所望の物性値を達成する化学グラフの推定

～環構造を持つグラフの場合～

mol-infer/Cyclic

2021 年 10 月 7 日

はじめに

概要

本研究の目標は, 指定された化学的性質に関して所望の物性値を達成するような化合物の化学グラフを推定するシステムを構築することである. このシステムは以下の四つのモジュールから構成される.

モジュール 1: SDF 形式で記述された化合物データを, 特徴ベクトルに変換するためのプログラム.

モジュール 2: モジュール 1 で得られた特徴ベクトルを訓練集合とし, 人工ニューラルネットワークを構築するためのプログラム.

モジュール 3: 所望の物性値を達成すると期待される化学グラフとその特徴ベクトルを, 混合整数線形計画 (mixed integer linear programming; MILP) に基づいて推定するプログラム.

モジュール 4: モジュール 3 で得られた化学グラフの構造異性体を生成するためのプログラム.

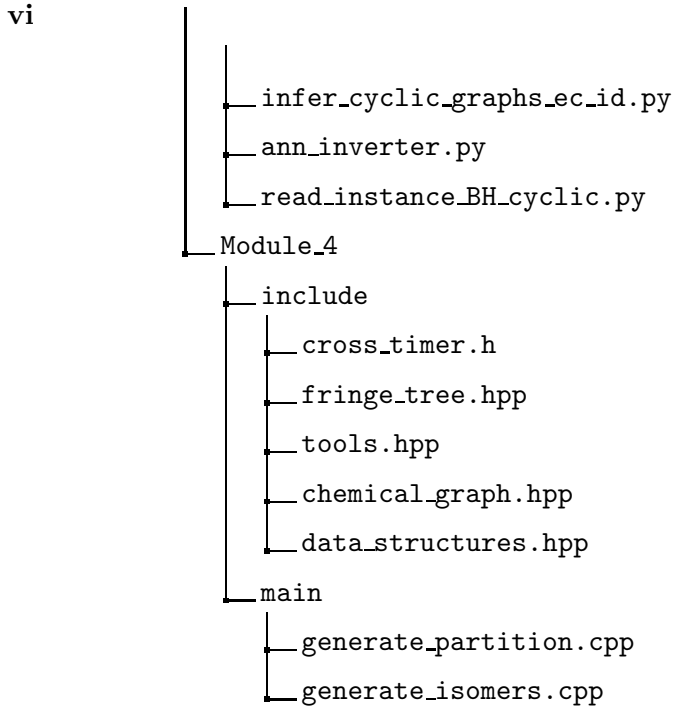
本稿では, システムの出力を得るために, 各モジュールのプログラムをどのように使えばよいかについて説明する.

フォルダとファイルの構成

```
├── LICENSE
├── Makefile
├── README.md
└── bin
    ├── cyclic_graphs_MILP_ec_id.py
    ├── .DS_Store
    ├── infer_cyclic_graphs_ec_id.py
    ├── ann_inverter.py
    ├── predict_values.py
    └── mol-infer_ANN.py
```

```
    ├── eliminate.py
    ├── read_instance_BH_cyclic.py
    ├── osx
    │   ├── generate_isomers
    │   ├── CHECKER
    │   ├── FV_ec
    │   ├── generate_partition
    │   └── FV_proj
    └── linux
        ├── generate_isomers
        ├── CHECKER
        ├── FV_ec
        ├── generate_partition
        └── FV_proj
├── doc
│   ├── cyclic_flow.pdf
│   ├── Manual_Cyclic_en.pdf
│   └── Manual_Cyclic_jp.pdf
└── instances
    ├── chemical_specification
    │   ├── instance_b4.txt
    │   ├── instance_a.txt
    │   ├── instance_c.txt
    │   ├── instance_b2.txt
    │   ├── instance_b3.txt
    │   ├── instance_b1.txt
    │   └── instance_d.txt
    └── BP
        ├── BP_desc.csv
        ├── BP_GEN.sdf
        ├── BP_values.txt
        ├── BP_MILP.LOG
        ├── BP_MILP_partition.txt
        ├── BP_ANN.LOG
        ├── BP.sdf
        ├── BP_biases.txt
        └── BP_weights.txt
```

```
        ├── BP_MILP.sdf
        ├── BP_GEN.LOG
    ├── KOW
    │   ├── KOW_GEN.sdf
    │   ├── KOW_MILP.LOG
    │   ├── KOW_values.txt
    │   ├── KOW_MILP_partition.txt
    │   ├── KOW_desc.csv
    │   ├── KOW_weights.txt
    │   ├── KOW_GEN.LOG
    │   ├── KOW_MILP.sdf
    │   ├── KOW_biases.txt
    │   ├── KOW.sdf
    │   └── KOW_ANN.LOG
    ├── MP
    │   ├── MP_values.txt
    │   ├── MP_GEN.LOG
    │   ├── MP.sdf
    │   ├── MP_ANN.LOG
    │   ├── MP_MILP.sdf
    │   ├── MP_weights.txt
    │   ├── MP_desc.csv
    │   ├── MP_biases.txt
    │   ├── MP_GEN.sdf
    │   ├── MP_MILP.LOG
    │   └── MP_MILP_partition.txt
└── src
    ├── Module_1
    │   ├── fv_proj.cpp
    │   ├── cycle_checker.cpp
    │   ├── fv_ec.cpp
    │   └── eliminate.py
    ├── Module_2
    │   ├── predict_values.py
    │   └── mol-infer_ANN.py
    ├── Module_3
    │   └── cyclic_graphs_MILP_ec_id.py
```


infer_cyclic_graphs_ec_id.py
ann_inverter.py
read_instance_BH_cyclic.py
Module_4
include
cross_timer.h
fringe_tree.hpp
tools.hpp
chemical_graph.hpp
data_structures.hpp
main
generate_partition.cpp
generate_isomers.cpp

目次

第 1 章

モジュール 1: SDF ファイルから特徴ベクトルを計算する

本稿では, 本プロジェクト (`mol-infer/Cyclic`) における モジュール 1 の手順を解説する. この モジュール 1 の入力と出力は以下の通りである.

> **入力**: 閉路を持つ化学グラフの集合 $D = \{G_1, G_2, \ldots, G_p\}$.
> **出力**: 特徴ベクトルの集合 $\mathcal{F}(D) \triangleq \{f(G_1), f(G_2), \ldots, f(G_p)\}$. ただし f は化学グラフを特徴ベクトルに変換する関数で, 論文 [3] において提案されたものである.

出力は, 特徴ベクトルが記載された csv ファイルとして与えられる. この csv ファイルは, 本プロジェクトの モジュール 2 で用いられる.

本稿の構成は以下の通りである.

- 第 1.1 章: 基本的な用語, およびパッケージのファイル構成の説明.
- 第 1.2 章: 実行例.
- 第 1.3 章: プログラムの入出力に関する詳細.

