




A Novel Method for the Inverse QSAR/QSPR based on Artificial Neural Networks and Mixed Integer Linear Programming with Guaranteed Admissibility

Naveed Ahmed Azam¹^a, Rachaya Chiewvanichakorn¹, Fan Zhang¹, Aleksandar Shurbevski¹^b, Hiroshi Nagamochi¹ and Tatsuya Akutsu²^c

¹Department of Applied Mathematics and Physics, Kyoto University, Kyoto, Japan

²Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji-city, Japan

Keywords: QSAR/QSPR, Artificial Neural Networks, Mixed Integer Programming, Feature Vectors, Chemical Graphs.


Abstract: Inverse QSAR/QSPR is a well-known approach for computer-aided drug design. In this study, we propose a novel method for inverse QSAR/QSPR using artificial neural network (ANNs) and mixed integer linear programming. In this method, we introduce a feature function f that converts each chemical compound G into a vector $f(G)$ of several descriptors of G . Next, given a set of chemical compounds along with their chemical properties, we construct a prediction function ψ with an ANN so that $\psi(f(G))$ takes a value nearly equal to a given chemical property for many chemical compounds G in the set. Then, given a target value y^* of the chemical property, we conversely infer a chemical structure G^* having the desired property y^* in the following way. We formulate the problem of finding a vector x^* such that (i) $\psi(x^*) = y^*$ and (ii) there exists a chemical compound G^* such that $f(G^*) = x^*$ (if one exists over all vectors x^* in (i)) as a mixed integer linear programming problem (MILP). In an existing method for the inverse QSAR/QSPR, the second condition (ii) was not guaranteed. For acyclic chemical compounds and some chemical properties such as heat of formation, boiling point, and retention time, we conducted computational experiments.


1 INTRODUCTION


Drug design is one of the major goals of bioinformatics and chemo-informatics. Many computational methods have been proposed for computer-aided drug design. Forward QSAR/QSPR (quantitative structure-activity and structure-property relationships) is the problem of computing a numerical relationship between a chemical structure and a biological activity from a given set of known data and using this relationship to predict the activity of unknown structures, whereas inverse QSAR/QSPR, given such a relationship, attempts to find a chemical compound that has a desired activity. Therefore, inverse QSAR/QSPR (quantitative structure-activity and structure-property relationships) is one of the major approaches for the purpose (Miyao et al., 2016; Skvortsova et al., 1993). In this approach, desired chemical activities and/or properties are speci-

fied along with some constraints and then chemical compounds satisfying these conditions are inferred, where chemical compounds are usually represented as chemical graphs. In many cases, it is also formulated as an optimization problem to find a chemical graph maximizing (or minimizing) an objective function under various constraints, where objective functions reflect certain chemical activities or properties. Objective functions are often derived from a set of training data consisting of known molecules and their activities/properties using statistical learning methods.

Recently, novel approaches have been proposed for inverse QSAR/QSPR using Artificial Neural Network (ANN) and deep learning technologies. For example, recurrent neural networks (Segler et al., 2017; Yang et al., 2017), variational autoencoders (Gómez-Bombarelli et al., 2018), and grammar variational autoencoders (Kusner et al., 2017) have been applied. In these approaches, neural networks are trained using known compound/activity data and then novel chemical graphs are obtained by solving a kind of inverse problems on neural networks, that is, by looking for

^a  <https://orcid.org/0000-0002-7941-3419>

^b  <https://orcid.org/0000-0001-9224-6929>

^c  <https://orcid.org/0000-0001-9763-797X>

input values for a trained neural network that will produce a desired output value. Usually, these inverse problems are solved using various statistical methods. However, the solutions found by statistical methods are generally not optimal. In order to cope with this issue, novel mixed integer linear programming (MILP)-based methods have been proposed (Akutsu and Nagamochi, 2019) for ANNs with rectified linear unit (ReLU) functions and sigmoid functions. In their methods, activation functions on neurons are efficiently encoded as piece-wise linear functions so as to represent ReLU functions exactly and sigmoid functions approximately.

Chiewvanichakorn et al. (2020) recently combined these previous approaches; efficient enumeration of tree-like graphs (Fujiwara et al., 2008), and MILP-based formulation of the inverse problem on ANNs (Akutsu and Nagamochi, 2019). This combined framework for QSAR/QSPR mainly consists of two phases. The first phase solves (I) PREDICTION PROBLEM, where each chemical compound G is transformed into a feature vector $f(G)$ and an ANN \mathcal{N} is trained from existing chemical compounds and their values $a(G)$ on a chemical property π to obtain a prediction function $\psi_{\mathcal{N}}$ so that $a(G)$ is predicted as $\psi_{\mathcal{N}}(f(G))$. The second phase solves (II) INVERSE PROBLEM, where (II-a) given a target value y^* of the chemical property π , a feature vector x^* is inferred from the trained ANN \mathcal{N} so that $\psi_{\mathcal{N}}(x^*)$ is close to y^* and (II-b) then a set of chemical structures G^* such that $f(G^*) = x^*$ is enumerated.

We call an inferred feature vector *admissible* if it admits a chemical structure whose feature vector is equal to the inferred feature vector. However the above-mentioned previous method (Chiewvanichakorn et al., 2020) has no guarantee that every inferred vector in (II-a) is admissible. In their experiment, the probability that an inferred vector is admissible was ranging from 20% to 50% on average. In this paper, we improve the method so that (i) every feature vector x^* inferred from a trained ANN \mathcal{N} in (II-a) becomes admissible; and (ii) no chemical structure exists for a given target value when no feature vector is inferred from the ANN \mathcal{N} . Since the new idea can be applied to any inference problem of a trained ANN, we describe the method in a more general setting (see Section 2.1). We conducted some preliminary computational experiments to infer acyclic chemical compounds on several chemical properties.

