rank and the adjacency-

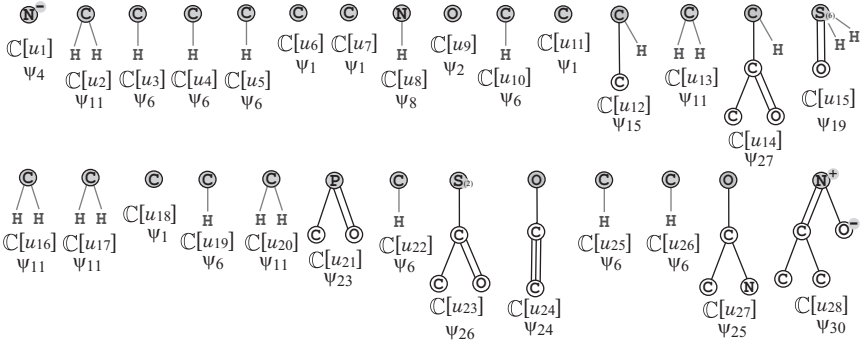


Figure 6.2. The set $\mathcal{T}(\mathbb{C})$ of 2-fringe-trees $\mathbb{C}[u_i], i \in [1, 28]$ of the example $\mathbb{C}$ in Figure 6.1, where the root of each tree is depicted with a gray circle and the hydrogens attached to non-root vertices are omitted in the figure.

configuration of leaf-edges as new descriptors in a feature vector of a chemical graph.

6.2.1 A Full Description of Descriptors

In Shi et al. [43], to represent a feature of the exterior of $\mathbb{C}$, a chemical rooted tree in $\mathcal{T}(\mathbb{C})$ is called a *fringe-configuration* of $\mathbb{C}$. In this chapter, we also represent leaf-edges in the exterior of $\mathbb{C}$. For a leaf-edge $uv \in E(\langle\mathbb{C}\rangle)$ with $\deg_{\langle\mathbb{C}\rangle}(u) = 1$, we define the *adjacency-configuration* of e to be an ordered tuple $(\alpha(u), \alpha(v), \beta(uv))$. Define

$$\Gamma_{\mathrm{ac}}^{\mathrm{lf}} \triangleq \{(\mathtt{a}, \mathtt{b}, m) \mid \mathtt{a}, \mathtt{b} \in \Lambda, m \in [1, \min\{\mathrm{val}(\mathtt{a}), \mathrm{val}(\mathtt{b})\}]\}$$

as a set of possible adjacency-configurations for leaf-edges.

Let π be a chemical property for which we will construct a prediction function η from a feature vector $f(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to a predicted value $y \in \mathbb{R}$ for the chemical property of $\mathbb{C}$.

We first choose a set Λ of chemical elements and then collect a data set D_π of chemical compounds C whose chemical elements belong to Λ, where we regard D_π as a set of chemical graphs $\mathbb{C}$ that represent the chemical compounds C in D_π. To define the interior/exterior of chemical graphs $\mathbb{C} \in D_\pi$, we next choose a branch-parameter ρ, where we recommend $\rho = 2$.

Let $\Lambda^{\mathrm{int}}(D_\pi) \subseteq \Lambda$ (resp., $\Lambda^{\mathrm{ex}}(D_\pi) \subseteq \Lambda$) denote the set of chemical elements used in the set $V^{\mathrm{int}}(\mathbb{C})$ of interior-vertices (resp., the set $V^{\mathrm{ex}}(\mathbb{C})$ of exterior-vertices) of $\mathbb{C}$ over all chemical graphs $\mathbb{C} \in D_\pi$, and $\Gamma^{\mathrm{int}}(D_\pi)$ denote the set of edge-configurations used in the set $E^{\mathrm{int}}(\mathbb{C})$ of interior-edges in $\mathbb{C}$ over all chemical

graphs $\mathbb{C} \in D_\pi$. Let $\mathcal{F}(D_\pi)$ denote the set of chemical rooted trees ψ r-isomorphic to a chemical rooted tree in $\mathcal{T}(\mathbb{C})$ over all chemical graphs $\mathbb{C} \in D_\pi$, where possibly a chemical rooted tree $\psi \in \mathcal{F}(D_\pi)$ consists of a single chemical element $\mathtt{a} \in \Lambda \backslash \{\mathtt{H}\}$.

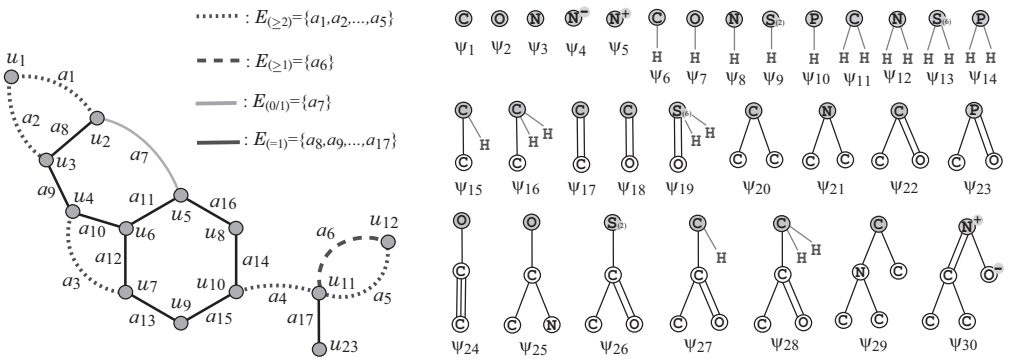
We define an integer encoding of a finite set A of elements to be a bijection $\sigma : A \to [1, |A|]$, where we denote by $[A]$ the set $[1, |A|]$ of integers. Introduce an integer coding of each of the sets $\Lambda^{\mathrm{int}}(D_\pi)$, $\Lambda^{\mathrm{ex}}(D_\pi)$, $\Gamma^{\mathrm{int}}(D_\pi)$ and $\mathcal{F}(D_\pi)$. Let $[\mathtt{a}]^{\mathrm{int}}$ (resp., $[\mathtt{a}]^{\mathrm{ex}}$) denote the coded integer of an element $\mathtt{a} \in \Lambda^{\mathrm{int}}(D_\pi)$ (resp., $\mathtt{a} \in \Lambda^{\mathrm{ex}}(D_\pi)$), $[\gamma]$ denote the coded integer of an element γ in $\Gamma^{\mathrm{int}}(D_\pi)$ and $[\psi]$ denote an element ψ in $\mathcal{F}(D_\pi)$.

Over 99% of chemical compounds $\mathbb{C}$ with up to 100 non-hydrogen atoms in PubChem have degree at most 4 in the hydrogen-suppressed graph $\langle \mathbb{C} \rangle$. We assume that a chemical graph $\mathbb{C}$ treated in this chapter satisfies $\deg_{\langle \mathbb{C} \rangle}(v) \leq 4$ in the hydrogen-suppressed graph $\langle \mathbb{C} \rangle$.

In our model, we use an integer $\mathrm{mass}^*(\mathtt{a}) = \lfloor 10 \cdot \mathrm{mass}(\mathtt{a}) \rfloor$, for each $\mathtt{a} \in \Lambda$.

We define the *feature vector* $f(\mathbb{C})$ of a chemical graph $\mathbb{C} = (H, \alpha, \beta) \in D_\pi$ to be a vector that consists of the following non-negative real descriptors $\mathrm{dcp}_i(\mathbb{C})$, $i \in [1, K]$, where $K = 14 + |\Lambda^{\mathrm{int}}(D_\pi)| + |\Lambda^{\mathrm{ex}}(D_\pi)| + |\Gamma^{\mathrm{int}}(D_\pi)| + |\mathcal{F}(D_\pi)| + |\Gamma^{\mathrm{lf}}_{\mathrm{ac}}|$.

1. $\mathrm{dcp}_1(\mathbb{C})$: the number $|V(H)| - |V_{\mathtt{H}}|$ of non-hydrogen atoms in $\mathbb{C}$.
2. $\mathrm{dcp}_2(\mathbb{C})$: the rank $\mathrm{r}(\mathbb{C})$ of $\mathbb{C}$.
3. $\mathrm{dcp}_3(\mathbb{C})$: the number $|V^{\mathrm{int}}(\mathbb{C})|$ of interior-vertices in $\mathbb{C}$.
4. $\mathrm{dcp}_4(\mathbb{C})$: the average $\overline{\mathrm{ms}}(\mathbb{C})$ of mass* over all atoms in $\mathbb{C}$; i.e., $\overline{\mathrm{ms}}(\mathbb{C}) \triangleq \frac{1}{|V(H)|} \sum_{v \in V(H)} \mathrm{mass}^*(\alpha(v))$.
5. $\mathrm{dcp}_i(\mathbb{C})$, $i = 4 + d, d \in [1, 4]$: the number $\mathrm{dg}^{\overline{\mathtt{H}}}_d(\mathbb{C})$ of non-hydrogen vertices $v \in V(H) \setminus V_{\mathtt{H}}$ of degree $\deg_{\langle \mathbb{C} \rangle}(v) = d$ in the hydrogen-suppressed chemical graph $\langle \mathbb{C} \rangle$.
6. $\mathrm{dcp}_i(\mathbb{C})$, $i = 8 + d, d \in [1, 4]$: the number $\mathrm{dg}^{\mathrm{int}}_d(\mathbb{C})$ of interior-vertices of interior-degree $\deg_{\mathbb{C}^{\mathrm{int}}}(v) = d$ in the interior $\mathbb{C}^{\mathrm{int}} = (V^{\mathrm{int}}(\mathbb{C}), E^{\mathrm{int}}(\mathbb{C}))$ of $\mathbb{C}$.
7. $\mathrm{dcp}_i(\mathbb{C})$, $i = 12 + m$, $m \in [2, 3]$: the number $\mathrm{bd}^{\mathrm{int}}_m(\mathbb{C})$ of interior-edges with bond multiplicity m in $\mathbb{C}$; i.e., $\mathrm{bd}^{\mathrm{int}}_m(\mathbb{C}) \triangleq \{e \in E^{\mathrm{int}}(\mathbb{C}) \mid \beta(e) = m\}$.
8. $\mathrm{dcp}_i(\mathbb{C})$, $i = 14 + [\mathtt{a}]^{\mathrm{int}}$, $\mathtt{a} \in \Lambda^{\mathrm{int}}(D_\pi)$: the frequency $\mathrm{na}^{\mathrm{int}}_{\mathtt{a}}(\mathbb{C}) = |V_{\mathtt{a}}(\mathbb{C}) \cap V^{\mathrm{int}}(\mathbb{C})|$ of chemical element $\mathtt{a}$ in the set $V^{\mathrm{int}}(\mathbb{C})$ of interior-vertices in $\mathbb{C}$.
9. $\mathrm{dcp}_i(\mathbb{C})$, $i = 14 + |\Lambda^{\mathrm{int}}(D_\pi)| + [\mathtt{a}]^{\mathrm{ex}}$, $\mathtt{a} \in \Lambda^{\mathrm{ex}}(D_\pi)$: the frequency $\mathrm{na}^{\mathrm{ex}}_{\mathtt{a}}(\mathbb{C}) = |V_{\mathtt{a}}(\mathbb{C}) \cap V^{\mathrm{ex}}(\mathbb{C})|$ of chemical element $\mathtt{a}$ in the set $V^{\mathrm{ex}}(\mathbb{C})$ of exterior-vertices in $\mathbb{C}$.



(a) A seed graph $G_C=(V_C,E_C)$ (b) A set $\mathcal{F}$ of chemical rooted trees

Figure 6.3. (a) An illustration of a seed graph G_{C} with $\mathrm{r}(G_{\mathrm{C}}) = 5$ where the vertices in V_{C} are depicted with gray circles, the edges in $E_{(\geq 2)}$ are depicted with dotted lines, the edges in $E_{(\geq 1)}$ are depicted with dashed lines, the edges in $E_{(0/1)}$ are depicted with gray bold lines and the edges in $E_{(=1)}$ are depicted with black solid lines; (b) A set $\mathcal{F} = \{\psi_1, \psi_2, \ldots, \psi_{30}\} \subseteq \mathcal{F}(D_\pi)$ of 30 chemical rooted trees $\psi_i, i \in [1, 30]$, where the root of each tree is depicted with a gray circle, where the hydrogens attached to non-root vertices are omitted in the figure.

10. $\mathrm{dcp}_i(\mathbb{C})$, $i = 14 + |\Lambda^{\mathrm{int}}(D_\pi)| + |\Lambda^{\mathrm{ex}}(D_\pi)| + [\gamma]$, $\gamma \in \Gamma^{\mathrm{int}}(D_\pi)$: the frequency $\mathrm{ec}_\gamma(\mathbb{C})$ of edge-configuration γ in the set $E^{\mathrm{int}}(\mathbb{C})$ of interior-edges in $\mathbb{C}$.

11. $\mathrm{dcp}_i(\mathbb{C})$, $i = 14 + |\Lambda^{\mathrm{int}}(D_\pi)| + |\Lambda^{\mathrm{ex}}(D_\pi)| + |\Gamma^{\mathrm{int}}(D_\pi)| + [\psi]$, $\psi \in \mathcal{F}(D_\pi)$: the frequency $\mathrm{fc}_\psi(\mathbb{C})$ of fringe-configuration ψ in the set of ρ-fringe-trees in $\mathbb{C}$.

12. $\mathrm{dcp}_i(\mathbb{C})$, $i = 14 + |\Lambda^{\mathrm{int}}(D_\pi)| + |\Lambda^{\mathrm{ex}}(D_\pi)| + |\Gamma^{\mathrm{int}}(D_\pi)| + |\mathcal{F}(D_\pi)| + [\nu]$, $\nu \in \Gamma^{\mathrm{lf}}_{\mathrm{ac}}$: the frequency $\mathrm{ac}^{\mathrm{lf}}_\nu(\mathbb{C})$ of adjacency-configuration ν in the set of leaf-edges in $\langle\mathbb{C}\rangle$.

6.2.2 Topological Specification

A topological specification is described as a set of the following rules proposed by Shi et al. [43] and modified by Tanaka et al. [49]:

(i) a *seed graph* G_{C} as an abstract form of a target chemical graph $\mathbb{C}$;

(ii) a set $\mathcal{F}$ of chemical rooted trees as candidates for a tree $\mathbb{C}[u]$ rooted at each exterior-vertex u in $\mathbb{C}$; and

(iii) lower and upper bounds on the number of components in a target chemical graph such as chemical elements, double/triple bounds and the interior-vertices in $\mathbb{C}$.

Figures 6.3(a) and (b) illustrate examples of a seed graph G_{C} and a set $\mathcal{F}$

of chemical rooted trees, respectively. Given a seed graph G_{C}, the interior of a target chemical graph $\mathbb{C}$ is constructed from G_{C} by replacing some edges $a = uv$ with paths P_a between the end-vertices u and v and by attaching new paths Q_v to some vertices v. For example, a chemical graph $\langle\mathbb{C}\rangle$ in Figure 6.1 is constructed from the seed graph G_{C} in Figure 6.3(a) as follows.

- First replace five edges $a_1 = u_1u_2, a_2 = u_1u_3, a_3 = u_4u_7, a_4 = u_{10}u_{11}$ and $a_5 = u_{11}u_{12}$ in G_{C} with new paths $P_{a_1} = (u_1, u_{13}, u_2)$, $P_{a_2} = (u_1, u_{14}, u_3)$, $P_{a_3} = (u_4, u_{15}, u_{16}, u_7)$, $P_{a_4} = (u_{10}, u_{17}, u_{18}, u_{19}, u_{11})$ and $P_{a_5} = (u_{11}, u_{20}, u_{21}, u_{22}, u_{12})$, respectively to obtain a subgraph G_1 of $\langle\mathbb{C}\rangle$.
- Next attach to this graph G_1 three new paths $Q_{u_5} = (u_5, u_{24})$, $Q_{u_{18}} = (u_{18}, u_{25}, u_{26}, u_{27})$ and $Q_{u_{22}} = (u_{22}, u_{28})$ to obtain the interior of $\langle\mathbb{C}\rangle$ in Figure 6.1.
- Finally attach to the interior 28 trees selected from the set $\mathcal{F}$ and assign chemical elements and bond-multiplicities in the interior to obtain a chemical graph $\mathbb{C}$ in Figure 6.1. In Figure 6.2, $\psi_1 \in \mathcal{F}$ is selected for $\mathbb{C}[u_i]$, $i \in \{6, 7, 11\}$. Similarly ψ_2 for $\mathbb{C}[u_9]$, ψ_4 for $\mathbb{C}[u_1]$, ψ_6 for $\mathbb{C}[u_i]$, $i \in \{3, 4, 5, 10, 19, 22, 25, 26\}$, ψ_8 for $\mathbb{C}[u_8]$, ψ_{11} for $\mathbb{C}[u_i]$, $i \in \{2, 13, 16, 17, 20\}$, ψ_{15} for $\mathbb{C}[u_{12}]$, ψ_{19} for $\mathbb{C}[u_{15}]$, ψ_{23} for $\mathbb{C}[u_{21}]$, ψ_{24} for $\mathbb{C}[u_{24}]$, ψ_{25} for $\mathbb{C}[u_{27}]$, ψ_{26} for $\mathbb{C}[u_{23}]$, ψ_{27} for $\mathbb{C}[u_{14}]$ and ψ_{30} for $\mathbb{C}[u_{28}]$.

Our definition of a topological specification is analogous with the one by Tanaka et al. [49] except for a necessary modification due to the introduction of multiple valences of chemical elements, cations and anions (see [63] for a full description of topological specification).

6.3 Linear Regression

Let D be a data set of chemical graphs $\mathbb{C}$ with an observed value $a(\mathbb{C}) \in \mathbb{R}$, where we denote by $a_i = a(\mathbb{C}_i)$ for an indexed graph $\mathbb{C}_i$.

Let f be a feature function that maps a chemical graph $\mathbb{C}$ to a vector $f(\mathbb{C}) \in \mathbb{R}^K$ where we denote by $x_i = f(\mathbb{C}_i)$ for an indexed graph $\mathbb{C}_i$.

For a feature space $\mathbb{R}^K$, a hyperplane is denoted by a pair (w, b) of a vector $w \in \mathbb{R}^K$ and a real $b \in \mathbb{R}$. Given a hyperplane $(w, b) \in \mathbb{R}^{K+1}$, a prediction function $\eta_{w,b} : \mathbb{R}^K \to \mathbb{R}$ is defined by setting

$$\eta_{w,b}(x) \triangleq w \cdot x + b = \sum_{j \in [1,K]} w(j)x(j) + b.$$

We wish to find a hyperplane (w, b) that minimizes the error function $\mathrm{Err}(\eta_{w,b}; D)$. In many cases, a feature vector f contains descriptors that do not play an essential role in constructing a good prediction function. When we solve the minimization

problem, the entries $w(j)$ for some descriptors $j \in [1, K]$ in the resulting hyperplane (w, b) become zero, which means that these descriptors were not necessarily important for finding a prediction function $\eta_{w,b}$. It is proposed that solving the minimization with an additional penalty term τ to the error function often results in more number of entries $w(j) = 0$, reducing a set of descriptors necessary for defining a prediction function $\eta_{w,b}$. For an error function with such a penalty term, a Ridge function $\frac{1}{2|D|}\mathrm{Err}(\eta_{w,b}; D) + \lambda[\sum_{j\in[1,K]} w(j)^2 + b^2]$ [21, 22] and a Lasso function $\frac{1}{2|D|}\mathrm{Err}(\eta_{w,b}; D) + \lambda[\sum_{j\in[1,K]} |w(j)| + |b|]$ [52] are known, where $\lambda \in \mathbb{R}_+$ is a given real number.

Given a prediction function $\eta_{w,b}$, we can simulate a process of computing the output $\eta_{w,b}(x)$ for an input $x \in \mathbb{R}^K$ as an MILP $\mathcal{M}(x, y; \mathcal{C}_1)$ in the framework. By solving such an MILP for a specified target value y^*, we can find a vector $x^* \in \mathbb{R}^K$ such that $\eta_{w,b}(x^*) = y^*$. Instead of specifying a single target value y^*, we use lower and upper bounds $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ on the value $a(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to be inferred. We can control the range between $\underline{y}^*$ and $\overline{y}^*$ for searching a chemical graph $\mathbb{C}$ by setting $\underline{y}^*$ and $\overline{y}^*$ to be close or different values. A desired MILP is formulated as follows.

$\mathcal{M}(x, y; \mathcal{C}_1)$: An MILP formulation for the inverse problem to prediction function

constants:

- A hyperplane (w, b) with $w \in \mathbb{R}^K$ and $b \in \mathbb{R}$;
- Real values $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ such that $\underline{y}^* < \overline{y}^*$;
- A set $I_{\mathbb{Z}}$ of indices $j \in [1, K]$ such that the j-th descriptor $\mathrm{dcp}_j(\mathbb{C})$ is always an integer;
- A set I_+ of indices $j \in [1, K]$ such that the j-th descriptor $\mathrm{dcp}_j(\mathbb{C})$ is always non-negative;
- $\ell(j), u(j) \in \mathbb{R}, j \in [1, K]$: lower and upper bounds on the j-th descriptor;

variables:

- Non-negative integer variable $x(j) \in \mathbb{Z}_+, j \in I_{\mathbb{Z}} \cap I_+$;
- Integer variable $x(j) \in \mathbb{Z}, j \in I_{\mathbb{Z}} \setminus I_+$;
- Non-negative real variable $x(j) \in \mathbb{R}_+, j \in I_+ \setminus I_{\mathbb{Z}}$;
- Real variable $x(j) \in \mathbb{R}, j \in [1, K] \setminus (I_{\mathbb{Z}} \cup I_+)$;

constraints:

$$\ell(j) \leq x(j) \leq u(j), j \in [1, K]; \qquad \underline{y}^* \leq \sum_{j\in[1,K]} w(j)x(j) + b \leq \overline{y}^*,$$

objective function:
none.

The number of variables and constraints in the above MILP formulation is $O(K)$. It is not difficult to see that the above MILP is an NP-hard problem. The entire MILP for Stage 4 consists of the two MILPs $\mathcal{M}(x, y; \mathcal{C}_1)$ and $\mathcal{M}(x, g; \mathcal{C}_2)$ with no objective function. The latter represents the computation process of our feature function f and a given topological specification. See [63] for the details of MILP $\mathcal{M}(x, g; \mathcal{C}_2)$.

6.4 Experimental Results

We implemented our method of Stages 1 to 5 for inferring chemical graphs under a given topological specification and conducted experiments to evaluate the computational efficiency. We executed the experiments on a PC with Processor: Core i7-9700 (3.0 GHz; 4.7 GHz at the maximum) and Memory: 16 GB RAM DDR4.

Results on Phase 1. We have conducted experiments of linear regression for 37 chemical properties among which we report the following 18 properties to which the test coefficient of determination R^2 attains at least 0.8: octanol/water partition coefficient (Kow), heat of combustion (Hc), vapor density (Vd), optical rotation (OptR), electron density on the most positive atom (EDPA), melting point (Mp), heat of atomization (Ha), heat of formation (Hf), internal energy at 0K (U0), energy of lowest unoccupied molecular orbital (Lumo), isotropic polarizability (Alpha), heat capacity at 298.15K (Cv), solubility (Sl), surface tension (SfT), viscosity (Vis), isobaric heat capacities in liquid phase (IhcLiq), isobaric heat capacities in solid phase (IhcSol) and lipophilicity (Lp).

We used data sets provided by HSDB from PubChem [27] for Kow, Hc, Vd and OptR, M. Jalali-Heravi and M. Fatemi [26] for EDPA, Roy and Saha [39] for Mp, Ha and Hf, MoleculeNet [54] for U0, Lumo, Alpha, Cv and Sl, Goussard et al. [19] for SfT and Vis, R. Naef [34] for IhcLiq and IhcSol, and Figshare [53] for Lp.

Properties U0, Lumo, Alpha and Cv share a common original data set D^* with more than 130,000 compounds, and we used a set D_π of 1,000 graphs randomly selected from D^* as a common data set of these four properties π in this experiment.

Stages 1, 2 and 3 in Phase 1 are implemented as follows.

Stage 1. We set a graph class $\mathcal{G}$ to be the set of all chemical graphs with any

graph structure, and set a branch-parameter ρ to be 2.

For each of the properties, we first select a set Λ of chemical elements and then collect a data set D_π on chemical graphs over the set Λ of chemical elements. During construction of the data set D_π, chemical compounds that do not satisfy one of the following are eliminated: the graph is connected, the number of non-hydrogen neighbors of each atom is at most four, and the number of carbon atoms is at least four.

Table 6.1 shows the size and range of data sets that we prepared for each chemical property in Stage 1, where we denote the following:

- Λ: the set of elements used in the data set D_π; Λ is one of the following 11 sets: $\Lambda_1 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}\}$; $\Lambda_2 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}\}$; $\Lambda_3 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{S}_{(2)}\}$; $\Lambda_4 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{Si}\}$; $\Lambda_5 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{Cl}, \mathtt{P}_{(3)}, \mathtt{P}_{(5)}\}$; $\Lambda_6 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{F}\}$; $\Lambda_7 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$; $\Lambda_8 = \{\mathtt{H}, \mathtt{C}_{(2)}, \mathtt{C}_{(3)}, \mathtt{C}_{(4)}, \mathtt{O}, \mathtt{N}_{(2)}, \mathtt{N}_{(3)}\}$; $\Lambda_9 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{S}_{(4)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$; $\Lambda_{10} = \{\mathtt{H}, \mathtt{C}_{(2)}, \mathtt{C}_{(3)}, \mathtt{C}_{(4)}, \mathtt{C}_{(5)}, \mathtt{O}, \mathtt{N}_{(1)}, \mathtt{N}_{(2)}, \mathtt{N}_{(3)}, \mathtt{F}\}$; and
  $\Lambda_{11} = \{\mathtt{H}, \mathtt{C}_{(2)}, \mathtt{C}_{(3)}, \mathtt{C}_{(4)}, \mathtt{O}, \mathtt{N}_{(2)}, \mathtt{N}_{(3)}, \mathtt{S}_{(2)}, \mathtt{S}_{(4)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$, where $\mathtt{e}_{(i)}$ for a chemical element $\mathtt{e}$ and an integer $i \geq 1$ means that a chemical element $\mathtt{e}$ with valence i.
- $|D_\pi|$: the size of data set D_π over Λ for the property π.
- $\underline{n}$, $\overline{n}$: the minimum and maximum values of the number $n(\mathbb{C})$ of non-hydrogen atoms in compounds $\mathbb{C}$ in D_π.
- $\underline{a}$, $\overline{a}$: the minimum and maximum values of $a(\mathbb{C})$ for π over compounds $\mathbb{C}$ in D_π.
- $|\Gamma|$: the number of different edge-configurations of interior-edges over the compounds in D_π.
- $|\mathcal{F}|$: the number of non-isomorphic chemical rooted trees in the set of all 2-fringe-trees in the compounds in D_π.
- K: the number of descriptors in a feature vector $f(\mathbb{C})$.

Stage 2. The newly defined feature function in our chemical model without suppressing hydrogen in Section 6.2 is used. We normalize the range of each descriptor and the range $\{t \in \mathbb{R} \mid \underline{a} \leq t \leq \overline{a}\}$ of property values $a(\mathbb{C}), \mathbb{C} \in D_\pi$.

Stage 3. For each chemical property π, we select a penalty value λ_π in the Lasso function from 36 different values from 0 to 100 by conducting linear regression as a preliminary experiment.

We conducted an experiment in Stage 3 to evaluate the performance of the prediction function based on cross-validation. For a property π, an execution of a *cross-validation* consists of five trials of constructing a prediction function as follows. First partition the data set D_π into five subsets $D^{(k)}$, $k \in [1, 5]$ randomly. For each $k \in [1, 5]$, the k-th trial constructs a prediction function $\eta^{(k)}$

Table 6.1. Results in Phase 1.

π	Λ	$\lvert D_\pi \rvert$	$\underline{n}$, $\overline{n}$	$\underline{a}$, $\overline{a}$	$\lvert\Gamma\rvert$	$\lvert\mathcal{F}\rvert$	K	λ_π	K'	test R^2
Kow	Λ_2	684	4, 58	-7.5, 15.6	25	166	223	6.4E−5	80.3	0.953
Kow	Λ_9	899	4, 69	-7.5, 15.6	37	219	303	5.5E−5	112.1	0.927
Hc	Λ_2	255	4, 63	49.6, 35099.6	17	106	154	1.9E−4	19.2	0.946
Hc	Λ_7	282	4, 63	49.6, 35099.6	21	118	177	1.9E−4	20.5	0.951
Vd	Λ_2	474	4, 30	0.7, 20.6	21	160	214	1.0E−3	3.6	0.927
Vd	Λ_5	551	4, 30	0.7, 20.6	24	191	256	5.5E−4	8.0	0.942
OptR	Λ_2	147	5, 44	-117.0, 165.0	21	55	107	4.6E−4	39.2	0.823
OptR	Λ_6	157	5, 69	-117.0, 165.0	25	62	123	7.3E−4	41.7	0.825
EDPA	Λ_1	52	11, 16	0.80, 3.76	9	33	64	1.0E−4	10.9	0.999
Mp	Λ_2	467	4, 122	-185.33, 300.0	23	142	197	3.7E−5	82.5	0.817
Ha	Λ_3	115	4, 11	1100.6, 3009.6	8	83	115	3.7E−5	39.0	0.997
Hf	Λ_1	82	4, 16	30.2, 94.8	5	50	74	1.0E−4	34.0	0.987
U0	Λ_{10}	977	4, 9	-570.6, -272.8	59	190	297	1.0E−7	246.7	0.999
Lumo	Λ_{10}	977	4, 9	-0.11, 0.10	59	190	297	6.4E−5	133.9	0.841
Alpha	Λ_{10}	977	4, 9	50.9, 99.6	59	190	297	1.0E−5	125.5	0.961
Cv	Λ_{10}	977	4, 9	19.2, 44.0	59	190	297	1.0E−5	165.3	0.961
Sl	Λ_9	915	4, 55	-11.6, 1.11	42	207	300	7.3E−5	130.6	0.808
SfT	Λ_4	247	5, 33	12.3, 45.1	11	91	128	6.4E−4	20.9	0.804
Vis	Λ_4	282	5, 36	-0.64, 1.63	12	88	126	8.2E−4	16.3	0.893
IhcLiq	Λ_2	770	4, 78	106.3, 1956.1	23	200	256	1.9E−5	82.2	0.987
IhcLiq	Λ_7	865	4, 78	106.3, 1956.1	29	246	316	8.2E−6	139.1	0.986
IhcSol	Λ_8	581	5, 70	67.4, 1220.9	33	124	192	2.8E−5	75.9	0.985
IhcSol	Λ_{11}	668	5, 70	67.4, 1220.9	40	140	228	2.8E−5	86.7	0.982
Lp	Λ_2	615	6, 60	-3.62, 6.84	32	116	186	1.0E−4	98.5	0.856
Lp	Λ_9	936	6, 74	-3.62, 6.84	44	136	231	6.4E−5	130.4	0.840

by conducting a linear regression with the penalty term λ_π using the set $D_\pi \setminus D^{(k)}$ as a training data set. We used scikit-learn version 0.23.2 with Python 3.8.5 for executing linear regression with Lasso function. For each property, we executed ten cross-validations and we show the median of test $R^2(\eta^{(k)}, D^{(k)}), k \in [1, 5]$ over all ten cross-validations. Recall that a subset of descriptors is selected in linear regression with Lasso function and let K' denote the average number of selected descriptors over all 50 trials. The running time per trial in a cross-validation was at most one second.