1.1 準備

■**化学グラフ**. **節点** (node) の集合と, 節点と節点を結ぶ**辺** (edge) の集合の対を**グラフ** (graph) という. グラフにおける**閉路** (cycle) とは, ある節点を出発し, 辺を次々になぞって得られる節点の系列 (ただし始点と終点以外の節点は二度以上訪れない) のうち, 始点と終点が一致するものをいう.

節点に対する元素 (炭素, 窒素, 酸素など) の割当, 辺に対する多重度 (一般に 1 以上 4 以下の整数) の割当が与えられたグラフを**化学グラフ** (chemical graph) という.

■**記述子**. 化学グラフの文脈における**記述子** (descriptor) とは, 化学グラフの特徴を表す指標をいう. 一般に, 化学グラフは一つの記述子に対して一つの数値を取る. 本プロジェクトで用いられる記述子の例として, 水素を除く原子の数, コアに属する原子の数, コアの高さ, などがある. 詳細は論文 [3] を参照のこと.

■**特徴ベクトル**. 化学グラフと記述子の系列が与えられたとき, その化学グラフが各記述子に対して取る数値を順に並べたベクトルを**特徴ベクトル** (feature vector) という.

1.2 実行例

1.2.1 データの妥当性の確認

化合物データは標準的な SDF ファイルによって与えられなければならない.
また各化学グラフ $G \in D$ は以下の条件を満たす必要がある.

(i) G は閉路を持つ.
(ii) G は 4 つ以上の炭素原子を持つ. また電荷のある原子, 価数が標準値[*1]と異なる原子を持たない.
(iii) 芳香辺 (aromatic edge) を含まない.

なお (i), (ii) については, すべての化学グラフが条件を満たすかどうか, パッケージ内のプログラムを用いて判定することができる.

■**閉路の有無を確認**. すべての化学グラフが閉路に関する条件 (i) を満たすどうかを確認するには `cycle_checker.cpp` を用いる.
コンパイルの方法は, `Makefile` が使用可能な環境では,

```
$ make CHECKER
```

とすればよい. そうでなければ

```
$ g++ -std=c++11 -Wall -O3 -o CHECKER cycle_checker.cpp
```

とする.
閉路を持たない化学グラフが SDF ファイル `input.sdf` 内に存在するか否かを確認するには,

[*1] 第 1.3.3 章を参照せよ.

```
$ ./CHECKER input.sdf
```

とする.

- すべての化学グラフが閉路を持つ (すなわち (i) を満たす) 場合は, プログラムは何も出力しない. この場合は次に進んで問題ない.
- 一方, 閉路を持たない化学グラフが存在する場合は当該化学グラフの CID が出力される. このような場合, 次に進むには SDF ファイルから手動で当該化学グラフのデータを取り除く必要がある.

■対象外の化学グラフを除去. すべての化学グラフが条件 (ii) を満たすか否かを判定し, 満たすもののみを別の SDF ファイルにまとめるには `eliminate.py` を用いる.

```
$ python eliminate.py input.sdf
```

もし条件 (ii) を満たさない化学グラフがあれば, その化学グラフの CID が出力される.

実行後, 条件 (ii) を満たす化学グラフのみをまとめた `input_eli.sdf` という SDF ファイルが生成される. もしすべての化学グラフが (ii) を満たす場合は, `input.sdf` と `input_eli.sdf` の中身は同一となる.

1.2.2 特徴ベクトルの生成

すべての化学グラフが上記 (i), (ii), (iii) を満たすような SDF ファイルに対して特徴ベクトルを生成するには `fv_ec.cpp` を用いる.

コンパイルの方法は, `Makefile` が使用可能な環境では,

```
$ make FV_ec
```

とすればよい. そうでなければ

```
$ g++ -std=c++11 -Wall -O3 -o FV_ec fv_ec.cpp
```

とする.

`input_eli.sdf` に対して特徴ベクトルを生成し, その結果を `output.csv` に出力するには,

```
$ ./FV_ec input_eli.sdf output.csv
```

とする. 引数を与えずに実行すれば (もしくは引数が適切に与えられなかった場合), 指定すべき引数が出力されて終了する.

1.2.3 別の化学グラフデータを特徴ベクトル化 (必須ではない)

本プロジェクトで構成する特徴ベクトル変換関数 f は, 元となる化学グラフデータ D に依存する. 別の化学グラフデータ $D' \neq D$ を, f を用いて特徴ベクトルに変換するには, `fv_proj.cpp` を用いる.

コンパイルの方法は, `Makefile` が使用可能な環境では,

```
$ make FV_proj
```

とすればよい. そうでなければ

```
$ g++ -std=c++11 -Wall -O3 -o FV_proj fv_proj.cpp
```

とする.

化学グラフデータ D を `FV_ec` によって変換して得られた特徴ベクトルファイルを `descriptor.csv` とする. すなわち変換写像 f は D から作られたものとする. 一方, D とは別の化学グラフデータ D' が記された SDF ファイルを `input.sdf` とする. $\mathcal{F}(D')$ を `output.csv` に出力するには以下を実行する.

```
$ ./FV_proj descriptor.csv input.sdf output.csv
```

なお `FV_proj` の実行は, モジュール 2 以降に進むために必須の操作ではない. `FV_proj` の用途として, すでに `descriptor.csv` からニューラルネットワークが作られていて, そのニューラルネットワークを使って, `input.sdf` に含まれる化学グラフの化学的性質の値を推定したい状況などが考えられる.

1.3 プログラムの入出力に関する詳細

ここでは特徴ベクトル生成のメインプログラム `FV_ec` (ソースコードは `fv_ec.cpp`) の入出力および実行に関する注意を述べる.

1.3.1 入力

このプログラムは入力のフォーマットとして標準的な SDF (Structure Data File) を使用している. SDF の書式について, 以下を参考資料として挙げておく.

- `http://help.accelrysonline.com/ulm/onelab/1.0/content/ulm_pdfs/direct/reference/ctfileformats2016.pdf` (2021 年 2 月 1 日 アクセス確認)
- `https://www.chem-station.com/blog/2012/04/sdf.html` (2021 年 2 月 1 日 アクセス確認)

1.3.2 出力

本プログラムの出力ファイルは, 独自の FV フォーマットを採用している. このテキストファイルは, カンマで区切った CSV ファイルであり, Excel などの表計算ソフトで開くことができる. 具体的には, 一行目には特徴ベクトルの記述子が, 二行目以降の各行には特徴ベクトルの数値データが記入される. 例として、`sample1.sdf` に対して `FV_ec` を実行した結果得られる `sample1.csv` を示す. それぞれの構成要素はそのあとに説明する.

```
CID,n,cs,ch,bl_2,ms,dg_co_1,dg_co_2,dg_co_3,dg_co_4,dg_nc_1,\
dg_nc_2,dg_nc_3,dg_nc_4,bd_co_2,bd_co_3,bd_in_2,bd_in_3,\
bd_ex_2,bd_ex_3,ns_co_C3,ns_co_C2,ns_nc_O1,ns_nc_N1,ns_nc_C2,ns_nc_C3,\
ec_co_C2_C3_2,ec_co_C2_C2_1,ec_co_C2_C3_1,ec_co_C2_C2_2,\
ec_in_C2_C3_1,ec_in_C3_C2_1,\
ec_ex_C3_N1_1,ec_ex_C3_C3_1,ec_ex_C3_O1_1,ec_ex_C3_O1_2,nsH
6140,12,6,4,1,128.333,0,5,1,0,3,1,2,0,3,0,0,0,1,0,1,5,2,\
1,1,2,1,2,1,2,1,1,1,1,1,1,11
```

なお上記のバックスラッシュ \ とそれに続く改行は紙面の都合上挿入したものであり, 実際のファイル内では\も現れなければ改行もされない. 記述子の概要は以下の通りである. 詳しい説明は論文 [3] を参照せよ.