2 PRELIMINARY

Let \mathbb{R} and \mathbb{Z} denote the sets of reals and non-negative integers, respectively. For two integers a and b , let $[a, b]$ denote the set of integers i with $a \leq i \leq b$.

2.1 Inference of Structures based on ANNs and MILPs

This section reviews the previous method for the inverse QSAR/QSPR (Chiewvanichakorn et al., 2020), and then shows how to improve the theoretical framework of the method so as to increase the chance of finding a larger number of solutions to the inverse QSAR/QSPR. We here describe their and our methods in a general setting where the object to be inferred with an ANN is not necessarily chemical compounds. Fig. 1 illustrates the general setting.

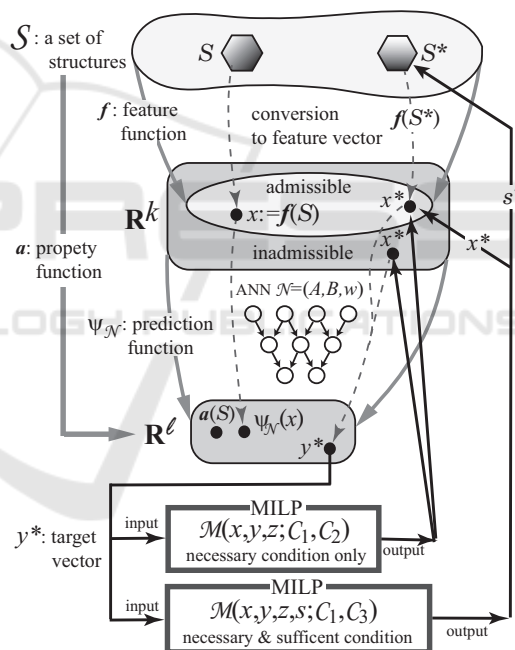


Figure 1: An illustration of a property function a , a feature function f , a prediction function $\psi_{\mathcal{N}}$ and MILPs $\mathcal{M}(x, y, z, s; C_1, C_2)$ and $\mathcal{M}(x, y, z, s; C_1, C_3)$.

We start to formulate a prediction problem and its inverse problem. Let \mathcal{S} be a set of structures, and $a : \mathcal{S} \rightarrow \mathbb{R}^{\ell}$ be a property function for an integer $\ell \geq 1$. We assume that the function a is not explicitly given and we are given the value $a(S)$ only for structures S in some subset S' of \mathcal{S} .

(I) PREDICTION PROBLEM: Given a structure $S \in \mathcal{S}$ (especially when $S \notin S'$), predict the value $a(S)$ based on the information $\{(S, a(S)) \mid S \in S'\}$;

(II) INVERSE PROBLEM: Given a vector $y \in \mathbb{R}^\ell$, generate a structure $S \in \mathcal{S}$ such that $a(S) = y$.

For Problem (I), we use artificial neural networks (ANN) in a standard way. We first define a feature space \mathbb{R}^k for an integer $k \geq 1$ and a feature function $f: \mathcal{S} \rightarrow \mathbb{R}^k$ that converts each structure $S \in \mathcal{S}$ into a vector $f(S) \in \mathbb{R}^k$, called the *feature vector* of S . We next construct an ANN $\mathcal{N} = (A, B, w)$, which consists of an architecture A , a set B of activation functions and a vector $w \in \mathbb{R}^h$ that consists of arc weights and node bias for some integer $h \geq 1$. An ANN $\mathcal{N} = (A, B, w)$ defines a prediction function $\Psi_{\mathcal{N}}: \mathbb{R}^k \rightarrow \mathbb{R}^\ell$. For some subset \mathcal{S}'' of \mathcal{S}' , we try to choose an ANN $\mathcal{N} = (A, B, w)$ so that the prediction $\Psi_{\mathcal{N}}(f(S))$ of each structure $S \in \mathcal{S}''$ is close to the value $a(S)$. There have been developed many learning algorithms that choose such an ANN.

For Problem (II), Akutsu and Nagamochi (Akutsu and Nagamochi, 2019) proposed a formulation based on MILP assuming that an ANN \mathcal{N} and its prediction function $\Psi_{\mathcal{N}}$ are given. Let $\mathcal{N} = (A, B, w)$ be an ANN such that all activation functions in B are continuous piece-wise linear functions, n_A denote the number of nodes in the architecture A in \mathcal{N} , and n_B denote the total number of break-points over all activation functions in B . (We approximate activation functions that are not piece-wise linear with piece-wise linear functions, where the approximation may cause some numerical errors in computing the predicted values with Ψ . See (Akutsu and Nagamochi, 2019) for details.) Then it is known (Akutsu and Nagamochi, 2019) that there is an MILP $\mathcal{M}(x, y, z; C_1)$ without an objective function that consists of variable vectors $x \in \mathbb{R}^k$, $y \in \mathbb{R}^\ell$ and $z \in \mathbb{R}^p$ and a set C_1 of integer restrictions and linear equalities on these variables such that

- $p = O(n_A + n_B)$ and $|C_1| = O(n_A + n_B)$;
- $\Psi_{\mathcal{N}}(x) = y$ if and only if there is a vector $z \in \mathbb{R}^p$ such that (x, y, z) is feasible to the system $\mathcal{M}(x, y, z; C_1)$.