Table 6.1 shows the results on Stages 2 and 3, where we denote the following:

- λ_π: the penalty value in the Lasso function selected for a property π, where

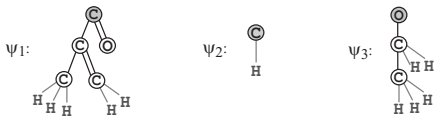


Figure 6.4. An illustration of chemical rooted trees ψ_1, ψ_1 and ψ_3 that are selected in Lasso linear regression for constructing a prediction function to property VD, where the root is depicted with a gray circle.

aEb means $a \times 10^b$.

- K': the average of the number of descriptors selected in the linear regression over all 50 trials in ten cross-validations.
- test R^2: the median of test R^2 over all 50 trials in ten cross-validations.

Recall that the adjacency-configuration for leaf-edges was introduced as a new descriptor in this chapter. Without including this new descriptor, the test R^2 for property VIS was 0.790, that for LUMO was 0.799 and that for MP was 0.796, while the test R^2 for each of the other properties in Table 6.1 was almost the same.

From Table 6.1, we observe that a relatively large number of properties admit a good prediction function based on linear regression. The number K' of descriptors used in linear regression is considerably small for some properties. For example of property VD, the four descriptors most frequently selected in the case of $\Lambda = \{\mathtt{H}, \mathtt{O}, \mathtt{C}, \mathtt{N}\}$ are the number of non-hydrogen atoms; the number of interior-vertices v with $\deg_{\mathbb{C}^{\rm int}}(v) = 1$; the number of fringe-trees r-isomorphic to the chemical rooted tree ψ_1 in Figure 6.4; and the number of leaf-edges with adjacency-configuration $(\mathtt{O}, \mathtt{C}, 2)$. The eight descriptors most frequently selected in the case of $\Lambda = \{\mathtt{H}, \mathtt{O}, \mathtt{C}, \mathtt{N}, \mathtt{Cl}, \mathtt{P}_{(3)}, \mathtt{P}_{(5)}\}$ are the number of non-hydrogen atoms; the number of interior-vertices v with $\deg_{\mathbb{C}^{\rm int}}(v) = 1$; the number of exterior-vertices v with $\alpha(v) = \mathtt{Cl}$; the number of interior-edges with edge-configuration $\gamma_i, i = 1, 2$, where $\gamma_1 = (\mathtt{C2}, \mathtt{C2}, 2)$ and $\gamma_2 = (\mathtt{C3}, \mathtt{C4}, 1)$; and the number of fringe-trees r-isomorphic to the chemical rooted tree $\psi_i, i = 1, 2, 3$ in Figure 6.4.

For the 18 properties listed in Table 6.1, we used ANN to construct prediction functions. For this purpose, we used our newly proposed feature vector and the experimental setup as explained in Tanaka et al. [49]. From these computation experiments, we observe that for the properties HC, VD, HA, HF, U0, ALPHA and CV, the test R^2 scores of the prediction functions obtained by Lasso linear regression is at least 0.05 more than those obtained by ANN. For the properties OPTR, SL and SFT, the test R^2 scores of the prediction functions obtained by ANN is at least 0.05 more than those obtained by Lasso linear regression. For

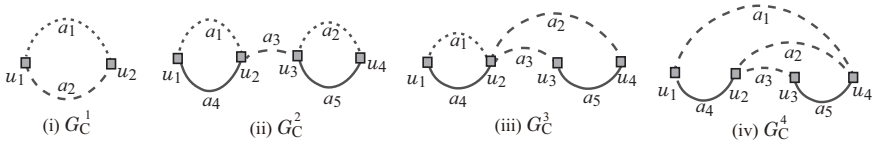


Figure 6.5. (i) Seed graph G_{C}^1 for I_{b}^1 and I_{d}; (ii) Seed graph G_{C}^2 for I_{b}^2; (iii) Seed graph G_{C}^3 for I_{b}^3; (iv) Seed graph G_{C}^4 for I_{b}^4.

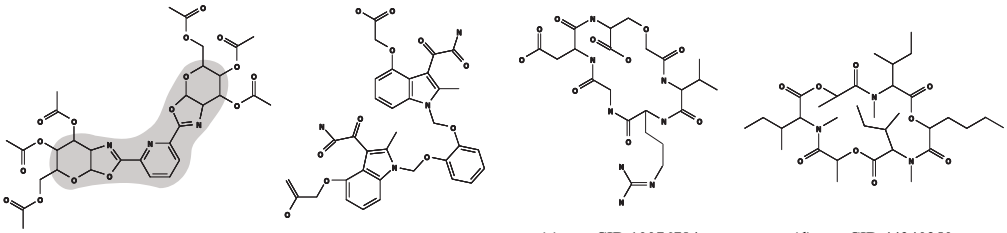


Figure 6.6. An illustration of chemical compounds for instances I_{c} and I_{d}: (a) $\mathbb{C}_A$: CID 24822711; (b) $\mathbb{C}_B$: CID 59170444; (c) $\mathbb{C}_A$: CID 10076784; (d) $\mathbb{C}_B$: CID 44340250, where hydrogens are omitted.

the other properties, the test R^2 scores obtained by Lasso linear regression and ANN are comparable.

Results on Phase 2. We used a set of seven instances I_{a}, $I_{\mathrm{b}}^i, i \in [1,4]$, I_{c} and I_{d} based on seed graphs prepared by Shi et al. [43] to execute Stages 4 and 5 in Phase 2. We here present their seed graphs G_{C} (see [63] for the details of instances I_{a}, $I_{\mathrm{b}}^i, i \in [1,4]$, I_{c} and I_{d}). The seed graph G_{C} of instance I_{a} is illustrated in Figure 6.3(a). The seed graph G_{C}^1 (resp., $G_{\mathrm{C}}^i, i = 2,3,4$) of instances I_{b}^1 and I_{d} (resp., $I_{\mathrm{b}}^i, i = 2,3,4$) is illustrated in Figure 6.5.

Instance I_{c} has been introduced by Shi et al. [43] in order to infer a chemical graph $\mathbb{C}^\dagger$ such that the core of $\mathbb{C}^\dagger$ is the same as the core of chemical graph $\mathbb{C}_A$: CID 24822711 in Figure 6.6(a) and the frequency of each edge-configuration in the non-core of $\mathbb{C}^\dagger$ is the same as that of chemical graph $\mathbb{C}_B$: CID 59170444 illustrated in Figure 6.6(b). This means that the seed graph G_{C} of I_{c} is the core of $\mathbb{C}_A$ which is indicated by a shaded area in Figure 6.6(a).

Instance I_{d} has been introduced by Shi et al. [43] in order to infer a monocyclic chemical graph $\mathbb{C}^\dagger$ such that the frequency vector of edge-configurations in $\mathbb{C}^\dagger$ is a vector obtained by merging those of two chemical graphs $\mathbb{C}_A$: CID 10076784 and $\mathbb{C}_B$: CID 44340250 illustrated in Figure 6.6(c) and (d), respectively.

Stage 4. We executed Stage 4 for five properties $\pi \in \{$Hc, Vd, OptR, IhcLiq,

Table 6.2. Results of Stages 4 and 5 for HC using Lasso linear regression.

inst.	$\underline{y}^*,\ \overline{y}^*$	#v	#c	I-time	n	$\mathrm{n}^{\mathrm{int}}$	$\eta(f(\mathbb{C}^\dagger))$	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	5950, 6050	9902	9255	4.6	44	25	5977.9	0.068	1	1
I_{b}^1	5950, 6050	9404	6776	1.7	36	10	6007.1	0.048	6	6
I_{b}^2	5950, 6050	11729	9891	16.7	50	25	6043.7	38.7	2.4×10^5	100
I_{b}^3	5950, 6050	11510	9894	16.3	47	25	6015.4	0.353	8724	100
I_{b}^4	5950, 6050	11291	9897	9.0	49	26	5971.6	0.304	84	84
I_{c}	13700, 13800	6915	7278	0.7	50	33	13703.3	0.016	1	1
I_{d}	13700, 13800	5535	6781	4.9	44	23	13704.7	0.564	4.3×10^5	100

VIS}.

For the MILP formulation $\mathcal{M}(x, y; \mathcal{C}_1)$ in Section 6.3, we use the prediction function $\eta_{w,b}$ that attained the median test R^2 in Table 6.1. We used `CPLEX` version 12.10 to solve an MILP in Stage 4. Tables 6.2 to 6.6 show the computational results of the experiment in Stage 4 for the five properties, where we denote the following:

- $\underline{y}^*,\ \overline{y}^*$: lower and upper bounds $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ on the value $a(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to be inferred;
- #v (resp., #c): the number of variables (resp., constraints) in the MILP in Stage 4;
- I-time: the time (sec.) to solve the MILP in Stage 4;
- n: the number $n(\mathbb{C}^\dagger)$ of non-hydrogen atoms in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4; and
- $\mathrm{n}^{\mathrm{int}}$: the number $\mathrm{n}^{\mathrm{int}}(\mathbb{C}^\dagger)$ of interior-vertices in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4;
- $\eta(f(\mathbb{C}^\dagger))$: the predicted property value $\eta(f(\mathbb{C}^\dagger))$ of the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4.

From Tables 6.2 to 6.6, we observe that an instance with a large number of variables and constraints takes more running time than those with a smaller size in general. We solved all instances in this experiment with our MILP formulation in a few seconds to around 30 seconds.

Figure 6.7(a)-(e) illustrate the chemical graphs $\mathbb{C}^\dagger$ inferred from I_{c} with $(\underline{y}^*, \overline{y}^*) = (13700, 13800)$ of HC, I_{b}^2 with $(\underline{y}^*, \overline{y}^*) = (21, 22)$ of VD, I_{b}^4 with $(\underline{y}^*, \overline{y}^*) = (70, 71)$ of OPTR, I_{d} with $(\underline{y}^*, \overline{y}^*) = (1190, 1210)$ of IHCLIQ, and I_{b}^3 with $(\underline{y}^*, \overline{y}^*) = (1.85, 1.90)$ of VIS, respectively.

Similarly, we executed Stage 4 for these seven instances I_{a}, $I_{\mathrm{b}}^i, i \in [1, 4]$,

Table 6.3. Results of Stages 4 and 5 for Vd using Lasso linear regression.

inst.	$\underline{y}^*, \overline{y}^*$	#v	#c	I-time	n	n^{int}	$\eta(f(\mathbb{C}^\dagger))$	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	16, 17	9481	9358	1.6	38	23	16.83	0.070	1	1
I_{b}^1	16, 17	9928	6986	1.5	35	12	16.68	0.206	48	48
I_{b}^2	21, 22	12373	10101	10.0	48	25	21.62	0.104	20	20
I_{b}^3	21, 22	12159	10104	6.5	48	25	21.95	3.65	8.6×10^5	100
I_{b}^4	21, 22	11945	10107	8.1	48	25	21.34	0.057	6	6
I_{c}	21, 22	7073	7438	0.7	50	34	21.89	0.016	1	1
I_{d}	17, 18	5693	6942	2.1	41	23	17.94	0.161	216	100

Table 6.4. Results of Stages 4 and 5 for OptR using Lasso linear regression.

inst.	$\underline{y}^*, \overline{y}^*$	#v	#c	I-time	n	n^{int}	$\eta(f(\mathbb{C}^\dagger))$	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	70, 71	8962	9064	3.5	40	23	70.1	0.061	1	1
I_{b}^1	70, 71	9432	6662	2.7	37	14	70.1	0.185	2622	100
I_{b}^2	70, 71	11818	9773	10.0	50	25	70.8	0.041	4	4
I_{b}^3	70, 71	11602	9776	10.2	50	25	70.2	0.241	60	60
I_{b}^4	70, 71	11386	9779	24.7	49	25	70.9	6.39	4.6×10^5	100
I_{c}	-112, -111	6807	7170	1.8	50	32	-111.9	0.016	1	1
I_{d}	70, 71	5427	6673	6.1	42	23	70.2	0.127	78768	100

Table 6.5. Results of Stages 4 and 5 for IhcLiq using Lasso linear regression.

inst.	$\underline{y}^*, \overline{y}^*$	#v	#c	I-time	n	n^{int}	$\eta(f(\mathbb{C}^\dagger))$	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	1190, 1210	10180	9538	3.9	48	26	1208.5	0.071	2	2
I_{b}^1	1190, 1210	10784	7191	2.4	35	14	1206.7	0.082	12	12
I_{b}^2	1190, 1210	13482	10302	14.1	47	25	1206.7	0.11	12	12
I_{b}^3	1190, 1210	13275	10301	9.0	49	27	1209.9	0.090	24	24
I_{b}^4	1190, 1210	13128	10306	16.5	50	25	1208.4	0.424	2388	100
I_{c}	1190, 1210	7193	7560	0.8	50	33	1196.5	0.016	1	1
I_{d}	1190, 1210	5813	7063	2.2	44	23	1198.8	5.63	5.2×10^5	100

I_{c} and I_{d} for five properties $\pi \in \{\text{Hc}, \text{Vd}, \text{OptR}, \text{IhcLiq}, \text{Vis}\}$ by using the prediction functions obtained by ANN. We list the running time to solve

Table 6.6. Results of Stages 4 and 5 for VIS using Lasso linear regression.

inst.	$\underline{y}^*,\ \overline{y}^*$	#v	#c	I-time	n	$\mathrm{n}^{\mathrm{int}}$	$\eta(f(\mathbb{C}^\dagger))$	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	$1.25, 1.30$	6847	8906	1.3	38	22	1.295	0.042	2	2
I_{b}^1	$1.25, 1.30$	7270	6397	2.5	36	15	1.272	0.155	140	100
I_{b}^2	$1.85, 1.90$	8984	9512	8.9	45	25	1.879	0.149	288	100
I_{b}^3	$1.85, 1.90$	8741	9515	16.2	45	26	1.880	0.137	4928	100
I_{b}^4	$1.85, 1.90$	8498	9518	8.1	45	25	1.851	0.13	660	100
I_{c}	$2.75, 2.80$	6813	7162	1.0	50	33	2.763	0.025	4	4
I_{d}	$1.85, 1.90$	5433	6665	2.7	41	23	1.881	0.138	4608	100

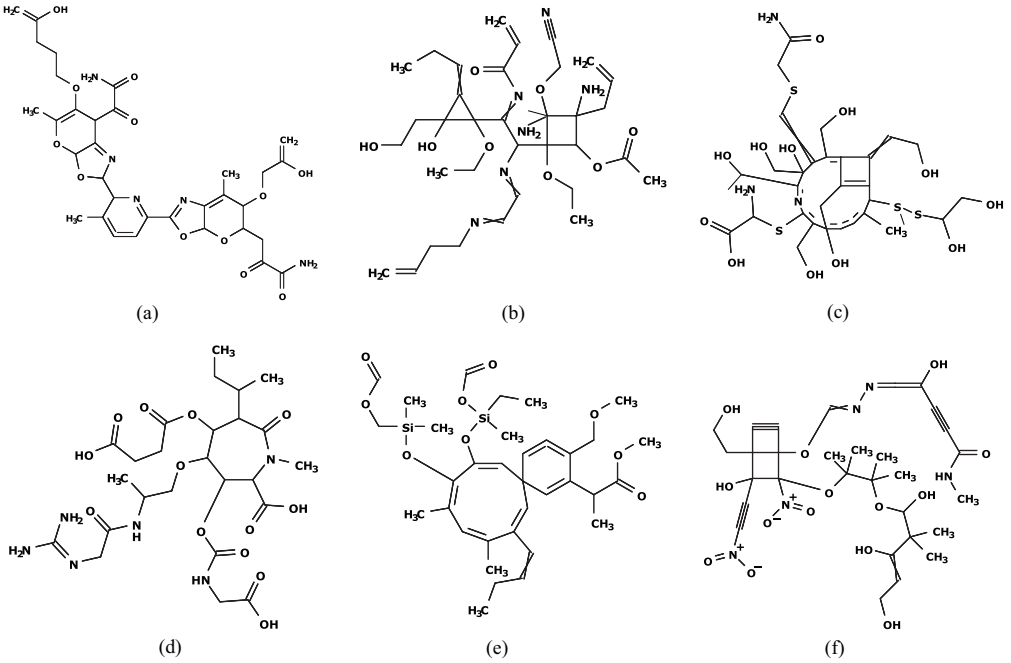


Figure 6.7. (a) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 13703.3$ inferred from I_{c} with $(\underline{y}^*, \overline{y}^*) = (13700, 13800)$ of HC; (b) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 21.62$ inferred from I_{b}^2 with $(\underline{y}^*, \overline{y}^*) = (21, 22)$ of VD; (c) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 70.9$ inferred from I_{b}^4 with $(\underline{y}^*, \overline{y}^*) = (70, 71)$ of OPTR; (d) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 1198.8$ inferred from I_{d} with $(\underline{y}^*, \overline{y}^*) = (1190, 1210)$ of IHCLIQ; (e) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 1.880$ inferred from I_{b}^3 with $(\underline{y}^*, \overline{y}^*) = (1.85, 1.90)$ of VIS; (f) $\mathbb{C}^\dagger$ inferred from I_{b}^4 with lower and upper bounds on the predicted property value $\eta_\pi(f(\mathbb{C}^\dagger))$ of property $\pi \in \{$KOW, LP, SL$\}$ in Table 6.9.

Table 6.7. Running time of Stage 4 for Hc, Vd and OptR using ANN.

Hc			Vd			OptR		
inst.	$\underline{y}^*,\ \overline{y}^*$	I-time	inst.	$\underline{y}^*,\ \overline{y}^*$	I-time	inst.	$\underline{y}^*,\ \overline{y}^*$	I-time
$I_{\rm a}$	13350, 13450	24.7	$I_{\rm a}$	18, 19	18.1	$I_{\rm a}$	62, 63	35.6
$I_{\rm b}^1$	9650, 9750	13.5	$I_{\rm b}^1$	13, 14	9.4	$I_{\rm b}^1$	109, 110	15.5
$I_{\rm b}^2$	16750, 16850	70.4	$I_{\rm b}^2$	15, 16	40.9	$I_{\rm b}^2$	23, 24	192.6
$I_{\rm b}^3$	12350, 12450	87.0	$I_{\rm b}^3$	20, 21	46.3	$I_{\rm b}^3$	-2, -1	936.4
$I_{\rm b}^4$	14250, 14350	70.9	$I_{\rm b}^4$	22, 23	27.1	$I_{\rm b}^4$	19, 20	63.9
$I_{\rm c}$	10400,10500	31.3	$I_{\rm c}$	20, 21	20.5	$I_{\rm c}$	86, 87	16.4
$I_{\rm d}$	12500, 12600	44.3	$I_{\rm d}$	18, 19	6.1	$I_{\rm d}$	30, 31	31.8

Table 6.8. Running time of Stage 4 for IhcLiq and Vis using ANN.

IhcLiq			Vis		
inst.	$\underline{y}^*,\ \overline{y}^*$	I-time	inst.	$\underline{y}^*,\ \overline{y}^*$	I-time
$I_{\rm a}$	980, 1000	56.6	$I_{\rm a}$	1.85, 1.90	2.0
$I_{\rm b}^1$	1000, 1020	40.4	$I_{\rm b}^1$	1.95, 2.00	3.5
$I_{\rm b}^2$	1130, 1150	71.6	$I_{\rm b}^2$	1.85, 1.90	19.7
$I_{\rm b}^3$	1240, 1260	45.0	$I_{\rm b}^3$	2.35, 2.40	26.0
$I_{\rm b}^4$	1240, 1260	105.7	$I_{\rm b}^4$	2.50, 2.55	9.3
$I_{\rm c}$	810, 830	9.7	$I_{\rm c}$	3.90, 3.95	1.8
$I_{\rm d}$	1100, 1120	25.8	$I_{\rm d}$	3.30, 3.35	8.3

MILP formulation for each of these instances in Tables 6.7 and 6.8. From the computation experiments, we observe that for many instances, the running time is significantly faster than that of Stage 4 based on ANN.

Inferring a chemical graph with target values in multiple properties Once we obtained prediction functions η_π for several properties π, include MILP formulations for these functions η_π into a single MILP $\mathcal{M}(x, y; \mathcal{C}_1)$ so as to infer a chemical graph that satisfies given target values y^* for these properties at the same time. As an additional experiment in Stage 4, we inferred a chemical graph that has a desired predicted value each of three properties Kow, Lp and Sl, where we used the prediction function η_π for each property $\pi \in \{$Kow, Lp, Sl$\}$

Table 6.9. Results of Stage 4 for instances $I_{\rm b}^i, i = 2, 3, 4$ with specified target values of three properties KOW, LP and SL using Lasso linear regression.

inst.	π	$\underline{y}^*_\pi$, $\overline{y}^*_\pi$	#v	#c	I-time	n	$\mathrm{n}^{\rm int}$	$\eta_\pi(f(\mathbb{C}^\dagger))$
	KOW	-7.50, -7.40						-7.41
$I_{\rm b}^2$	LP	-1.40, -1.30	14574	11604	62.7	50	30	-1.33
	SL	-11.6, -11.5						-11.52
	KOW	-7.40, -7.30						-7.38
$I_{\rm b}^3$	LP	-2.90, -2.80	14370	11596	35.5	48	25	-2.81
	SL	-11.6, -11.4						-11.52
	KOW	-7.50, -7.40						-7.48
$I_{\rm b}^4$	LP	-0.70, -0.60	14166	11588	71.7	49	26	-0.63
	SL	-11.4, -11.2						-11.39

constructed in Stage 3. Table 6.9 shows the result of Stage 4 for inferring a chemical graph $\mathbb{C}^\dagger$ from instances $I_{\rm b}^2$, $I_{\rm b}^3$ and $I_{\rm b}^4$ with $\Lambda = \{\mathtt{H}, \mathtt{C}, \mathtt{N}, \mathtt{O}, \mathtt{S}_{(2)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$, where we denote the following:

- π: one of the three properties KOW, LP and SL used in the experiment;
- $\underline{y}^*_\pi$, $\overline{y}^*_\pi$: lower and upper bounds $\underline{y}^*_\pi, \overline{y}^*_\pi \in \mathbb{R}$ on the predicted property value $\eta_\pi(f(\mathbb{C}^\dagger))$ of property $\pi \in \{$KOW, LP, SL$\}$ for a chemical graph $\mathbb{C}^\dagger$ to be inferred;
- #v (resp., #c): the number of variables (resp., constraints) in the MILP in Stage 4;
- I-time: the time (sec.) to solve the MILP in Stage 4;
- n: the number $n(\mathbb{C}^\dagger)$ of non-hydrogen atoms in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4;
- $\mathrm{n}^{\rm int}$: the number $\mathrm{n}^{\rm int}(\mathbb{C}^\dagger)$ of interior-vertices in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4; and
- $\eta_\pi(f(\mathbb{C}^\dagger))$: the predicted property value $\eta_\pi(f(\mathbb{C}^\dagger))$ of property $\pi \in \{$KOW, LP, SL$\}$ for the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4.

Figure 6.7(f) illustrates the chemical graph $\mathbb{C}^\dagger$ inferred from $I_{\rm b}^4$ with $(\underline{y}^*_{\pi_1}, \overline{y}^*_{\pi_1}) = (-7.50,\ -7.40)$, $(\underline{y}^*_{\pi_2}, \overline{y}^*_{\pi_2}) = (-0.70, -0.60)$ and $(\underline{y}^*_{\pi_3}, \overline{y}^*_{\pi_3}) = (-11.4, -11.2)$ for $\pi_1 =$KOW, $\pi_2 =$LP and $\pi_3 =$SL, respectively.

Stage 5. We executed Stage 5 to generate more target chemical graphs $\mathbb{C}^*$, where a chemical graph $\mathbb{C}^*$ is called a *chemical isomer* of a target chemical graph $\mathbb{C}^\dagger$ of a topological specification σ if $f(\mathbb{C}^*) = f(\mathbb{C}^\dagger)$ and $\mathbb{C}^*$ also satisfies the same

topological specification σ. We computed chemical isomers $\mathbb{C}^*$ of each target chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4. We executed an algorithm to generate chemical isomers of $\mathbb{C}^\dagger$ up to 100 when the number of all chemical isomers exceeds 100. We can obtain such an algorithm from the dynamic programming proposed by Tanaka et al. [49] with a slight modification. The algorithm first decomposes $\mathbb{C}^\dagger$ into a set of acyclic chemical graphs, next replaces each acyclic chemical graph T with another acyclic chemical graph T' that admits the same feature vector as that of T, and finally assembles the resulting acyclic chemical graphs into a chemical isomer $\mathbb{C}^*$ of $\mathbb{C}^\dagger$. Also, a lower bound on the total number of all chemical isomers of $\mathbb{C}^\dagger$ can be computed by the algorithm without generating all of them.

Tables 6.2 to 6.6 show the computational results of the experiment in Stage 5 for the five properties, where we denote the following:

- D-time: the running time (sec.) to execute the dynamic programming algorithm in Stage 5 to compute a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ and generate all (or up to 100) chemical isomers $\mathbb{C}^*$;
- $\mathbb{C}$-LB: a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$; and
- $\#\mathbb{C}$: the number of all (or up to 100) chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ generated in Stage 5.

From Tables 6.2 to 6.6, we observe that for many cases the running time for generating up to 100 target chemical graphs in Stage 5 is less than 0.4 seconds. For some chemical graph $\mathbb{C}^\dagger$, no chemical isomer was found by our algorithm. This is because each acyclic chemical graph in the decomposition of $\mathbb{C}^\dagger$ has no alternative acyclic chemical graph than the original one. On the other hand, some chemical graph $\mathbb{C}^\dagger$ such as the one in $I_{\rm d}$ in Table 6.2 admits an extremely large number of chemical isomers $\mathbb{C}^*$. Remember that we know such a lower bound $\mathbb{C}$-LB on the number of chemical isomers without generating all of them.

6.5 Concluding Remarks

In this chapter, we studied the problem of inferring chemical structures from desired chemical properties and constraints, based on the framework proposed and developed in [1, 4, 59]. In the previous applications of the framework of inferring chemical graphs, artificial neural network (ANN) and decision tree have been used for the machine learning of Stage 3. In this chapter, we used linear regression in Stage 3 for the first time and derived an MILP formulation that simulates the computation process of linear regression. We also extended a way of specifying a target value y^* in a property so that the predicted value $\eta(f(\mathbb{C}^\dagger))$ of a target chemical graph $\mathbb{C}^\dagger$ is required to belong to an interval between two

specified values $\underline{y}^*$ and $\overline{y}^*$. Furthermore, we modified a model of chemical compounds so that multi-valence chemical elements, cation and anion are treated, and introduced the rank and the adjacency-configuration of leaf-edges as new descriptors in a feature vector of a chemical graph.

We implemented the new system of the framework and conducted computational experiments for Stages 1 to 5. We found 18 properties for which linear regression delivers a relatively good prediction function by using our feature vector based on the two-layered model. We also observed that an MILP formulation for inferring a chemical graph in Stage 4 can be solved efficiently over different types of test instances with complicated topological specifications. The experimental result suggests that our method can infer chemical graphs with up to 50 non-hydrogen atoms. Therefore, combination of linear regression and integer programming is a potentially useful approach to computational molecular design.

It is an interesting future work to use other learning methods such as graph convolution networks, random forest and an ensemble method to construct a prediction function and derive the corresponding MILP formulations in Stages 3 and 4 in the framework.

7 Two-Layered Model with Adjustive Linear Regression[1]

7.1 Introduction

In this chapter, we develop a novel prediction function and its machine learning method that can be used in the framework. Let us compare linear regression and ANNs. The former uses a hyperplane to explain a given data set and the latter can represent a more complex subspace than a hyperplane. Importantly a best hyperplane that minimizes an error function can be found exactly in the former whereas a local optimum solution to an error function is constructed by an iterative procedure in the latter and different local optimum solutions often appear depending on how we have tuned many parameters in ANNs. Linear regression can be regarded as an ANN on an architecture with an input layer and an output layer of a single node with a linear activation function. We consider an ANN on the same architecture such that each node in the input and output layers is equipped with a set of activation functions. Given a data set, we consider a problem of minimizing an error function on the data set by choosing a weight of each arc, a bias of the output node and a best activation function for each node simultaneously. With some restriction on the set of activation functions and the definition of an error function, we show that such an minimization problem can be formulated as a linear program, which is much easier than an MILP to solve exactly. We call this new method "adjustive linear regression" and implemented it in the two phase framework. We compared adjustive linear regression with Lasso linear regression in constructing prediction functions for several chemical properties. From the results of our computational experiments, we observe that a prediction function constructed by adjustive linear regression for some chemical properties drastically outperforms that by Lasso linear regression.

