Instructions on How to use the Programs for Generating Acyclic Chemical Graphs Based on 2-Branch Structure Characterization

August 19, 2020

Contents

1 Outline

In this text we explain the two programs that implement algorithms for generating acyclic chemical graphs based on 2-branch structure characterization [1]. One program takes a feature vector as input and outputs acyclic chemical graphs with 2-branch-number 2 that satisfies the feature vector. The other program takes a feature vector as input and outputs acyclic chemical graphs with 2-branch-number 4 that satisfies the feature vector.

In addition to this document, `README.txt` and `License.txt`, the following folders and files are included in the same folder:

- folder `2_2-branches`
  The folder including the program for generating acyclic chemical graphs with 2-branch-number 2.
  - folder `include`
    - cross_timer.h
      A header file used for calculating the computational time.
    - data_structures.hpp
      A header file containing the settings of data structures.
    - debug.h
      A header file used for debug.
    - fringe_tree.hpp
      A header file containing the functions of enumerating fringe trees [1].
    - tools.hpp
      A header file containing several minor functions used in this program.
  - folder `instances`
    - `ha_tv3500_n15_dia10_dmax3_k2_bn2_bh1.txt`
      An input file describing a feature vector satisfying Ha (Heat of Atomization) property value 3500, vertex number 15, diameter 10, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 2, 2-branch-height 1.
    - `ha_tv4500_n20_dia12_dmax4_k2_bn2_bh1.txt`
      An input file describing a feature vector satisfying Ha property value 4500, vertex number 20, diameter 12, vertex degree upper bound 4, branch-parameter 2, 2-branch-number 2, 2-branch-height 1.
    - `ha_tv7000_n30_dia15_dmax3_k2_bn2_bh1.txt`
      An input file describing a feature vector satisfying Ha property value 7000, vertex number 30, diameter 15, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 2, 2-branch-height 1.
    - `ha_tv8500_n40_dia30_dmax4_k2_bn2_bh1.txt`
      An input file describing a feature vector satisfying Ha property value 8500, vertex number 40, diameter 30, vertex degree upper bound 4, branch-parameter 2, 2-branch-number 2, 2-branch-height 1.

    * `ha_tv11000_n50_dia40_dmax3_k2_bn2_bh1.txt`
      An input file describing a feature vector satisfying Ha property value 11000, vertex number 50, diameter 40, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 2, 2-branch-height 1.
  - `main.cpp`
    Source file of the program generating acyclic chemical graphs with 2-branch-number 2.

- folder `4_2-branches`
  The folder including the program for generating acyclic chemical graphs with 2-branch-number 4.
  - folder `include`
    * cross_timer.h
      A header file used for calculating the computational time.
    * data_structures.hpp
      A header file containing the settings of data structures.
    * debug.h
      A header file used for debug.
    * fringe_tree.hpp
      A header file containing the functions of enumerating fringe trees.
    * tools.hpp
      A header file containing several minor functions used in this program.
  - folder `instances`
    * `ha_tv3500_n15_dia10_dmax3_k2_bn4_bh2.txt`
      An input file describing a feature vector satisfying Ha property value 3500, vertex number 15, diameter 10, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 4, 2-branch-height 2.
    * `ha_tv4500_n20_dia12_dmax4_k2_bn4_bh2.txt`
      An input file describing a feature vector satisfying Ha property value 4500, vertex number 20, diameter 12, vertex degree upper bound 4, branch-parameter 2, 2-branch-number 4, 2-branch-height 2.
    * `ha_tv8000_n30_dia20_dmax3_k2_bn4_bh2.txt`
      An input file describing a feature vector satisfying Ha property value 8000, vertex number 30, diameter 20, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 4, 2-branch-height 2.
    * `ha_tv8000_n35_dia20_dmax4_k2_bn4_bh2.txt`
      An input file describing a feature vector satisfying Ha property value 8000, vertex number 35, diameter 20, vertex degree upper bound 4, branch-parameter 2, 2-branch-number 4, 2-branch-height 2.
    * `ha_tv8500_n40_dia20_dmax3_k2_bn4_bh2.txt`
      An input file describing a feature vector satisfying Ha property value 8500, vertex number 40, diameter 20, vertex degree upper bound 3, branch-parameter 2, 2-branch-number 4, 2-branch-height 2.