When a target value $y^* \in \mathbb{R}^\ell$ is specified for the variable vector $y \in \mathbb{R}^\ell$, we can find a vector $x^* \in \mathbb{R}^k$ such that $\Psi_{\mathcal{N}}(x^*) = y^*$ by solving MILP $\mathcal{M}(x, y, z; C_1)$ with $y = y^*$. However, in general, there may be no structure $S^* \in \mathcal{S}$ such that $f(S^*) = x^*$, since no information on the structure of \mathcal{S} is included in the formulation of MILP $\mathcal{M}(x, y, z; C_1)$. We call a vector $x \in \mathbb{R}^k$ *admissible* if it admits a structure $S^* \in \mathcal{S}$ such that $f(S^*) = x^*$. To increase the chance to have an admissible vector $x \in \mathbb{R}^k$, we impose a set C_2 of linear constraints on $x \in \mathbb{R}^k$ that are "necessary conditions" for a vector $x \in \mathbb{R}^k$ to be admissible. Let $\mathcal{M}(x, y, z; C_1, C_2)$ denote the MILP obtained from $\mathcal{M}(x, y, z; C_1)$ by adding the set C_2 of linear constraints

on $x \in \mathbb{R}^k$. The previous method (Chiewvanichakorn et al., 2020) for the inverse QSAR/QSPR followed this idea. However there was no guarantee that every inferred vector x^* by this method is admissible

In this paper, we impose to MILP $\mathcal{M}(x, y, z; C_1)$ a new set C_3 of constraints so that, for a given target value y^* , (i) every feature vector x^* inferred from the new system becomes admissible; and (ii) no structure exists in \mathcal{S} when the new system has no feasible solution. For this, we introduce a new variable vector $s \in \mathbb{R}^h$ for some integer h , and prepare a set C_3 of linear constraints on $x \in \mathbb{R}^k$ and $s \in \mathbb{R}^h$ to obtain an MILP $\mathcal{M}(x, s; C_3)$ that consists of variable vectors $x \in \mathbb{R}^k$ and $s \in \mathbb{R}^h$ and a set C_3 of integer restrictions and linear inequalities such that

- $x \in \mathbb{R}^k$ is admissible if and only if there is a vector $s \in \mathbb{R}^h$ such that (x, s) is feasible to MILP $\mathcal{M}(x, s; C_3)$; and
- $s \in \mathbb{R}^h$ forms a structure $S \in \mathcal{S}$ if and only if there is a vector $x \in \mathbb{R}^k$ such that (x, s) is feasible to MILP $\mathcal{M}(x, s; C_3)$.

That is, the constraint set C_3 is "necessary and sufficient condition" for a vector $x \in \mathbb{R}^k$ to be admissible. Now we combine the two systems $\mathcal{M}(x, y, z; C_1)$ and $\mathcal{M}(x, s; C_3)$ to obtain an MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$ that consists of variable vectors $x \in \mathbb{R}^k$, $y \in \mathbb{R}^\ell$, $z \in \mathbb{R}^p$ and $s \in \mathbb{R}^h$ and a set C_1 of constraints on x , y and z and a set C_3 of constraints on x and s . Hence given a target vector $y^* \in \mathbb{R}^\ell$, we solve MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$ to find a feasible vector (x^*, y^*, z^*, s^*) , where the vector $x^* \in \mathbb{R}^k$ satisfies $\Psi_{\mathcal{N}}(x^*) = y^*$, and the vector $s^* \in \mathbb{R}^h$ forms a structure $S^* \in \mathcal{S}$ such that $f(S^*) = x^*$. Based on the new idea, we design a method for the inverse QSAR/QSPR in Section 3. Our $\mathcal{M}(x, s; C_3)$ in the method is presented in Section 4.

2.2 Modeling of Chemical Compounds

Graphs. Let $H = (V, E)$ be a graph with a set V of vertices and a set E of edges. For a vertex $v \in V$, the set of neighbors of v in H is denoted by $N_H(v)$, and the *degree* $\deg_H(v)$ of v is defined to be $|N_H(v)|$. The length of a path is defined to be the number of edges in the path. The *distance* $\text{dist}_H(u, v)$ between two vertices $u, v \in V$ is defined to be the minimum length of a path connecting u and v in H . The *diameter* $\text{dia}(H)$ of H is defined to be the maximum distance between two vertices in H . The *sum-distance* $\text{smdt}(H)$ of H is defined to be the sum of distances over all vertex pairs.

Chemical Graphs. 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. Let Λ be a set of labels each of which represents a chemical element such as C (carbon), O (oxygen), N (nitrogen), S (sulfur) and so on, where we assume that Λ does not contain H (hydrogen). Let $\text{mass}(a)$ and $\text{val}(a)$ denote the mass and valence of a chemical element $a \in \Lambda$, respectively.

A *chemical graph* in a hydrogen-suppressed model is defined to be a tuple $G = (H, \alpha, \beta)$ of a graph $H = (V, E)$, a function $\alpha : V \rightarrow \Lambda$ (called an *atom-function*) and a function $\beta : E \rightarrow \{1, 2, 3\}$ (called a *bond-function*) such that

(i) H is connected; and

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

Figure 2 illustrates an example of a chemical graph $G = (H, \alpha, \beta)$, where the structure of the vector $f(G)$ is explained in Section 2.3.