The rest of this chapter is organized as follows. Section 7.2 reviews the idea of prediction functions based on linear regression and ANNs and designs "ad-

[1] ©2022 SCITEPRESS. Reprinted, with permission, from J. Zhu, K. Haraguchi, H. Nagamochi, and T. Akutsu. Adjustive linear regression and its application to the inverse QSAR. In *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOINFORMATICS*, pages 144–151. INSTICC, SciTePress, 2022.

justive linear regression", a new method for constructing a prediction function by solving a linear program to optimize a choice of weights/bias together with activation functions in an ANN with no hidden layers. Section 7.3 reports the results on some computational experiments conducted for the framework of inferring chemical graphs by using our new method of adjustive linear regression. Section 7.4 makes some concluding remarks. We refer [64] for the complete MILP formulations.

7.2 Constructing Prediction Functions

7.2.1 Linear Prediction Functions

For the feature space $\mathbb{R}^K$, a hyperplane is defined to be a pair (w, b) of a vector $w \in \mathbb{R}^K$ and a real $b \in \mathbb{R}$. A prediction function η is called *linear* if η is given by $\eta_{w,b}(x) = w \cdot x + b, x \in \mathbb{R}^K$ for a hyperplane (w, b). The linear regression is to find a hyperplane (w, b) that minimizes $\mathrm{Err}(\eta_{w,b}; \mathcal{X}) = \sum_{i\in[1,m]}(a_i - (w \cdot x_i + b))^2$.

In many cases, a feature vector f contains descriptors that do not play an essential role in constructing a good prediction function. When we solve the minimization problem, the entries $w(j)$ for some descriptors $j \in [1, K]$ in the resulting hyperplane (w, b) become zero, which means that these descriptors were not necessarily important for finding a prediction function $\eta_{w,b}$. It is proposed that solving the minimization with an additional penalty term to the error function often results in a more number of entries $w(j) = 0$, reducing a set of descriptors necessary for defining a prediction function $\eta_{w,b}$. For an error function with such a penalty term, a Ridge function $\frac{1}{2m}\mathrm{Err}(\eta_{w,b}; \mathcal{X}) + \lambda[\sum_{j\in[1,K]} w(j)^2 + b^2]$ [21, 22] and a Lasso function $\frac{1}{2m}\mathrm{Err}(\eta_{w,b}; \mathcal{X}) + \lambda[\sum_{j\in[1,K]} |w(j)| + |b|]$ [52] are known, where $\lambda \in \mathbb{R}$ is a given real number. As a hybridization of Ridge linear regression and Lasso linear regression, a linear regression that minimizes an error function defined to be $\frac{1}{2m}\mathrm{Err}(\eta_{w,b}; \mathcal{X}) + \lambda_2[\sum_{j\in[1,K]} w(j)^2 + b^2] + \lambda_1[\sum_{j\in[1,K]} |w(j)| + |b|]$ is called elastic net linear regression [68], where $\lambda_1, \lambda_2 \in \mathbb{R}$ are given real numbers.

7.2.2 ANNs for Linear Prediction Functions

It is not difficult to see that a linear prediction function η with a hyperplane (w, b) can be represented by an ANN $\mathcal{N}$ with an input layer $L_{\mathrm{in}} = \{u_1, u_2, \ldots, u_K\}$ of K input nodes and an output layer $L_{\mathrm{out}} = \{v\}$ of a single output node v such that the weight of an arc (u_j, v) from an input node u_j to the output node v is given by $w(j), j \in [1, K]$; the bias at node v is given by b; and the activation function at node v is linear. See Figure 7.1(a) for an illustration of an ANN $\mathcal{N}$ that represents a linear prediction function η with a hyperplane (w, b). Given a

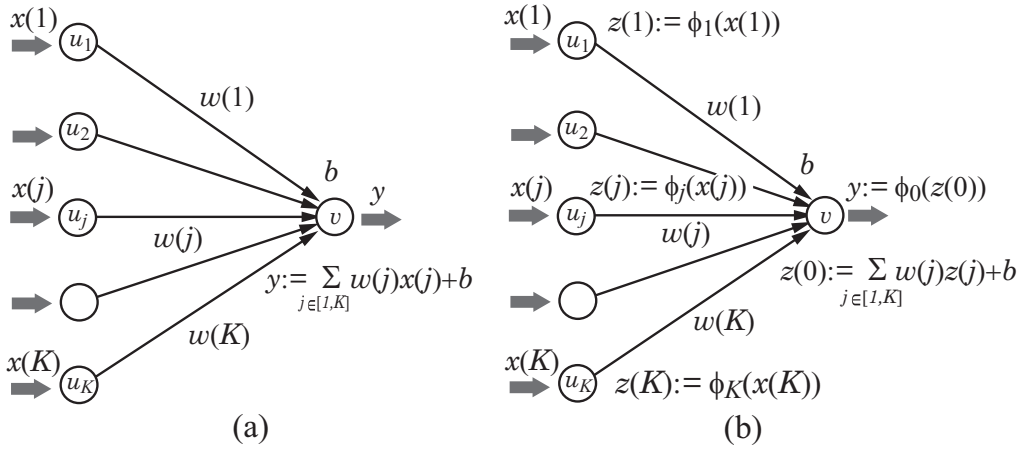


Figure 7.1. An illustration of the process in ANNs with no hidden layers: (a) An ANN $\mathcal{N}$ that represents a linear prediction function η with a hyperplane (w, b); (b) an ANN $\mathcal{N}_\phi$ with activation functions $\phi_j, j \in [0, K]$ at all nodes.

vector $x \in \mathbb{R}^K$, the ANN $\mathcal{N}$ outputs $y := \sum_{j\in[1,K]} w(j)x(j) + b$.

We consider an ANN $\mathcal{N}_\phi$ with the same architecture with the ANN $\mathcal{N}$ and introduce activation functions ϕ_j at nodes $u_j, j \in [1, K]$ and an activation function ϕ_0 at node v. Given a vector $x \in \mathbb{R}^K$, the ANN $\mathcal{N}_\phi$ outputs $y := \phi_0(z(0))$ for $z(0) := \sum_{j\in[1,K]} w(j)z(j) + b$ and $z(j) := \phi_j(x(j)), j \in [1, K]$.

In a standard method of a prediction function $\eta_{\mathcal{N}_\phi}$ with the above ANN $\mathcal{N}_\phi$, we specify each activation function ϕ_j and determine weights w and a bias b by executing an iterative procedure that tries to minimize an error function between the real values a_i and the predicted values $\eta_{\mathcal{N}_\phi}(x_i)$.

7.2.3 Adjustive Linear Regression

In this chapter, we design a new method of constructing a prediction function with the above ANN $\mathcal{N}_\phi$ so that (i) not only weights w and a bias b but also prediction functions ϕ_j are chosen so as to minimize an error function and (ii) the minimization problem is formulated as a linear programming problem.

We introduce a class Φ_j of functions for a choice of each activation function $\phi_j, j \in [0, K]$. When we choose a function $\phi_j \in \Phi_j$ for each $j \in [0, K]$ and a hyperplane (w, b), we define a prediction function $\eta_{\Psi,w,b}$ such that

$$\eta_{\Psi,w,b}(x) \triangleq \phi_0\Big(\sum_{j\in[1,K]} w(j)(\phi_j(x(j))) - b\Big)$$

for the set $\Psi = \{\phi_j \mid j \in [0, K]\}$ of the functions.

In this chapter, we use a function $\xi(t) = ct + c't^2 + c''(1-(t-1)^2), 0 \le t \le 1$ for a function $\phi_j, j \in [1,K]$ or the inverse ϕ_0^{-1} of a function ϕ_0, where c, c' and c'' are nonnegative constant constants with $c+c'+c''=1$ which will be determined for each $j \in [0,K]$ by our method. Note that, for a domain $0 \le t \le 1$, $\xi(t)$ is a monotone increasing function such that $\xi(0)=0$ and $\xi(1)=1$ and admits an inverse function $\xi^{-1}(t)$.

We introduce a class Φ_j of functions in the following way.

1. Normalize the set $\{x_i(j) \mid x_i \in \mathcal{X}\}, j \in [1,K]$ and the set $\{a_i(j) \mid x_i \in \mathcal{X}\}$ so that the minimum and maximum in the set become 0 and 1.

2. For each index $j \in [0,K]$, define a class Φ_j of functions to be
$\Phi_j \triangleq \{c_0(j)t + c_1(j)t^2 + c_2(j)(1-(t-1)^2),\ 0 \le t \le 1 \mid c_q(j) \ge 0, q \in [0,2], \sum_{q\in[0,2]} c_q(j) = 1\},\ j \in [1,K],$
and define
$\widetilde{\Phi}_0 \triangleq \{c_0(0)t + c_1(0)t^2 + c_2(0)(1-(t-1)^2),\ 0 \le t \le 1 \mid c_q(0) \ge 0, q \in [0,2], \sum_{q\in[0,2]} c_q(0) = 1\},\ \Phi_0 \triangleq \{\xi^{-1}(t), 0 \le t \le 1 \mid \xi(t) \in \widetilde{\Phi}_0\}.$

To use linear programming, we measure an error of a prediction function η over a data set $\mathcal{X}$ by the sum of the absolute errors: $\mathrm{SAE}(\eta;\mathcal{X}) \triangleq \sum_{x_i\in\mathcal{X}} |a_i - \eta(x_i)|$.

Now our aim is to find a prediction function $\eta_{\Psi,w,b}$ that minimizes the sum of the absolute errors $\mathrm{SAE}(\eta_{\Psi,w,b};\mathcal{X})$ over all functions $\phi_0 \in \widetilde{\Phi}_0, \phi_j \in \Phi_j, j \in [1,K]$ and hyperplanes (w,b).

To formulate this minimization problem as a linear programming problem, we predetermine the sign of $w(j)$ for each descriptor j in a hyperplane (w,b) that we will choose. Compute the correlation coefficient $\sigma(X_j, A)$ between $X_j = \{x_i(j) \mid i \in [1,m]\}$ and $A = \{a_i \mid i \in [1,m]\}$ and partition the set of descriptors into two sets $I^+ := \{j \in [1,K] \mid \sigma(X_j,A) \ge 0\}$ and $I^- := \{j \in [1,K] \mid \sigma(X_j,A) < 0\}$. We impose an additional constraint that $w(j) \ge 0, j \in I^+$ and $w(j) \le 0, j \in I^-$. Then the objective function is described as follows, where we rewrite each term $w(j), j \in I^+$ (resp., $-w(j), j \in I^-$) as $w'(j)$:

$$\sum_{i\in[1,m]} \Big|c_0(0)a_i + c_1(0)a_i^2 + c_2(0)(1-(a_i-1)^2) - \sum_{j\in I^+}[w'(j)\big(c_0(j)x_i(j) + c_1(j)x_i(j)^2 + c_2(j)(1-(x_i(j)-1)^2)\big)] + \sum_{j\in I^-}[w'(j)\big(c_0(j)x_i(j) + c_1(j)x_i(j)^2 + c_2(j)(1-(x_i(j)-1)^2)\big)] - b\Big|.$$

We minimize this over all nonnegative reals $c_q(j), q \in [0,2], j \in [1,K]$, nonnegative reals $w(j), j \in [1,K]$ and a real $b \in \mathbb{R}$ such that $\sum_{q\in[0,2]} c_q(j) = 1, j \in [1,K]$.

By introducing a penalty term for the weights $w(j), j \in [1,K]$, we consider the following problem which we call *adjustive linear regression* $\mathrm{ALR}(\mathcal{X},\lambda)$, where

$w'(j)c_q(j), q \in [0,2]$ is rewritten as $w_q(j)$.

$$\text{min: } \frac{1}{2m}\sum_{i\in[1,m]} |c_0(0)a_i + c_1(0)a_i^2 + c_2(0)(1-(a_i-1)^2)$$
$$-\sum_{j\in I^+}[w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1-(x_i(j)-1)^2)]$$
$$+\sum_{j\in I^-}[w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1-(x_i(j)-1)^2)]$$
$$- b| + \lambda \sum_{j\in[1,K]} w_0(j) + \lambda|b|$$

subject to $c_0(0) + c_1(0) + c_2(0) = 1$. See Appendix B.1 for the details of the formulation.

We observe that adjustive linear regression is an extension of the Lasso linear regression except that the error function is the sum of absolute errors in the former and the sum of square errors in the latter. It is not difficult to see that the above minimization can be formulated as a linear program with $O(m + K)$ variables and constraints. In our experiment, we also penalize each weight $w_q(j), q \in [1,2]$ with the same constant λ in a similar fashion to Lasso linear regression..

We solve the above minimization problem to construct a prediction function $\eta_{\Psi,w,b}$. Let $c_q^*(0), q \in [0,2]$, $w_q^*(j), q \in [0,2], j \in [1,K]$ and b^* denote the values of variables $c_q(0), q \in [0,2]$, $w_q(j), q \in [0,2], j \in [1,K]$ and b in an optimal solution, respectively. Let K' denote the number of descriptors $j \in [1,K]$ with $w_0^*(j) > 0$ and $I_{K'}$ denote the set of $j \in [1,K]$ with $w_0^*(j) > 0$. Then we set
$w^*(j) := 0, j \in [1,K]$ with $w_0^*(j) = 0$,
$w^*(j) := w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^+ \cap I_{K'}$,
$w^*(j) := -w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^- \cap I_{K'}$,
$c_q^*(j) := w_q^*(j)/w^*(j), q \in [0,2], j \in I_{K'}$ and
$w^* := (w_0^*(1), w_0^*(2), \ldots, w_0^*(K)) \in \mathbb{R}^K$.

For a set Ψ^* of selected functions $\phi_j(t) = c_0^*(j)t + c_1^*(j)t^2 + c_2^*(j)(1-(t-1)^2), j \in I_{K'}$ with and $\phi_0(t)$ with $\phi_0^{-1}(t) = c_0^*(0)t + c_1^*(0)t^2 + c_2^*(0)(1-(t-1)^2)$ and a hyperplane (w^*, b^*), we construct a prediction function η_{Ψ^*,w^*,b^*}.

We propose the following scheme of executing ALR for constructing a prediction function and evaluating the performance.

1. Given a data set $\mathcal{X} = \{x_i \in \mathbb{R}^K \mid i \in [1,m]\}$ of normalized feature vectors and a set $A = \{a_i \in \mathbb{R} \mid i \in [1,m]\}$ of normalized observed values, we choose a real $\lambda > 0$ possibly from a set of candidates for $\lambda > 0$ so that the performance of a prediction function η_{Ψ^*,w^*,b^*} obtained from an optimal solution (Ψ^*, w^*, b^*) to the ALR $(\mathcal{X}, \lambda)$ attains a criterion, where we may use cross-validation and the test coefficient of determination to know the performance.

2. With the real λ determined in 1, we evaluate the performance of a prediction function obtained with ALR based on cross-validation. We divide the entire set $\mathcal{X}$ into five subsets $\mathcal{X}^{(k)}, k \in [1,5]$. For each $k \in [1,5]$, we use the set $\mathcal{X} \backslash \mathcal{X}^{(k)}$ as a training data to construct a prediction function $\eta_{\Psi,w,b}$ with ALR $(\mathcal{X} \setminus \mathcal{X}^{(k)}, \lambda)$ and compute the coefficient of determination $\mathrm{R}^2(\eta_{\Psi,w,b}, \mathcal{X}^{(k)})$.

7.2.4 MILP Formulation for the Inverse Problem

This section introduces an MILP that simulates the computation process of a prediction function constructed with adjustive linear regression.

Let $x = (x(1), x(2), \ldots, x(K)) \in \mathbb{R}^K$ denote the feature function $f(\mathbb{C})$ of a chemical graph $\mathbb{C}$. Let $c_{\min}(j)$ (resp., $c_{\max}(j)$) denote the minimum (resp., maximum) values of the j-th descriptor in a data set D_π for a chemical property π. Let $\mathrm{atm}_{\mathrm{LB}} \in \mathbb{Z}_+$ (resp., $\mathrm{atm}_{\mathrm{UB}} \in \mathbb{Z}_+$) be a lower bound (resp., an upper bound) on the number of atoms in a chemical graph $\mathbb{C}$ to be inferred. Let $\mathrm{mass}(\mathtt{a})$ denote the observed mass of a chemical element $\mathtt{a} \in \Lambda$, and define $\mathrm{mass}^*(\mathtt{a}) \triangleq \lfloor 10 \cdot \mathrm{mass}(\mathtt{a}) \rfloor$. Let $\mathrm{Ms}_{\mathrm{LB}} \in \mathbb{Z}_+$ (resp., $\mathrm{Ms}_{\mathrm{UB}} \in \mathbb{Z}_+$) be a lower bound (resp., an upper bound) on the sum $\frac{1}{|V(H)|}\sum_{v \in V(H)} \mathrm{mass}^*(\alpha(v))$ in a chemical graph $\mathbb{C}$ to be inferred. Let j_{ms} denote the index $j \in [1,K]$ such that the j-th descriptor $\mathrm{dcp}_j(\mathbb{C})$ is the average mass $\overline{\mathrm{ms}}(\mathbb{C}) = \frac{1}{|V(H)|}\sum_{v \in V(H)} \mathrm{mass}^*(\alpha(v))$. Assume that all other descriptors $\mathrm{dcp}_j(\mathbb{C}), j \in [1,K] \setminus \{j_{\mathrm{ms}}\}$ are integers.

Let $\eta_{\Psi,w,b}$ be a prediction function obtained by adjustive linear regression, where $\Psi = \{\phi_j \mid j \in [0,K]\}$ and (w,b) is a hyperplane.

We first normalize each of the sets of descriptors $x(j), j \in [1,K]$ and the set of observed values $a(\mathbb{C})$ before we apply the prediction function to compute a predicted value $\eta_{\Psi,w,b}(f(\mathbb{C}))$ of a chemical graph $\mathbb{C}$, where the set $\{a_i \mid i \in [1,m]\}$ of observed values in the data set D_π is converted into a set $\{\phi_0^{-1}(\frac{a_i - \underline{a}}{\overline{a} - \underline{a}}) \mid i \in [1,m]\}$, where $\underline{a}$ (resp., $\overline{a}$) denotes the minimum (resp., maximum) value of $a(\mathbb{C})$ over the chemical graphs $\mathbb{C} \in D_\pi$.

Let $\underline{y}^*$ and $\overline{y}^*$ be lower and upper bounds on the predicted value $\eta_{\Psi,w,b}(f(\mathbb{C}))$ of a target chemical graph $\mathbb{C}$, respectively.

We first converted them into $\phi_0^{-1}(\frac{\underline{y}^* - \underline{a}}{\overline{a} - \underline{a}})$ and $\phi_0^{-1}(\frac{\overline{y}^* - \underline{a}}{\overline{a} - \underline{a}})$. We denote by $\psi_j(t)$ the function $\phi_j((t - c_{\min}(j))/(c_{\max}(j) - c_{\min}(j)))$. We pre-compute the values $\psi_j(s)$ for all integers $s \in [c_{\min}(j), c_{\max}(j)]$, $j \in [1,K] \setminus \{j_{\mathrm{ms}}\}$ (resp., $\phi_{j_{\mathrm{ms}}}(\frac{s/i - c_{\min}(j_{\mathrm{ms}})}{c_{\max}(j_{\mathrm{ms}}) - c_{\min}(j_{\mathrm{ms}})})$ for all integers $s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}]$ and $i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]$) as constants.

An MILP that simulates the computation process of a prediction function $\eta_{\Psi,w,b}$ is described as follows.

$\mathcal{M}(x, y; \mathcal{C}_1)$**:**
constants:

- A hyperplane (w, b) with $w \in \mathbb{R}^K$ and $b \in \mathbb{R}$;
- Activation functions $\phi_j : \mathbb{R} \to \mathbb{R}, j \in [0, K]$;
- Real values $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ such that $\underline{y}^* < \overline{y}^*$; Set $\underline{y}^{**} := \phi_0^{-1}(\frac{\underline{y}^* - \underline{a}}{\overline{a} - \underline{a}})$ and $\overline{y}^{**} := \phi_0^{-1}(\frac{\overline{y}^* - \underline{a}}{\overline{a} - \underline{a}})$.
- $c_{\min}(j), c_{\max}(j) \in \mathbb{R}, j \in [1, K]$: the minimum and maximum values of the j-th descriptor in the data set D_π, respectively;
- Reals $\Delta(j, s) \in \mathbb{R}, j \in [1, K] \setminus \{j_{\mathrm{ms}}\}, s \in [c_{\min}(j), c_{\max}(j)]$: $\Delta(j, c_{\min}(j)) := \psi_j(c_{\min}(j))$ and $\Delta(j, s) := \psi_j(s) - \psi_j(s-1), s \in [c_{\min}(j) + 1, c_{\max}(j)]$;
- $\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}} \in \mathbb{Z}_+$: lower and upper bounds on the number of atoms in a chemical graph $\mathbb{C}$ to be inferred; Set $\mathrm{atm}_{\mathrm{LB}} := n_{\mathrm{LB}} + \mathrm{na}_{\mathrm{LB}}(\mathtt{H})$ and $\mathrm{atm}_{\mathrm{UB}} := n^* + \mathrm{na}_{\mathrm{UB}}(\mathtt{H})$;
- $\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}} \in \mathbb{Z}_+$: lower and upper bounds on the sum $\sum_{v \in V(H)} \mathrm{mass}^*(\alpha(v))$; For example, set $\mathrm{Ms}_{\mathrm{LB}} := \lfloor \min\{\mathrm{mass}^*([\mathtt{a}]) \mid \mathtt{a} \in \Lambda, \mathrm{val}(a) = 1\} \cdot (3n_{\mathrm{LB}}/4) + \min\{\mathrm{mass}^*([\mathtt{a}]) \mid \mathtt{a} \in \Lambda, \mathrm{val}(a) \geq 2\} \cdot (n_{\mathrm{LB}}/4) + \mathrm{mass}^*([\mathtt{H}])\mathrm{na}_{\mathrm{LB}}(\mathtt{H}) \rfloor$ and $\mathrm{Ms}_{\mathrm{UB}} := n^* \max\{\mathrm{mass}^*([\mathtt{a}]) \mid \mathtt{a} \in \Lambda\} + \mathrm{mass}^*([\mathtt{H}])\mathrm{na}_{\mathrm{UB}}(\mathtt{H})$;
- $M \in \mathbb{R}_+$: an upper bound on $\zeta(x(j_{\mathrm{ms}}))$; For example, set $M := 2\phi_{j_{\mathrm{ms}}}(1)$;
- Define $\zeta(s, i) \triangleq \phi_{j_{\mathrm{ms}}}(\frac{s/i - c_{\min}(j_{\mathrm{ms}})}{c_{\max}(j_{\mathrm{ms}}) - c_{\min}(j_{\mathrm{ms}})}), s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}], i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]$, where $\psi_{j_{\mathrm{ms}}}(s/i) = \zeta(s, i)$;
- Reals $\Delta_{\mathrm{Ms}}(s, i) \in \mathbb{R}, s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}], i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]$: $\Delta_{\mathrm{Ms}}(\mathrm{Ms}_{\mathrm{LB}}, i) := \zeta(\mathrm{Ms}_{\mathrm{LB}}, i)$ and $\Delta_{\mathrm{Ms}}(s, i) := \zeta(s, i) - \zeta(s-1, i), s \in [\mathrm{Ms}_{\mathrm{LB}} + 1, \mathrm{Ms}_{\mathrm{UB}}], i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]$;
- A real $\varepsilon(j) > 0, j \in [1, K]$: a tolerance. For example, set $\varepsilon(j) := \frac{1}{10^5} \min\{\Delta(j, s) \mid s \in [c_{\min}(j), c_{\max}(j)]\}, j \in [1, K] \setminus \{j_{\mathrm{ms}}\}$ and $\varepsilon(j_{\mathrm{ms}}) := \frac{1}{10^5} \min\{\Delta_{\mathrm{Ms}}(s, i) \mid s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}], i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]\}$:

variables:

- Real variables $\widehat{x}(j) \in \mathbb{R}, j \in [1, K]$: $\widehat{x}(j)$ represents $\psi_j(x(j))$;
- Integer variables $x(j) \in [c_{\min}(j), c_{\max}(j)], j \in [1, K] \setminus \{j_{\mathrm{ms}}\}$: $x(j)$ represents the j-th descriptor in an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$;
- A real variable $x(j_{\mathrm{ms}}) \in \mathbb{R}_+$ with $c_{\min}(j_{\mathrm{ms}}) \leq x(j_{\mathrm{ms}}) \leq c_{\max}(j_{\mathrm{ms}})$: $x(j_{\mathrm{ms}})$ represents the average mass $\overline{\mathrm{ms}}(\mathbb{C})$ in an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$;
- Binary variables $\delta(j, s) \in [0, 1], j \in [1, K] \setminus \{j_{\mathrm{ms}}\}, s \in [c_{\min}(j), c_{\max}(j)]$: $\delta(j, s) = 1 \Leftrightarrow x(j) \geq s$;
- Binary variables $\delta_{\mathrm{atm}}(i) \in [0, 1], i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]$: $\delta_{\mathrm{atm}}(i) = 1 \Leftrightarrow |V(H)| = i$;
- Binary variables $\delta_{\mathrm{Ms}}(s) \in [0, 1], s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}]$ $\delta_{\mathrm{Ms}}(i) = 1 \Leftrightarrow \sum_{v \in V(H)} \mathrm{mass}^*(\alpha(v)) \geq s$;

constraints:

$$\underline{y}^{**} \leq \sum_{j\in[1,K]} w(j)\widehat{x}(j) + b \leq \overline{y}^{**}, \tag{7.2.1}$$

$$\sum_{s\in[c_{\min}(j),c_{\max}(j)]} \delta(j,s) + c_{\min}(j) - 1 = x(j),$$
$$\delta(j,s) \geq \delta(j,s+1), \qquad s \in [c_{\min}(j), c_{\max}(j)-1],$$

$$\sum_{s\in[c_{\min}(j),c_{\max}(j)]} \Delta(j,s)\delta(j,s) - \epsilon(j) \leq \widehat{x}(j) \leq \sum_{s\in[c_{\min}(j),c_{\max}(j)]} \Delta(j,s)\delta(j,s) + \epsilon(j),$$
$$j \in [1,K] \setminus \{j_{\mathrm{ms}}\}, \tag{7.2.2}$$

$$\sum_{i\in[\mathrm{atm}_{\mathrm{LB}},\mathrm{atm}_{\mathrm{UB}}]} \delta_{\mathrm{atm}}(i) = 1,$$
$$\sum_{i\in[\mathrm{atm}_{\mathrm{LB}},\mathrm{atm}_{\mathrm{UB}}]} i\cdot\delta_{\mathrm{atm}}(i) = n_G + \mathrm{na}^{\mathrm{ex}}([\mathtt{H}]^{\mathrm{ex}}),$$
$$\sum_{\mathtt{a}\in\Lambda} \mathrm{mass}^*(\mathtt{a})\cdot\mathrm{na}([\mathtt{a}]) = \sum_{s\in[\mathrm{Ms}_{\mathrm{LB}},\mathrm{Ms}_{\mathrm{UB}}]} \delta_{\mathrm{Ms}}(s) + \mathrm{Ms}_{\mathrm{LB}} - 1, \tag{7.2.3}$$

$$\delta_{\mathrm{Ms}}(s) \geq \delta_{\mathrm{Ms}}(s+1), \qquad s \in [\mathrm{Ms}_{\mathrm{LB}}, \mathrm{Ms}_{\mathrm{UB}}-1], \tag{7.2.4}$$

$$\sum_{s\in[\mathrm{Ms}_{\mathrm{LB}},\mathrm{Ms}_{\mathrm{UB}}]} \Delta_{\mathrm{Ms}}(s,i)\delta_{\mathrm{Ms}}(s) - M\cdot(1\text{–}\delta_{\mathrm{atm}}(i)) - \epsilon(j_{\mathrm{ms}}) \leq \widehat{x}(j_{\mathrm{ms}}) \leq$$
$$\sum_{s\in[\mathrm{Ms}_{\mathrm{LB}},\mathrm{Ms}_{\mathrm{UB}}]} \Delta_{\mathrm{Ms}}(s,i)\delta_{\mathrm{Ms}}(s) + M\cdot(1\text{–}\delta_{\mathrm{atm}}(i)) + \epsilon(j_{\mathrm{ms}}),$$
$$i \in [\mathrm{atm}_{\mathrm{LB}}, \mathrm{atm}_{\mathrm{UB}}]. \tag{7.2.5}$$

7.3 Experimental Results

We implemented our method of Stages 1 to 5 for inferring chemical graphs under a given topological specification and conducted experiments to evaluate the computational efficiency. We executed the experiments on a PC with Processor: Core i7-9700 (3.0GHz; 4.7 GHz at the maximum) and Memory: 16 GB RAM DDR4. We used scikit-learn version 0.23.2 with Python 3.8.5 for executing linear

regression with Lasso function or constructing an ANN. To solve an LP or MILP instance, we used `CPLEX` version 12.10.

Results on Phase 1 We implemented Stages 1, 2 and 3 in Phase 1 as follows.