- `main.cpp`
  Source file of the program generating acyclic chemical graphs with 2-branch-number 4.

The remainder of this text is organized as follows: Section 2 gives basic terminology used throughout the text. Section 3 explains the format of the input and output of the program. Section 4 shows the results of an experiment example.

2 Terminology

In this section, the terminology used throughout the text and the programs are explained.

- Feature vector
  A vector of numerical values describing a chemical compound such as the number of atoms of each type, or numerical values calculated based on the topology of graph representation of a chemical compound such as the diameter of the graph. The contents of feature vectors used in these two programs are explained with some examples in Section 3.2.

- Acyclic chemical graphs
  A chemical graph constructed by a set of vertices and a set of edges represents the structure of a chemical compound. Each vertex is assigned an atom type, while each edge is assigned a multiplicity of a chemical bond. An acyclic chemical graph represents a chemical compound without any cycle in its molecular structure. This text deals with hydrogen-suppressed chemical graphs.

- Please refer to the reference [1] for related terminology:
  2-branch, 2-branch-number, fringe trees, internal vertices, external vertices, internal edges, external edges, adjacency-configurations, bond-configurations.

3 Program Input and Output

In this section, the input and output of the programs are explained. Section 3.1 describes the input information of the programs. Section 3.2 gives the details of the input data format. Section 3.3 and Section 3.4 explain the output information and the output data format of the programs, respectively.

3.1 Program Input

These two programs each take four arguments as input. The first argument is a text file describing a feature vector. The format of this text file is explained in Section 3.2. The second argument is an upper bound of computational time input in seconds. The third argument is an upper bound of the number of output chemical graphs. The last argument is the name of an SDF file used to store the output chemical graphs.

3.2 Input Data Format

This section describes the data format of the first text file input to the programs as explained in Section 3.1. An example of the input file is as follows:

Input Data Format

```
2
C 120 4 5 8
O 160 2 2 0
3
C C 2 0 3
C O 1 4 0
C C 1 2 5
0 6
4 1
3 1
0 0
10
3
1 2 1 0 1
1 2 2 0 0
1 2 3 0 0
1 3 1 0 2
1 3 2 0 3
1 4 1 0 0
2 2 1 2 0
2 2 2 0 0
2 2 3 0 0
2 3 1 4 1
2 3 2 0 0
2 4 1 0 0
3 3 1 0 1
3 3 2 0 0
3 4 1 0 0
4 4 1 0 0
```

We use this instance to explain the contents of each row of input text files in Table 1.

Table 1: Meaning of the input text file

Numerical values	Contents
2	#types of atoms
C 120 4 5 8 O 160 2 2 0	Atom symbol, 10 times of atomic mass, Valence, #internal vertices, #external vertices
3	#types of atom bonds
C C 2 0 3 C O 1 4 0 C C 1 2 5	AC (atom, atom, multiplicity), #internal AC, #external AC
0 6	#internal vertices of degree 1, # external vertices of degree 1
4 1	#internal vertices of degree 2, # external vertices of degree 2
3 1	#internal vertices of degree 3, # external vertices of degree 3
0 0	#internal vertices of degree 4, # external vertices of degree 4
10	Diameter
3	Upper bound on the vertex degree
1 2 1 0 1 1 2 2 0 0 1 2 3 0 0 1 3 1 0 2 1 3 2 0 3 1 4 1 0 0 2 2 1 2 0 2 2 2 0 0 2 2 3 0 0 2 3 1 4 1 2 3 2 0 0 2 4 1 0 0 3 3 1 0 1 3 3 2 0 0 3 4 1 0 0 4 4 1 0 0	BC (degree, degree, multiplicity), #internal BC, #external BC

#△ stands for the number of △.
AC stands for adjacency-configuration.
BC stands for bond-configuration.