We introduce a total order $<$ over the elements in Λ according to their mass values; i.e., we write $a < b$ for chemical elements $a, b \in \Lambda$ with $\text{mass}(a) < \text{mass}(b)$. Choose a set $\Gamma_<$ of tuples $\gamma = (a, b, k) \in \Lambda \times \Lambda \times [1, 3]$ such that $a < b$. For a tuple $\gamma = (a, b, k) \in \Lambda \times \Lambda \times [1, 3]$, let $\bar{\gamma}$ denote the tuple (b, a, k) . Set $\Gamma_> = \{\bar{\gamma} \mid \gamma \in \Gamma_<\}$, $\Gamma_ = \{(a, a, k) \mid a \in \Lambda, k \in [1, 3]\}$ and $\Gamma = \Gamma_< \cup \Gamma_ =$. A pair of two atoms a and b joined with a bond of multiplicity k is denoted by a tuple $\gamma = (a, b, k) \in \Gamma$.

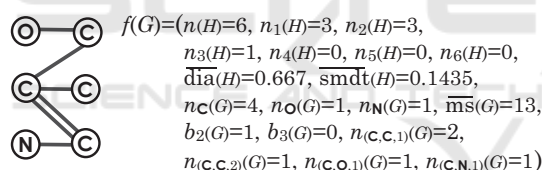


Figure 2: An example of a chemical graph $G = (H = (V, E), \alpha, \beta)$ and its feature vector $f(G)$.

2.3 Descriptors in Feature Vectors

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 $G = (H = (V, E), \alpha, \beta)$, we define a *feature vector* $f(G)$ that consists of the following eight kinds of descriptors: $n(H)$: the number of vertices; $n_d(H)$ ($d \in [1, 6]$): the number of vertices of degree d in H ; $\text{dia}(H)$: the diameter of H divided by $|V|$; $\text{smdt}(H)$: the sum of distances of H divided by $|V|^3$; $n_a(G)$ ($a \in \Lambda$): the number of vertices with label $a \in \Lambda$; $\text{ms}(G)$: the average mass of atoms in G ; $b_i(G)$ ($i = 2, 3$): the number of double and triple bonds; $n_\gamma(G)$ ($\gamma = (a, b, k) \in \Gamma$): the number of label pairs $\{a, b\}$ with multiplicity k ; Figure 2 illustrates an example of a feature vector $f(G)$.

3 A METHOD FOR INFERRING CHEMICAL GRAPHS

Analogously with the previous method (Chiewvanichakorn et al., 2020), our new method for the inverse QSAR/QSPR also solves two problems: (I) PREDICTION PROBLEM and (II) INVERSE PROBLEM introduced in Section 2.1. For a specified chemical property such as boiling point, we denote by $a(G)$ the value of the specified chemical property for a given chemical compound G , which is represented by a chemical graph $G = (H, \alpha, \beta)$. We solve Problem (I) for the inverse QSAR/QSPR as follows.

Phase 1.

1. Prepare a data set $D = \{(G_i, a(G_i)) \mid i = 1, 2, \dots, p\}$ for a specified chemical property, where G_i is a chemical graph. Set reals $\underline{a}, \bar{a} \in \mathbb{R}$ such that $\underline{a} \leq \min\{a(G_i) \mid i = 1, 2, \dots, p\}$ and $\bar{a} \geq \max\{a(G_i) \mid i = 1, 2, \dots, p\}$.
2. Set a graph class \mathcal{G} to be a set of chemical graphs such that $\mathcal{G} \supseteq \{G_i \mid i = 1, 2, \dots, p\}$. Introduce a function $f : \mathcal{G} \rightarrow \mathbb{R}^K$ for a positive integer K , as we have observed in Section 2.3. 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 .
3. Select an architecture A and a set B of activation functions to construct an ANN \mathcal{N} that, given a vector in \mathbb{R}^K , returns a real in the range $[\underline{a}, \bar{a}]$. Using the data set D as training data, choose a vector w of arc weights and node biases of \mathcal{N} to construct a prediction function $\psi_{\mathcal{N}}$ with $\mathcal{N} = (A, B, w)$ so that $\psi_{\mathcal{N}}(f(G))$ takes a value nearly equal to $a(G)$ for many chemical graphs in D .

In this paper, we solve Problem (II) for the inverse QSAR/QSPR by treating the following inference problems.

(II-a) Inference of Vectors

Input: A real $y^* \in [\underline{a}, \bar{a}]$.

Output: Vectors $x^* \in \mathbb{R}^K$ and $s^* \in \mathbb{R}^h$ such that $\psi_{\mathcal{N}}(x^*) = y^*$ and s^* forms a chemical graph $G^* \in \mathcal{G}$ with $f(G^*) = x^*$.

(II-b) Inference of Graphs

Input: A vector $x^* \in \mathbb{R}^K$.

Output: All graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$.

In the previous method (Chiewvanichakorn et al., 2020), they tried to find a vector $x^* \in \mathbb{R}^K$ such that $\psi_{\mathcal{N}}(x^*) = y^*$ in (II-a), which may not be admissible. In this paper, we include into (II-a) another vector s that represents a chemical graph so that an inferred vector x^* becomes admissible. To treat Problem (II-a), we assume that an ANN \mathcal{N} with a piecewise-linear

activation function (such as a ReLU) is given. As observed in Section 2.1, there is an MILP $\mathcal{M}(x, y, z; C_1)$ that consists of variable vectors $x \in \mathbb{R}^K$, $y \in \mathbb{R}$ and $z \in \mathbb{R}^p$ for some integer p and a set C_1 of constraints on these variables such that $\Psi_{\mathcal{N}}(x^*) = y^*$ if and only if there is a vector (x^*, y^*, z^*) feasible to $\mathcal{M}(x, y, z; C_1)$. We here show how to construct an MILP $\mathcal{M}(x, s; C_3)$ that consists of variable vectors $x \in \mathbb{R}^K$ and $s \in \mathbb{R}^h$ and a constraint set C_3 such that $x^* \in \mathbb{R}^K$ is admissible and $s^* \in \mathbb{R}^h$ forms a chemical graph $G^* \in \mathcal{G}$ if and only if (x^*, s^*) is feasible to MILP $\mathcal{M}(x, s; C_3)$. A more precise description of MILP $\mathcal{M}(x, s; C_3)$ will be given in Section 4. Given a value $y^* \in \mathbb{R}$, we can find an admissible vector $x^* \in \mathbb{R}^K$ such that $\Psi_{\mathcal{N}}(x^*) = y^*$ (if one exists) by solving the MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$.