We have conducted experiments of adjustive linear regression and for 37 chemical properties of monomers (resp., ten chemical properties of polymers) and we found that the test coefficient of determination R^2 of ALR exceeds 0.6 for 28 properties of monomers: isotropic polarizability (Alpha) and boiling point (Bp), critical pressure (Cp); critical temperature (Ct); heat capacity at 298.15K (Cv); dissociation constants (Dc); electron density on the most positive atom (EDPA); flash point (Fp); energy difference between the highest and lowest unoccupied molecular orbitals (Gap); heat of atomization (Ha); heat of combustion (Hc); heat of formation (Hf); energy of highest occupied molecular orbital (Homo); heat of vaporization (Hv); isobaric heat capacities in liquid phase (IhcL); isobaric heat capacities in solid phase (IhcS); Kováts retention index (KvI); octanol/water partition coefficient (Kow); lipophilicity (Lp); energy of lowest unoccupied molecular orbital (Lumo); melting point (Mp); optical rotation (OptR); refractive index (Rf); solubility (Sl); surface tension (SfT); internal energy at 0K (U0); viscosity (Vis); and vapor density (Vd) and that the test coefficient of determination R^2 of ALR exceeds 0.8 for eight properties of polymers: experimental amorphous density (AmD); characteristic ratio (ChaR); dielectric constant(DeC); heat capacity liquid (HcL); heat capacity solid (HcS); mol volume (MlV); refractive index (RfId); and glass transition (Tg), where we include the result of property permittivity (Prm) for a comparison with Lasso linear regression and ANN.

We used data sets are provided by HSDB from PubChem [27] for Cp, Ct, Dc, Fp, Hc, Hv, Kow, OptR, Rf and Vd M. Jalali-Heravi and M. Fatemi [26] for EDPA and KvI, Roy and Saha [39] for Bp, Ha, Hf and Mp, Ramakrishnan et al. [37] for Alpha, Cv, Lumo and U0, Goussard et al. [18] for SfT, Goussard et al. [19] for Vis, R. Naef [34] for IhcL and IhcS, Xiao [53] for Lp and Delaney [54] for Sl. Properties Alpha, Cv, Homo, Lumo and U0 share a common original data set D^* with more than 130,000 compounds, and we used a set D_π of 1,000 compounds randomly selected from D^* as a common data set of these four properties π in this experiment.

We used data sets of polymers provided by Bicerano [8], where we did not include any polymer whose chemical formula could not be found by its name in the book. For property ChaR (resp., RfId), we remove the following polymer as an outlier from the original data set:
ethyleneTerephthalate, oxy(2-methyl-6-phenyl-1_4-phenylene) and N-vinylCarbazole

(resp., 2-decyl-1_4-butadiene).

Stage 1 We set a graph class $\mathcal{G}$ to be the set of all chemical graphs with any graph structure, and set a branch-parameter ρ to be 2.

For each of the properties, we first select a set Λ of chemical elements and then collect a data set D_π on chemical graphs over the set Λ of chemical elements.

Table 7.1 shows the size and range of data sets that we prepared for each chemical property in Stage 1, where we denote the following: $|\Lambda|$: the size $|\Lambda|$ of Λ used in the data set D_π; $|D_\pi|$: the size of data set D_π over Λ for the property π; and K: the number of descriptors in a feature vector $f(\mathbb{C})$.

Stage 2 We used the feature function defined in our chemical model without suppressing hydrogen. We standardize the range of each descriptor and the range of property values $a(\mathbb{C}), \mathbb{C} \in D_\pi$.

Stage 3 For each chemical property π, we select a penalty value λ_π for a constant λ in ALR$(\mathcal{X}, \lambda)$ by conducting linear regression as a preliminary experiment.

We conducted an experiment in Stage 3 to evaluate the performance of the prediction function based on cross-validation. For a property π, an execution of a *cross-validation* consists of five trials of constructing a prediction function as follows.

Tables 7.1 and 7.2 show the results on Stages 2 and 3 for the properties on monomers and polymers, respectively, where we denote the following: time: the average time (sec.) to construct a prediction function with ALR by solving an LP with $O(|D_\pi| + K)$ variables and constraints over all 50 trials in ten cross-validations; ALR: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with ALR; LLR: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with Lasso linear regression; and ANN: the median of test R^2 over all 50 trials in ten cross-validations for prediction functions constructed with ANNs (see Appendix B.2 for the details of constructing a prediction function with ANNs).

From Tables 7.1 and 7.2, we see that ALR performs well for most of the properties in our experiments, The performance by ALR is inferior to that by LLR or ANN for some properities such as GAP, HOMO, LUMO, OPTR, SL, SFT and PRM, whereas ALR outperforms LLR and ANN for properties BP, CT, HV, KVI, VD, CHAR, RFID and TG. It should be noted that ALR drastically improves the result for properties CT and HV.

Results on Phase 2 To execute Stages 4 and 5 in Phase 2, we used a set of seven instances $I_{\rm a}$, $I^i_{\rm b}, i \in [1, 4]$, $I_{\rm c}$ and $I_{\rm d}$ based on the seed graphs prepared in [43].

Table 7.1. Results in Phase 1 for monomers.

π	$\lvert\Lambda\rvert$	$\lvert D_\pi\rvert$	K	time	ALR	LLR	ANN
Alpha	10	977	297	3.00	0.953	0.961	0.888
Bp	4	370	184	1.42	0.816	0.599	0.765
Bp	7	444	230	2.02	0.832	0.663	0.720
Cp	4	125	112	0.15	0.650	0.445	0.694
Cp	6	131	119	0.12	0.690	0.555	0.727
Ct	4	125	113	0.24	0.900	0.037	0.357
Ct	6	132	121	0.28	0.895	0.048	0.356
Cv	10	977	297	4.57	0.966	0.970	0.911
Dc	7	161	130	0.35	0.602	0.574	0.622
EDPA	3	52	64	0.06	0.999	0.999	0.992
Fp	4	36	183	1.31	0.719	0.589	0.746
Fp	7	424	229	1.92	0.684	0.571	0.745
Gap	10	977	297	4.77	0.755	0.784	0.795
Ha	4	115	115	0.29	0.998	0.997	0.926
Hc	4	255	154	0.74	0.986	0.946	0.848
Hc	7	282	177	0.84	0.986	0.951	0.903
Hf	3	82	74	0.05	0.982	0.987	0.928
Homo	10	977	297	4.95	0.689	0.841	0.689
Hv	4	95	105	0.19	0.626	-13.7	-8.44
IhcL	4	770	256	3.24	0.987	0.986	0.974
IhcL	7	865	316	1.98	0.989	0.985	0.971
IhcS	7	581	192	1.72	0.971	0.985	0.971
IhcS	11	668	228	2.21	0.974	0.982	0.968
KvI	3	52	64	0.05	0.838	0.677	0.727
Kow	4	684	223	3.13	0.964	0.953	0.952
Kow	8	899	303	4.95	0.952	0.927	0.937
Lp	4	615	186	1.81	0.844	0.856	0.867
Lp	8	936	231	3.37	0.807	0.840	0.859
Lumo	10	977	297	2.75	0.833	0.841	0.860
Mp	4	467	197	1.78	0.831	0.810	0.799
Mp	8	577	255	2.99	0.807	0.810	0.820
OptR	4	147	107	0.24	0.876	0.825	0.919
OptR	6	157	123	0.27	0.870	0.825	0.878
Rf	4	166	142	0.24	0.685	0.619	0.521
Sl	4	673	217	1.21	0.784	0.808	0.848
Sl	8	915	300	2.33	0.828	0.808	0.861
SfT	4	247	128	0.67	0.847	0.927	0.859
U0	10	977	297	2.40	0.995	0.999	0.890
Vis	4	282	126	0.37	0.911	0.893	0.929
Vd	4	474	214	2.24	0.985	0.927	0.912
Vd	7	551	256	2.28	0.980	0.942	0.889

Table 7.2. Results in Phase 1 for polymers.

π	$\|\Lambda\|$	$\|D_\pi\|$	K	time	ALR	LLR	ANN
AMD	4	86	83	0.09	0.933	0.914	0.885
AMD	7	93	94	0.10	0.917	0.918	0.823
CHAR	3	24	56	0.02	0.904	0.650	0.616
CHAR	4	27	67	0.03	0.835	0.431	0.641
DEC	7	37	72	0.04	0.918	0.761	0.641
HCL	4	52	67	0.06	0.996	0.990	0.969
HCL	7	55	81	0.05	0.992	0.987	0.970
HCS	4	54	75	0.07	0.963	0.968	0.893
HCS	7	59	92	0.09	0.983	0.961	0.880
MLV	4	86	83	0.10	0.998	0.996	0.931
MLV	7	93	94	0.09	0.997	0.994	0.894
PRM	4	112	69	0.09	0.505	0.801	0.801
PRM	5	131	73	0.09	0.489	0.784	0.735
RFID	5	91	96	0.15	0.953	0.852	0.871
RFID	7	124	124	0.21	0.956	0.832	0.891
TG	4	204	101	0.23	0.923	0.902	0.883
TG	7	232	118	0.54	0.927	0.894	0.881

Stage 4 We executed Stage 4 for heat of vaporization (HV). Table 7.3 shows the computational results of the experiment in Stage 4 for the two properties, where we denote the following: $\underline{y}^*$, $\overline{y}^*$: lower and upper bounds $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ on the value $a(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to be inferred; I-time: the time (sec.) to solve the MILP in Stage 4; and n: the number $n(\mathbb{C}^\dagger)$ of non-hydrogen atoms in the chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4. The result suggests that ALR is useful to infer relatively large size chemical graphs from given chemical properties. Note that hydrogen atoms can be recovered after getting hydrogen-suppressed chemical graphs.

Stage 5 We executed Stage 5 to generate a more number of target chemical graphs $\mathbb{C}^*$, where we call a chemical graph $\mathbb{C}^*$ a *chemical isomer* of a target chemical graph $\mathbb{C}^\dagger$ of a topological specification σ if $f(\mathbb{C}^*) = f(\mathbb{C}^\dagger)$ and $\mathbb{C}^*$ also satisfies the same topological specification σ. We computed chemical isomers $\mathbb{C}^*$ of each target chemical graph $\mathbb{C}^\dagger$ inferred in Stage 4. We execute the algorithm due to [43] to generate chemical isomers of $\mathbb{C}^\dagger$ up to 100 when the number of all chemical isomers exceeds 100. The algorithm can compute a lower bound on the total number of all chemical isomers $\mathbb{C}^\dagger$ without generating all of them.

Table 7.3. Results of Stages 4 and 5 for Hv.

inst.	$\underline{y}^*$, $\overline{y}^*$	I-time	n	D-time	$\mathbb{C}$-LB	$\#\mathbb{C}$
$I_{\rm a}$	145, 150	24.9	37	0.0632	2	2
$I_{\rm b}^1$	190, 195	146.6	35	0.121	30	30
$I_{\rm b}^2$	290, 295	188.8	46	0.154	604	100
$I_{\rm b}^3$	165, 170	1167.2	45	36.8	7.5×10^6	100
$I_{\rm b}^4$	250, 255	313.7	50	0.166	2208	100
$I_{\rm c}$	285, 290	102.5	50	0.016	1	1
$I_{\rm d}$	245, 250	351.9	40	5.53	3.9×10^5	100

Table 7.3 shows the computational results of the experiment in Stage 5 for property Hv, where we denote the following: D-time: the running time (sec.) to execute the dynamic programming algorithm in Stage 5 to compute a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ and generate all (or up to 100) chemical isomers $\mathbb{C}^*$; $\mathbb{C}$-LB: a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$; and $\#\mathbb{C}$: the number of all (or up to 100) chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ generated in Stage 5. The result suggests that ALR is useful not only for inference of chemical graphs but also for enumeration of chemical graphs.

7.4 Concluding Remarks

In this chapter, we proposed a new machine learning method, adjustive linear regression (ALR), which has the following feature: (i) ALR is an extension of the Lasso linear regression except for the definition of error functions; (ii) ALR is a special case of an ANN except that a choice of activation functions is also optimized differently from the standard ANNs and the definition of error functions; and (iii) ALR can be executed exactly by solving the equivalent linear program with $O(m+K)$ variables and constraints for a set of m data with K descriptors. Even though ALR is a special case of an ANN with non-linear activation functions, we still can read the relationship between cause and effect from a prediction function due to the simple structure of ALR.

In this chapter, we used a quadratic function for a set Ψ of activation functions ϕ. We can use many different functions such as sigmoid function and ramp functions, where the non-linearity of a function does not affect to derive a linear program for ALR. The proposed method/system is available at GitHub `https://github.com/ku-dml/mol-infer/tree/master/ALR` (Accessed: May 24, 2023).

8 Two-Layered Model with Quadratic Descriptors

8.1 Introduction

A novel framework for inferring chemical graphs has been developed [6, 23, 43, 66] based on an idea of formulating as a mixed integer linear programming (MILP) the computation process of a prediction function constructed by a machine learning method. Given a prediction function η and a topology specification σ, the task of the second phase is to infer chemical graphs $\mathbb{C}^* \in \mathcal{G}_\sigma$ such that $\underline{y}^* \leq \eta(f(\mathbb{C}^*)) \leq \overline{y}^*$. For this, we formulate an MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ that represents (i) the computation process of $x := f(\mathbb{C})$ from a chemical graph $\mathbb{C}$ in the feature function f; (ii) that of $y := \eta(x)$ from a vector $x \in \mathbb{R}^K$ in the prediction function η; and (iii) the constraint $\mathbb{C} \in \mathcal{G}_\sigma$. Given an interval with $\underline{y}^*, \overline{y}^* \in \mathbb{R}$, we solve the MILP $\mathcal{M}_{f,\eta,\sigma}$ to find a feature vector $x^* \in \mathbb{R}^K$ and a chemical graph $\mathbb{C}^\dagger \in \mathcal{G}_\sigma$ such that $f(\mathbb{C}^\dagger) = x^*$ and $\underline{y}^* \leq \eta(x^*) \leq \overline{y}^*$ (where if the MILP instance is infeasible then this suggests that $\mathcal{G}_\sigma$ does not contain such a desired chemical graph). In the second phase, we next generate some other desired chemical graphs based on the solution $\mathbb{C}^\dagger$. For this, the following two methods have been designed.

The first method constructs isomers of $\mathbb{C}^\dagger$ without solving any new MILP. In this method, we first decompose the chemical graph $\mathbb{C}^\dagger$ into a set of chemical acyclic graphs $T_1^\dagger, T_2^\dagger, \ldots, T_q^\dagger$, and next construct a set $\mathcal{T}_i$ of isomers T_i^* of each tree $T_i^\dagger$ such that $f(T_i^*) = f(T_i^\dagger)$ by a dynamic programming algorithm due to Azam et al. [7]. Finally we choose an isomer $T_i^* \in \mathcal{T}_i$ for each $i = 1, 2, \ldots, q$ and assemble them into an isomer $\mathbb{C}^* \in \mathcal{G}_\sigma$ of $\mathbb{C}^\dagger$ such that $f(\mathbb{C}^*) = x^* = f(\mathbb{C}^\dagger)$. The first method generates such isomers $\mathbb{C}_1^*, \mathbb{C}_2^*, \ldots$ which we call *recombination solutions* of $\mathbb{C}^\dagger$.

The second method constructs new solutions by solving the MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ with an additional set Θ of new linear constraints [6]. We first prepare arbitrary p_{dim} linear functions $\theta_j : \mathbb{R}^K \to \mathbb{R}, j = 1, 2, \ldots, p_{\mathrm{dim}}$ and consider a neighbor of $\mathbb{C}^\dagger$ defined by a set of chemical graphs $\mathbb{C}^*$ that satisfy linear constraints $k\delta \leq |\theta_j(f(\mathbb{C}^*)) - \theta_j(f(\mathbb{C}^\dagger))| \leq (k+1)\delta, j = 1, 2, \ldots, p_{\mathrm{dim}}$ for a small real $\delta > 0$ and an integer $k \geq 1$. By changing the integer k systematically, we can search

for new solutions $\mathbb{C}_1^\dagger, \mathbb{C}_2^\dagger, \ldots \in \mathcal{G}_\sigma$ of MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ with constraint Θ such that the feature vectors $x^* = f(\mathbb{C}^\dagger), x_1^* = f(\mathbb{C}_1^\dagger), x_2^* = f(\mathbb{C}_2^\dagger), \ldots$ are all slightly different. We call these chemical graphs $\mathbb{C}_1^\dagger, \mathbb{C}_2^\dagger, \ldots$ *neighbor solutions* of $\mathbb{C}^\dagger$, where a neighbor solution is not an isomer of $\mathbb{C}^\dagger$.

The main reason why the framework can infer a chemical compound with 50 non-hydrogen atoms is that the descriptors of a chemical graph are defined on local graph structures in the two-layered model and thereby an MILP necessary to represent a chemical graph can be formulated as a considerably compact form that is efficiently solvable by a standard solver.

In the framework, all descriptors $x(1), x(2), \ldots, x(K)$ in the feature vector $x = f(\mathbb{C})$ are mainly the frequencies of local graph structures based on the two-layered model by which a chemical graph $\mathbb{C}$ is regarded as a pair of interior and exterior structures (see Section 6.2 for details). To derive a compact MILP formulation to infer a chemical graph, it is important to use the current definition of descriptors. However, there are some chemical properties for which the performance of a prediction function constructed with the feature function f remains rather low. To improve the learning performance with the same two-layered model, we add as a new descriptor the product $x(i)x(j)$ (or $x(i)(1-x(j))$ of two descriptors (where each descriptor is assumed to be normalized within 0 and 1) and call such a new descriptor a *quadratic descriptor*. This drastically increases the number of descriptors, which would take extra running time in learning or cause overfitting to the data set. Moreover, computing quadratic descriptors cannot be directly formulated as a set of linear constraints in the original MILP. For this, we introduce a method of reducing a set of descriptors into a smaller set that delivers a prediction function with a higher performance. We also design an MILP formulation for representing a quadratic term $x(i)x(j)$. Based on the same MILP $\mathcal{M}(x, g; \mathcal{C}_2)$ formulation proposed by Zhu et al. [66], we implemented the framework to treat the feature function with quadratic descriptors. From the results of our computational experiments on over 40 chemical properties, we observe that our new method of utilizing quadratic descriptors improved the performance of a prediction function for many chemical properties.

The rest of this chapter is organized as follows. Section 8.2 introduces the quadratic descriptors and a heuristic to reduce the size of descriptors. Section 8.3 introduces a formulation for computing a quadratic descriptor in an MILP. Section 8.4 reports the results on computational experiments conducted for 42 chemical properties such as critical pressure, dissociation constants and lipophilicity for monomers and characteristic ratio and refractive index for polymers. Section 8.5 makes some concluding remarks. We refer [65] for the complete MILP

formulations.

8.2 Quadratic Descriptors

In the framework with the two-layered model, the feature vector f mainly consists of the frequency of edge-configurations of the interior-edges and the frequency of chemical rooted trees among the set of chemical rooted trees $\mathbb{C}[u]$ over all interior-vertices u. See Section 6.2.1 for all these descriptors $x(1), x(2), \ldots, x(K_1)$, which are called *linear descriptors*.

In the framework for polymers [23], a polymer is treated as a chemical graph of its repeating unit, where we call an edge e a *link-edge* when it lays on any path between the two joint-points of the repeating unit, and call the end-vertices of a link-edge *connecting-vertices*. The set of descriptors for a polymer is defined analogously with the above set for a monomer except for $\mathrm{dcp}_2(\mathbb{C})$ is replaced with the number of link-edges and the following two kinds of descriptors are added: the frequency $\mathrm{ec}_\gamma(\mathbb{C})$ of edge-configuration γ of link-edges; and the frequency of chemical symbols of connecting-vertices (see [23] for the details).

We denote by $D_\pi^{(1)} := \{x(k) \mid k \in [1, K_1]\}$ the set of descriptors constructed over a data set for a property π. In this chapter, we also use a quadratic term $x(i)x(j)$ (or $x(i)(1 - x(j))$), $1 \leq i \leq j \leq K_1$ as a new descriptor, where we assume that each $x(i)$ is normalized between 0 and 1. We call such a term $x(i)x(j)$ (or $x(i)(1-x(j))$), $1 \leq i \leq j \leq K_1$ a *quadratic descriptor* and denote by $D_\pi^{(2)} := \{x(i)x(j) \mid 1 \leq i \leq j \leq K_1\} \cup \{x(i)(1 - x(j)) \mid 1 \leq i, j \leq K_1\}$ the set of quadratic descriptors.

To construct a prediction function, we use the union $D_\pi^{(1)} \cup D_\pi^{(2)}$. This set of descriptors is usually excessive in constructing a prediction function, and we reduce it to a smaller set of descriptors to construct a *feature function* $f : \mathbb{R}^K \to \mathbb{R}$, where K is the number of resulting descriptors. See Section 8.2.1 for methods of reducing descriptors.

8.2.1 Methods for Reducing Descriptors

Let $\mathcal{C}$ be a set of chemical compounds, D be a set of all descriptors and $K^* \in [1, |D|]$ be a number of descriptors we want to choose from D.

Given a data set $\mathcal{C}$, a set D of descriptors and a real $\lambda > 0$, let Des-set-LLR$(\mathcal{C}, D, \lambda)$ denote the set S of descriptors $d \in D$ such that $w(d) = 0$ for the hyperplane (w, b) output by LLR$(\mathcal{C}, D, \lambda)$ (where we numerically treat $w(d)$ with $|w(d)| \leq 10^{-6}$ as 0 in our experiment).

A method based on Lasso linear regression

Since the Lasso linear regression finds some number of descriptors $d \in D$ with $w(d) = 0$ in the output hyperplane (w, b), we can reduce a given set of descriptors by applying the Lasso linear regression repeatedly. Choose parameters $c_{\max}$ and $d_{\max}$ so that $\mathrm{LLR}(\mathcal{C}, D, \lambda)$ can be executed in a reasonable running time when $|\mathcal{C}| \leq c_{\max}$ and $|D| \leq d_{\max}$. Let $\widetilde{K} \in [1, |D|]$ be an integer for the number of descriptors that we choose from a given set D of descriptors. The method is described in Algorithm 1.

Algorithm 1 LLR-Reduce$(\mathcal{C}, D)$

Input: A data set $\mathcal{C}$ and a set D of descriptors;
Output: A subset $\widetilde{D} \subseteq D$ with $|\widetilde{D}| = \widetilde{K}$.

1: Initialize $D' := D$;
2: **while** $|D'| > \widetilde{K}$ **do**
3: Partition D' randomly into disjoint subsets $D_1, D_2, \ldots, D_p$ such that $|D_i| \leq d_{\max}$ for each i;
4: **for each** $i = 1, 2, ..., p$ **do**
5: Choose a subset $\mathcal{C}_i$ with $|\mathcal{C}_i| = \min\{c_{\max}, |\mathcal{C}|\}$ of $\mathcal{C}$ randomly;
6: $D'_i :=$Des-set-LLR$(\mathcal{C}_i, D_i, \lambda)$ for some $\lambda > 0$
7: **end for**
8: $D' := D'_1 \cup D'_2 \cup \cdots \cup D'_p$
9: **end while**
10: Output $\widetilde{D} := D'$ after adding to D' extra $\widetilde{K} - |D'|$ descriptors from the previous set D' when $|D'| < \widetilde{K}$ by using the K-best method.

Here, we set $c_{\max} := 200$, $d_{\max} := 200$ and $\widetilde{K} := 5000$ in our computational experiment.

A method based on backward stepwise procedure

A backward stepwise procedure [13] reduces the number of descriptors one by one choosing the one removal of which maximizes the learning performance and outputs a subset with the maximum learning performance among all subsets during the reduction iteration.

For a subset $S \subseteq D$ and a positive integer p, let $\mathrm{R}^2_{\mathrm{CV,MLR}}(\mathcal{C}, S, p)$ denote $\mathrm{R}^2_{\mathrm{CV}}(\mathcal{C}, S, p)$ for constructing a prediction function $\eta_{w,b}$ by $\mathrm{MLR}(\mathcal{C}, S)$. We define a performance evaluation function $g_p : 2^D \to \mathbb{R}$ for an integer $p \geq 1$ such that $g_p(S) = \mathrm{R}^2_{\mathrm{CV,MLR}}(\mathcal{C}, S, p)$. The backward stepwise procedure with this function g_p is described in Algorithm 2.

Algorithm 2 BS-Reduce($\mathcal{C}, D, p$)

Input: A data set $\mathcal{C}$, a set D of descriptors, an integer $p \geq 1$ and a performance evaluation function $g_p : 2^D \to \mathbb{R}$ defined above;

Output: A subset $D^* \subseteq D$.

1: Compute $\ell_{\text{best}} := g_p(D)$; Initialize $D_{\text{best}} := D' := D$;
2: **while** $D' \neq \emptyset$ **do**
3: Compute $\ell(d) := g_p(D' \setminus \{d\})$ for each descriptor $d \in D'$;
4: Set $d^* \in D'$ to be a descriptor that maximizes $\ell(d)$ over all $d \in D'$;
5: Update $D' := D' \setminus \{d^*\}$;
6: **if** $\ell(d^*) > \ell_{\text{best}}$ **then**
7: Update $D_{\text{best}} := D'$ and $\ell_{\text{best}} := \ell(d^*)$
8: **end if**
9: **end while**
10: Output $D^* := D_{\text{best}}$.

Based on the Lasso linear regression and the backward stepwise procedure, we design the following method described in Algorithm 3 for choosing a subset D^* of a given set D of descriptors. We are given a give set A of 17 real numbers and a set $B(a)$ of 16 real numbers close to each number $a \in A$. The method first choose a best parameter $\lambda_{\text{best}} \in A$ to construct a prediction function by LLR and then choose a subset $D_i \subseteq D$ for each $\lambda_i \in B(\lambda_{\text{best}})$ by the backward stepwise procedure. The procedure takes $O(|D|^2)$ iterations which may take a large amount of running time. We introduce an upper bound $s_{\max}$ on the size of an input descriptor D for the backward stepwise procedure. Let p_1, p_2 and p_3 be integer parameters that control the number of executions of cross-validations to evaluate the learning performance in the method.

Algorithm 3 Select-Des-set($\mathcal{C}, D$)

Input: A data set $\mathcal{C}$, a set D of descriptors, a set $A = \{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.05, 0.1, 0.5, 0.75, 1, 2, 5, 10, 25, 50, 100\}$, and a set $B(\lambda)$ of 16 real numbers close to each $\lambda \in A$;

Output: A subset D^* of D.

1: **for each** $\lambda \in A$ **do**
2: Compute $D_\lambda :=$Des-set-LLR($\mathcal{C}, D, \lambda$) and $\ell_\lambda := \mathrm{R}^2_{\mathrm{CV,MLR}}(\mathcal{C}, D_\lambda, p_1)$
3: **end for**
4: Set λ_{best} to be a $\lambda \in A$ that maximizes ℓ_λ;
5: Denote $B(\lambda_{\text{best}})$ by $\{\lambda_1, \lambda_2, \ldots, \lambda_{16}\}$;
6: **for each** $i \in [1, 16]$ **do**
7: Compute $D_i :=$ Des-set-LLR($\mathcal{C}, D, \lambda_i$) and let $(w, b), w \in \mathbb{R}^{|D|}, b \in \mathbb{R}$ be

the hyperplane obtained by this LLR;
8: **if** $|D_i| \leq s_{\max}$ **then**
9: $D'_i := D_i$
10: **else**
11: Let D'_i consist of $s_{\max}$ descriptors $d \in D_i$ that have the $s_{\max}$ largest absolute values $|w(d)|$ in the weight sets $\{w(d) \mid d \in D_i\}$ of the hyperplane (w, b)
12: **end if**
13: $D_i^\dagger$:=BS-Reduce$(\mathcal{C}, D'_i, p_2)$;
14: $\ell_i := \mathrm{R}^2_{\mathrm{CV,MLR}}(\mathcal{C}, D_i^\dagger, p_3)$
15: **end for**
16: Set D^* to be a set $D_i^\dagger$ that maximizes $\ell_i, i \in [1, 16]$.

In our computational experiment in this chapter, we set $p_1 := p_2 := p_3 := 5$ and $s_{\max} := 150 + 10^4/(|\mathcal{C}| + 200)$.

8.3 Compute a Quadratic Term in an MILP

This section introduces an MILP formulation for computing the product of two descriptors in an MILP.

Given two real values x and y with $0 \leq x \leq 1$ and $0 \leq y \leq 1$, the computing process of the product $z = xy$ can be approximately formulated as the following MILP. First regard $(2^{p+1} - 1)x$ as an integer with a binary expression of $p + 1$ bits, where $x^{(j)} \in [0, 1]$ denotes the value of the j-th bit. Then compute $y \cdot x^{(j)}$ which becomes the j-th bit $z^{(j)}$ of $(2^{p+1} - 1)z$.

constants:

- x, y: reals with $0 \leq x, y \leq 1$;

- p: a positive integer;

variables:

- z, $z^{(j)}, j \in [0, p]$: reals with $0 \leq z, z^{(j)} \leq 1$;

- $x^{(j)} \in [0, 1], j \in [0, p]$: binary variables;

constraints:

$$\begin{aligned}
\sum_{j\in[0,p]} 2^j x^{(j)} - 1 \le (2^{p+1}-1)x \le \sum_{j\in[0,p]} 2^j x^{(j)}, & \\
z^{(j)} \le x^{(j)}, & \quad j \in [0,p], \\
y - (1 - x^{(j)}) \le z^{(j)} \le y + (1 - x^{(j)}), & \quad j \in [0,p], \\
z = \frac{1}{2^{p+1}-1} \sum_{j\in[0,p]} 2^j z^{(j)}. & \qquad (8.3.1)
\end{aligned}$$

Note that the necessary number of integer variables for computing xy for one pair of x and y is p. In this chapter, we set $p := 6$ in our computational experiment. The relative error by $p = 6$ in the above method is at most $\frac{1}{2^{p+1}-1} = 1/127$, which is around 0.8%.

8.4 Experimental Results

With our new method of choosing descriptors and formulating an MILP to treat quadratic descriptors in the two-layered model, we implemented the framework for inferring chemical graphs and conducted experiments to evaluate the computational efficiency. We executed the experiments on a PC with Processor: Core i7-9700 (3.0GHz; 4.7 GHz at the maximum) and Memory: 16 GB RAM DDR4. To construct a prediction function by LLR, MLR or ANN, we used `scikit-learn` version 1.0.2 with Python 3.8.12, MLPRegressor and ReLU activation function for ANN, Lasso function for LLR, and LinearRegression function for MLR.

8.4.1 Results on the First Phase of the Framework

Chemical properties We implemented the first phase for the following 32 chemical properties of monomers and ten chemical properties of polymers.