- **CID:** 化合物の識別子 (Compound ID). この例 (`sample1.sdf`) では 6140 だが, これは `https://pubchem.ncbi.nlm.nih.gov/compound/6140` から取得可能なフェニルアラニン (Phenylalanine) である.
- **n:** 水素を除く原子の数.
- **cs:** コアに属する原子の数.
- **ch:** コアの高さ.

- **bl:** 2-leaf の個数.
- **ms:** 平均分子質量 ms $\triangleq \frac{1}{n}\sum_a \lfloor 10 \cdot \mathrm{mass}(a)$, ただし $\mathrm{mass}(a)$ は原子 a の原子量を表す.
- **dg_co_1, . . . , dg_co_4:** コアに属する原子で, 次数が 1,2,3,4 のものの個数.
- **dg_nc_1, . . . , dg_nc_4:** コアに属さない原子で, 次数が 1,2,3,4 のものの個数.
- **bd_co_2, bd_co_3:** コア二重結合およびコア三重結合の数.
- **bd_in_2, bd_in_3:** 内部二重結合および内部三重結合の数.
- **bd_ex_2, bd_ex_3:** 外部二重結合および外部三重結合の数.
- **ns_co_Xd:** コアに属する元素記号 X の原子で, 次数が d のものの個数. たとえば `ns_co_C3` は, コアに属する炭素原子 C で次数が 3 のものの個数を示す.
- **ns_nc_Xd:** コアに属さない元素記号 X の原子で, 次数が d のものの個数.
- **ec_co_Xx_Yy_2, ec_co_Xx_Yy_3:** コア二重結合およびコア三重結合であって (無向辺), 次数 x の原子 X と次数 y の原子 Y を結ぶものの本数. たとえば `ec_co_C2_C3_2` は, 次数 2 の炭素原子 C および次数 3 の炭素原子 C を結ぶコア二重結合の本数を示す.
- **ec_in_Xx_Yy_2, ec_in_Xx_Yy_3:** 内部二重結合および内部三重結合であって (親から子への有向辺), 次数 x の原子 X と次数 y の原子 Y を結ぶものの本数.
- **ec_ex_Xx_Yy_2, ec_ex_Xx_Yy_3:** 外部二重結合および外部三重結合であって (親から子への有向辺), 次数 x の原子 X と次数 y の原子 Y を結ぶものの本数.
- **nsH:** 水素原子の個数.

なお ns, ec で始まる記述子, 入力 SDF 内の化合物に現れるもののみが FV ファイルに出力される.

1.3.3 注意

原子の質量は、プログラムの中に記載するハードコード仕様になっている. `fv_ec.cpp` 内の関数 `init_MassMap()` で以下のように定められているが, 質量の変更や, ほかの原子が必要な場合には, 編集して再度コンパイルすることで利用できる.

```
M["B"]  = 108;
M["C"]  = 120;
M["O"]  = 160;
M["N"]  = 140;
M["F"]  = 190;
M["Si"] = 280;
M["P"]  = 310;
M["S"]  = 320;
M["Cl"] = 355;
M["V"]  = 510;
M["Br"] = 800;
M["Cd"] = 1124;
M["I"]  = 1270;
M["Hg"] = 2006;
M["Pb"] = 2072;
M["Al"] = 269;
```

第 2 章

モジュール 2: ニューラルネットワークの学習

本稿では，本プロジェクト (`mol-infer/Cyclic`) における モジュール 2 の手順を解説する．

与えられた化学グラフの集合を $D_\pi = \{G_1, G_2, \ldots, G_p\}$, 化学グラフを特徴ベクトルに変換する写像を f とする．$\mathcal{F}(D_\pi) \triangleq \{f(G_1), f(G_2), \ldots, f(G_p)\}$ と定義する．また, 注目する化学的性質を π と書くことにする．この π は，たとえば沸点，燃焼熱，水分配係数など様々である．モジュール 2 の入力と出力は以下の通りである．

入力: 特徴ベクトルの集合 $\mathcal{F}(D_\pi) = \{x_1, x_2, \ldots, x_p\}$，各化合物 $G_i \in D_\pi$ もといその特徴ベクトル $x_i = f(G_i) \in \mathcal{F}(D_\pi)$ が化学的性質 π に関して持つ値の集合 $\{a(x_1), a(x_2), \ldots, a(x_p)\}$，ニューラルネットワークの各種パラメータの値 (隠れ素子の個数など).

出力: 入力で指定された構造を持ち，$\mathcal{F}(D_\pi)$ における「 多くの」特徴ベクトル $x \in \mathcal{F}(D_\pi)$ に対して化学的性質の値 $a(x)$ を「 良く」推定するようなニューラルネットワーク．

出力の具体的な中身は，学習されたニューラルネットワークにおける各枝の重みと各ノードのバイアスである．

本稿の構成は以下の通りである．

- 第 2.1 章: 基本的な用語，およびパッケージのファイル構成の説明.
- 第 2.2 章: 簡単な実行例.
- 第 2.3 章: プログラムの入出力に関する詳細.