We design an algorithm to Problem (II-b) based on the branch-and-bound method (see (Fujiwara et al., 2008) for enumerating acyclic chemical compounds).

Phase 2.

4. Formulate Problem (II-a) as an MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$ based on \mathcal{N} . Find a set F^* of admissible vectors $x^* \in \mathbb{R}^K$ such that $(1 - \epsilon)y^* \leq \Psi_{\mathcal{N}}(x^*) \leq (1 + \epsilon)y^*$ for a tolerance ϵ set to be a small positive real. A vector $s^* \in \mathbb{R}^h$ obtained together with x^* forms a chemical graph $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$.
5. To solve Problem (II-b), enumerate all graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$ for each admissible vector $x^* \in F^*$.

In our experiment in Section 5, we further impose another set C_4 of constraints on the descriptors in vector x not to change the feasibility of MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$ but to try to reduce the search space required when we solve the MILP with some solver. Also we fix the numbers of vertices and vertices of degree 1 and the diameter to specified integers n^* , n_1^* and dia^* , respectively in Section 5.

In Step 5 of the previous method (Chiewvanichakorn et al., 2020), some vector $x^* \in F^*$ may not be admissible; i.e., there may not exist a graph $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$. As discussed in Section 2.1, we formulate an MILP in Phase 2 so that every vector $x^* \in F^*$ obtained in Step 4 is admissible.

4 MILPs FOR REPRESENTING ACYCLIC CHEMICAL GRAPHS

In this section, we propose a formulation of the MILP $\mathcal{M}(x, s; C_3)$ introduced in Section 2.1. For space limitation, we show a sketch of the MILP. In our MILP, we construct a tree H with n vertices and diameter

dia^* by selecting $n - 1$ pairs of vertices in an $n \times n$ adjacency matrix (or a complete graph K_n with n vertices). For this, we introduce the following variables and constraints:

- $e(i, j) \in \{0, 1\}$, $1 \leq i, j \leq n$ ($e(i, j) = 1$ implies that v_i is the parent of v_j in a tree H rooted at v_1);

$$\begin{aligned} e(j, i) &= 0, & 1 \leq i \leq j \leq n, \\ e(i, i+1) &= 1, & i \in [1, \text{dia}^*], \\ e(\text{dia}^* + 1, i) &= 0, & i \in [\text{dia}^* + 2, n], \\ \sum_{1 \leq i < j} e(i, j) &= 1, & j \in [2, n]. \end{aligned}$$

For a tree H , we select functions α and β to define a chemical graph $G = (H, \alpha, \beta)$ with the following variables and constraints:

- $\tilde{\alpha}(i) \in \{[a] \mid a \in \Lambda\}$, $i \in [1, n]$ ($\tilde{\alpha}(i) = [a]$ implies $\alpha(v_i) = a$ in G);

- $\delta_\alpha(i, a) \in \{0, 1\}$, $i \in [1, n]$, $a \in \Lambda$ ($\delta_\alpha(i, a) = 1$ implies $\alpha(v_i) = a$ in G);

- $\tilde{\beta}(i, j) \in [0, 3]$, $1 \leq i < j \leq n$ ($\tilde{\beta}(i, j) = k$ implies $\beta(v_i v_j) = k$ for $k \geq 1$ and $v_i v_j \notin E$ for $k = 0$);

$$\begin{aligned} \sum_{a \in \Lambda} \delta_\alpha(i, a) &= 1, & i \in [1, n], \\ \sum_{a \in \Lambda} [a] \cdot \delta_\alpha(i, a) &= \tilde{\alpha}(i), & i \in [1, n], \\ e(i, j) &\leq \tilde{\beta}(i, j) \leq 3e(i, j), & 1 \leq i < j \leq n, \\ \sum_{h < i} \tilde{\beta}(h, i) + \sum_{i < j} \tilde{\beta}(i, j) &\leq \sum_{a \in \Lambda} \text{val}(a) \cdot \delta_\alpha(i, a), & i \in [1, n]. \end{aligned}$$

Among the descriptors in our feature vector, we show how the number $n_\gamma(G)$ of tuple $\gamma \in \Gamma$ is represented in our MILP. We omit to show the linear constraints for the other descriptors. Let Γ_{fbn} denote the set of forbidden tuples; i.e., $\Gamma_{\text{fbn}} = (\Lambda \times \Lambda \times \{1, 2, 3\}) \setminus (\Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>})$. We introduce the following variables and constraints for descriptor $n_\gamma(G)$:

- $\delta_{\beta'}(i, k) \in \{0, 1\}$, $i = 2, 3, \dots, n$, $k \in [1, 3]$ (for a function $\beta' : [2, n] \rightarrow [1, 3]$, $\delta_{\beta'}(i, k) = 1$ implies that v_h is the parent of v_i and $\beta(v_h v_i) = k$ in G);

- $\delta_\tau(i, \gamma) \in \{0, 1\}$, $i \in [2, n]$, $\gamma \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}$ (for a function $\tau : \{2, 3, \dots, n\} \rightarrow \{\gamma \mid \gamma \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}\}$, $\delta_\tau(i, \gamma) = 1$ implies that v_h is the parent of v_i and $(\alpha(v_h), \alpha(v_i), \beta(v_h v_i)) = \gamma \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}$ in G);