3.3 Program Output

The output information of these two programs contains a lower bound on the number of chemical graphs that satisfy the input feature vector, the number of generated graphs, the computational time of the program and several chemical graphs that satisfy the input feature vector. The chemical graphs are saved in an SDF file with filename that has been provided as input.

3.4 Output Data Format

About the format of SDF files please refer to `https://www.chem-station.com/blog/2012/04/sdf.html` for an explanation in Japanese or the official definition (in English) `http://help.accelrysonline.com/ulm/onelab/1.0/content/ulm_pdfs/direct/reference/ctfileformats2016.pdf` for detail.

4 Program Execution and a Computational Example

This section shows how to execute the programs and the computational results of an example.

4.1 Executing the Program

Both programs are executed as follows:

- *Confirm the development environment*
  Any compiler compatible with the ISO C++ 2011 standard should work. The g++ compiler version 5 & 7 on Linux Mint 18 & 19 have been tested, which can be installed from the command line by the next command.
  `$ sudo apt install g++`

- *Compile*
  `$ g++ -o main main.cpp -O3 -std=c++11`
  (The option `-std=c++11` can be omitted for g++ ver 7.)

- *Run*
  `$ ./main instance.txt a b output.sdf`
  The first argument, `instance.txt`, specifies the input text file, the second argument, `a`, gives an upper bound of computational time in seconds, the third argument, `b`, gives an upper bound of the number of output chemical graphs and `output.sdf` specifies the filename where to save the output chemical graphs.

4.2 Computational Example

Here we show the results of running the program of 2-branch-number 2 with the following input information:

- Input file: `ha_tv3500_n15_dia10_dmax3_k2_bn2_bh1.txt` in the folder `instances`
- Upper bound of computational time: 10 seconds
- Upper bound of the number of output chemical graphs: 2
- Name of the SDF file to save the output chemical graphs: `output.sdf`

The commands to run the program is as follows:

```
./main ../instances/ha_tv3500_n15_dia10_dmax3_k2_bn2_bh1.txt 10 2 output.sdf
```

The output results on the terminal and in the SDF file `output.sdf` are as follows:

Results shown on the terminal

```
A lower bound on the number of graphs = 56
Number of generated graphs = 2
Time :  0.00842078s.
```

Content of output.sdf

```
1
2-branches
2-branches
15 14 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 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
1 2 2 0 0 0 0
1 3 1 0 0 0 0
1 6 1 0 0 0 0
3 4 1 0 0 0 0
3 5 1 0 0 0 0
6 7 1 0 0 0 0
7 8 2 0 0 0 0
7 9 1 0 0 0 0
9 10 1 0 0 0 0
```

```
10 11 1 0 0 0 0
11 12 1 0 0 0 0
12 13 2 0 0 0 0
12 14 1 0 0 0 0
14 15 1 0 0 0 0
M END
$$$$
2
2-branches
2-branches
15 14 0 0 0 0 0 0 0 0999 V2000
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 O 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
0.0000 0.0000 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0
1 2 1 0 0 0 0
1 4 1 0 0 0 0
1 5 1 0 0 0 0
2 3 1 0 0 0 0
5 6 1 0 0 0 0
6 7 2 0 0 0 0
6 8 1 0 0 0 0
8 9 1 0 0 0 0
9 10 1 0 0 0 0
10 11 1 0 0 0 0
11 12 2 0 0 0 0
11 13 1 0 0 0 0
13 14 2 0 0 0 0
13 15 1 0 0 0 0
M END
$$$$
```

References

[1] 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. *In preparation*.