2.1 準備

2.1.1 用語の説明

■特徴ベクトル． 各元素の種類の原子数等の化学物質を説明する数値，あるいはグラフの直径等の化学物質のグラフ表現のトポロジーに基づいて計算される数値のベクトル．

■人工ニューラルネットワーク（ANN）． 人工ニューラルネットワーク（artifician neural network, ANN），または単にニューラルネットワーク（NN）とは，機械学習で最も確立した手法の 1 つである．これらは入力ベクトルに基づいて値を予測するために用いられる．この冊子では，ニューラルネットワークへの入力は，化合物の特徴ベクトルであり，出力は特定の化学的性質の予測値である．

本プロジェクトで用いるのはフィードフォーワード型のニューラルネットワークであり，これは非巡回有向グラフによって表すことができる．図 2.1 に例を示す．

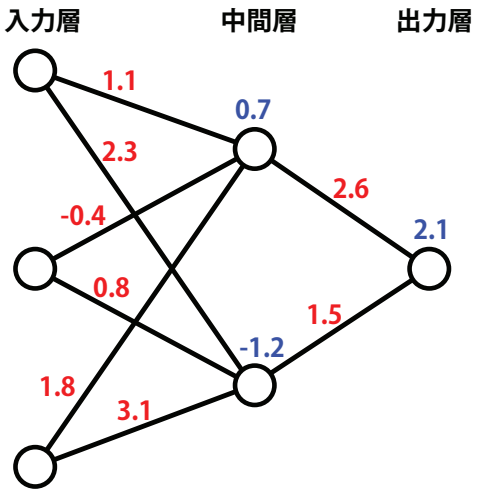


図 2.1: ニューラルネットワークの具体例．赤色の数値が重み，青色の数値がバイアスを表している．枝はすべて左から右に向いている．

■入力層，隠れ層，出力層． 人工ニューラルネットワークの多層パーセプトロンモデルを仮定する．このモデルでは，ニューラルネットワークはいくつかの層で構成されている．最初の層は入力層で，入力層は場合によっては特徴ベクトルから数値データを得るため，特徴ベクトルの要素と同じ数のノードがある．次に数値は隠れ層を介して伝播される．隠れ層は 1 つの層の計算が次の層への入力として用いられる．最後に出力層が入力ベクトルに基づいた予測値を与える．

図 2.1 では，入力層は 3 つのノードを持つ．したがって入力する特徴ベクトルは 3 次元のものでなければならない．また 1 つの隠れ層を有し，この隠れ層は 2 つのノードを持つ．そして出力層は 1 つのノードを持つ．

■重み． ニューラルネットワークでは，ノード間を接続する枝（有向枝）にはそれぞれ数値が割り当てられ，その値を重みと呼ぶ．入力層から出力層への値の伝播には，これらの重みに基づく計算が含まれている．

図 2.1 では，枝の重みは赤によって示される．

■バイアス． ニューラルネットワークの隠れ層の各ノードにはバイアスと呼ばれる数値が割り当てられている．この数値は重みと共に，入力ベクトルに基づいて出力値を計算する過程で使用される．

図 2.1 では，ノードのバイアスは青によって示される．

■活性化関数． 活性化関数はニューラルネットワークの各ノードに割り当てられており，与えられた入力ベクトルから出力値を計算する際に用いられる．特に各ノードの出力値は，重み付けされた対応する枝の重みと前の層からのノードの出力の線形結合を入力として与えられた活性化関数の値である．

本プロジェクトでは，隠れ層の各ノードの活性化関数として Rectified Linear-Unit (ReLU) 関数を限定して用いる．これは次の モジュール 3 (この モジュール 2 で学習したニューラルネットワークに基づいた化合物推定) では，活性化関数が ReLU であることを仮定しているからである．

2.2 クイックスタート

■ニューラルネットワークの学習． 次のコマンドを入力すれば，`instances/BP/BP_desc.csv` を特徴ベクトル，`instances/BP/BP_values.csv` を化学的性質（この場合は BP，すなわち沸点）の値とするようなデータセットに対し，2 つの隠れ層を持ち，それぞれの隠れ層におけるノード数を 20, 10 とするようなニューラルネットワークが学習され，そのニューラルネットワークにおける各枝の重みは `output_weights.txt`，各ノードのバイアスは `output_biases.txt` にそれぞれ出力される．

```
$ python scikit_chemgraph_learning_lim.py ../instances/BP/BP_desc.csv\
 ../instances/BP/BP_values.txt output 20 10
```

学習されたニューラルネットワークの**重みおよびバイアスに関するファイルは，モジュール 3 でも使用**する．

■化学的性質の値の推定． 学習したニューラルネットワークを用いて，化合物の化学的性質の値を推定することができる．

次のコマンド[*1]を入力すれば,学習済ニューラルネットワーク(枝の重みは output_weights.txt, ノードのバイアスは output_biases.txt に保持されている)を用いて,instances/BP/BP_desc.csv に記述された特徴ベクトル(に対応する化合物) の化学的性質の値が推定され，その結果は predicted.txt に出力される.

```
$ python predict_values.py output_weights.txt output_biases.txt \
    instances/BP/BP_desc.csv predicted.txt
```

- 化学的性質の値を推定したい化合物の特徴ベクトルは，モジュール 1 の特徴ベクトル生成器を用いて生成することができる.
- この推定機能は補助的なものに過ぎず，モジュール 3 以降で使用することはない.

2.3 プログラムの入出力に関する詳細

2.3.1 入力

特徴ベクトル

特徴ベクトルは，我々が **FV 形式**と呼ぶフォーマットに基づく csv ファイルに記述されている必要がある．モジュール 1 の特徴ベクトル生成器は化合物に関する SDF ファイルから FV 形式の csv ファイルを生成するため，当該生成器を用いて生成されたファイルを用いれば問題はない.

FV 形式の記述ルールは第 1.3 章で述べたとおりである.

化学的性質の値

化学的性質の値は，CID と値を羅列した csv ファイルに記述されなければならない.

```
CID,a
307,11.2
244,-0.5
657014,98.124
16704,-12.8
117,5.3
```

- 先頭行は CID,a でなければならない.
- 二行目以降に，一つの行に一つの化合物の CID および化学的性質の値をコンマ区切りで記

[*1] 一行目の最後のバックスラッシュ \ は，実際に入力するときには改行してはならないことを示す.

述する．

- 化合物が CID の昇順あるいは降順に並んでいる必要はない．

2.3.2 実行

第 2.2 章で示したとおり，ニューラルネットワークを学習するには scikit_chemgraph_learning_lim.py を用いる．

引数

第 2.2 章で示したコマンドを再掲する．

```
$ python scikit_chemgraph_learning_lim.py ../instances/BP/BP_desc.csv\
 ../instances/BP/BP_values.txt output 20 10
```

各引数には以下を指定する．

- 第 1 引数: 特徴ベクトルに関する csv ファイル
- 第 2 引数: 化学的性質の値に関する csv ファイル
- 第 3 引数: 学習されたニューラルネットワークの重み· バイアスを書き込むファイルの名前
- 第 4 引数以降: 隠れ層（ 中間層）のノードの個数

引数が適切に与えられると，ニューラルネットワークの学習が始まる．

5 分割交差検定が行われ，試験集合に対して最も高い決定係数 (R^2 値) を実現するニューラルネットワークが採用され，その重みとバイアスが，ファイルに出力される．上記のコマンドの場合，そのファイルの名前は第 3 引数で指定された output に基づく，