- $n(\gamma) \in [0, n - 1]$, $\gamma \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}$ ($n(\gamma)$ represents $n_\gamma(G)$);

$$\begin{aligned} \sum_{k \in [1, 3]} \delta_{\beta'}(i, k) &\leq 1, & i \in [2, n], \\ \sum_{k \in [1, 3]} k \delta_{\beta'}(i, k) &= \sum_{h < i} \tilde{\beta}(h, i), & i \in [2, n], \end{aligned}$$

$$\begin{aligned}
 \delta_\tau(j, (a, b, k)) &\geq \delta_\alpha(i, a) + \delta_\alpha(j, b) \\
 &\quad + \delta_{\beta'}(j, k) + e(i, j) - 3, \\
 &\quad 1 \leq i < j \leq n, (a, b, k) \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}, \\
 \delta_\alpha(i, a) + \delta_\alpha(j, b) + \delta_{\beta'}(j, k) + e(i, j) &\leq 3, \\
 &\quad 1 \leq i < j \leq n, (a, b, k) \in \Gamma_{\text{bn}}, \\
 \sum_{\gamma \in \Gamma_{<} \cup \Gamma_{=} \cup \Gamma_{>}} \delta_\tau(i, \gamma) &= 1, \quad i \in [2, n], \\
 \sum_{i \in [2, n]} (\delta_\tau(i, \gamma) + \delta_\tau(i, \bar{\gamma})) &= n(\gamma), \quad \gamma \in \Gamma_{<}, \\
 \sum_{i \in [2, n]} \delta_\tau(i, \gamma) &= n(\gamma), \quad \gamma \in \Gamma_{=}.
 \end{aligned}$$

To reduce the number of graph-isomorphic solutions in this MILP, we require the indexing v_1, v_2, \dots, v_n of vertices in a tree H to be a depth-first search ordering from v_1 . This can be attained with the following variables and constraints:

- $\text{dist}(i, j) \in [0, n]$, $1 \leq i < j \leq n$ ($\text{dist}(i, j)$ represents $\text{dist}_H(v_i, v_j)$);
- $\text{dist}_{\text{lca}}(i, j)$, $1 \leq i < j \leq n$ ($\text{dist}_{\text{lca}}(i, j)$ represents $\text{dist}_H(w, v_1)$ for the least common ancestor w of v_i and v_j);

$$\begin{aligned}
 \text{dist}_{\text{lca}}(j, i) &\geq \text{dist}(1, j) - n(1 - e(j, h)), \\
 &\quad 1 \leq j < i < h \leq n.
 \end{aligned}$$

5 EXPERIMENTAL RESULTS

We implemented our method for inferring acyclic chemical graphs and executed on a PC with Intel Core i5 1.6 GHz CPU and 8GB of RAM running under the Mac OS operating system version 10.14.4. We select five chemical properties: heat of atomization (HA), heat of formation (HF), boiling point (BP), octanol/water partition coefficient (K_{ow}) and retention time (RT).

Results on Phase 1. In Step 1, we used a data set D of acyclic chemical graphs in (Roy and Saha, 2003) (resp., (Jalali-Heravi and Fatemi, 2001)) for HA, HF, BP and K_{ow} (resp., RT). Table 1 shows the size and range of data sets that we prepared for each chemical property, where we denote the following:

π : one of the chemical properties HA, HF, BP, K_{ow} and RT;

$|D|$: the size of data set D for a chemical property π ;

$[\underline{n}, \bar{n}]$: the minimum and maximum number of vertices in H over data set D .

The set Γ of all tuples γ and the number $K = |\Lambda| + |\Gamma| + 12$ of descriptors in $f(G)$ in D are $\Gamma = \{(C, 0, 1), (C, S, 1), (C, C, 1), (C, C, 2)\}$ and

$K = 19$ for HA, HF, BP and K_{ow} and $\Gamma = \{(C, 0, 1), (C, 0, 2), (C, C, 1), (C, C, 2)\}$ and $K = 18$ for RT. Some of the above parameters may be denoted with a suffix π such as a_π to indicate the underlying property $\pi \in \{\text{HA}, \text{HF}, \text{BP}, K_{\text{ow}}, \text{RT}\}$.

Table 1: The results on Steps 1 to 3 in Phase 1.

π	$ D $	Λ	$[\underline{n}, \bar{n}]$	L-time	R^2
HA	128	C, 0, S	[2, 11]	11.240	0.999
HF	88	C, 0, S	[2, 16]	1.437	0.985
BP	131	C, 0, S	[2, 16]	7.335	0.974
K_{ow}	62	C, 0, S	[2, 16]	0.534	0.969
RT	39	C, 0	[11, 16]	9.529	0.890

In Step 2, we set a graph class \mathcal{G} to be the set of all acyclic chemical graphs over the sets Λ and Γ in Table 1.

In Step 3, we used `scikit-learn` version 0.21.3 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 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 into five subsets D_i , $i = 1, 2, 3, 4, 5$ randomly, where $D \setminus D_i$ is used for a training set and D_i is used for a test set in five trials $i = 1, 2, 3, 4, 5$. Table 1 shows the results on Step 3, where

L-time: the average time (sec) to construct ANNs for each trial;

R^2 : the average 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 $\mathcal{M}(x, y, z; C_1)$ in the second phase.