For monomers, we used the following data sets:

- biological half life (BHL), boiling point (Bp), critical temperature (Ct), critical pressure (Cp), dissociation constants (Dc), flash point in closed cup (Fp), heat of combustion (Hc), heat of vaporization (Hv), octanol/water partition coefficient (Kow), melting point (Mp), optical rotation (OptR), refractive index of trees (RfIdT), vapor density (Vd) and vapor pressure (Vp), provided by HSDB from PubChem [27];

- electron density on the most positive atom (EDPA) and Kovats retention index (Kov) by M. Jalali-Heravi and M. Fatemi [26];

- entropy (ET) by P. Duchowicz et al. [14];
- heat of atomization (HA) and heat of formation (HF) by K. Roy and A. Saha [39];
- surface tension (SFT) by V. Goussard et al. [18];
- viscosity (VIS) by V. Goussard et al. [19];
- isobaric heat capacities liquid (LHCL) and isobaric heat capacities solid (LHCS) by R. Naef [34];
- lipophilicity (LP) by N. Xiao [53];
- flammable limits lower of organics (FLMLO) by S. Yuan et al. [58];
- molar refraction at 20 degree (MR) by Y. M. Ponce [36]; and
- solubility (SL) by ESOL [54], energy of highest occupied molecular orbital (HOMO), energy of lowest unoccupied molecular orbital (LUMO), the energy difference between HOMO and LUMO (GAP), isotropic polarizability (ALPHA), heat capacity at 298.15K (CV), internal energy at 0K (U0) and electric dipole moment (MU) provided by ESOL [54], where the properties from HOMO to MU are based on a common data set QM9.

The data set QM9 contains more than 130,000 compounds. In our experiment, we use a set of 1,000 compounds randomly selected from the data set. For property HV, we remove the chemical compound with CID=7947 as an outlier from the original data set.

For polymers, we used the following data provided by J. Bicerano [8]:

- experimental amorphous density (AMD), characteristic ratio (CHAR), dielectric constant (DIEC), dissipation factor (DISF), heat capacity in liquid (HCL), heat capacity in solid (HCS), mol volume (MLV), permittivity (PRM), refractive index of polymers (RFIDP) and glass transition (TG),

where we excluded from our test data set every polymer whose chemical formula could not be found by its name in the book [8]. We remark that the previous learning experiments for $\pi \in \{\text{CHAR}, \text{RFIDP}\}$ based on the two-layered model due to Azam et al. [6] and Zhu et al. [67] excluded some number of polymers as outliers. In our experiments, we do not exclude any polymer from the original data set as outliers for these properties.

Setting data sets For each property π, we first select a set Λ of chemical elements and then collect a data set $\mathcal{C}_\pi$ on chemical graphs over the set Λ of chemical

Table 8.1. Results of setting data sets for monomers.

π	Λ	$\lvert\mathcal{C}_\pi\rvert$	$\underline{n},\ \overline{n}$	$\underline{a},\ \overline{a}$	$\lvert\Gamma\rvert$	$\lvert\mathcal{F}\rvert$	K_1
BHL	Λ_7	514	$5, 36$	$0.03, 732.99$	26	101	166
Bp	Λ_2	370	$4, 67$	$-11.7, 470.0$	22	130	184
Bp	Λ_7	444	$4, 67$	$-11.7, 470.0$	26	163	230
Cp	Λ_5	131	$4, 63$	$4.7\times10^{-6}, 5.52$	8	79	119
Ct	Λ_2	125	$4, 63$	$56.1, 3607.5$	8	76	113
Ct	Λ_5	132	$4, 63$	$56.1, 3607.5$	8	81	121
Dc	Λ_2	141	$5, 44$	$0.5, 17.11$	20	62	111
Dc	Λ_7	161	$5, 44$	$0.5, 17.11$	25	69	130
ET	Λ_7	17	$5, 12$	$64.34, 96.21$	5	17	53
Fp	Λ_2	368	$4, 67$	$-82.99, 300.0$	20	131	183
Fp	Λ_7	424	$4, 67$	$-82.99, 300.0$	25	161	229
FlmLO	Λ_{16}	1046	$1, 49$	$0.185, 4.3$	34	282	376
Hv	Λ_2	94	$4, 16$	$19.12, 210.3$	12	63	105
Kov	Λ_1	52	$11, 16$	$1422.0, 1919.0$	9	33	64
Kow	Λ_2	684	$4, 58$	$-7.5, 15.6$	25	166	223
Kow	Λ_8	899	$4, 69$	$-7.5, 15.6$	37	219	303
Lp	Λ_2	615	$6, 60$	$-3.62, 6.84$	32	116	186
Lp	Λ_8	936	$6, 74$	$-3.62, 6.84$	44	136	231
Mp	Λ_2	467	$4, 122$	$-185.33, 300.0$	23	142	197
Mp	Λ_8	577	$4, 122$	$-185.33, 300.0$	32	176	255
OptR	Λ_2	147	$5, 44$	$-117.0, 165.0$	21	55	107
OptR	Λ_4	157	$5, 69$	$-117.0, 165.0$	25	62	123
RfIdT	Λ_{10}	191	$4, 26$	$0.919, 1.613$	17	115	168
Sl	Λ_2	673	$4, 55$	$-9.332, 1.11$	27	154	217
Sl	Λ_8	915	$4, 55$	$-11.6, 1.11$	42	207	300
SfT	Λ_3	247	$5, 33$	$12.3, 45.1$	11	91	128
Vis	Λ_3	282	$5, 36$	$-0.64, 1.63$	12	88	126
Homo	Λ_9	977	$6, 9$	$-0.3335,\ -0.1583$	59	190	297
Lumo	Λ_9	977	$6, 9$	$-0.1144, 0.1026$	59	190	297
Gap	Λ_9	977	$6, 9$	$0.1324, 0.4117$	59	190	297
Alpha	Λ_9	977	$6, 9$	$50.9, 99.6$	59	190	297
Cv	Λ_9	977	$6, 9$	$19.2, 44.0$	59	190	297
mu	Λ_9	977	$6, 9$	$0.04, 6.897$	59	190	297

Table 8.2. Results of setting data sets for polymers.

π	Λ	$\lvert\mathcal{C}_\pi\rvert$	$\underline{n},\ \overline{n}$	$\underline{a},\ \overline{a}$	$\lvert\Gamma\rvert$	$\lvert\mathcal{F}\rvert$	K_1
AMD	Λ_2	86	$4, 45$	$0.838, 1.34$	16	25	83
AMD	Λ_{13}	93	$4, 45$	$0.838, 1.45$	18	30	94
CHAR	Λ_2	30	$4, 18$	$3.7, 15.9$	15	17	68
CHAR	Λ_{12}	32	$4, 18$	$3.7, 15.9$	15	18	71
CHAR	Λ_6	35	$4, 18$	$3.7, 15.9$	18	21	83
DEIC	Λ_{12}	36	$4, 22$	$2.13, 3.4$	11	18	67
DISF	Λ_{13}	132	$4, 45$	$7\times10^{-5}, 0.07$	15	18	78
PRM	Λ_2	112	$4, 45$	$2.23, 4.91$	14	15	69
PRM	Λ_{13}	132	$4, 45$	$2.23, 4.91$	15	18	78
RFIDP	Λ_{11}	92	$4, 29$	$0.4899, 1.683$	15	35	96
RFIDP	Λ_{14}	125	$4, 29$	$0.4899, 1.683$	19	50	124
RFIDP	Λ_{15}	135	$4, 29$	$0.4899, 1.71$	23	56	144
TG	Λ_2	204	$4, 58$	$171, 673$	19	36	101
TG	Λ_7	232	$4, 58$	$171, 673$	21	43	118

elements. To construct the data set $\mathcal{C}_\pi$, we eliminated chemical compounds that do not satisfy one of the following: the graph is connected, the number of carbon atoms is at least four, and the number of non-hydrogen neighbors of each atom is at most 4.

We set a branch-parameter ρ to be 2, introduce linear descriptors defined by the two-layered graph in the chemical model without suppressing hydrogen and use the set $D_\pi^{(1)} \cup D_\pi^{(2)}$ of linear and quadratic descriptors (see Section 8.2 for the details). We normalize the range of each linear descriptor and the range $\{t \in \mathbb{R} \mid \underline{a} \leq t \leq \overline{a}\}$ of property values $a(\mathbb{C}), \mathbb{C} \in \mathcal{C}_\pi$.

We compare the following four methods of constructing a prediction function.

(i) **LLR**: use Lasso linear regression on the set the $D_\pi^{(1)}$ of linear descriptors (see [66] for the detail of the implementation);

(ii) **ANN**: use ANN on the set the $D_\pi^{(1)}$ of linear descriptors (see [66] for the detail of the implementation);

(iii) **ALR**: use adjustive linear regression on the set the $D_\pi^{(1)}$ of linear descriptors (see Chapter 7 for the detail of the implementation); and

(iv) **R-MLR**: apply our method (see Section 8.2.1) of reducing descriptors to

the set $D_\pi^{(1)} \cup D_\pi^{(2)}$ of linear and quadratic descriptors and use multi-linear regression for the resulting set of descriptors.

Among the above properties, we found that the median of test coefficient of determination R^2 of the prediction function constructed by LLR [66] or ALR [67] exceeds 0.98 for the following nine properties of monomers (resp., three properties of polymers): EDPA, Hc, Ha, Hf, LhcL, LhcS, Mr, Vd and U0 (resp., HcL, HcS and MLV). We excluded the above properties in the following experiment, and used the rest of 23 chemical properties of monomers and seven chemical properties of polymers to compare the four methods (i)-(iv).

Tables 8.1 and 8.2 show the size and range of data sets that we prepared for each chemical property to construct a prediction function, where we denote the following:

- π: the name of a chemical property used in the experiment.
- Λ: a set of selected elements used in the data set $\mathcal{C}_\pi$; Λ is one of the following 19 sets:
  $\Lambda_1 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}\}$; $\Lambda_2 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}\}$; $\Lambda_3 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{Si}_{(4)}\}$; $\Lambda_4 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{F}\}$;
  $\Lambda_5 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{Cl}, \mathtt{Pb}\}$; $\Lambda_6 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{Si}_{(4)}, \mathtt{Cl}, \mathtt{Br}\}$; $\Lambda_7 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$;
  $\Lambda_8 = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{S}_{(2)}, \mathtt{S}_{(4)}, \mathtt{S}_{(6)}, \mathtt{Cl}\}$; $\Lambda_9 = \{\mathtt{H}, \mathtt{C}_{(2)}, \mathtt{C}_{(3)}, \mathtt{C}_{(4)}, \mathtt{C}_{(5)}, \mathtt{O}, \mathtt{N}_{(1)}, \mathtt{N}_{(2)}, \mathtt{N}_{(3)}, \mathtt{F}\}$;
  $\Lambda_{10} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{P}_{(2)}, \mathtt{P}_{(5)}, \mathtt{Cl}\}$; $\Lambda_{11} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}_{(1)}, \mathtt{O}_{(2)}, \mathtt{N}\}$; $\Lambda_{12} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{Cl}\}$;
  $\Lambda_{13} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}, \mathtt{N}, \mathtt{Cl}, \mathtt{S}_{(2)}\}$; $\Lambda_{14} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}_{(1)}, \mathtt{O}_{(2)}, \mathtt{N}, \mathtt{Cl}, \mathtt{Si}_{(4)}, \mathtt{F}\}$;
  $\Lambda_{15} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}_{(1)}, \mathtt{O}_{(2)}, \mathtt{N}, \mathtt{Si}_{(4)}, \mathtt{Cl}, \mathtt{F}, \mathtt{S}_{(2)}, \mathtt{S}_{(6)}, \mathtt{Br}\}$;
  $\Lambda_{16} = \{\mathtt{H}, \mathtt{C}, \mathtt{O}_{(2)}, \mathtt{N}, \mathtt{Cl}, \mathtt{P}_{(3)}, \mathtt{P}_{(5)}, \mathtt{S}_{(2)}, \mathtt{S}_{(4)}, \mathtt{S}_{(6)}, \mathtt{Si}_{(4)}, \mathtt{Br}, \mathtt{I}\}$, where $\mathtt{a}_{(i)}$ for a chemical element $\mathtt{a}$ and an integer $i \geq 1$ means that a chemical element $\mathtt{a}$ with valence i.
- $|\mathcal{C}_\pi|$: the size of data set $\mathcal{C}_\pi$ over Λ for the property π.
- $\underline{n}$, $\overline{n}$: the minimum and maximum values of the number $n(\mathbb{C})$ of non-hydrogen atoms in the compounds $\mathbb{C}$ in $\mathcal{C}_\pi$.
- $\underline{a}$, $\overline{a}$: the minimum and maximum values of $a(\mathbb{C})$ for π over the compounds $\mathbb{C}$ in $\mathcal{C}_\pi$.
- $|\Gamma|$: the number of different edge-configurations of interior-edges over the compounds in $\mathcal{C}_\pi$.
- $|\mathcal{F}|$: the number of non-isomorphic chemical rooted trees in the set of all 2-fringe-trees in the compounds in $\mathcal{C}_\pi$.
- K_1: the size $|D_\pi^{(1)}|$ of a set $D_\pi^{(1)}$ of linear descriptors, where $|D_\pi^{(2)}| = (3(K_1)^2 + K_1)/2$ holds.

Constructing prediction functions For each chemical property π, we construct a prediction function by one of the four methods (i)-(iv).

For methods (i)-(iii), we used the same implementation due to Zhu et al. [66,

67] and omit the details.

In method (iv), we use our new procedures LLR-Reduce and Select-Des-set for reducing the number of descriptors (see Appendix 8.2.1 for the details). Method (iv) for property π is implemented as follows. If π is a monomer property and $|D_\pi^{(1)} \cup D_\pi^{(2)}| > 5000$ then first execute LLR-Reduce($\mathcal{C}_\pi, D_\pi^{(1)} \cup D_\pi^{(2)}$) to find a subset $\widetilde{D}$ of $D_\pi^{(1)} \cup D_\pi^{(2)}$ with 5000 descriptors. Otherwise set $\widetilde{D} := D_\pi^{(1)} \cup D_\pi^{(2)}$. Next execute Select-Des-set($\mathcal{C}_\pi, \widetilde{D}$) to obtain a subset D^* of $\widetilde{D}$. Construct a prediction function by MLR on the selected descriptor set D^*.

Tables 8.3 and 8.4 show the results on constructing prediction functions, where we denote the following:

- π: the name of a chemical property used in the experiment.
- Λ: the set of elements selected from the data set $\mathcal{C}_\pi$.
- LLR: the median of test R^2 in ten 5-fold cross-validations for prediction functions constructed by method (i).
- ANN: the median of test R^2 in ten 5-fold cross-validations for prediction functions constructed by method (ii).
- ALR: the median of test R^2 in ten 5-fold cross-validations for prediction functions constructed by method (iii).
- R-MLR: the median of test R^2 in ten 5-fold cross-validations for prediction functions constructed by method (iv).
- the score of LLR, ANN, ALR or R-MLR marked with "*" indicates the best performance among the four methods for the property π;
- K_1^*, K_2^*: the numbers K_1^* and K_2^* of linear and quadratic descriptors, respectively in the set D^* selected by our method (iv) from the set $D_\pi^{(1)} \cup D_\pi^{(2)}$ before a prediction function is constructed by MLR in (iv).

The running time of choosing a descriptor set D^* in method (iv) was around from 80 to 4×10^4 seconds and the time for constructing a prediction function to D^* is around 0.03 to 0.46 second.

There are 47 instances for constructing prediction functions in Tables 8.3 and 8.4. From these tables, we observe that method (iv) using quadratic descriptors performs better than methods (i)-(iii) with linear descriptors only in 43 out of the 47 instances. The averages of the median test R^2 of the method (i)-(iv) over the 47 instances are 0.634, 0.733, 0.764 and 0.913, respectively. In particular, method (iv) considerably improved the performance for $\pi \in \{$BP, CP, DC, ET, FP, HV, KOV, LP, RFIDT, GAP, CHAR, DISF, PRM, RFIDP$\}$. We also see that most descriptors in the resulting descriptor set D^* in R-MLR are quadratic.

Table 8.3. Results of constructing prediction functions for monomers.

π	Λ	LLR	ANN	ALR	R-MLR	K_1^*, K_2^*
BHL	Λ_7	0.483	0.622	0.265	*0.659	$0, 27$
Bp	Λ_2	0.599	0.765	0.816	*0.935	$1, 59$
Bp	Λ_7	0.663	0.720	0.832	*0.899	$0, 38$
Cp	Λ_5	0.555	0.727	0.690	*0.841	$0, 67$
Ct	Λ_2	0.037	0.357	0.900	*0.937	$1, 47$
Ct	Λ_5	0.048	0.357	*0.895	0.860	$0, 13$
Dc	Λ_2	0.489	0.651	0.488	*0.908	$0, 58$
Dc	Λ_7	0.574	0.622	0.602	*0.829	$0, 26$
ET	Λ_7	0.132	0.479	0.464	*0.996	$0, 13$
Fp	Λ_2	0.589	0.746	0.719	*0.899	$0, 42$
Fp	Λ_7	0.571	0.745	0.684	*0.846	$0, 32$
FlmLO	Λ_{16}	0.819	0.928	0.604	*0.949	$0, 77$
Hv	Λ_2	0.864	0.778	0.816	*0.970	$0, 22$
Kov	Λ_1	0.677	0.727	0.838	*0.953	$2, 19$
Kow	Λ_2	0.953	0.952	0.964	*0.967	$0, 55$
Kow	Λ_8	0.927	0.937	*0.952	0.950	$0, 64$
Lp	Λ_2	0.856	0.867	0.844	*0.928	$0, 89$
Lp	Λ_8	0.840	0.859	0.807	*0.914	$0, 109$
Mp	Λ_2	0.810	0.800	0.831	*0.873	$0, 51$
Mp	Λ_8	0.810	0.820	0.807	*0.898	$0, 58$
OptR	Λ_2	0.825	0.918	0.876	*0.970	$0, 85$
OptR	Λ_4	0.825	0.878	0.870	*0.970	$0, 69$
RfIdT	Λ_{10}	0.000	0.453	0.425	*0.775	$0, 43$
Sl	Λ_2	0.808	0.848	0.784	*0.894	$0, 82$
Sl	Λ_8	0.808	0.861	0.828	*0.897	$0, 74$
SfT	Λ_3	0.927	0.859	0.847	*0.941	$0, 36$
Vis	Λ_3	0.893	0.929	0.911	*0.973	$0, 43$
Homo	Λ_9	*0.841	0.689	0.689	0.804	$0, 87$
Lumo	Λ_9	0.841	0.860	0.833	*0.920	$0, 102$
Gap	Λ_9	0.784	0.795	0.755	*0.876	$0, 83$
Alpha	Λ_9	0.961	0.888	0.953	*0.980	$0, 104$
Cv	Λ_9	0.970	0.911	0.966	*0.978	$0, 83$
mu	Λ_9	0.367	0.409	0.403	*0.645	$0, 112$

Table 8.4. Results of constructing prediction functions for polymers.

π	Λ	LLR	ANN	ALR	R-MLR	K_1^*, K_2^*
AMD	Λ_2	0.914	0.885	*0.933	0.906	$0, 5$
AMD	Λ_{13}	0.918	0.824	0.917	*0.953	$0, 6$
CHAR	Λ_2	0.210	0.642	0.863	*0.938	$0, 10$
CHAR	Λ_{12}	0.088	0.640	0.835	*0.924	$0, 9$
CHAR	Λ_6	-0.073	0.527	0.766	*0.950	$0, 12$
DEIC	Λ_{12}	0.761	0.641	0.918	*0.956	$3, 41$
DISF	Λ_{13}	0.623	0.801	0.308	*0.906	$1, 23$
PRM	Λ_2	0.801	0.801	0.505	*0.967	$0, 26$
PRM	Λ_{13}	0.784	0.735	0.489	*0.977	$0, 34$
RFIDP	Λ_{11}	0.104	0.423	0.853	*0.962	$2, 52$
RFIDP	Λ_{14}	0.373	0.560	0.848	*0.953	$2, 43$
RFIDP	Λ_{15}	0.346	0.492	0.883	*0.947	$5, 53$
TG	Λ_2	0.902	0.883	0.923	*0.958	$1, 33$
TG	Λ_7	0.894	0.860	0.927	*0.957	$0, 32$

8.4.2 Results on the Second Phase of the Framework

To execute the second phase, we used a set of seven instances $I_{\rm a}$, $I_{\rm b}^i, i \in [1, 4]$, $I_{\rm c}$ and $I_{\rm d}$ based on the seed graphs prepared by Zhu et al. [66].

Solving an MILP for the inverse problem We executed the stage of solving an MILP to infer a chemical graph for two properties $\pi \in \{\text{BP}, \text{DC}\}$.

For the MILP formulation $\mathcal{M}_{f,\eta,\sigma}$, we use the prediction function η for each $\pi \in \{\text{BP}, \text{DC}\}$ by method (iv), R-MLR that attained the median test R^2 in Table 8.3. To solve an MILP with the formulation, we used `CPLEX` version 12.10. Tables 8.5 and 8.6 show the computational results of the experiment in this stage for the two properties, where we denote the following:

- $n_{\rm LB}$: a lower bound on the number of non-hydrogen atoms in a chemical graph $\mathbb{C}$ to be inferred;
- $\underline{y}^*$, $\overline{y}^*$: lower and upper bounds $\underline{y}^*, \overline{y}^* \in \mathbb{R}$ on the value $a(\mathbb{C})$ of a chemical graph $\mathbb{C}$ to be inferred;
- #v (resp., #c): the number of variables (resp., constraints) in the MILP;
- I-time: the time (sec.) to solve the MILP;
- n: the number $n(\mathbb{C}^\dagger)$ of non-hydrogen atoms in the chemical graph $\mathbb{C}^\dagger$ inferred by solving the MILP;

Table 8.5. Results of inferring a chemical graph $\mathbb{C}^\dagger$ and generating recombination solutions for Bp with Λ_7.

inst.	n_{LB}	$\underline{y}^*,\ \overline{y}^*$	#v	#c	I-time	n	$\mathrm{n}^{\mathrm{int}}$	η	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	30	$225, 235$	10502	10240	4.29	49	26	233.92	0.072	3	3
I_{b}^1	35	$285, 295$	10507	7793	2.27	35	10	286.52	0.034	6	6
I_{b}^2	45	$365, 375$	13000	10913	11.9	49	25	370.70	0.14	3202	100
I_{b}^3	45	$305, 315$	12788	10920	7.07	48	25	309.39	0.22	6304	100
I_{b}^4	45	$260, 270$	12576	10928	10.7	49	27	266.26	0.17	376	100
I_{c}	50	$340, 350$	7515	8270	0.867	50	33	344.98	0.019	2	2
I_{d}	40	$320, 330$	6135	7773	8.22	45	23	329.85	8.3	6733440	100

Table 8.6. Results of inferring a chemical graph $\mathbb{C}^\dagger$ and generating recombination solutions for Dc with Λ_7.

inst.	n_{LB}	$\underline{y}^*,\ \overline{y}^*$	#v	#c	I-time	n	$\mathrm{n}^{\mathrm{int}}$	η	D-time	$\mathbb{C}$-LB	#$\mathbb{C}$
I_{a}	30	$0.55, 0.60$	10194	9787	3.91	41	25	0.558	0.069	2	2
I_{b}^1	35	$1.10, 1.15$	10415	7368	4.73	35	11	1.104	0.10	16	16
I_{b}^2	45	$6.00, 6.05$	12976	10481	57.4	45	25	6.04	0.12	2040	100
I_{b}^3	45	$1.45, 1.50$	2767	10488	39.7	49	26	1.488	0.28	21600	100
I_{b}^4	45	$6.10, 6.15$	12558	10494	26.4	46	25	6.10	0.027	2	2
I_{c}	50	$12.35, 12.40$	7207	7819	1.75	50	34	12.38	0.020	2	2
I_{d}	40	$3.15, 3.20$	5827	7325	14.9	41	23	3.199	0.079	18952	100

- $\mathrm{n}^{\mathrm{int}}$: the number $\mathrm{n}^{\mathrm{int}}(\mathbb{C}^\dagger)$ of interior-vertices in the chemical graph $\mathbb{C}^\dagger$; and
- η: the predicted property value $\eta(f(\mathbb{C}^\dagger))$ of the chemical graph $\mathbb{C}^\dagger$.

Figure 8.1(a) illustrates the chemical graph $\mathbb{C}^\dagger$ inferred from I_{c} with $(\underline{y}^*, \overline{y}^*) = (340, 350)$ of Bp in Table 8.5.

Figure 8.1(b) (resp., Figure 8.1(c)) illustrates the chemical graph $\mathbb{C}^\dagger$ inferred from I_{a} with $(\underline{y}^*, \overline{y}^*) = (0.55, 0.60)$ (resp., I_{d} with $(\underline{y}^*, \overline{y}^*) = (3.15, 3.20)$) of Dc in Table 8.6.

In this experiment, we prepared several different types of instances: instances I_{a} and I_{c} have restricted seed graphs, the other instances have abstract seed graphs and instances I_{c} and I_{d} have restricted set of fringe-trees. From Tables 8.5 and 8.6, we observe that an instance with a large number of variables and constraints takes more running time than those with a smaller size in general. All instances in this experiment are solved in a few seconds to around 60 seconds with our MILP formulation.

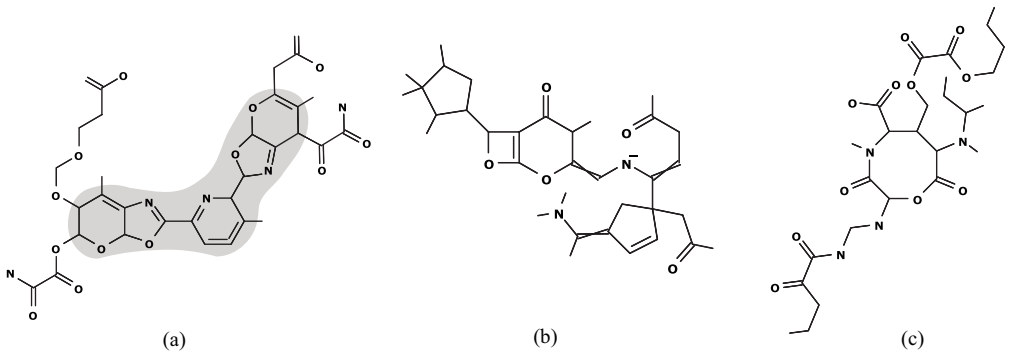


Figure 8.1. (a) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 344.98$ inferred from $I_{\rm c}$ with $(\underline{y}^*, \overline{y}^*) = (340, 350)$ of BP; (b) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 0.558$ inferred from $I_{\rm a}$ with $(\underline{y}^*, \overline{y}^*) = (0.55, 0.60)$ of DC; and (c) $\mathbb{C}^\dagger$ with $\eta(f(\mathbb{C}^\dagger)) = 3.199$ inferred from $I_{\rm d}$ with $(\underline{y}^*, \overline{y}^*) = (3.15, 3.20)$ of DC.

Generating recombination solutions. Let $\mathbb{C}^\dagger$ be a chemical graph obtained by solving the MILP $\mathcal{M}_{f,\eta,\sigma}$ for the inverse problem. We here execute a stage of generating recombination solutions $\mathbb{C}^* \in \mathcal{G}_\sigma$ of $\mathbb{C}^\dagger$ such that $f(\mathbb{C}^*) = x^* = f(\mathbb{C}^\dagger)$.

We execute an algorithm for generating chemical isomers of $\mathbb{C}^\dagger$ up to 100 when the number of all chemical isomers exceeds 100. For this, we use a dynamic programming algorithm [66]. The algorithm first decomposes $\mathbb{C}^\dagger$ into a set of acyclic chemical graphs, next replaces each acyclic chemical graph T with another acyclic chemical graph T' that admits the same feature vector as that of T and finally assembles the resulting acyclic chemical graphs into a chemical isomer $\mathbb{C}^*$ of $\mathbb{C}^\dagger$. The algorithm can compute a lower bound on the total number of all chemical isomers $\mathbb{C}^\dagger$ without generating all of them.

Tables 8.5 and 8.6 show the computational results of the experiment in this stage for the two properties $\pi \in \{$BP, DC$\}$, where we denote the following:

- D-time: the running time (sec.) to execute the dynamic programming algorithm to compute a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ and generate all (or up to 100) chemical isomers $\mathbb{C}^*$;
- $\mathbb{C}$-LB: a lower bound on the number of all chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$; and
- $\#\mathbb{C}$: the number of all (or up to 100) chemical isomers $\mathbb{C}^*$ of $\mathbb{C}^\dagger$ generated in this stage.

From Tables 8.5 and 8.6, we observe the running time and the number of generated recombination solutions in this stage.

The chemical graph $\mathbb{C}^\dagger$ in $I_{\rm b}^2$, $I_{\rm b}^3$ and $I_{\rm d}$ admits a large number of chemical isomers $\mathbb{C}^*$ in some cases, where a lower bound $\mathbb{C}$-LB on the number of chemical isomers is derived without generating all of them. For the other instances, the

running time for generating up to 100 target chemical graphs in this stage is less than 0.03 second. For some chemical graph $\mathbb{C}^\dagger$, the number of chemical isomers found by our algorithm was small. This is because some of acyclic chemical graphs in the decomposition of $\mathbb{C}^\dagger$ has no alternative acyclic chemical graph other than the original one.

Generating neighbor solutions Let $\mathbb{C}^\dagger$ be a chemical graph obtained by solving the MILP $\mathcal{M}_{f,\eta,\sigma}$ for the inverse problem. We executed a stage of generating neighbor solutions of $\mathbb{C}^\dagger$.

We select an MILP for the inverse problem with a prediction function η such that a solution $\mathbb{C}^\dagger$ of the MILP admits only two isomers $\mathbb{C}^*$ in the stage of generating recombination solutions; i.e., instance $I_{\rm c}$ for property BP with Λ_7 and instances $I_{\rm a}$, $I_{\rm b}^4$ and $I_{\rm c}$ for property DC with Λ_7.