- output_weights.txt（ 重み）
- output_biases.txt（ バイアス）

である．これら出力されたファイルは，モジュール 3 の実行に必要となる．

■データセットに関する注意. データセットは，

- 特徴ベクトルに関する csv ファイル，および
- 化学的性質の値に関する csv ファイル

の二つのファイルから構成されるが，CID は前者ファイルに記されたものすべて，かつそれのみが計算の対象となる．したがって，

前者ファイルに記された **CID** は，すべて後者ファイルに記されていなければならない．

しかし逆は成り立たなくともよい．つまり，化学的性質の値に関する csv ファイルには，特徴ベクトルの csv ファイルに記されていない,「余計な」CID に関する値が記されていても構わない．そのような値は無視される．

ハイパーパラメータの調整

ニューラルネットワークの学習は `scikit-learn` ライブラリ[*2] における `MLPRegressor` を用いて行われる．`scikit_chemgraph_learning_lim.py` の 135 行目以降で `MLPRegressor` インスタンスの初期化が行われるが，ハイパーパラメータの調整はここで行うことができる．一部のパラメータを以下のように設定している．

- `activation`: `'relu'` **注意:** 学習されたニューラルネットワークを モジュール 3 以降で用いるには, `'relu'` (ReLU 関数) でなければならない．
- `alpha`: 10^{-5}
- `early_stopping`: `False`
- `hidden_layer_sizes`: 実行時に引数で指定した個数に基づく
- `max_iter`: 10^{10}
- `random_state`: 1
- `solver`: `'adam'`

2.3.3 出力

ふたたび以下のコマンドを取り上げ，その実行によって得られる出力について説明する．

```
$ python scikit_chemgraph_learning_lim.py ../instances/BP/BP_desc.csv\
 ../instances/BP/BP_values.txt output 20 10
```

標準出力

上記コマンドを実行すると端末（標準出力）に計算過程が出力される．この出力例がパッケージ内のファイル `instances/BP/BP_ANN.LOG` に記されている．

枝の重み

`scikit_chemgraph_learning_lim.py` が出力する枝重みファイルの書式について説明する．

[*2] `https://scikit-learn.org/stable/`

簡単のため，図 2.1 に示したニューラルネットワークが学習されたとする．このニューラルネットワークの枝重みは以下のようにファイルに出力される．

```
3 2 1
1.1 2.3
-0.4 0.8
1.8 3.1
2.6
1.5
```

- 最初の行はニューラルネットワークの構造，つまり入力層のノード数，各隠れ層のノード数，最後に出力層のノード数である．
- 2 行目以降は枝重みの値である．各行は 1 つのノードから出る枝重みの値を示す．

ノードのバイアス

同じく，scikit_chemgraph_learning_lim.pyが出力するノードのバイアスに関するファイルの書式について説明する．

やはり簡単のため，図 2.1 に示したニューラルネットワークが学習されたとする．このニューラルネットワークのノードのバイアスは以下のようにファイルに出力される．

```
0.7
-1.2
2.1
```

1 行につき 1 つのノードのバイアスの値が示されている．入力層のノードにはバイアスの値がないことに注意せよ．

第 3 章

モジュール 3: MILP を用いた環構造化学グラフの推定

この章では，学習済み人工ニューラルネットワーク (ANN) の重みとバイアスと目標値を与えられた時に，デスクリプタ (記述子) のベクトルを推定できる混合整数線形計画法 (Mixed Integer Linear Programming; MILP) の使い方を説明する．

MILP を定義する機能は Python で実装されていて，COIN-OR パッケージの PuLP モデリングモジュール [6, 7, 8, 9] を使用する．

この章は次のように構成されている．第 3.1 節では使用される用語と表記法を説明する．第 3.2 節はプログラムの入力データと出力データを説明し，第 3.3 節は入力データと計算結果の具体例を示す．

3.1 用語と表記法

この節では，用語と表記法について説明する．

- **特徴ベクトル**
  特徴ベクトルは，デスクリプタと呼ばれる特定のパラメータの数値を格納する．本研究では，非水素原子の数，ある程度の頂点の数など，グラフ理論のデスクリプタを選択する．
- **人工ニューラルネットワーク - ANN**
  人工ニューラルネットワークは，機械学習の方法の一つである．入力の特徴ベクトルと出力の目標データのペア間の相関関数を構築する方法を提供する．
- **入力層， 中間層， 出力層**
  順伝播型ニューラルネットワークの多層パーセプトロンモデルを扱う．これらのニューラルネットワークは、いくつかの層で構成されている．最初に入力層があり，各ニューロンは特徴ベクトルの一つの値を入力として受け取る．次に隠れ層がある．ここでは，入力層からの値が順伝播的に伝播され，一つの層の各ノードが次の層の全てのノードに接続され

る．最後に，出力は出力層で与えられる．単一の目標値の予測を扱うため，出力層は単一のノードで構成されていると仮定する．

- **重み**
  ANN 内の二つのノードを接続する各辺には，重みと呼ばれる実数値が割り当てられる．ANN の学習プロセスの一部で，既知の特徴ベクトルと目標値のペアに基づいて，それぞれの重みの値を決定する．
- **バイアス**
  入力層のノードを除く ANN の各ノードには，バイアスと呼ばれる実数値が割り当てられる．バイアスは辺の重みと同様に，学習プロセスを通じて決定される．
- **活性化関数**
  ANN では，各ノードは入力の活性化関数と呼ばれる関数として出力を生成する．各ノードには，活性化関数として Rectified Linear-Unit（ ReLU）関数があると仮定する．これは ANN の逆問題の MILP 定式化 [2] で正確に表すことができる．
- **混合整数計画問題 (MILP)**
  すべての制約が線形式として与えられ，いくつかの決定変数が整数値のみを取る必要がある数理計画問題の一種．MILP の標準的な文献として，[4] を挙げておく．
- **グラフ**
  頂点の有限集合とエッジの有限集合で構成される抽象的な組合せ構造．ここで各辺は頂点のペアである．無向グラフを扱う．すなわち，辺が順序付けられていない頂点のペアになっているグラフを扱う．グラフ理論の初等的な内容に関しては様々な文献 (たとえば [5] など) が知られている．

3.2 プログラムの入力と出力

この節では，プログラムの入力と出力の形式について説明する．第 3.2.1 節ではプログラムの入力形式の例を示し，第 3.2.2 節では具体的な計算例を示す．次に，第 3.2.3 節ではプログラムの出力形式の例を示し，第 3.2.4 節では具体的な計算例を示す．

3.2.1 プログラムの入力

この節では、プログラムへの入力について説明する．最初の入力には、以下を含む三つのテキストファイルが必要となる．

- CSV 形式のデスクリプタ名，
- テキスト 形式の学習済み ANN の重みと
- テキスト 形式の学習済み ANN のバイアス．

プログラム実行のコマンドラインパラメータとして用いる共通の接頭辞が TT の場合，学習済み ANN のディスクリプタ名，重み，バイアスが格納されたファイルはそれぞれ TT_desc.csv，TT_weights.txt， TT_biases.txt というファイル名で保存されていなければならない．

次に，学習済み ANN に基づいて化学グラフを推定するための目標値を指定する．その次は，[3] で説明されているように、テキストファイルで指定された化学仕様と、第 3.2.4 節で説明されている出力ファイルのファイル名の接頭辞を指定する．

最後に，使用する MILP ソルバープログラムを選択する．以下から選択できる．

- 1: CPLEX(商用 MILP ソルバー) [10].