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

Step 4. In this step, we solve the MILP $\mathcal{M}(x, y, z, s; C_1, C_3)$ formulated based on the ANN \mathcal{N} obtained in Phase 1. In our computational experiment, we further impose a set C_4 of several other necessary conditions on the descriptors of our feature vector $x \in \mathbb{R}^K$ so as to reduce the search space for solving this MILP with a solver. Thus we solve an MILP $\mathcal{M}(x, y, z, s; C_1, C_3, C_4)$. We here omit describing the additional constraints in C_4 . To solve an MILP in Step 4, we use CPLEX (ILOG CPLEX version 12.8).

In Step 4, we conducted the following two options in our experiments.

(a) Choose some target $y^* \in [\underline{a}, \bar{a}]$ for each chemical property in $\{\text{HA, HF, BP, Kow, RT}\}$; and
 (b) For two combinations $(\pi_1, \pi_2) \in \{(\text{HA, Kow}), (\text{BP, HF})\}$, choose a pair of target values $y_{\pi_i}^* \in [\underline{a}_{\pi_i}, \bar{a}_{\pi_i}]$, $i = 1, 2$. When the same feature function f is used for two chemical properties $\pi_1, \pi_2 \in \{\text{HA, HF, BP, Kow, RT}\}$, we can combine the corresponding MILP $_{\pi_1}$ and MILP $_{\pi_2}$ to find a vector x^* for a pair of target values $y_{\pi_i}^* \in [\underline{a}_{\pi_i}, \bar{a}_{\pi_i}]$, $i = 1, 2$. In our experiment, we choose a target value y^* and fix or bound some descriptors in our feature vector as follows:

- Choose two target values $y_{\pi}^* \in [\underline{a}, \bar{a}]$ for each chemical property π and one pair of target values $y_{\pi_i}^* \in [\underline{a}_{\pi_i}, \bar{a}_{\pi_i}]$, $i = 1, 2$ for each pair $(\pi_1, \pi_2) \in \{(\text{HA, Kow}), (\text{BP, HF})\}$;
- Choose some two values $n^* \in [\underline{n}, \bar{n}]$ to fix the number $n(H)$ of vertices;
- Fix the number $n_1(H)$ of vertices of degree 1 to each $n_1^* \in \{4, 5, 6\}$ if $n^* = 10, 11, 12, 13$; $n_1^* \in \{5, 6, 7\}$ if $n^* = 14, 15$; and $n_1^* \in \{6, 7, 8\}$ if $n^* = 16$;
- Choose some three values dia^* to fix the diameter $\text{dia}(H)$ so that $\text{dia}^* \in \{4, 5, 6\}$ if $n^* = 10$; $\text{dia}^* \in \{4, 6, 7\}$ if $n^* = 11$; $\text{dia}^* \in \{5, 7, 8\}$ if $n^* = 12$; and $\text{dia}^* \in \{6, 7, 8\}$ if $n^* = 13, 14, 15, 16$; and
- Bound the number $n_{(\text{c,c},2)}(G)$ of double bonds between two carbon atoms by choosing each of the two ranges $r \in \{[0, 1/2], (1/2, 1]\}$ so that $n_{(\text{c,c},2)}(G)/(|V| - 1) \in r$.

This scheme results in $3 \times 3 \times 2 = 18$ MILPs with different combinations of specifications for each pair (y_{π}^*, n^*) (resp., each tuple $(y_{\pi_1}^*, y_{\pi_2}^*, n^*)$) of specified y_{π}^* , $\pi \in \{\text{HA, HF, BP, Kow, RT}\}$ and n (resp., $(y_{\pi_1}^*, y_{\pi_2}^*)$, $(\pi_1, \pi_2) \in \{(\text{HA, Kow}), (\text{BP, HF})\}$ and n). Each of these MILPs is either feasible or infeasible and we find one feasible vector x^* to each feasible MILP. We set $\epsilon = 0.02$ in Step 4.

Tables 2 to 4 show the results on Step 4, where we denote the following:

y_{π}^* : a target value in $[\underline{a}, \bar{a}]$ for a property π ;
 n^* : a specified number of vertices in $[\underline{n}, \bar{n}]$;
 $|F^*|$: the size of set F^* of vectors x^* generated from feasible instances of these 18 MILPs in Step 4;
 IP-time: the time (sec.) to solve the 18 MILP instances to find a set F^* of vectors x^* .

We observe that MILPs with some restrictions were infeasible. For example, $|F^*| = 5$ for $(y_{\text{Ha}}^* = 2700, n^* = 10)$ in Table 2 means that six instances out of the 18 MILPs were infeasible.

Step 5. In this step, we modified the algorithm proposed in (Fujiwara et al., 2008) to enumerate all

acyclic graphs. For each $x^* \in F^*$, we enumerate all graphs $G^* \in \mathcal{G}$ such that $f(G^*) = x^*$.

Tables 2 to 4 show the results on Step 5, where we denote the following:

$\#G^*(\cdot)$: the number of acyclic chemical graphs G^* such that $f(G^*) = x^*$, where the number of chemical graphs already registered in the database PubChem among the generated chemical graphs is indicated in parentheses;
 G-time: the time (sec.) to generate all chemical graphs G^* such that $f(G^*) \in F^*$.

Table 2: Results on Steps 4 and 5 with $\epsilon = 0.02$.