In this experiment, we add to the MILP $\mathcal{M}_{f,\eta,\sigma}$ an additional set Θ of two linear constraints on linear and quadratic descriptors as follows. For the two constraints, we use the prediction functions η_π constructed by R-MLR for properties $\pi \in \{\text{LP, SL}\}$ with Λ_8 in Table 8.3.

Let D_π^* denote the set of descriptors selected in the construction of prediction function for properties $\pi \in \{\text{BP,DC}\}$ with Λ_7 and $\pi \in \{\text{LP, SL}\}$ with Λ_8 in Table 8.3 and let $D_\pi^{\rm union}, \pi \in \{\text{BP, DC}\}$ denote the union $D_\pi^* \cup D_{\rm LP}^* \cup D_{\rm SL}^*$. We regard each of $\eta_{\rm LP}$ and $\eta_{\rm SL}$ as a function from $\mathbb{R}^{|D_\pi^{\rm union}|}$ to $\mathbb{R}$ for $\pi \in \{\text{BP, DC}\}$. We set $p_{\rm dim} := 2$ and let Θ consist of two linear constraints $\theta_1 := \eta_{\rm LP}$ and $\theta_2 := \eta_{\rm SL}$. We set $\delta := 0.1$ or 0.05 which defines a two-dimensional grid space where $\mathbb{C}^\dagger$ is mapped to the origin (see [6] for the detail on the neighbors). We choose a set N_0 of 48 neighbors of the origin $\mathbb{C}^\dagger$ in the grid search space. For each instance, we check the feasibility of neighbors in N_0. in a non-decreasing order of the distance between the neighbor and the origin. For each feasible neighbor $z \in N_0$, output a feasible solution $\mathbb{C}_z^\dagger$ of the augmented MILP instance. We set a time limit for checking the feasibility of a neighbor to be 300 seconds, and we skip a neighbor when the corresponding MILP is not solved within the time limit. We also ignore any neighbor $z \in N_0$ without testing the feasibility of z if we find an infeasible neighbor $z' \in N_0$ such that z' is closer to the origin than z is.

Table 8.7 shows the computational results of the experiment for the three instances, where we denote the following:

- (inst., π): topological specification I and property π;
- n: the number of non-hydrogen atoms in the tested instance;
- δ: the size of a sub-region in the grid search space;
- #sol: the number of new chemical graphs obtained from the neighbor set N_0;
- #infs: the number of neighbors in N_0 that are found to be infeasible during

Table 8.7. Results of generating neighbor solutions of $\mathbb{C}^{\dagger}$.

(inst., π)	n	δ	#sol	#infs	#ign	#TO
(I_{c},BP)	50	0.1	5	1	3	39
(I_{a},DC)	30	0.1	40	1	0	7
(I_{b}^{4},DC)	45	0.1	2	0	0	46
(I_{c},DC)	40	0.05	0	0	0	48

the search procedure;

- #ign: the number of neighbors in N_0 that are ignored during the search procedure;
- #TO: the number of neighbors in N_0 such that the time for feasibility check exceeds the time limit of 300 seconds during the search procedure.

The branch-and-bound method for solving an MILP sometimes takes an extremely large execution time for the same size of instances. We introduce a time limit to bound an entire running time to skip such instances during an execution of testing the feasibility of neighbors in N_0. From Table 8.7, we observe that some number of neighbor solutions of the solution $\mathbb{C}^{\dagger}$ to the MILP $\mathcal{M}_{f,\eta,\sigma}$ could be generated for each of the four instances.

8.5 Concluding Remarks

In the framework of inferring chemical graphs, the descriptors of a prediction function were mainly defined to be the frequencies of local graph structures in the two-layered model and such definition was important to derive a compact MILP formulation for inferring a desired chemical graph. To improve the performance of prediction functions in the framework, this chapter introduced a multiple of two of these descriptors as a new descriptor and examined the effectiveness of the new set of descriptors. For this, we designed a method for reducing the size of a descriptor set not to lose the learning performance in constructing prediction functions and gave a compact formulation to compute a product of two values in an MILP. From the results of our computational experiments, we observe that a prediction function constructed by our new method performs considerably better than the previous prediction functions for many chemical properties. We also found that the modified MILP in the second phase of the framework still can infer a chemical graph with around 50 non-hydrogen atoms.

9 Conclusion

In this thesis, we mainly focused on applying the mixed integer linear programming (MILP)-based framework for the inverse QSAR problem. We formulated both the inverse problem of machine learning and the necessary constraints for constructing a valid chemical graph with a given feature vector and an abstract graph topology into an integrated MILP together, to guarantee the exactness and optimality of the inferred chemical compounds, which is not realized by most of the existing methods.

Following the works in [4, 11, 59, 60], we extended the existing framework and proposed MILP formulations for certain classes of chemical graphs (rank-2 chemical graphs in Chapter 4, and arbitrary cyclic chemical graphs in Chapter 5, respectively), which contained nearly 97% of the registered chemical compounds in PubChem [27]. Since in this approach, we use only graph-theoretical descriptors to define a feature vector for tractability in MILP, it also becomes possible to specify a prescribed topological substructure of the inferred chemical compound, which allows the possibility to include domain knowledge in the way of specifying graph structures, which can be of importance for drug discovery. We also employed a new mechanism to generate a large number of target chemical graphs with a DP-based graph enumeration algorithms. The experimental results showed that the proposed approach is available to infer a chemical graph with around 50 non-hydrogen atoms in a reasonably short time. Both the capable size of the inferred chemical compounds and the time efficiency drastically improved when compared to the existing methods.

Since the learning performance of the QSAR phase contributes much to the quality of the inferred chemical compounds, we then proposed several more machine learning methods whose inverse problems can be formulated into a MILP formulation (Lasso linear regression in Chapter 6, adjustive linear regression in Chapter 7, and multiple linear regression with quadratic descriptors in Chapter 8, respectively) and thus can be applied to this framework instead of ANN. From the results of our computational experiments, we observe that these methods are available to improve the learning performance of the QSAR phase a lot for most of the chemical property datasets we have while the capable size of the inferred chemical compounds and time efficiency is preserved.

It is left as an interesting future work for the following points:

- Find more graph-theoretical descriptors that can be employed in the framework. Using more descriptors usually means extracting more information from the chemical graph, and has the possibility to improve the learning performance in the QSAR phase. We note that the descriptors are necessary to be available to be expressed as variables and constraints in MILP formulations in our proposed methods, and also the graph enumeration algorithms may need to be modified. These make finding new descriptors a challenging task.

- Use other machine learning methods to construct a prediction function that can obtain a good learning performance and derive the corresponding MILP formulation in the framework. Graph neural networks (GNN) [28], a learning model that does not require designing a feature vector, but directly uses the graph as the input and can extract the intrinsic features from the graph structure itself, is a promising method for the framework. It has the potential to achieve higher learning performance and the inverse problem can be written down as an MILP formulation. Also, an ensemble method, such as random forests, is another kind of possible method for the framework, because the MILP formulations can be derived relatively easily.

- Find new ways of feature selection during the learning process. Since the time to solve an MILP formulation depends on its number of variables and constraints, it will be desirable to use less number of descriptors in the constructed prediction function. Just as the Lasso linear regression introduced in Chapter 6 has the ability to select a subset of descriptors, it will be interesting to design some heuristics to reduce the size of the descriptor set while keeping or even improving the learning performance for some learning methods. It will also be interesting to use some clustering methods to select some representative features and chemical compounds from a dataset to reduce the computation cost for the learning stage.

- Improve the MILP formulations, especially by reducing the size of variables and constraints. Just as the MILP formulations in Chapter 5 can be improved based on a characterization of a chemical acyclic graph [62], it is very possible that there is still space for our MILP formulations to improve, by introducing a more compact way for the constraints and removing unnecessary variables.

- Try to include 3D information on the chemical compounds. The model of

chemical graphs used in this work completely disregards the 3D information of the corresponding chemical compounds, such as the chirality and the cis-trans isomerism of the double bond. These pieces of information sometimes play an important role in drug design and biochemical properties. Modifying the model to include the 3D information of chemical compounds and the corresponding MILP formulations will be a challenging future task for this study.

- Finally, develop a completely new model to describe the chemical compound, to meet the more sophisticated need in practice.

We hope that the results obtained in this study will be useful in future research and development in computational chemo-informatics, bioinformatics, machine learning, and other related fields.

Bibliography

[1] T. Akutsu and H. Nagamochi. A mixed integer linear programming formulation to artificial neural networks. In *Proceedings of the 2nd International Conference on Information Science and Systems*, pages 215–220, 2019.

[2] T. Akutsu and H. Nagamochi. A novel method for inference of chemical compounds with prescribed topological substructures based on integer programming. *arXiv preprint arXiv:2010.09203*, 2020.

[3] T. Akutsu, D. Fukagawa, J. Jansson, and K. Sadakane. Inferring a graph from path frequency. *Discrete Applied Mathematics*, 160(10-11):1416–1428, 2012.

[4] N. A. Azam, R. Chiewvanichakorn, F. Zhang, A. Shurbevski, H. Nagamochi, and T. Akutsu. A novel method for the inverse QSAR/QSPR based on artificial neural networks and mixed integer linear programming with guaranteed admissibility. In *Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOINFORMATICS*, pages 101–108. INSTICC, SciTePress, 2020.

[5] N. A. Azam, J. Zhu, R. Ido, H. Nagamochi, and T. Akutsu. Experimental results of a dynamic programming algorithm for generating chemical isomers based on frequency vectors. In *Fourth International Workshop on Enumeration Problems and Applications (WEPA)*, pages 7–10, Israel, WEPA2020 15, December 2020. Israel.

[6] N. A. Azam, J. Zhu, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. Molecular design based on artificial neural networks, integer programming and grid neighbor search. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 360–363. IEEE, 2021.

[7] N. A. Azam, J. Zhu, Y. Sun, Y. Shi, A. Shurbevski, L. Zhao, H. Nagamochi, and T. Akutsu. A novel method for inference of acyclic chemical compounds with bounded branch-height based on artificial neural networks and integer programming. *Algorithms for Molecular Biology*, 16:1–39, 2021.

[8] J. Bicerano. *Prediction of polymer properties*. CRC Press, 2002.

[9] R. S. Bohacek, C. McMartin, and W. C. Guida. The art and practice of structure-based drug design: A molecular modeling perspective. *Medicinal Research Reviews*, 16(1):3–50, 1996.

[10] A. Cherkasov, E. N. Muratov, D. Fourches, A. Varnek, I. I. Baskin, M. Cronin, J. Dearden, P. Gramatica, Y. C. Martin, R. Todeschini, et al. QSAR modeling: where have you been? where are you going to? *Journal of Medicinal Chemistry*, 57(12):4977–5010, 2014.

[11] R. Chiewvanichakorn, C. Wang, Z. Zhang, A. Shurbevski, H. Nagamochi, and T. Akutsu. A method for the inverse QSAR/QSPR based on artificial neural networks and mixed integer linear programming. In *Proceedings of the 10th International Conference on Bioscience, Biochemistry and Bioinformatics*, pages 40–46, 2020.

[12] N. De Cao and T. Kipf. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[13] N. R. Draper and H. Smith. *Applied regression analysis*, volume 326. John Wiley & Sons, 1998.

[14] P. Duchowicz, E. A. Castro, and A. A. Toropov. Improved QSPR analysis of standard entropy of acyclic and aromatic compounds using optimized correlation weights of linear graph invariants. *Computers & Chemistry*, 26 (4):327–332, 2002.

[15] 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.

[16] F. Ghasemi, A. Mehridehnavi, A. Pérez-Garrido, and H. Pérez-Sánchez. Neural network and deep-learning algorithms used in QSAR studies: merits and drawbacks. *Drug Discovery Today*, 23(10):1784–1790, 2018.

[17] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4 (2):268–276, 2018.

[18] V. Goussard, F. Duprat, V. Gerbaud, J.-L. Ploix, G. Dreyfus, V. Nardello-Rataj, and J.-M. Aubry. Predicting the surface tension of liquids: Comparison of four modeling approaches and application to cosmetic oils. *Journal of Chemical Information and Modeling*, 57(12):2986–2995, 2017.

[19] V. Goussard, F. Duprat, J.-L. Ploix, G. Dreyfus, V. Nardello-Rataj, and J.-M. Aubry. A new machine-learning tool for fast estimation of liquid viscosity.

application to cosmetic oils. *Journal of Chemical Information and Modeling*, 60(4):2012–2023, 2020.

[20] R. Gugisch, A. Kerber, A. Kohnert, R. Laue, M. Meringer, C. Rücker, and A. Wassermann. MOLGEN 5.0, a molecular structure generator. In *Advances in Mathematical Chemistry and Applications*, pages 113–138. Elsevier, 2015.

[21] A. Hoerl and R. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[22] A. Hoerl and R. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, 12(1):69–82, 1970.

[23] R. Ido, S. Cao, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A method for inferring polymers based on linear regression and integer programming. *arXiv preprint arXiv:2109.02628*, 2021.

[24] H. Ikebata, K. Hongo, T. Isomura, R. Maezono, and R. Yoshida. Bayesian molecular design with a chemical language model. *Journal of Computer-aided Molecular Design*, 31:379–391, 2017.

[25] R. Ito, N. A. Azam, C. Wang, A. Shurbevski, H. Nagamochi, and T. Akutsu. A novel method for the inverse QSAR/QSPR to monocyclic chemical compounds based on artificial neural networks and integer programming. In *Advances in Computer Vision and Computational Biology: Proceedings from IPCV'20, HIMS'20, BIOCOMP'20, and BIOENG'20*, pages 641–655. Springer, 2021.

[26] M. Jalali-Heravi and M. H. Fatemi. Artificial neural network modeling of kovats retention indices for noncyclic and monocyclic terpenes. *Journal of Chromatography A*, 915(1-2):177–183, 2001.

[27] S. Kim, J. Chen, T. Cheng, A. Gindulyte, J. He, S. He, Q. Li, B. A. Shoemaker, P. A. Thiessen, B. Yu, et al. Pubchem 2023 update. *Nucleic Acids Research*, 51(D1):D1373–D1380, 2023.

[28] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] M. J. Kusner, B. Paige, and J. M. Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR, 2017.

[30] J. Li, H. Nagamochi, and T. Akutsu. Enumerating substituted benzene isomers of tree-like chemical graphs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15(2):633–646, 2016.

[31] Y.-C. Lo, S. E. Rensi, W. Torng, and R. B. Altman. Machine learning in chemoinformatics and drug discovery. *Drug Discovery Today*, 23(8):1538–1546, 2018.

[32] K. Madhawa, K. Ishiguro, K. Nakago, and M. Abe. GraphNVP: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

[33] T. Miyao, H. Kaneko, and K. Funatsu. Inverse QSPR/QSAR analysis for chemical structure generation (from y to x). *Journal of Chemical Information and Modeling*, 56(2):286–299, 2016.

[34] R. Naef. Calculation of the isobaric heat capacities of the liquid and solid phase of organic compounds at and around 298.15 k based on their "true" molecular volume. *Molecules*, 24(8):1626, 2019.

[35] T. I. Netzeva, A. P. Worth, T. Aldenberg, R. Benigni, M. T. Cronin, P. Gramatica, J. S. Jaworska, S. Kahn, G. Klopman, C. A. Marchant, et al. Current status of methods for defining the applicability domain of (quantitative) structure-activity relationships: The report and recommendations of ecvam workshop 52. *Alternatives to Laboratory Animals*, 33(2):155–173, 2005.

[36] Y. M. Ponce. Total and local quadratic indices of the molecular pseudograph's atom adjacency matrix: applications to the prediction of physical properties of organic compounds. *Molecules*, 8(9):687–726, 2003.

[37] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1 (1):1–7, 2014.

[38] J.-L. Reymond. The chemical space project. *Accounts of Chemical Research*, 48(3):722–730, 2015.

[39] K. Roy and A. Saha. Comparative QSPR studies with molecular connectivity, molecular negentropy and tau indices: Part i: Molecular thermochemical properties of diverse functional acyclic compounds. *Journal of Molecular Modeling*, 9:259–270, 2003.

[40] C. Rupakheti, A. Virshup, W. Yang, and D. N. Beratan. Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of Chemical Information and Modeling*, 55(3):529–537, 2015.

[41] M. H. Segler, T. Kogej, C. Tyrchan, and M. P. Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131, 2018.

[42] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang. GraphAF: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*, 2020.

[43] Y. Shi, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on a two-layered model and integer programming. *International Journal of Molecular Sciences*, 22(6): 2847, 2021.

[44] M. I. Skvortsova, I. I. Baskin, O. L. Slovokhotova, V. A. Palyulin, and N. S. Zefirov. Inverse problem in QSAR/QSPR studies for the case of topological indexes characterizing molecular shape (kier indices). *Journal of Chemical Information and Computer Sciences*, 33(4):630–634, 1993.

[45] M. Sumita, X. Yang, S. Ishihara, R. Tamura, and K. Tsuda. Hunting for organic molecules with artificial intelligence: molecules optimized for desired excitation energies. *ACS Central Science*, 4(9):1126–1133, 2018.

[46] M. Suzuki, H. Nagamochi, and T. Akutsu. Efficient enumeration of monocyclic chemical graphs with given path frequencies. *Journal of Cheminformatics*, 6(1):1–18, 2014.

[47] S. Takeda, T. Hama, H.-H. Hsu, V. A. Piunova, D. Zubarev, D. P. Sanders, J. W. Pitera, M. Kogoh, T. Hongo, Y. Cheng, et al. Molecular inverse-design platform for material industries. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2961–2969, 2020.

[48] Y. Tamura, Y. Nishiyama, C. Wang, Y. Sun, A. Shurbevski, H. Nagamochi, and T. Akutsu. Enumerating chemical graphs with mono-block 2-augmented tree structure from given upper and lower bounds on path frequencies. *arXiv preprint arXiv:2004.06367*, 2020.

[49] K. Tanaka, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on decision tree and integer

programming. In *Intelligent Computing Theories and Application: 17th International Conference, ICIC 2021, Shenzhen, China, August 12–15, 2021, Proceedings, Part II*, pages 628–644. Springer, 2021.

[50] I. V. Tetko and O. Engkvist. From big data to artificial intelligence: chemoinformatics meets new challenges. *Journal of Cheminformatics*, 12:1–3, 2020.

[51] Y. Tezuka and H. Oike. Topological polymer chemistry. *Progress in Polymer Science*, 27(6):1069–1122, 2002.

[52] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[53] J.-B. Wang, D.-S. Cao, M.-F. Zhu, Y.-H. Yun, N. Xiao, and Y.-Z. Liang. In silico evaluation of logd7. 4 and comparison with other prediction methods. *Journal of Chemometrics*, 29(7):389–398, 2015.

[54] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

[55] J. Xiong, Z. Xiong, K. Chen, H. Jiang, and M. Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

[56] K. Yamashita, R. Masui, X. Zhou, C. Wang, A. Shurbevski, H. Nagamochi, and T. Akutsu. Enumerating chemical graphs with two disjoint cycles satisfying given path frequency specifications. *arXiv preprint arXiv:2004.08381*, 2020.

[57] X. Yang, J. Zhang, K. Yoshizoe, K. Terayama, and K. Tsuda. ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976, 2017.

[58] S. Yuan, Z. Jiao, N. Quddus, J. S.-I. Kwon, and C. V. Mashuga. Developing quantitative structure–property relationship models to predict the upper flammability limit using machine learning. *Industrial & Engineering Chemistry Research*, 58(8):3531–3537, 2019.

[59] F. Zhang, J. Zhu, R. Chiewvanichakorn, A. Shurbevski, H. Nagamochi, and T. Akutsu. A new integer linear programming formulation to the inverse QSAR/QSPR for acyclic chemical compounds using skeleton trees. In

Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices: 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2020, Kitakyushu, Japan, September 22-25, 2020, Proceedings, pages 433–444. Springer, 2020.

[60] F. Zhang, J. Zhu, R. Chiewvanichakorn, A. Shurbevski, H. Nagamochi, and T. Akutsu. A new approach to the design of acyclic chemical compounds using skeleton trees and integer linear programming. *Applied Intelligence*, 52(15):17058–17072, 2022.

[61] J. Zhu, C. Wang, A. Shurbevski, H. Nagamochi, and T. Akutsu. A novel method for inference of chemical compounds of cycle index two with desired properties based on artificial neural networks and integer programming. *Algorithms*, 13(5):124, 2020.

[62] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An improved integer programming formulation for inferring chemical compounds with prescribed topological structures. In *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part I 34*, pages 197–209. Springer, 2021.

[63] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on linear regression and integer programming. *arXiv preprint arXiv:2107.02381*, 2021.

[64] J. Zhu, K. Haraguchi, H. Nagamochi, and T. Akutsu. Adjustive linear regression and its application to the inverse qsar. Department of Applied Mathematics and Physics, Kyoto University, Technical Report, TR:2021-002 http://www.amp.i.kyoto-u.ac.jp/tecrep, 2021.

[65] J. Zhu, N. A. Azam, S. Cao, R. Ido, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. Molecular design based on integer programming and quadratic descriptors in a two-layered model. *arXiv preprint arXiv:2209.13527*, 2022.

[66] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A method for molecular design based on linear regression and integer programming. In *Proceedings of the 12th International Conference on Bioscience, Biochemistry and Bioinformatics*, pages 21–28, 2022.

[67] J. Zhu, K. Haraguchi, H. Nagamochi, and T. Akutsu. Adjustive linear regression and its application to the inverse QSAR. In *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOINFORMATICS*, pages 144–151. INSTICC, SciTePress, 2022.

[68] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix A Appendix for Chapter 4

A.1 All Constraints in an MILP Formulation for Rank-2 Chemical Graphs

To formulate an MILP that represents a chemical graph $G = (H, \alpha, \beta)$, we distinguish a tuple $(\mathtt{a}, \mathtt{b}, k)$ from a tuple $(\mathtt{b}, \mathtt{a}, k)$. For a tuple $\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Lambda \times \Lambda \times \{1, 2, 3\}$, let $\overline{\gamma}$ denote the tuple $(\mathtt{b}, \mathtt{a}, k)$. Let $\Gamma_< \triangleq \{\overline{\gamma} \mid \gamma \in \Gamma_>\}$. We call a tuple $\gamma = (\mathtt{a}, \mathtt{b}, k) \in \Lambda \times \Lambda \times \{1, 2, 3\}$ *proper* if

$$k \leq \min\{\mathrm{val}(\mathtt{a}), \mathrm{val}(\mathtt{b})\} \text{ and } k \leq \max\{\mathrm{val}(\mathtt{a}), \mathrm{val}(\mathtt{b})\} - 1,$$

where the latter is assumed because otherwise G must consist of two atoms of $\mathtt{a} = \mathtt{b}$. Assume that each tuple $\gamma \in \Gamma$ is proper. Let ϵ be a fictitious chemical element that represents null, call a tuple $(\mathtt{a}, \mathtt{b}, 0)$ with $\mathtt{a}, \mathtt{b} \in \Lambda \cup \{\epsilon\}$ *fictitious*, and define Γ_0 to be the set of all fictitious tuples; i.e., $\Gamma_0 = \{(\mathtt{a}, \mathtt{b}, 0) \mid \mathtt{a}, \mathtt{b} \in \Lambda \cup \{\epsilon\}\}$. To represent chemical elements $\mathtt{e} \in \Lambda \cup \{\epsilon\} \cup \Gamma$ in an MILP, we encode these elements $\mathtt{e}$ into some integers denoted by $[\mathtt{e}]$. Assume that, for each element $\mathtt{a} \in \Lambda$, $[\mathtt{a}]$ is a positive integer and that $[\epsilon] = 0$.

A.1.1 Applicability Domain

We use the range-based method to define an applicability domain for our method. For this, we find the range (the minimum and maximum) of each descriptor over all relevant chemical compounds and represent each range as a set of linear constraints in the constraint set $\mathcal{C}_1$ of our MILP formulation. Recall that D_π stands for a set of chemical graphs used for constructing a prediction function. However, the number of examples in D_π may not be large enough to capture a general feature on the structure of chemical graphs. For this, we also use some data set from the whole set DB of chemical graphs in a database. Let $\mathrm{DB}^{(i)}_{\mathcal{G}}$ denote the set of chemical graphs $G \in \mathrm{DB} \cap \mathcal{G}$ such that $n(G) = i$ for each integer $i \geq 1$. Formally the set of variables and constraints is given as follows.

AD constraints in $\mathcal{C}_1$:
constants:

Integers $\mathrm{cs}^* \geq 3$ and $\mathrm{ch}^* \geq 1$;
An integer $d_{\max} \in \{3, 4\}$;
An integer $n^* \in [\mathrm{cs}^* + 1, \mathrm{cs}^* \cdot (d_{\max} - 1)^{\mathrm{ch}^*}]$;

variables for descriptors in x:
A real variable $\kappa_1 \geq 0$: κ_1 represents $\kappa_1(H)$;
$\mathrm{dg}(i) \in [0, n^*]$ ($i \in [1, 4]$): $\mathrm{dg}(i)$ represents the number of vertices of degree i in H;
Mass $\in \mathbb{Z}$: Mass represents $\sum_{v \in V} \mathrm{mass}^*(\alpha(v))$;
$\mathrm{ce}^{\mathrm{co}}(\mathtt{a}) \in [0, n^*]$, $\mathtt{a} \in \Lambda$: $\mathrm{ce}^{\mathrm{co}}(\mathtt{a})$ represents the number of vertices of chemical element $\mathtt{a}$ in the core of H;
$\mathrm{ce}^{\mathrm{nc}}(\mathtt{a}) \in [0, n^*]$, $\mathtt{a} \in \Lambda$: $\mathrm{ce}^{\mathrm{nc}}(\mathtt{a})$ represents the number of vertices of chemical element $\mathtt{a}$ in the non-core of H;
$\mathrm{b}^{\mathrm{co}}(k) \in [0, 2n^*]$, $k \in [1, 3]$: $\mathrm{b}^{\mathrm{co}}(k)$ represents the number of k-bonds in the core of H;
$\mathrm{b}^{\mathrm{nc}}(k) \in [0, 2n^*]$, $k \in [1, 3]$: $\mathrm{b}^{\mathrm{nc}}(k)$ represents the number of k-bonds in the non-core of H;
$\mathrm{ac}^{\mathrm{co}}(\gamma) \in [0, n^*]$, $\gamma \in \Gamma_< \cup \Gamma_=$: $\mathrm{ac}^{\mathrm{co}}(\gamma)$ represents the number of core edges in H that are assigned tuple $\gamma \in \Gamma_<$;
$\mathrm{ac}^{\mathrm{nc}}(\gamma) \in [0, n^*]$, $\gamma \in \Gamma_< \cup \Gamma_=$: $\mathrm{ac}^{\mathrm{nc}}(\gamma)$ represents the number of non-core edges in H that are assigned tuple $\gamma \in \Gamma_<$;

constraints:

$$n^* \min_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\kappa_1(G)}{n(G)} \leq \kappa_1 \leq n^* \max_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\kappa_1(G)}{n(G)}, \tag{A.1.1}$$

$$n^* \min_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{dg}_i(G)}{n(G)} \leq \mathrm{dg}(i) \leq n^* \max_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{dg}_i(G)}{n(G)}, \qquad i \in [1, 4], \tag{A.1.2}$$

$$n^* \min_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \overline{\mathrm{ms}}(G) \leq \mathrm{Mass} \leq n^* \max_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \overline{\mathrm{ms}}(G), \tag{A.1.3}$$

$$n^* \min_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{ce}_{\mathtt{a}}^{\mathrm{co}}(G)}{n(G)} \leq \mathrm{ce}^{\mathrm{co}}(\mathtt{a}) \leq n^* \max_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{ce}_{\mathtt{a}}^{\mathrm{co}}(G)}{n(G)}, \qquad \mathtt{a} \in \Lambda, \tag{A.1.4}$$

$$n^* \min_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{ce}_{\mathtt{a}}^{\mathrm{nc}}(G)}{n(G)} \leq \mathrm{ce}^{\mathrm{nc}}(\mathtt{a}) \leq n^* \max_{G \in D_\pi \cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}} \frac{\mathrm{ce}_{\mathtt{a}}^{\mathrm{nc}}(G)}{n(G)}, \qquad \mathtt{a} \in \Lambda, \tag{A.1.5}$$

$$(n^*+1)\min_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{b_k^{\mathrm{co}}(G)}{n(G)+1}\le \mathrm{b}^{\mathrm{co}}(k)\le (n^*+1)\max_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{b_k^{\mathrm{co}}(G)}{n(G)+1},\quad k\in[2,3], \tag{A.1.6}$$

$$(n^*+1)\min_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{b_k^{\mathrm{nc}}(G)}{n(G)+1}\le \mathrm{b}^{\mathrm{nc}}(k)\le (n^*+1)\max_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{b_k^{\mathrm{nc}}(G)}{n(G)+1},\quad k\in[2,3], \tag{A.1.7}$$

$$(n^*+1)\min_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{\mathrm{ac}_\gamma^{\mathrm{co}}(G)}{n(G)+1}\le \mathrm{ac}^{\mathrm{co}}(\gamma)\le (n^*+1)\max_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{\mathrm{ac}_\gamma^{\mathrm{co}}(G)}{n(G)+1},\qquad \gamma\in\Gamma, \tag{A.1.8}$$

$$(n^*+1)\min_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{\mathrm{ac}_\gamma^{\mathrm{nc}}(G)}{n(G)+1}\le \mathrm{ac}^{\mathrm{nc}}(\gamma)\le (n^*+1)\max_{G\in D_\pi\cup \mathrm{DB}_{\mathcal{G}}^{(n^*)}}\frac{\mathrm{ac}_\gamma^{\mathrm{nc}}(G)}{n(G)+1},\qquad \gamma\in\Gamma. \tag{A.1.9}$$

In the following, we derive an MILP $\mathcal{M}(x,g;\mathcal{C}_2)$ that satisfies the condition in Theorem 4.2. Let $d_{\max}\in\{3,4\}$, $n^*\ge 3$, $\mathrm{cs}^*\ge 3$ $\mathrm{ch}^*\ge 0$ and θ^* be given integers. We describe the set $\mathcal{C}_2$ with several sets of constraints.