ファイル infer_acyclic_graphs.py のパラメータ CPLEX_PATH は、CPLEX プログラム実行可能ファイルの正しいパスに設定する必要があることに注意せよ．

- 2: CBC(無償のオープンソース MILP ソルバー)．Python 用の PuLP パッケージ [6] に付属している．

3.2.2 入力データ形式

この節では，プログラムの実際の入力例を示す．特に，第 3.2.1 節で述べた三つの入力ファイルの具体例を示す．

このプログラムの目的は，与えられた学習済み ANN から所望の出力を生成する特徴ベクトルを計算することである．図 2.1 は学習済み ANN の例を示す．

学習済み ANN の重みとバイアスの情報はそれぞれ二つのテキストファイルに書き込まれる．まず，重みの情報を含むテキストファイルの構造を示す．このテキストファイルの最初の行には，ANN のアーキテクチャに関する情報 (各レイヤーのノードの数) が書き込まれる．2 行目以降は、ANN の重みが書き込まれる．各行は，ＡＮＮの1 つのノードに接続する辺の重みが書き込まれており，最初は入力層のノードの重み，次に隠れ層のノードの重みを表している．以下に，図 2.1 に示されてる ANN のテキストファイルの例を示す．

重みのデータ形式

```
3 2 1
1.1 2.3
-0.4 0.8
1.8 3.1
2.6
1.5
```

次は ANN のバイアスのテキストファイルである．Fig. 2.1 のバイアスを以下に示す．

バイアスのデータ形式

0.7
-1.2
2.1

最後に， 特徴ベクトルのデータが含まれたテキストファイルの形式について説明する．このファイルの最初の行には，特徴ベクトルで使用されるデスクリプタの名前が書き込まれている．2 行目以降は，トレーニングデータセット内の各化学グラフのデスクリプタの数値が書き込まれている．一つの化学グラフ毎に 1 行書き込まれる．例えば，ANN フォルダの TT_desc.csv を確認せよ．ここで，TT $\in$ {BP, MP, KOW} である．

3.2.3 プログラムの出力

この節では，プログラムの出力について説明する．与えられた学習済み ANN の予測として与えられた目標値となるような特徴ベクトルを持つ非環式化合グラフが存在する場合，プログラムは特徴ベクトルを出力する．そのような化学グラフが存在しない場合，存在しないと出力する．次の節では，プログラムの出力について説明する．

3.2.4 出力データ形式

この節では，パソコン上で得られたプログラムの出力データについて説明する．プログラムを実行すると，ターミナルに表示される標準エラーストリームにいくつかのメッセージが出力される．MILP ソルバーが計算を完了すると，計算のステータスがターミナルに表示される．

ターミナル上の出力結果のデータ形式

```
Initializing Time: 0.529          # stderr への書き込み
Start Solving Using CPLEX...      # stderr への書き込み
Status: Feasible                  # 解のステータス
MILP y*: 223.856                  # MILP で計算された目標値
ANN propagated y*: 223.856        # 学習済み ANN によって計算された目標値
Solving Time: 52.534              # MILP ソルバーの所要時間
```

最後に，実行可能解が存在する場合、プログラムは二つのテキストファイルを出力する．プログラムは出力ファイルのファイル名を入力の一部として要求することに注意せよ．出力ファイル名として与えられたパラメータが `filename` であると仮定する．結果として得られる二つのテキストファイルのファイル名を以下に示す．

- `filename.sdf`
- `filename_partition.sdf`.

`filename.sdf` は推定された化学グラフの情報が SDF(Structure Data File) 形式で書き込まれている．詳細については，公式ドキュメント（ 英語）を参照せよ．

`http://help.accelrysonline.com/ulm/onelab/1.0/content/ulm_pdfs/direct/reference/`
`ctfileformats2016.pdf`

`filename_partition.sdf` には，[3] で説明されているように，`filename.sdf` から非巡回部分グラフへの閉路グラフの分解が含まれている．

3.3 プログラムの実行と計算例

この節では，プログラムを実行する方法と具体的な計算例について説明する．以下は，`infer_cyclic_graphs_ec_id.py` を実行する例である．

3.3.1 プログラムの実行

```
python infer_cyclic_graphs_ec_id.py trained_ann_filename_prefix target_value
        chemical_specification output_file_name solver_type
```

例として目標特性の融点を使用する．ファイル名の接頭辞が MP となる学習済み ANN、目標値を 220、化学仕様のためのファイルを `instance_1.txt`，MILP ソルバー (パラメータ値は 1) を CPLEX[10] とする．

```
python infer_cyclic_graphs_ec_id.py ANN/MP 220
       chemical_specification/instance_a.txt result 1
```

上記のコマンドを実行すると，ターミナルプロンプトに次のテキストが表示される．

ターミナル上の出力

Initializing Time: 0.529
Start Solving Using CPLEX...
Status: Feasible
MILP y*: 223.856
ANN propagated y*: 223.856
Solving Time: 52.534

出力ファイル `result.sdf` および `result_partition.txt` の内容を以下に示す．

File `result.sdf`

```
 1
MILP_cyclic

 48 51  0  0  0  0  0  0  0  0999 V2000
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 O   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
    0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
```

0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	N	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	N	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	N	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	N	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	C	0	0	0	0	0	0	0	0	0	0	0	0
0.0000	0.0000	0.0000	O	0	0	0	0	0	0	0	0	0	0	0	0

```
   0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
   0.0000    0.0000    0.0000 C   0  0  0  0  0  0  0  0  0  0  0  0
  1  2  1  0  0  0  0
  1 28  1  0  0  0  0
  1 29  1  0  0  0  0
  2  3  1  0  0  0  0
  2  4  2  0  0  0  0
  5  6  1  0  0  0  0
  5 28  1  0  0  0  0
  5 43  1  0  0  0  0
  6 29  2  0  0  0  0
  7  8  1  0  0  0  0
  7 27  1  0  0  0  0
  7 38  1  0  0  0  0
  8  9  1  0  0  0  0
  9 10  1  0  0  0  0
  9 31  1  0  0  0  0
 10 28  2  0  0  0  0
 11 21  1  0  0  0  0
 11 24  1  0  0  0  0
 12 13  1  0  0  0  0
 12 14  2  0  0  0  0
 12 18  1  0  0  0  0
 14 15  1  0  0  0  0
 15 16  2  0  0  0  0
 15 20  1  0  0  0  0
 17 18  1  0  0  0  0
 17 19  1  0  0  0  0
 19 20  1  0  0  0  0
 20 21  1  0  0  0  0
 21 23  1  0  0  0  0
 22 23  1  0  0  0  0
 22 25  1  0  0  0  0
 22 46  1  0  0  0  0
 23 24  2  0  0  0  0
```

```
 24 26  1  0  0  0  0
 25 27  1  0  0  0  0
 26 27  1  0  0  0  0
 29 30  1  0  0  0  0
 31 32  2  0  0  0  0
 31 33  1  0  0  0  0
 33 34  1  0  0  0  0
 33 35  1  0  0  0  0
 35 36  1  0  0  0  0
 36 37  3  0  0  0  0
 38 39  1  0  0  0  0
 39 40  1  0  0  0  0
 40 41  1  0  0  0  0
 40 42  1  0  0  0  0
 43 44  1  0  0  0  0
 44 45  3  0  0  0  0
 46 47  1  0  0  0  0
 47 48  3  0  0  0  0
M  END
$$$$
```