π	y^*	n^*	$ F^* $	IP-time	$\#G^*$	G-time
HA	2700	10	7	0.503	50 (10)	0.0045
HA	2700	11	6	0.799	293 (4)	0.0067
HA	2900	10	7	0.362	8 (8)	0.0042
HA	2900	11	6	0.614	208 (29)	0.0043
HF	70	11	5	0.503	57 (45)	0.0033
HF	70	12	2	0.501	4 (4)	0.0016
HF	90	12	7	1.293	374 (0)	0.0147
HF	90	13	9	1.679	663 (1)	0.0126
BP	190	11	10	2.021	339 (6)	0.0334
BP	190	12	13	3.807	496 (0)	0.1423
BP	220	12	13	4.261	245 (0)	0.1240
BP	220	13	17	5.884	1675 (0)	0.3110
Kow	5	11	6	1.883	94 (4)	0.0047
Kow	5	12	7	4.196	356 (2)	0.0095
Kow	7	14	2	7.058	11 (0)	0.0041
Kow	7	15	9	16.174	975 (0)	0.0216
RT	1600	14	17	15.815	880 (0)	0.5046
RT	1600	15	17	13.933	6152 (0)	1.3607
RT	1900	15	16	14.829	6616 (2)	0.4782
RT	1900	16	18	24.100	19299 (0)	1.6018

Table 3: Results on Steps 4 and 5 for (HA, Kow) with $y_{\text{Ha}}^* = 2700$ and $y_{\text{Kow}}^* = 5$.

n^*	$ F^* $	IP-time	$\#G^*$	G-time
10	5	0.817	19 (6)	0.0027
11	3	1.298	25 (0)	0.0020

Table 4: Results on Step 4 and 5 for (HF, BP) with $y_{\text{HF}}^* = 90$ and $y_{\text{BP}}^* = 220$.

n^*	$ F^* $	IP-time	$\#G^*$	G-time
12	7	2.194	1056 (0)	0.0130
13	3	3.345	18 (1)	0.0090

From these tables, we observe that, for all chemical properties we tested, each set of the 18 instances for finding an acyclic chemical graph with specified sizes $n^* \leq 16$, $n_1^* (< n^*)$ and $\text{dia}^* (< n^*)$ from a target y^* was solved within 2 seconds based on our novel MILP formulation; and that, for each inferred feature

vector x^* , enumeration of all acyclic chemical graphs G^* inferred from x^* was completed in 2 milli seconds to 2 seconds. Note that our MILP outputs a feature vector for which we enumerated all structurally unique chemical graphs as defined in Section 2.2, and hence we were able to generate some chemical graphs which are not registered in the PubChem database. In these results, each inferred vector is admissible (i.e., has one or more corresponding chemical structures), whereas only around 20% to 50% of the feature vectors were admissible under a tolerance $\epsilon = 0.02$ in the previous method (Chiewvanichakorn et al., 2020). Note that, for any small tolerance $\epsilon > 0$, our new method will infer an admissible vector, if one exists.

6 CONCLUDING REMARKS

In this paper, we proposed an improved method over the previous method (Chiewvanichakorn et al., 2020) for the inverse QSAR/QSPR by extending the MILP so that, for a given target y^* , either (i) any feasible solution to the MILP provides an admissible vector $x^* \in \mathbb{R}^K$ and a chemical graph $G = (H, \alpha, \beta)$ such that $\Psi_{\mathcal{G}}(x^*) = y^*$ and $n^* = n(H)$, and $n_1^* = n_1(H)$ and $\text{dia}^* = \text{dia}(H)$ or (ii) the infeasibility of the MILP tells us that there exists no such chemical graph G . Although this paper presented such an MILP for the class \mathcal{G} of acyclic chemical graphs, it is not difficult to modify the MILP to treat any class of chemical graphs. Note that our MILP formulation provides a vector s that forms a chemical graph, which means that we can find an inferred chemical graph in Step 4 without designing an algorithm for Step 5 separately.

REFERENCES

- Akutsu, T. and Nagamochi, H. (2019). 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.
- Chiewvanichakorn, R., Wang, C., Zhang, Z., Shurbevski, A., Nagamochi, H., and Akutsu, T. (2020). A method for the inverse QSAR/QSPR based on artificial neural networks and mixed integer linear programming. *ICBBB2020* (to appear).
- Fujiwara, H., Wang, J., Zhao, L., Nagamochi, H., and Akutsu, T. (2008). Enumerating treelike chemical graphs with given path frequency. *Journal of Chemical Information and Modeling*, 48(7):1345–1357.
- Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. (2018). Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276.
- IBM ILOG CPLEX Optimization Studio 12.8. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf.
- Jalali-Heravi, M. and Fatemi, M. H. (2001). Artificial neural network modeling of Kovats retention indices for noncyclic and monocyclic terpenes. *Journal of Chromatography A*, 915(1-2):177–183.
- Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1945–1954.
- Miyao, T., Kaneko, H., and Funatsu, K. (2016). Inverse QSPR/QSAR analysis for chemical structure generation (from y to x). *Journal of Chemical Information and Modeling*, 56(2):286–299.
- Roy, K. and Saha, A. (2003). Comparative QSPR studies with molecular connectivity, molecular negentropy and TAU indices. *Journal of Molecular Modeling*, 9(4):259–270.
- Segler, M. H., Kogej, T., Tyrchan, C., and Waller, M. P. (2017). Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS Central Science*, 4(1):120–131.
- Skvortsova, M. I., Baskin, I. I., Slovokhotova, O. L., Palyulin, V. A., and Zefirov, N. S. (1993). Inverse problem in QSPR/QSAR studies for the case of topological indices characterizing molecular shape (Kier indices). *Journal of Chemical Information and Computer Sciences*, 33(4):630–634.
- Yang, X., Zhang, J., Yoshizoe, K., Terayama, K., and Tsuda, K. (2017). ChemTS: an efficient python library for de novo molecular generation. *Science and Technology of Advanced Materials*, 18(1):972–976.