A.1.2 Construction of Scheme Graph and Tree-Extension

We infer a subgraph H such that the maximum degree is $d_{\max}\in\{3,4\}$, $n(H)=n^*$, $\mathrm{cs}(H)=\mathrm{cs}^*$ and $\mathrm{ch}(H)=\mathrm{ch}^*$. For this, we first construct the $(t^*,\mathrm{ch}^*,d_{\max})$-tree-extension of the scheme graph $(K=(V_K=\{u_1,\ldots,u_{s^*}\},E_K=\{a_1,a_2,\ldots,a_m\})$, $\mathcal{E}=(E_1,E_2,E_3))$. We use the following notations: For $j\in[1,3]$ and $s\in[1,s^*]$, let $E_j^+(s)$ (resp., $E_j^-(s)$) denote the set of indices i of edges $a_i\in E_i$ such that the tail (resp., head) of a_i is $u_{s,1}$. Let $E_{j,k}^+(s)\triangleq E_j^+(s)\cup E_k^+(s)$, $E_{j,k}^-(s)\triangleq E_j^-(s)\cup E_k^-(s)$, $E_j(s)\triangleq E_j^+(s)\cup E_j^-(s)$ and $E_{j,k}(s)\triangleq E_j(s)\cup E_k(s)$.

As described in Section 4.3, some edge $a(i)\in E_1\cup E_2$ may be replaced with a subpath P_i of $(v_{1,1},v_{1,2},\ldots,v_{t^*,1})$, which consists of the roots of trees $T_1,T_2,\ldots,T_{t^*}$. We assign color i to the vertices in such a subpath P_i by setting a variable $\chi(t)$ of each vertex $v_{t,1}\in V(P_i)$ to be i. For each edge $u_{s,1}v_{t,1}$, we prepare a binary variable $e(s,t)$ to denote that edge $u_{s,1}v_{t,1}$ is used (resp., not used) in a selected graph H when $e(s,t)=1$ (resp., $e(s,t)=0$). We also include constraints necessary for the variables to satisfy a degree condition at each of the vertices $u_{s,1}$, $s\in[1,s^*]$ and $v_{t,1}$, $t\in[1,t^*]$.

constants:
Integers $s^*=|V_K|$, $c^*=|E_1\cup E_2|$, cs^* $(\ge s^*)$, n^* $(\ge \mathrm{cs}^*)$ and $\mathrm{ch}^*\ge 0$;
$\underline{d}^+(s)$, $s\in[1,s^*]$: a lower bound on the out-degree of vertex $u_{s,1}$ in H;
$\underline{d}^-(s)$, $s\in[1,s^*]$: a lower bound on the in-degree of vertex $u_{s,1}$ in H;

$\overline{d}^{+}(s)$, $s \in [1, s^*]$: an upper bound on the out-degree of vertex $u_{s,1}$ in H;
$\overline{d}^{-}(s)$, $s \in [1, s^*]$: an upper bound on the in-degree of vertex $u_{s,1}$ in H;

variables:
$a(i) \in \{0, 1\}$, $i \in E_1 \cup E_3$: $a(i)$ represents edge $a_i \in E_1 \cup E_3$ ($a(i) = 1$, $i \in E_1$) ($a(i) = 1 \Leftrightarrow$ edge a_i is used in H);
$e(s,t), e(t,s) \in \{0, 1\}$, $s \in [1, s^*]$, $t \in [1, t^*]$: $e(s,t)$ (resp., $e(t,s)$) represents direction $(u_{s,1}, v_{t,1})$ (resp., $(v_{t,1}, u_{s,1})$), where $e(s,t) = 1$ (resp., $e(t,s) = 1$) $\Leftrightarrow$ edge $u_{s,1}, v_{t,1}$ is used in H and direction $(u_{s,1}, v_{t,1})$ (resp., $(v_{t,1}, u_{s,1})$) is assigned to edge $u_{s,1}, v_{t,1}$;
$\chi(t) \in [1, c^*]$, $t \in [1, t^*]$: $\chi(t)$ represents the color assigned to vertex $v_{t,1}$ ($\chi(t) = c \Leftrightarrow$ vertex $v_{t,1}$ is assigned color c);
$\mathrm{clr}(c) \in [0, n^* - s^*]$, $c \in [1, c^*]$: the number of vertices $v_{t,i}$ with color c;
$\deg^{\mathrm{co}+}(s) \in [1, 4]$, $s \in [1, s^*]$: the out-degree of vertex $u_{s,1}$ in the core of H;
$\deg^{\mathrm{co}-}(s) \in [1, 4]$, $s \in [1, s^*]$: the in-degree of vertex $u_{s,1}$ in the core of H;
$\delta_{\mathrm{clr}}(t, c) \in \{0, 1\}$, $t \in [1, t^*]$, $c \in [1, c^*]$ ($\delta_{\mathrm{clr}}(t, c) = 1 \Leftrightarrow \chi(t) = c$);

constraints:

$$\sum_{c \in [1, c^*]} \delta_{\mathrm{clr}}(t, c) = 1, \qquad t \in [1, t^*], \tag{A.1.10}$$

$$\sum_{c \in [1, c^*]} c \cdot \delta_{\mathrm{clr}}(t, c) = \chi(t), \qquad t \in [1, t^*], \tag{A.1.11}$$

$$\sum_{t \in [1, t^*]} \delta_{\mathrm{clr}}(t, c) = \mathrm{clr}(c), \qquad c \in [1, c^*], \tag{A.1.12}$$

$$e(s,t) + e(t,s) \leq 1, \qquad s \in [1, s^*], t \in [1, t^*], \tag{A.1.13}$$

$$\sum_{s \in [1, s^*] \setminus \{\mathrm{head}(c)\}} e(t,s) \leq 1 - \delta_{\mathrm{clr}}(t, c), \qquad c \in [1, c^*], t \in [1, t^*], \tag{A.1.14}$$

$$\sum_{s \in [1, s^*] \setminus \{\mathrm{tail}(c)\}} e(s,t) \leq 1 - \delta_{\mathrm{clr}}(t, c), \qquad c \in [1, c^*], t \in [1, t^*], \tag{A.1.15}$$

$$\sum_{i \in E^{-}_{1,3}(s)} a(i) + \sum_{t \in [1, t^*]} e(t,s) = \deg^{\mathrm{co}-}(s), \qquad s \in [1, s^*], \tag{A.1.16}$$

$$\sum_{i \in E^{+}_{1,3}(s)} a(i) + \sum_{t \in [1, t^*]} e(s,t) = \deg^{\mathrm{co}+}(s), \qquad s \in [1, s^*], \tag{A.1.17}$$

$$\underline{d}^{+}(s) \leq \deg^{\mathrm{co}+}(s) \leq \overline{d}^{+}(s), \qquad s \in [1, s^*], \tag{A.1.18}$$

$$\underline{d}^{-}(s) \leq \deg^{\mathrm{co}-}(s) \leq \overline{d}^{-}(s), \qquad s \in [1, s^*]. \tag{A.1.19}$$

A.1.3 Specification for Chemical Graphs with Rank 2

To generate any of the three rank-2 polymer topologies in $\mathcal{PT}(2,4)$, we use the scheme graph $(K = (V_K = \{u_1, u_2, u_3, u_4\}, E_K), \mathcal{E} = (E_1, E_2, E_3))$ in Figure 4.2d, where $s^* = |V(K)| = 4$, $c^* = |E_1 \cup E_2| = 5$, $E_1 = \{a_1 = (u_1, u_4), a_2 = (u_2, u_3), a_3 = (u_2, u_4)\}$, $E_2 = \{a_4 = (u_1, u_2), a_5 = (u_3, u_4)\}$ and $E_3 = \{a_6 = (u_1, u_2), a_7 = (u_3, u_4)\}$. Recall that each color $i \in [1, c^*]$ is assigned to edge $a_i \in E_1 \cup E_2$. We impose some more constraints on the degree of each of the vertices $u_{s,1}$, $s \in [1, s^*]$ and $v_{t,1}$, $t \in [1, t^*]$ so that the core of a selected graph H satisfies one of the three least simple graphs in Figure 4.2a–c. We also let a variable θ mean the topological parameter $\theta(H)$ of a selected subgraph H.

constants:
$s^* = 4$, $c^* = 5$,
$E_1^-(1) = \emptyset$, $E_2^-(1) = \emptyset$, $E_3^-(1) = \emptyset$, $E_1^+(1) = \{1\}$, $E_2^+(1) = \{4\}$, $E_3^+(1) = \{6\}$,
$E_1^-(2) = \emptyset$, $E_2^-(2) = \{4\}$, $E_3^-(2) = \{6\}$, $E_1^+(2) = \{2,3\}$, $E_2^+(2) = \emptyset$, $E_3^+(2) = \emptyset$,
$E_1^-(3) = \{2\}$, $E_2^-(3) = \emptyset$, $E_3^-(3) = \emptyset$, $E_1^+(3) = \emptyset$, $E_2^+(3) = \{5\}$, $E_3^+(3) = \{7\}$,
$E_1^-(4) = \{1,3\}$, $E_2^-(4) = \{5\}$, $E_3^-(4) = \{7\}$, $E_1^+(4) = \emptyset$, $E_2^+(4) = \emptyset$, $E_3^+(4) = \emptyset$,
$\underline{d}^-(1) = 0$, $\overline{d}^-(1) = 0$, $\underline{d}^+(1) = 2$, $\overline{d}^+(1) = 2$,
$\underline{d}^-(2) = 1$, $\overline{d}^-(2) = 2$, $\underline{d}^+(2) = 1$, $\overline{d}^+(2) = 2$,
$\underline{d}^-(3) = 1$, $\overline{d}^-(3) = 1$, $\underline{d}^+(3) = 1$, $\overline{d}^+(3) = 2$,
$\underline{d}^-(4) = 2$, $\overline{d}^-(4) = 3$, $\underline{d}^+(4) = 0$, $\overline{d}^+(4) = 0$,

variables:
$\theta \in [-n^*, n^*]$: The topology-parameter $\theta(H)$ for rank 2;

constraints:

$$a(2) + \mathrm{clr}(2) \geq 1, \tag{A.1.20}$$
$$a(3) + \mathrm{clr}(3) + \mathrm{clr}(4) \geq 1, \tag{A.1.21}$$
$$\mathrm{clr}(4) \geq \mathrm{clr}(5), \tag{A.1.22}$$
$$\mathrm{clr}(3) \leq \mathrm{clr}(2) + 1, \tag{A.1.23}$$
$$\mathrm{clr}(3) \leq \mathrm{clr}(1) + 1 + n^*(3 - \deg^{\mathrm{co}-}(4)), \tag{A.1.24}$$
$$-\theta \leq 1 + \mathrm{clr}(2) + n^*(2 - \deg^{\mathrm{co}+}(3)), \tag{A.1.25}$$
$$-\theta \geq 1 + \mathrm{clr}(2) - n^*(2 - \deg^{\mathrm{co}+}(3)), \tag{A.1.26}$$
$$\theta \leq n^*(4 - \deg^{\mathrm{co}+}(2) - \deg^{\mathrm{co}-}(2)), \tag{A.1.27}$$
$$\theta \geq -n^*(4 - \deg^{\mathrm{co}+}(2) - \deg^{\mathrm{co}-}(2)), \tag{A.1.28}$$
$$\theta \leq 1 + \mathrm{clr}(3) + n^*(3 - \deg^{\mathrm{co}-}(4)), \tag{A.1.29}$$
$$\theta \geq 1 + \mathrm{clr}(3) - n^*(3 - \deg^{\mathrm{co}-}(4)). \tag{A.1.30}$$

A.1.4 Selecting A Subgraph

We prepare a binary variable $u(s,i)$ (resp., $v(t,i)$) for each vertex $u_{s,i}$ in tree S_s (resp., $v_{t,i}$ in tree T_t). We include constraints so that the path $(v_{1,1}, v_{1,2}, \ldots, v_{t^*,1})$ is partitioned into subpaths P_c, $c \in [1, c^*]$, where possibly some P_c is empty, and the resulting subgraph H becomes a connected rank-2 graph with $n(H) = n^*$, $\mathrm{cs}(H) = \mathrm{cs}^*$, $\mathrm{ch}(H) = \mathrm{ch}^*$ and $\theta(H) = \theta^*$.

constants:
Integers $d_{\max} \in \{3, 4\}$, $\mathrm{ch}^* \geq 0$;
Prepare the set $\mathrm{Cld}(i)$ of the indices of children of a vertex v_i
the index $\mathrm{prt}(i)$ of the parent of a non-root vertex v_i, and
the set $\mathrm{Dst}(h)$ of indices i such that the height of a vertex v_i is h
in the rooted tree $T(2, d_{\max} - 1, \mathrm{ch}^*)$;

variables:
$u(s,i) \in \{0,1\}$, $s \in [1, s^*]$, $i \in [1, n_{\mathrm{tree}}]$: $u(s,i)$ represents vertex $u_{s,i}$
($u(s,i) = 1 \Leftrightarrow$ vertex $u_{s,i}$ is used in H and edge $e'_{s,i}$ $(i \geq 2)$ is used in H);
$v(t,i) \in \{0,1\}$, $t \in [1, t^*]$, $i \in [1, n_{\mathrm{tree}}]$: $v(t,i)$ represents vertex $v_{t,i}$
($v(t,i) = 1 \Leftrightarrow$ vertex $v_{t,i}$ is used in H and edge $e_{t,i}$ $(i \geq 2)$ is used in H);
$e(t) \in \{0,1\}$, $t \in [1, t^* + 1]$: $e(t)$ represents edge $e_t = v_{t-1,1}v_{t,i}$,
where $e_{1,1}$ and $e_{t^*+1,1}$ are fictitious edges ($e(t) = 1 \Leftrightarrow$ edge e_t is used in H);

constraints:

$$u(s,1) = 1, \qquad s \in [1, s^*], \tag{A.1.31}$$

$$d_{\max} \cdot u(t,i) \geq \sum_{j \in \mathrm{Cld}(i)} u(t,j), \quad t \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{in}}], \tag{A.1.32}$$

$$v(t,1) = 1, \qquad t \in [1, t^*], \tag{A.1.33}$$

$$d_{\max} \cdot v(t,i) \geq \sum_{j \in \mathrm{Cld}(i)} v(t,j), \quad t \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{in}}], \tag{A.1.34}$$

$$\sum_{s \in [1,s^*], i \in [1, n_{\mathrm{tree}}]} u(s,i) + \sum_{t \in [1,t^*], i \in [1, n_{\mathrm{tree}}]} v(t,i) = n^*, \tag{A.1.35}$$

$$\sum_{s \in [1,s^*], i \in \mathrm{Dst}(\mathrm{ch}^*)} u(s,i) + \sum_{t \in [1,t^*], i \in \mathrm{Dst}(\mathrm{ch}^*)} v(t,i) \geq 1, \tag{A.1.36}$$

$$e(1) = e(t^* + 1) = 0, \tag{A.1.37}$$

$$e(t+1) + \sum_{s \in [1, s^*]} e(t, s) = 1, \qquad t \in [1, t^*], \tag{A.1.38}$$

$$e(t) + \sum_{s \in [1, s^*]} e(s, t) = 1, \qquad t \in [1, t^*], \tag{A.1.39}$$

$$c^* \geq \chi(1) \geq \chi(2) \geq \cdots \geq \chi(t^*) \geq 1, \tag{A.1.40}$$

$$e(t+1) \geq 1 + \chi(t+1) - \chi(t), \qquad t \in [1, t^* - 1], \tag{A.1.41}$$

$$c^* \cdot (1 - e(t+1)) \geq \chi(t) - \chi(t+1), \qquad t \in [1, t^* - 1]. \tag{A.1.42}$$

A.1.5 Assigning Multiplicity

We prepare an integer variable $\widetilde{\beta}(e)$ or $\widehat{\beta}(e)$ for each edge e in the $(t^*, \mathrm{ch}^*, d_{\max})$-tree-extension of the scheme graph to denote the multiplicity of e in a selected graph H and include necessary constraints for the variables to satisfy in H.

variables:
$\widetilde{\beta}(i) \in [0, 3]$, $i \in E_1 \cup E_3$: $\widetilde{\beta}(i)$ represents the multiplicity of edge a_i,
where $\widetilde{\beta}(i) = 0$ if edge a_i is not in H;
$\widetilde{\beta}(p, i) \in [0, 3]$, $p \in [1, \mathrm{cs}^*]$, $i \in [2, n_{\mathrm{tree}}]$: $\widetilde{\beta}(p, i)$ with $p \leq s^*$ (resp., $p > s^*$) represents the multiplicity of edge $e'_{p,i}$ (resp., $e_{p-s^*,i}$);
$\widetilde{\beta}(t, 1) \in [0, 3]$, $t \in [1, t^* + 1]$: $\widetilde{\beta}(t, 1)$ represents the multiplicity of edge e_t;
$\widehat{\beta}(s, t) \in [0, 3]$, $s \in [1, s^*]$, $t \in [1, t^*]$: $\widehat{\beta}(s, t)$ represents the multiplicity of edge $u_{s,1}v_{t,1}$;

constraints:

$$a(i) = 1, \qquad i \in E_3, \tag{A.1.43}$$

$$a(i) \leq \widetilde{\beta}(i) \leq 3a(i), \qquad i \in E_1 \cup E_3, \tag{A.1.44}$$

$$u(s, i) \leq \widetilde{\beta}(s, i) \leq 3u(s, i), \qquad s \in [1, s^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.45}$$

$$v(t, i) \leq \widetilde{\beta}(s^* + t, i) \leq 3v(t, i), \qquad t \in [1, t^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.46}$$

$$e(t) \leq \widetilde{\beta}(t, 1) \leq 3e(t), \qquad t \in [1, t^* + 1], \tag{A.1.47}$$

$$e(s, t) + e(t, s) \leq \widehat{\beta}(s, t) \leq 3e(s, t) + 3e(t, s), \qquad s \in [1, s^*], t \in [1, t^*]. \tag{A.1.48}$$

A.1.6 Assigning Chemical Elements and Valence Condition

We include constraints so that each vertex v in a selected graph H satisfies the valence condition; i.e., $\sum_{uv \in E(H)} \beta(uv) \leq \mathrm{val}(\alpha(u))$. With these constraints, a rank-2 chemical graph $G = (H, \alpha, \beta)$ on a selected subgraph H will be constructed.

constants:
A set $\Lambda \cup \{\epsilon\}$ of chemical elements, where ϵ denotes null;
A coding $[\mathtt{a}]$, $\mathtt{a} \in \Lambda \cup \{\epsilon\}$ such that $[\epsilon] = 0$; $[\mathtt{a}] \geq 1$, $\mathtt{a} \in \Lambda$; and $[\mathtt{a}] \neq [\mathtt{b}]$ if $\mathtt{a} \neq \mathtt{b}$;
Let $[\Lambda]$ and $[\Lambda \cup \{\epsilon\}]$ denote $\{[\mathtt{a}] \mid \mathtt{a} \in \Lambda\}$ and $\{[\mathtt{a}] \mid \mathtt{a} \in \Lambda \cup \{\epsilon\}\}$, respectively;
A valence function: $\mathrm{val} : \Lambda \to [1, 4]$;

variables:
$\widetilde{\alpha}(p, i) \in [\Lambda \cup \{\epsilon\}]$, $p \in [1, \mathrm{cs}^*]$, $i \in [1, n_{\mathrm{tree}}]$:
$\widetilde{\alpha}(p, i)$ with $p \leq s^*$ (resp., $p > s^*$) represents $\alpha(u_{p,i})$ (resp., $\alpha(v_{p-s^*,i})$);
$\delta_\alpha(p, i, \mathtt{a}) \in \{0, 1\}$, $p \in [1, \mathrm{cs}^*]$, $i \in [1, n_{\mathrm{tree}}]$, $\mathtt{a} \in \Lambda \cup \{\epsilon\}$:
$\delta_\alpha(p, i, \mathtt{a}) = 1 \Leftrightarrow \alpha(u_{p,i}) = \mathtt{a}$ for $p \leq s^*$ and $\alpha(v_{p-s^*,i}) = \mathtt{a}$ for $p > s^*$;
$\delta_{\widetilde{\beta}}(i, k) \in \{0, 1\}$, $p \in [1, \mathrm{cs}^*]$, $i \in E_1 \cup E_3$, $k \in [0, 3]$:
$\delta_{\widetilde{\beta}}(i, k) = 1 \Leftrightarrow$ the multiplicity of edge a_i in H is k;
$\delta_{\widetilde{\beta}}(p, i, k) \in \{0, 1\}$, $p \in [1, \mathrm{cs}^*]$, $i \in [2, n_{\mathrm{tree}}]$, $k \in [0, 3]$:
$\delta_{\widetilde{\beta}}(p, i, k) = 1 \Leftrightarrow$ the multiplicity of edge $e'_{p,i}$, $p \leq s^*$ (or $e_{p-s^*,i}$, $p > s^*$) in H is k;
$\delta_{\widetilde{\beta}}(t, 1, k) \in \{0, 1\}$, $t \in [1, t^* + 1]$, $k \in [0, 3]$:
$\delta_{\widetilde{\beta}}(t, 1, k) = 1 \Leftrightarrow$ the multiplicity of edge e_t in H is k;
$\delta_{\widetilde{\beta}}(s, t, k) \in \{0, 1\}$, $s \in [1, s^*]$, $t \in [1, t^*]$, $k \in [0, 3]$:
$\delta_{\widetilde{\beta}}(s, t, k) = 1 \Leftrightarrow$ the multiplicity of edge $u_{s,1}v_{t,1}$ in H is k;

constraints:

$$\sum_{\mathtt{a} \in \Lambda \cup \{\epsilon\}} \delta_\alpha(p, i, \mathtt{a}) = 1, \qquad p \in [1, \mathrm{cs}^*], i \in [1, n_{\mathrm{tree}}], \tag{A.1.49}$$

$$\sum_{\mathtt{a} \in \Lambda \cup \{\epsilon\}} [\mathtt{a}] \cdot \delta_\alpha(p, i, \mathtt{a}) = \widetilde{\alpha}(p, i), \qquad p \in [1, \mathrm{cs}^*], i \in [1, n_{\mathrm{tree}}], \tag{A.1.50}$$

$$\sum_{k \in [0,3]} \delta_{\widetilde{\beta}}(i, k) = 1, \qquad i \in E_1 \cup E_3, \tag{A.1.51}$$

$$\sum_{k \in [1,3]} k \cdot \delta_{\widetilde{\beta}}(i, k) = \widetilde{\beta}(i), \qquad i \in E_1 \cup E_3, \tag{A.1.52}$$

$$\sum_{k\in[0,3]} \delta_{\widetilde{\beta}}(p,i,k) = 1, \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.53}$$

$$\sum_{k\in[1,3]} k \cdot \delta_{\widetilde{\beta}}(p,i,k) = \widetilde{\beta}(p,i), \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.54}$$

$$\sum_{k\in[0,3]} \delta_{\widetilde{\beta}}(t,1,k) = 1, \qquad t \in [1, t^*+1], \tag{A.1.55}$$

$$\sum_{k\in[1,3]} k \cdot \delta_{\widetilde{\beta}}(t,1,k) = \widetilde{\beta}(t,1), \qquad t \in [1, t^*+1], \tag{A.1.56}$$

$$\sum_{k\in[0,3]} \delta_{\widehat{\beta}}(s,t,k) = 1, \qquad s \in [1, s^*], t \in [1, t^*], \tag{A.1.57}$$

$$\sum_{k\in[0,3]} k\delta_{\widehat{\beta}}(s,t,k) = \widehat{\beta}(s,t), \qquad s \in [1, s^*], t \in [1, t^*], \tag{A.1.58}$$

$$\sum_{i\in E_{1,3}(s)} \widetilde{\beta}(i) + \sum_{t\in[1,t^*]} \widehat{\beta}(s,t) + \sum_{j\in \mathrm{Cld}(1)} \widetilde{\beta}(s,j) \le \sum_{\mathtt{a}\in\Lambda} \mathrm{val}(\mathtt{a}) \cdot \delta_\alpha(s,1,\mathtt{a}), \qquad s \in [1, s^*], \tag{A.1.59}$$

$$\sum_{s\in[1,s^*]} \widehat{\beta}(s,t) + \widetilde{\beta}(t,1) + \widetilde{\beta}(t+1,1) + \sum_{j\in \mathrm{Cld}(1)} \widetilde{\beta}(s^*+t,j) \le \sum_{\mathtt{a}\in\Lambda} \mathrm{val}(\mathtt{a}) \cdot \delta_\alpha(s^*+t,1,\mathtt{a}), \quad t \in [1, t^*], \tag{A.1.60}$$

$$\widetilde{\beta}(p,i) + \sum_{j\in \mathrm{Cld}(i)} \widetilde{\beta}(p,j) \le \sum_{\mathtt{a}\in\Lambda} \mathrm{val}(\mathtt{a}) \cdot \delta_\alpha(p,i,\mathtt{a}), \quad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}]. \tag{A.1.61}$$

A.1.7 Descriptors for Mass, the Numbers of Elements and Bonds

We include constraints to compute descriptors $\overline{\mathrm{ms}}(G)$ $\mathrm{ce}^{\mathrm{co}}_{\mathtt{a}}(G)$, $\mathrm{ce}^{\mathrm{nc}}_{\mathtt{a}}(G)$ ($\mathtt{a} \in \Lambda$), $b_k(G)$ ($k \in [2,3]$) and $n_{\mathtt{H}}(G)$ according to the definitions in Section 4.2.2.

constants:

A function $\mathrm{mass}^* : \Lambda \to \mathbb{Z}$; Let $\mathrm{mass}(\mathtt{a})$ denote the observed mass of a chemical element $\mathtt{a} \in \Lambda$, and
define $\mathrm{mass}^*(\mathtt{a}) = \lfloor 10 \cdot \mathrm{mass}(\mathtt{a}) \rfloor$;

variables:

$\mathrm{ce}^{\mathrm{co}}(\mathtt{a}) \in [0, n^*]$, $\mathtt{a} \in \Lambda$;

$\mathrm{ce}^{\mathrm{nc}}(\mathtt{a}) \in [0, n^*]$, $\mathtt{a} \in \Lambda$;
Mass $\in \mathbb{Z}$;
$\mathrm{b}^{\mathrm{co}}(k) \in [0, 2n^*]$, $k \in [1, 3]$;
$\mathrm{b}^{\mathrm{nc}}(k) \in [0, 2n^*]$, $k \in [1, 3]$;
$\mathrm{n}_{\mathtt{H}} \in [0, 4n^*]$: the number of hydrogen atoms to be included in G;

constraints:

$$\sum_{p \in [1, \mathrm{cs}^*]} \delta_\alpha(p, 1, \mathtt{a}) = \mathrm{ce}^{\mathrm{co}}(\mathtt{a}), \qquad \mathtt{a} \in \Lambda, \quad \text{(A.1.62)}$$

$$\sum_{p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}]} \delta_\alpha(p, i, \mathtt{a}) = \mathrm{ce}^{\mathrm{nc}}(\mathtt{a}), \qquad \mathtt{a} \in \Lambda, \quad \text{(A.1.63)}$$

$$\sum_{\mathtt{a} \in \Lambda} \mathrm{mass}^*(\mathtt{a})(\mathrm{ce}^{\mathrm{co}}(\mathtt{a}) + \mathrm{ce}^{\mathrm{nc}}(\mathtt{a})) = \mathrm{Mass}, \quad \text{(A.1.64)}$$

$$\sum_{i \in E_1 \cup E_3} \delta_{\widetilde{\beta}}(i, k) + \sum_{s \in [1, s^*], t \in [1, t^*]} \delta_{\widehat{\beta}}(s, t, k) + \sum_{t \in [2, t^*]} \delta_{\widetilde{\beta}}(t, 1, k) = \mathrm{b}^{\mathrm{co}}(k), \quad k \in [1, 3], \quad \text{(A.1.65)}$$

$$\sum_{p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}]} \delta_{\widetilde{\beta}}(p, i, k) = \mathrm{b}^{\mathrm{nc}}(k), \quad k \in [1, 3], \quad \text{(A.1.66)}$$

$$\sum_{\mathtt{a} \in \Lambda} \mathrm{val}(\mathtt{a})(\mathrm{ce}^{\mathrm{co}}(\mathtt{a}) + \mathrm{ce}^{\mathrm{nc}}(\mathtt{a})) - 2(n^* + 1 + \mathrm{b}^{\mathrm{co}}(2) + \mathrm{b}^{\mathrm{nc}}(2) + 2\mathrm{b}^{\mathrm{co}}(3) + 2\mathrm{b}^{\mathrm{nc}}(3)) = \mathrm{n}_{\mathtt{H}}. \quad \text{(A.1.67)}$$

A.1.8 Descriptor for the Number of Specified Degree

We include constraints to compute descriptors $\mathrm{dg}_i(G)$ ($i \in [1, 4]$) according to the definitions in Section 4.2.2. We also add constraints so that the maximum degree of a non-core vertex in H is at most 3 (resp., equal to 4) when $d_{\max} = 3$ (resp., $d_{\max} = 4$).