File `result_partition.sdf`

```
12
18
0 1
19
0 0
20
0 0
21
```

```
0 0
22
1 3
23
0 0
24
0 1
25
0 1
26
0 0
27
0 1
28
0 2
29
0 4
15
18 12 14 15 20
1 3
18 17 19
0 3
19 20
0 0
20 21
0 0
21 11 24
0 1
21 23
0 0
22 23
0 0
22 25
0 0
23 24
```

```
0 0
24 26
0 0
25 27
0 0
26 27
0 0
27 7 8 9 10 28
4 6
28 1 29
0 2
28 5 6 29
3 5
```

第 4 章

モジュール 4: 構造異性体の列挙

この章では，与えられた 2-lean 環状化学グラフの構造異性体を列挙する [11] プログラムの使い方を説明する．

次に，この章の構成について説明する．第 4.1 節では，この冊子およびプログラム内で使用している用語について説明する．第 4.2 節では, 環状化学グラフを非環状部分グラフに分割するプログラムの入力と出力について説明し , 実際の計算例を示す．第 4.3 節では，与えられた 2-lean 環状化学グラフの構造異性体を生成するプログラムの入力と出力について説明し , 実際の計算例を示す．

4.1　用語の説明

この節では，用語について説明する．

- 化学グラフ
  化学グラフは節点集合と枝集合の組で化合物の構造を表すものである．各節点には原子の種類が割り当てられており，各枝には結合の多重度が割り当てられている．この冊子では水素原子が省略された化学グラフを扱っていく．
- 特徴ベクトル
  化学グラフにおける各化学元素の数などの情報を与える数値ベクトル．本プロジェクトの特徴ベクトルに使われるディスクリプターの詳細な情報は, [11] を参照.
- 分割情報
  入力する化学グラフの基底節点･ 枝 (base vertex/edge) の指定とその節点･ 枝成分 (vertex/edge component)[1] を固定するか可変にするかの情報．より詳細な情報は，[11] を参照.

4.2　非環状部分グラフへ分割するプログラム

4.2.1　入力と出力

この節では, 環状化学グラフを非環状部分グラフに分割するために使うプログラムの入力と出力情報にすいて説明する．以下ではこのプログラムのことを**分割プログラム**と呼ぶ．4.2.1 節では，プログラムの入力情報について説明する．4.2.1 節では，プログラムの出力情報について説明する．4.2.1 節では，計算機上でプログラムを実行した際に出力されるデータ形式について説明する．

プログラムの入力

分割プログラムでは二つの情報を入力とする．一つ目は化学グラフである．化学グラフは SDF ファイルの形式で入力する．この分割プログラムの入力する SDF ファイルのフォーマットについて，

https://www.chem-station.com/blog/2012/04/sdf.html

などの解説が分かりやすい．さらに，正確な定義書として，公式資料

http://help.accelrysonline.com/ulm/onelab/1.0/content/ulm_pdfs/direct/reference/ctfileformats2016.pdf

を参照するとよい．

二つ目は出力される分割情報を保存する txt ファイル名である．

プログラムの出力

分割プログラムの出力は，入力された化学グラフの分割情報である．分割情報はファイル名が入力された txt ファイルに出力される．

出力データの形式

出力データの形式

```
4
7 # C
0 0 0
15 # C
0 0 0
10 # C
0 0 0
```

```
16 # C
0 0 0
5
7 4 3 5 6 9 2 15 # C1C1N1C1C1C1O1C
0 1 0
7 10 # C2C
0 0 1
10 12 14 13 11 7 # C1C2C1C2C1C
0 0 1
15 16 # C2C
0 0 1
16 18 20 19 17 15 # C1C2C1C2C1C
0 0 1
```

この具体例を用いて，各行の内容を説明する．数値例とそれぞれの内容の対応を表 4.1 に示す．

表 4.1: 出力ファイルの構成

数値例	内容
4	基底節点の数
7 # C 0 0 0 15 # C 0 0 0 10 # C 0 0 0 16 # C 0 0 0	入力した SDF ファイル内での, 基底節点の指標とその元素 核高の下限と上限, そして節点成分を固定するかしないか (0/1)
5	基底枝の数
7 4 3 5 6 9 2 15 # C1C1N1C1C1C1O1C 0 1 0 7 10 # C2C 0 0 1 10 12 14 13 11 7 # C1C2C1C2C1C 0 0 1 15 16 # C2C 0 0 1 16 18 20 19 17 15 # C1C2C1C2C1C 0 0 1	入力した SDF ファイル内での, 基底枝の指標とその元素, そして隣り合う元素同士の価数 核高の下限と上限, そして枝成分を固定するかしないか (0/1)

4.2.2 プログラムの実行と計算例

ここでは入力グラフを非環状部分グラフに分割するためのプログラムの実行方法と，サンプルファイルを入力として用いた場合の計算結果を示す.

コンパイルされた実行ファイルは, 以下の OS 上で動作確認済みである.

- linux
- osx
- windows (cygwin)

実行方法

- 環境確認

  ISO C++ 2011 標準に対応する C++ コンパイラーがあれば問題ないと考えられる．

- コンパイル

  ターミナル上で files のフォルダへ移動する．もしシステム上で make のコマンドが使用可能であれば，以下のコマンドで簡易コンパイルできる．$ make generate_partition
  もし make のコマンドが使えなければ，以下のコマンドでコンパイルできる．
  $ g++ -o generate_partition ./main/generate_partition.cpp -O3 -std=c++11

- 実行

  $ ./generate_partition instance.sdf instance_partition.txt
  instance.sdf で入力の化学グラフファイル，instance_partition.txt で分割情報の出力 txt ファイルを指定する．

計算例

具体例として，以下のような条件で実行する．

- 入力ファイル: フォルダ instances 内の sample_1.sdf
- 分割情報の指定出力ファイル: partition.txt

実行のコマンドは以下のようになる．

```
./generate_isomers ./instances/sample_1.sdf partition.txt
```

