

Module 4における入力した 2-Lean 化学グラフの構造異性体を生成するプログラムの使用説明書

`mol-infer/Cyclic`

2021 年 3 月 25 日

目 次

1	概要	1
2	用語の説明	3
3	非環状部分グラフへ分割するプログラム	4
3.1	入力と出力	4
3.1.1	プログラムの入力	4
3.1.2	プログラムの出力	4
3.1.3	出力データの形式	4
3.2	プログラムの実行と計算例	6
3.2.1	実行方法	6
3.2.2	計算例	6
4	構造異性体生成プログラムについて	7
4.1	プログラムの入力と出力	7
4.1.1	プログラムの入力	7
4.1.2	プログラムの出力	8
4.2	異性体生成プログラムの実行と計算例	8
4.2.1	実行方法	8
4.2.2	計算例	8
	参考文献	11

1 概要

この冊子では、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 [1] プログラムの使い方を説明する。

当モジュールに含まれているファイル及びフォルダの構造は以下の通りである。

```
./Module_4
├── Manual_Module_4_Cyclic_jp.pdf
├── Manual_Module_4_Cyclic_jp.tex
├── Manual_Module_4_Cyclic_en.pdf
├── Manual_Module_4_Cyclic_en.tex
├── Pseudocode_Graph_Generation.pdf
└── files
    ├── Makefile
    ├── executables
    │   ├── linux
    │   │   ├── generate_isomers
    │   │   └── generate_partition
    │   ├── osx
    │   │   ├── generate_isomers
    │   │   └── generate_partition
    │   └── windows
    │       ├── generate_partition.exe
    │       └── generate_isomers.exe
    ├── instances
    │   ├── sample_1_partition.txt
    │   ├── sample_1.png
    │   └── sample_1.sdf
    ├── main
    │   ├── readme.txt
    │   ├── generate_partition.cpp
    │   ├── output.sdf
    │   └── generate_isomers.cpp
    └── include
        ├── chemical_graph.hpp
        ├── data_structures.hpp
        ├── tools.hpp
        ├── cross_timer.h
        └── fringe_tree.hpp
```

また、このパッケージに含まれているファイルとフォルダの内容は以下のとおりである。

- Manual_Module_4_Cyclic_jp.pdf
この冊子.
- Manual_Module_4_Cyclic_jp.tex
この冊子の L^AT_EX のソースファイル.

- `Manual_Module_4_Cyclic_en.pdf`

この冊子の英語版.

- `Manual_Module_4_Cyclic_en.tex`

英語版の冊子の \LaTeX のソースファイル.

- `Pseudocode_Graph_Generation.pdf`

グラフ探索アルゴリズムの疑似コードを記した pdf ファイル.

- フォルダ `files`

- `Makefile`

当モジュールに含まれている C++ プログラムを簡易コンパイルするためのファイル

- フォルダ `executables`

コンパイルされた実行ファイルが入ったフォルダ. サブフォルダとして三つのメイン OS: linux, osx, 及び windows に分かれている.

- フォルダ `instances`

入力のインスタンスのフォルダである.

- * `sample_1.sdf`

節点数が 20, 核サイズが 18, 核高が 1 となる化合物が記載されている入力データである.

- * `sample_1_partition.txt`

`sample_1` の分割情報ファイルである.

- * `sample_2.sdf`

節点数が 50, 核サイズが 24, 核高が 6 となる化合物が記載されている入力データである.

- * `sample_2_partition.txt`

`sample_2` の分割情報ファイルである.

- * `sample_3.sdf`

節点数が 60, 核サイズが 31, 核高が 4 となる化合物が記載されている入力データである.

- * `sample_3_partition.txt`

`sample_3` の分割情報ファイルである.

- * `sample_4.sdf`

節点数が 120, 核サイズが 60, 核高が 4 となる化合物が記載されている入力データである.

- * `sample_4_partition.txt`

`sample_4` の分割情報ファイルである.

- フォルダ `main`

プログラムを走らせる `main` フォルダである.

- * `generate_partition.cpp`

環状化学グラフを, 非環状部分グラフに分割するためのプログラムである.

* `generate_isomers.cpp`

2-lean 環状化学グラフの構造異性体を列挙するためのプログラムである.

– フォルダ `include`

このプログラムに使われるヘッダファイルのフォルダである.

* `chemical_graph.hpp`

化学グラフのためのヘッダファイルである.

* `cross_timer.h`

計算時間を計測するためのヘッダファイルである.

* `data_structures.hpp`

化学グラフのデータ構造を定義するヘッダファイルである.

* `debug.h`

デバッグのためのヘッダファイルである.

* `fringe_tree.hpp`

外縁木 (fringe tree) [1] を列挙する関数のヘッダファイルである.

* `tools.hpp`

便利なツール関数のヘッダファイルである.

次に、この冊子の構成について説明する. 第2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3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.

2 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、[1] 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、[1] を参照.

3 非環状部分グラフへ分割するプログラム

3.1 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ついて説明する。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。3.1.1 節では、プログラムの入力情報について説明する。3.1.2 節では、プログラムの出力情報について説明する。3.1.3 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。

3.1.1 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.accelrys.com/ulm/one/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。

3.1.2 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。

3.1.3 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 # C1C1N1C1C1C101C
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 1 に示す．

表 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 # C1C1N1C1C1C101C 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)

3.2 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)

3.2.1 実行方法

- 環境確認

ISO C++ 2011 標準に対応する C++コンパイラ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。

3.2.2 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 構造異性体生成プログラムについて

4.1 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。4.1.1 節では、プログラムの入力情報について説明する。4.1.2 節では、プログラムの出力情報について説明する。

4.1.1 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。オプションの入力が与えた場合、最後にかくこと。

4.1.2 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。

4.2 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)

4.2.1 実行方法

- 環境確認

ISO C++ 2011 標準に対応する C++コンパイラ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.txt a b c d output.sdf instance_partition.txt`
`instance.txt` で入力のテキストファイル, `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。

4.2.2 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た場合, 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] 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