variables:
$\deg(p, i) \in [0, 4]$, $p \in [1, \mathrm{cs}^*]$, $i \in [1, n_{\mathrm{tree}}]$:
$\deg(p, i)$ represents $\deg_H(u_{p,i})$ for $p \leq s^*$ or $\deg_H(v_{p-s^*,i})$ for $p > s^*$;
$\delta_{\deg}(p, i, d) \in \{0, 1\}$, $p \in [1, \mathrm{cs}^*]$, $i \in [1, n_{\mathrm{tree}}]$, $d \in [0, 4]$:
$\delta_{\deg}(p, i, d) = 1 \Leftrightarrow \deg(p, i) = d$;
$\mathrm{dg}(d) \in [0, n^*]$, $d \in [1, 4]$;

constraints:

$$\sum_{i\in E_{1,3}(s)} a(i) + \sum_{t\in[1,t^*]} (e(s,t)+e(t,s)) + \sum_{j\in\mathrm{Cld}(1)} u(s,j) = \deg(s,1), \qquad s\in[1,s^*], \tag{A.1.68}$$

$$u(s,i) + \sum_{j\in\mathrm{Cld}(i)} u(s,j) = \deg(s,i), \qquad s\in[1,s^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.69}$$

$$2 + \sum_{j\in\mathrm{Cld}(1)} v(t,j) = \deg(s^*+t,1), \qquad t\in[1,t^*], \tag{A.1.70}$$

$$v(t,i) + \sum_{j\in\mathrm{Cld}(i)} v(t,j) = \deg(s^*+t,i), \qquad t\in[1,t^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.71}$$

$$\sum_{d\in[0,4]} \delta_{\deg}(p,i,d) = 1, \qquad p\in[1,\mathrm{cs}^*], i\in[1,n_{\mathrm{tree}}], \tag{A.1.72}$$

$$\sum_{d\in[1,4]} d\cdot\delta_{\deg}(p,i,d) = \deg(p,i), \qquad p\in[1,\mathrm{cs}^*], i\in[1,n_{\mathrm{tree}}], \tag{A.1.73}$$

$$\sum_{p\in[1,\mathrm{cs}^*],i\in[1,n_{\mathrm{tree}}]} \delta_{\deg}(p,i,d) = \mathrm{dg}(d), \qquad d\in[1,4], \tag{A.1.74}$$

$$\sum_{p\in[1,\mathrm{cs}^*],i\in[2,n_{\mathrm{tree}}]} \delta_{\deg}(p,i,4) \geq 1 \text{ (resp., } = 0) \quad \text{when } d_{\max}=4 \text{ (resp., } =3). \tag{A.1.75}$$

A.1.9 Descriptor for the Number of Adjacency-Configurations

We include constraints to compute descriptors $\mathrm{ac}^{\mathrm{co}}_\gamma(G)$ and $\mathrm{ac}^{\mathrm{nc}}_\gamma(G)$ ($\gamma=(\mathtt{a},\mathtt{b},k)\in\Gamma$) according to the definitions in Section 4.2.2.

constants:

A set $\Gamma=\Gamma_<\cup\Gamma_=\cup\Gamma_>$ of proper tuples $(\mathtt{a},\mathtt{b},k)\in\Lambda\times\Lambda\times[1,3]$;

The set $\Gamma_0=\{(\mathtt{a},\mathtt{b},0)\mid \mathtt{a},\mathtt{b}\in\Lambda\cup\{\epsilon\}\}$;

variables:

$\delta_\tau(i,\gamma)\in\{0,1\}$, $i\in E_1\cup E_3$, $\gamma\in\Gamma\cup\Gamma_0$:

$\delta_\tau(i,\gamma)=1\Leftrightarrow$ edge a_i is assigned tuple γ; i.e., $\gamma=(\widetilde{\alpha}(\mathrm{tail}(i),1),\widetilde{\alpha}(\mathrm{head}(i),1),\widetilde{\beta}(i))$;

$\delta_\tau(t,1,\gamma)\in\{0,1\}$, $t\in[2,t^*]$, $\gamma\in\Gamma\cup\Gamma_0$:

$\delta_\tau(t,1,\gamma) = 1 \Leftrightarrow$ edge e_t is assigned tuple γ; i.e., $\gamma = (\widetilde{\alpha}(s^* + t - 1, 1), \widetilde{\alpha}(s^* + t, 1), \widetilde{\beta}(t,1))$;
$\delta_\tau(t,i,\gamma) \in \{0,1\}$, $p \in [1, \mathrm{cs}^*]$, $i \in [2, n_{\mathrm{tree}}]$, $\gamma \in \Gamma \cup \Gamma_0$:
$\delta_\tau(t,i,\gamma) = 1 \Leftrightarrow$ edge $e'_{p,i}$, $p \le s^*$ (or $e_{p-s^*,i}$, $p > s^*$) is assigned tuple γ; i.e.,
$\gamma = (\widetilde{\alpha}(p, \mathrm{prt}(i)), \widetilde{\alpha}(p,i), \widetilde{\beta}(p,i))$;
$\delta_{\widehat{\tau}}(s,t,\gamma) \in \{0,1\}$, $s \in [1, s^*]$, $t \in [1, t^*]$, $\gamma \in \Gamma \cup \Gamma_0$:
$\delta_{\widehat{\tau}}(s,t,\gamma) = 1 \Leftrightarrow$ edge $u_{s,1}v_{t,1}$ is assigned tuple γ; i.e., $\gamma = (\widetilde{\alpha}(s,1), \widetilde{\alpha}(s^* + t, 1), \widehat{\beta}(s,t))$;
$\mathrm{ac}^{\mathrm{co}}(\gamma) \in [0, n^*]$, $\gamma \in \Gamma_< \cup \Gamma_=$;
$\mathrm{ac}^{\mathrm{nc}}(\gamma) \in [0, n^*]$, $\gamma \in \Gamma_< \cup \Gamma_=$;

constraints:

$$\sum_{\gamma \in \Gamma \cup \Gamma_0} \delta_\tau(i,\gamma) = 1, \qquad i \in E_1 \cup E_3, \tag{A.1.76}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} [\mathtt{a}]\delta_\tau(i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(\mathrm{tail}(i),1), \qquad i \in E_1 \cup E_3, \tag{A.1.77}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} [\mathtt{b}]\delta_\tau(i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(\mathrm{head}(i),1), \qquad i \in E_1 \cup E_3, \tag{A.1.78}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} k \cdot \delta_\tau(i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\beta}(i), \qquad i \in E_1 \cup E_3, \tag{A.1.79}$$

$$\sum_{\gamma \in \Gamma \cup \Gamma_0} \delta_\tau(t,1,\gamma) = 1, \qquad t \in [2, t^*], \tag{A.1.80}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} [\mathtt{a}]\delta_\tau(t,1,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(s^* + t - 1, 1), \qquad t \in [2, t^*], \tag{A.1.81}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} [\mathtt{b}]\delta_\tau(t,1,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(s^* + t, 1), \qquad t \in [2, t^*], \tag{A.1.82}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} k \cdot \delta_\tau(t,1,(\mathtt{a},\mathtt{b},k)) = \widetilde{\beta}(t,1), \qquad t \in [2, t^*], \tag{A.1.83}$$

$$\sum_{\gamma \in \Gamma \cup \Gamma_0} \delta_\tau(p,i,\gamma) = 1, \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.84}$$

$$\sum_{(\mathtt{a},\mathtt{b},k) \in \Gamma \cup \Gamma_0} [\mathtt{a}]\delta_\tau(p,i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(p, \mathrm{prt}(i)), \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{tree}}], \tag{A.1.85}$$

$$\sum_{(\mathtt{a},\mathtt{b},k)\in\Gamma\cup\Gamma_0} [\mathtt{b}]\delta_\tau(p,i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(p,i), \quad p\in[1,\mathrm{cs}^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.86}$$

$$\sum_{(\mathtt{a},\mathtt{b},k)\in\Gamma\cup\Gamma_0} k\cdot\delta_\tau(p,i,(\mathtt{a},\mathtt{b},k)) = \widetilde{\beta}(p,i), \quad p\in[1,\mathrm{cs}^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.87}$$

$$\sum_{\gamma\in\Gamma\cup\Gamma_0} \delta_{\widehat{\tau}}(s,t,\gamma) = 1, \quad s\in[1,s^*], t\in[1,t^*], \tag{A.1.88}$$

$$\sum_{(\mathtt{a},\mathtt{b},k)\in\Gamma\cup\Gamma_0} [\mathtt{a}]\delta_{\widehat{\tau}}(s,t,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(s,1), \quad s\in[1,s^*], t\in[1,t^*], \tag{A.1.89}$$

$$\sum_{(\mathtt{a},\mathtt{b},k)\in\Gamma\cup\Gamma_0} [\mathtt{b}]\delta_{\widehat{\tau}}(s,t,(\mathtt{a},\mathtt{b},k)) = \widetilde{\alpha}(s^*+t,1), \quad s\in[1,s^*], t\in[1,t^*], \tag{A.1.90}$$

$$\sum_{(\mathtt{a},\mathtt{b},k)\in\Gamma\cup\Gamma_0} k\cdot\delta_{\widehat{\tau}}(s,t,(\mathtt{a},\mathtt{b},k)) = \widehat{\beta}(s,t), \quad s\in[1,s^*], t\in[1,t^*], \tag{A.1.91}$$

$$\begin{aligned}
&\sum_{i\in E_1\cup E_3} (\delta_\tau(i,\gamma)+\delta_\tau(i,\overline{\gamma})) \\
&+ \sum_{s\in[1,s^*],t\in[1,t^*]} (\delta_{\widehat{\tau}}(s,t,\gamma)+\delta_{\widehat{\tau}}(s,t,\overline{\gamma})) \\
&+ \sum_{t\in[2,t^*]} (\delta_\tau(t,1,\gamma)+\delta_\tau(t,1,\overline{\gamma})) = \mathrm{ac}^{\mathrm{co}}(\gamma), \qquad \gamma\in\Gamma_<,
\end{aligned} \tag{A.1.92}$$

$$\begin{aligned}
&\sum_{i\in E_1\cup E_3} \delta_\tau(i,\gamma) + \sum_{s\in[1,s^*],t\in[1,t^*]} \delta_{\widehat{\tau}}(s,t,\gamma) \\
&+ \sum_{t\in[2,t^*]} \delta_\tau(t,1,\gamma) = \mathrm{ac}^{\mathrm{co}}(\gamma), \qquad \gamma\in\Gamma_=,
\end{aligned} \tag{A.1.93}$$

$$\sum_{p\in[1,\mathrm{cs}^*],i\in[2,n_{\mathrm{tree}}]} (\delta_\tau(p,i,\gamma)+\delta_\tau(p,i,\overline{\gamma})) = \mathrm{ac}^{\mathrm{nc}}(\gamma), \qquad \gamma\in\Gamma_<, \tag{A.1.94}$$

$$\sum_{p\in[1,\mathrm{cs}^*],i\in[2,n_{\mathrm{tree}}]} \delta_\tau(p,i,\gamma) = \mathrm{ac}^{\mathrm{nc}}(\gamma), \qquad \gamma\in\Gamma_=. \tag{A.1.95}$$

A.1.10 Descriptor for 1-Path Connectivity

We include constraints to compute descriptor $\kappa_1(G)$ according to the definition.

variables:

A real variable $\kappa_1 \geq 0$;
$\delta_{\mathrm{dd}}(i,d,d',\mu)\in\{0,1\}$, $i\in E_1\cup E_3$, $d,d'\in[0,4]$, $\mu\in\{0,1\}$:
$\delta_{\mathrm{dd}}(i,d,d',\mu)=1 \Leftrightarrow \deg_H(u_{\mathrm{tail}}(i))=d$ and $\deg_H(u_{\mathrm{head}}(i))=d'$,
where a_i is in H if and only if $\mu=1$;
$\delta_{\mathrm{dd}}(t,1,d,d',\mu)\in\{0,1\}$, $t\in[2,t^*]$, $d,d'\in[0,4]$: $\delta_{\mathrm{dd}}(t,1,d,d',\mu)=1 \Leftrightarrow$
$\deg_H(v_{t-1,1})=d$ and $\deg_H(v_{t,1})=d'$ where e_t is in H if and only if $\mu=1$;

$\delta_{\mathrm{dd}}(p,i,d,d',\mu) \in \{0,1\}$, $p \in [1,\mathrm{cs}^*]$, $i \in [2,n_{\mathrm{tree}}]$, $d,d' \in [0,4]$: $\delta_{\mathrm{dd}}(p,i,d,d',\mu) = 1 \Leftrightarrow$
$\deg_H(u_{p,\mathrm{prt}(i)}) = d$ and $\deg_H(u_{p,i}) = d'$ for $p \leq s^*$
(or $\deg_H(v_{p-s^*,\mathrm{prt}(i)}) = d$ and $\deg_H(v_{p-s^*,i}) = d'$ for $p > s^*$),
where edge $e'_{p,i}$ or $e_{p-s^*,i}$ is in H if and only if $\mu = 1$;
$\delta_{\widehat{\mathrm{dd}}}(s,t,d,d',\mu) \in \{0,1\}$, $s \in [1,s^*]$, $t \in [1,t^*]$, $d,d' \in [0,4]$, $\mu \in \{0,1\}$:
$\delta_{\widehat{\mathrm{dd}}}(s,t,d,d',1) = 1 \Leftrightarrow \deg_H(u_{s,1}) = d$ and $\deg_H(v_{t,1}) = d'$,
where $u_{s,1}v_{t,1}$ is in H if and only if $\mu = 1$;

constraints:

$$\sum_{d,d' \in [0,4], \mu \in \{0,1\}} \delta_{\mathrm{dd}}(i,d,d',\mu) = 1, \quad i \in E_1 \cup E_3, \tag{A.1.96}$$

$$\sum_{d,d' \in [0,4], \mu \in \{0,1\}} \mu \cdot \delta_{\mathrm{dd}}(i,d,d',\mu) = a(i), \quad i \in E_1 \cup E_3, \tag{A.1.97}$$

$$\sum_{d \in [1,4], d' \in [0,4], \mu \in \{0,1\}} d \cdot \delta_{\mathrm{dd}}(i,d,d',\mu) = \deg(\mathrm{tail}(i),1), \quad i \in E_1 \cup E_3, \tag{A.1.98}$$

$$\sum_{d \in [0,4], d' \in [1,4], \mu \in \{0,1\}} d' \cdot \delta_{\mathrm{dd}}(i,d,d',\mu) = \deg(\mathrm{head}(i),1), \quad i \in E_1 \cup E_3, \tag{A.1.99}$$

$$\sum_{d,d' \in [0,4], \mu \in \{0,1\}} \delta_{\mathrm{dd}}(t,1,d,d',\mu) = 1, \qquad t \in [2,t^*], \tag{A.1.100}$$

$$\sum_{d,d' \in [0,4], \mu \in \{0,1\}} \mu \cdot \delta_{\mathrm{dd}}(t,1,d,d',\mu) = e(t), \qquad t \in [2,t^*], \tag{A.1.101}$$

$$\sum_{d \in [1,4], d' \in [0,4], \mu \in \{0,1\}} d \cdot \delta_{\mathrm{dd}}(t,1,d,d',\mu) = \deg(s^*+t-1,1), \qquad t \in [2,t^*], \tag{A.1.102}$$

$$\sum_{d \in [0,4], d' \in [1,4], \mu \in \{0,1\}} d' \cdot \delta_{\mathrm{dd}}(t,1,d,d',\mu) = \deg(s^*+t,1), \qquad t \in [2,t^*], \tag{A.1.103}$$

$$\sum_{d,d'\in[0,4],\mu\in\{0,1\}} \delta_{\mathrm{dd}}(p,i,d,d',\mu) = 1, \quad p\in[1,\mathrm{cs}^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.104}$$

$$\sum_{d,d'\in[0,4],\mu\in\{0,1\}} \mu\cdot\delta_{\mathrm{dd}}(s,i,d,d',\mu) = u(s,i), \quad s\in[1,s^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.105}$$

$$\sum_{d,d'\in[0,4],\mu\in\{0,1\}} \mu\cdot\delta_{\mathrm{dd}}(s^*+t,i,d,d',\mu) = v(t,i), \quad t\in[1,t^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.106}$$

$$\sum_{d\in[1,4],d'\in[0,4],\mu\in\{0,1\}} d\cdot\delta_{\mathrm{dd}}(p,i,d,d',\mu) = \deg(p,\mathrm{prt}(i)), \quad p\in[1,\mathrm{cs}^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.107}$$

$$\sum_{d\in[0,4],d'\in[1,4],\mu\in\{0,1\}} d'\cdot\delta_{\mathrm{dd}}(t,i,d,d',\mu) = \deg(p,i), \quad p\in[1,\mathrm{cs}^*], i\in[2,n_{\mathrm{tree}}], \tag{A.1.108}$$

$$\sum_{d,d'\in[1,4],\mu\in\{0,1\}} \delta_{\widehat{\mathrm{dd}}}(s,t,d,d',\mu) = 1, \quad s\in[1,s^*], t\in[1,t^*], \tag{A.1.109}$$

$$\sum_{d,d'\in[1,4],\mu\in\{0,1\}} \mu\cdot\delta_{\widehat{\mathrm{dd}}}(s,t,d,d',\mu) = e(s,t)+e(t,s), s\in[1,s^*], t\in[1,t^*], \tag{A.1.110}$$

$$\sum_{d\in[1,4],d'\in[0,4],\mu\in\{0,1\}} d\cdot\delta_{\widehat{\mathrm{dd}}}(s,t,d,d',\mu) = \deg(s,1), s\in[1,s^*], t\in[1,t^*], \tag{A.1.111}$$

$$\sum_{d\in[0,4],d'\in[1,4],\mu\in\{0,1\}} d'\cdot\delta_{\widehat{\mathrm{dd}}}(s,t,d,d',\mu) = \deg(s^*+t,1), s\in[1,s^*], t\in[1,t^*], \tag{A.1.112}$$

$$\begin{aligned}(1-\xi)\kappa_1 \le & \sum_{i\in E_1\cup E_3,d,d'\in[1,4]} \delta_{\mathrm{dd}}(i,d,d',1)/\sqrt{dd'} \\ & + \sum_{t\in[2,t^*],d,d'\in[1,4]} \delta_{\mathrm{dd}}(t,1,d,d',1)/\sqrt{dd'} \\ & + \sum_{\substack{p\in[1,\mathrm{cs}^*],i\in[2,n_{\mathrm{tree}}],\\ d,d'\in[1,4]}} \delta_{\mathrm{dd}}(p,i,d,d',1)/\sqrt{dd'} \\ & + \sum_{\substack{s\in[1,s^*],t\in[1,t^*],\\ d,d'\in[1,4]}} \delta_{\widehat{\mathrm{dd}}}(s,t,d,d',1)/\sqrt{dd'} \le (1+\xi)\kappa_1,\end{aligned} \tag{A.1.113}$$

where a tolerance ξ is set to be 0.001.

A.1.11 Constraints for Left-Heavy Trees

To reduce the number of rank-2 chemical graphs G that are isomorphic to each other, we include in $\mathcal{C}_2$ some additional constraints so that each subtree T' selected from tree S_p or T_t satisfies the following property:

> for any two siblings $u(p, j_1)$ and $u(p, j_2)$, $j_1 < j_2$ in T', the number of descendants of $u(p, j_1)$ is not smaller than that of $u(p, j_2)$.

For this, we define $\mathrm{dsn}(p, i)$ to be the number of descendants of a vertex $u_{p,i}$ (or $v_{p-s^*,i}$) in a selected graph H and $\eta(p, i) \triangleq 21|\Lambda|\mathrm{dsn}(p, i) + 20\widetilde{\alpha}(p, i) + 4\mathrm{deg}(p, i)+\widetilde{\beta}(p, i)$, $p \in [1, \mathrm{cs}^*]$, $i \in [2, n_{\mathrm{tree}}]$. We include constraints that compute the values of dsn recursively.

variables:

$\mathrm{dsn}(p, i) \in [1, n_{\mathrm{tree}}]$, $p \in [1, \mathrm{cs}^*]$, $i \in [1, n_{\mathrm{tree}}]$: the number of descendants of vertex $u_{p,i}$
in tree S_p for $p \leq s^*$ and vertex $v_{p-s^*,i}$ in tree T_{p-s^*} for $p > s^*$;

constraints:

$$\mathrm{dsn}(s, i) \geq \sum_{j \in \mathrm{Cld}(i)} \mathrm{dsn}(s, j) + u(s, i), \qquad s \in [1, s^*], i \in [1, n_{\mathrm{tree}}], \tag{A.1.114}$$

$$\mathrm{dsn}(s^* + t, i) \geq \sum_{j \in \mathrm{Cld}(i)} \mathrm{dsn}(s^* + t, j) + v(t, i), \quad t \in [s^* + 1, \mathrm{cs}^*], i \in [1, n_{\mathrm{tree}}], \tag{A.1.115}$$

$$\sum_{p \in [1, \mathrm{cs}^*]} \mathrm{dsn}(p, 1) \leq n^*, \tag{A.1.116}$$

$$\eta(p, j_1) \geq \eta(p, j_2), \qquad p \in [1, \mathrm{cs}^*], j_1, j_2 \in \mathrm{Cld}(1), j_1 < j_2, \tag{A.1.117}$$

$$\eta(p, j_1) \geq \eta(p, j_2), \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{in}}], j_1, j_2 \in \mathrm{Cld}(i), \; j_1 < j_2, \text{ for } d_{\max} = 3, \tag{A.1.118}$$

$$\eta(p, j_1) \geq \eta(p, j_2) \geq \eta(p, j_3), \qquad p \in [1, \mathrm{cs}^*], i \in [2, n_{\mathrm{in}}], j_1, j_2, j_3 \in \mathrm{Cld}(i), \; j_1 < j_2 < j_3, \text{ for } d_{\max} = 4. \tag{A.1.119}$$

Appendix B Appendix for Chapter 7

B.1 An LP formulation for Adjustive Linear Regression

We formulate a linear programming problem LP($\mathcal{X}, \lambda$) to the adjustive linear regression ALR($\mathcal{X}, \lambda$).

LP($\mathcal{X}, \lambda$):

constants:

- A set $\mathcal{X} = \{x_i \in \mathbb{R}^K \mid i \in [1, m]\}$ of feature vectors and a set $A = \{a_i \in \mathbb{R} \mid i \in [1, m]\}$ of observed values. Assume that each of the sets $X_j = \{x_i(j) \mid i \in [1, m]\}$, $j \in [1, K]$ and A is standardized;
- A positive real $\lambda \in \mathbb{R}$: a coefficient for the penalty term;

variables:

- Nonnegative reals $c_q(0) \in \mathbb{R}, q \in [0, 2]$;
- Nonnegative vectors $w_q \in \mathbb{R}^K, q \in [0, 2]$ and a real $b \in \mathbb{R}$;
- Nonnegative real $\overline{b} \in \mathbb{R}$;
- Nonnegative reals $\Delta_i \geq 0, i \in [1, m]$;

constraints:

$$c_0(0) + c_1(0) + c_2(0) = 1, \tag{B.1.1}$$

$$\begin{aligned} &\Delta_i \geq c_0(0)a_i + c_1(0)a_i^2 + c_2(0)(1 - (a_i - 1)^2) \\ &- \sum_{j \in I^+} [w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1 - (x_i(j) - 1)^2)] \\ &+ \sum_{j \in I^-} [w_0(j)x_i(j) + w_1(j)x_i(j)^2 + w_2(j)(1 - (x_i(j) - 1)^2)] - b \geq -\Delta_i, \quad i \in [1, m], \end{aligned} \tag{B.1.2}$$

$$\overline{b} \geq b \geq -\overline{b}, \tag{B.1.3}$$

objective function:

$$\text{Minimize} \quad \frac{1}{2m} \sum_{i \in [1,m]} \Delta_i + \lambda \sum_{q \in [0,2], j \in [1,K]} w_q(j) + \lambda \bar{b}.$$

We see that the numbers of variables and constraints in the linear program $\mathrm{LP}(\mathcal{X}, \lambda)$ are both $O(m + K)$.

Let $w_q^*(j), q \in [0,2], j \in [1,K]$ and b^* denote the values of variables $w_q(j), q \in [0,2], j \in [1,K]$ and b in an optimal solution to linear program $\mathrm{LP}(\mathcal{X}, \lambda)$, respectively. Let K' denote the number of descriptors $j \in [1,K]$ with $w_0^*(j) > 0$ and $I_{K'}$ denote the set of $j \in [1,K]$ with $w_0^*(j) > 0$. Then we obtain an optimal solution to the adjustive linear regression by setting
$w^*(j) := w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^+ \cap I_{K'}$,
$w^*(j) := -w_0^*(j)/(w_0^*(j) + w_1^*(j) + w_2^*(j)), j \in I^- \cap I_{K'}$, and
$c_q^*(j) := w_q^*(j)/w^*(j), q \in [1,2], j \in I_{K'}$.

B.2 A Procedure for Constructing a Prediction Function with ANNs

For each of the properties, we first select a set Λ of chemical elements and then collect a data set D_π on chemical graphs over the set Λ of chemical elements.

For each chemical property π, we conducted a preliminary experiment to choose the following: a subset S_π of the original set of K descriptors; an architecture A_π with at most five hidden layers; a nonnegative real $\rho_\pi^{\mathrm{stp}} \leq 1$; and an integer $\mathrm{ite}_\pi^{\mathrm{stp}}$, where we will use ρ_π^{stp} and $\mathrm{ite}_\pi^{\mathrm{stp}}$ as parameters to execute an early stopping in constructing a prediction function with a training data set. Let f_π denote the feature vector that consists of the descriptors in the set S_π.

For each property π, we conducted ten 5-fold cross-validations. In a 5-fold cross-validation, we construct five prediction functions $\eta^{(k)}, k \in [1,5]$ as follows. Partition a data set D_π into five subsets $D_\pi^{(k)}$, $k \in [1,5]$ randomly. For each $k \in [1,5]$, use the set $D_{\mathrm{train}} := D_\pi \setminus D_\pi^{(k)}$ as a training set and construct an ANN on the selected architecture A_π with the feature vector f_π by the MLPRegressor of `scikit-learn`, where we stop updating weights/biases on A_π during an execution of the iterative algorithm when the coefficient of determination $\mathrm{R}^2(\eta, D_{\mathrm{train}})$ of the prediction function η by the current weights/biases exceeds ρ_π^{stp} (where we terminate the execution when the number of iterations exceeds $1.5 \times \mathrm{ite}_\pi^{\mathrm{stp}}$ even if $\mathrm{R}^2(\eta, D_{\mathrm{train}})$ does not reach ρ_π^{stp}). Set $\eta^{(k)}$ to be the prediction function η by the resulting weights/biases on A_π. We evaluate the performance of the prediction function $\eta^{(k)}$ with the coefficient $\mathrm{R}^2(\eta^{(k)}, D_{\mathrm{test}})$ of determination for the test set $D_{\mathrm{test}} := D_\pi^{(k)}$.

List of the Author's Work

Publications Related to the Dissertation

[1] J. Zhu, C. Wang, A. Shurbevski, H. Nagamochi, and T. Akutsu. A novel method for inference of chemical compounds of cycle index two with desired properties based on artificial neural networks and integer programming. *Algorithms*, 13(5):124, 2020.

[2] J. Zhu, N. A. Azam, F. Zhang, A. Shurbevski, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A novel method for inferring chemical compounds with prescribed topological substructures based on integer programming. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(6):3233–3245, 2021.

[3] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An improved integer programming formulation for inferring chemical compounds with prescribed topological structures. In *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part I 34*, pages 197–209. Springer, 2021.

[4] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A method for molecular design based on linear regression and integer programming. In *Proceedings of the 12th International Conference on Bioscience, Biochemistry and Bioinformatics*, pages 21–28, 2022.

[5] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on linear regression and integer programming. *Frontiers in Bioscience-Landmark*, 27(6):188, 2022.

[6] J. Zhu, K. Haraguchi, H. Nagamochi, and T. Akutsu. Adjustive linear regression and its application to the inverse QSAR. In *Proceedings of the 15th International Joint Conference on Biomedical Engineering Systems and Technologies - BIOINFORMATICS*, pages 144–151. INSTICC, SciTePress, 2022.

[7] J. Zhu, N. A. Azam, S. Cao, R. Ido, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. Molecular design based on integer programming and quadratic descriptors in a two-layered model. *arXiv preprint arXiv:2209.13527*, 2022. (submitted to BMC Bioinformatics)

[8] N. A. Azam, J. Zhu, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. Molecular design based on artificial neural networks, integer programming and grid neighbor search. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 360–363. IEEE, 2021.

[9] N. A. Azam, J. Zhu, R. Ido, H. Nagamochi, and T. Akutsu. Experimental results of a dynamic programming algorithm for generating chemical isomers based on frequency vectors. In *Fourth International Workshop on Enumeration Problems and Applications (WEPA)*, pages 7–10, Israel, WEPA2020 15, December 2020. Israel.

[10] N. A. Azam, J. Zhu, Y. Sun, Y. Shi, A. Shurbevski, L. Zhao, H. Nagamochi, and T. Akutsu. A novel method for inference of acyclic chemical compounds with bounded branch-height based on artificial neural networks and integer programming. *Algorithms for Molecular Biology*, 16:1–39, 2021.

[11] R. Ido, S. Cao, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A method for inferring polymers based on linear regression and integer programming. *arXiv preprint arXiv:2109.02628*, 2021. (submitted to IEEE/ACM Transactions on Computational Biology and Bioinformatics)

[12] K. Tanaka, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on decision tree and integer programming. In *Intelligent Computing Theories and Application: 17th International Conference, ICIC 2021, Shenzhen, China, August 12–15, 2021, Proceedings, Part II*, pages 628–644. Springer, 2021.

[13] Y. Shi, J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. An inverse QSAR method based on a two-layered model and integer programming. *International Journal of Molecular Sciences*, 22(6): 2847, 2021.

[14] F. Zhang, J. Zhu, R. Chiewvanichakorn, A. Shurbevski, H. Nagamochi, and T. Akutsu. A new integer linear programming formulation to the inverse QSAR/QSPR for acyclic chemical compounds using skeleton trees. In

Trends in Artificial Intelligence Theory and Applications. Artificial Intelligence Practices: 33rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2020, Kitakyushu, Japan, September 22-25, 2020, Proceedings, pages 433–444. Springer, 2020.

[15] F. Zhang, J. Zhu, R. Chiewvanichakorn, A. Shurbevski, H. Nagamochi, and T. Akutsu. A new approach to the design of acyclic chemical compounds using skeleton trees and integer linear programming. *Applied Intelligence*, 52(15):17058–17072, 2022.