このコマンドを実行して場合，以下のような出力結果が得られる．

partition.txt の内容

```
4
7 # C
0 0 0
15 # C
0 0 0
10 # C
0 0 0
16 # C
0 0 0
```

```
5
7 4 3 5 6 9 2 15 # C1C1N1C1C1C1O1C
0 1 0
7 10 # C2C
0 0 0
10 12 14 13 11 7 # C1C2C1C2C1C
0 0 0
15 16 # C2C
0 0 0
16 18 20 19 17 15 # C1C2C1C2C1C
0 0 0
```

4.3 構造異性体生成プログラム

4.3.1 プログラムの入力と出力

この節では, 与えられた 2-lean 化学グラフの構造異性体を生成するプログラム入力と出力について説明する. 以下ではこのプログラムのことを, **異性体生成プログラム**と呼ぶ. 4.3.1 節では，プログラムの入力情報について説明する．4.3.1 節では，プログラムの出力情報について説明する．

プログラムの入力

異性体生成プログラムでは六つの情報を入力を必要とし, 加えて一つのオプションがある．
一つ目は 2-lean 化学グラフの情報 (SDF フォーマット) である．
二つ目はプログラムの計算時間の上限秒である．
三つ目は特徴ベクト ルサイズの上限である．
四つ目は特徴ベクト ルあたり のサンプル木の数である．
五つ目は出力する化学グラフの個数の上限である．
六つ目は出力される化学グラフを保存する SDF ファイル名である．
オプションして入力する化学グラフの分割情報である. イプションの入力が与えた場合，最後にかくこと．

プログラムの出力

異性体生成プログラムの出力は，入力された化学グラフと同型な化学グラフの個数の下限，生成された化学グラフの個数，プログラムの計算時間と入力された化学グラフと同型な化学グラフ

である．化学グラフはファイル名が入力された SDF ファイルに出力される．

4.3.2 異性体生成プログラムの実行と計算例

ここで異性体生成プログラムの実行方法と，サンプルファイルを入力として実行した場合の計算結果を示す．

コンパイルされた実行ファイルは, 以下の OS 上で動作確認済みである．

- linux
- osx
- windows (cygwin)

実行方法

- **環境確認**

  ISO C++ 2011 標準に対応する C++ コンパイラーがあれば問題ないと考えられる．

- **コンパイル**

  ターミナル上で `files` のフォルダへ移動する．もしシステム上で `make` のコマンドが使用可能であれば，以下のコマンドで簡易コンパイルできる．`$ make generate_isomers`
  もし `make` のコマンドが使えない状態であれば，以下のコマンドでコンパイルできる．
  `$ g++ -o generate_isomers ./main/generate_isomers.cpp -O3 -std=c++11`

- **実行**

  `$ ./generate_isomers instance.sdf a b c d output.sdf instance_partition.txt`
  `instance.sdf` で入力の SDF ファイル，`a` で計算時間の上限，`b` で特徴ベクトルサイズの上限, `c` で特徴ベクトルあたりのサンプルツリーの数, `d` で出力する化学グラフの個数の上限，`output.txt` で化学グラフの出力の SDF ファイル, `instance_partition.txt` で化学グラフの分割情報を指定する．

計算例

具体例として，異性体生成プログラムを以下のような条件で実行する．

- 入力ファイル: フォルダ `instances` 内の `sample_1.sdf`
- 計算時間上限: 10 秒
- 特徴ベクトルサイズの上限: 10000000
- 特徴ベクトルあたりのサンプルツリーの数: 5
- 出力する化学グラフの個数上限: 2

- 化学グラフの指定出力ファイル: output.sdf
- 分割情報ファイル: フォルダ instances 内の sample_1_partition.txt

実行のコマンドは以下のようになる.

```
./generate_isomers ./instances/sample_1.sdf 10 10000000 5 2
                    output.sdf ./instances/sample_1_partition.txt
```

このコマンドを実行して場合，以下のような出力結果が得られる.

ターミナルに実行結果の具体例

```
A lower bound on the number of graphs = 72
Number of generated graphs = 72
Total time :  0.00649s.
```

output.sdf の内容

```
1
BH-cyclic
BH-cyclic
20 21 0 0 0 0 0 0 0 0999 V2000
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1 3 2 0 0 0 0
1 5 1 0 0 0 0
1 16 1 0 0 0 0
2 4 2 0 0 0 0
2 11 1 0 0 0 0
2 20 1 0 0 0 0
3 13 1 0 0 0 0
4 17 1 0 0 0 0
5 6 1 0 0 0 0
6 7 1 0 0 0 0
7 8 1 0 0 0 0
8 9 1 0 0 0 0
8 10 1 0 0 0 0
10 11 1 0 0 0 0
11 12 1 0 0 0 0
13 14 2 0 0 0 0
14 15 1 0 0 0 0
15 16 2 0 0 0 0
17 18 2 0 0 0 0
18 19 1 0 0 0 0
19 20 2 0 0 0 0
M END
$$$$
2
BH-cyclic
BH-cyclic
20 21 0 0 0 0 0 0 0 0999 V2000
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1 3 2 0 0 0 0
1 5 1 0 0 0 0
1 16 1 0 0 0 0
2 4 2 0 0 0 0
2 11 1 0 0 0 0
2 20 1 0 0 0 0
3 13 1 0 0 0 0
4 17 1 0 0 0 0
5 6 1 0 0 0 0
6 7 1 0 0 0 0
7 8 1 0 0 0 0
8 9 1 0 0 0 0
8 10 1 0 0 0 0
10 11 1 0 0 0 0
11 12 1 0 0 0 0
13 14 2 0 0 0 0
14 15 1 0 0 0 0
```

```
15 16 2 0 0 0 0
17 18 2 0 0 0 0
18 19 1 0 0 0 0
19 20 2 0 0 0 0
M END
$$$$
```

参考文献

[1] HSDB in PubChem https://pubchem.ncbi.nlm.nih.gov (2021 年 2 月 1 日 アクセス確認)

[2] T. Akutsu and H. Nagamochi. A Mixed Integer Linear Programming Formulation to Artificial Neural Networks, in Proceedings of the 2019 2nd International Conference on Information Science and Systems, pp. 215–220, https://doi.org/10.1145/3322645.3322683.

[3] 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

[4] J. Matousek and B. Gärtner. Understanding and Using Linear Programming. Springer, 2007.

[5] M. S. Rahman. Basic Graph Theory. Springer, 2017.

[6] A Python Linear Programming API, https://github.com/coin-or/pulp.

[7] Optimization with PuLP, http://coin-or.github.io/pulp/.

[8] The Python Papers Monograph, https://ojs.pythonpapers.org/index.php/tppm/article/view/111.

[9] Optimization with PuLP, https://pythonhosted.org/PuLP/.

[10] IBM ILOG CPLEX Optimization Studio 12.8 User Manual. https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf.

[11] H. Nagamochi and T. Akutsu. A Novel Method for Inference of Chemical Compounds with Prescribed Topological Substructures Based on Integer Programming. Arxiv preprint, arXiv:2010.09203