Chemical Graph Project: Inferring Chemical Graphs Based on Grid Neighbor Search

`mol-infer/Grid-neighbor-search`

December 10, 2021

Introduction

Abstract

Our research goal is to develop a new procedure called grid neighbor search (GNS) that can infer many chemical graphs with desired value for a prescribed chemical property as feasible solutions of an MILP as an alternative to Module 3 of the model 2LMM-LLR [8] which can infer at most one graph for a given instance. Similar to Module 3 of 2L-model [9], in this procedure we use prediction functions constructed by an artificial neural network (ANN). The process of generation of the feature vectors is exactly the same as that of Module 1 of the model 2LMM-LLR.

This booklet explains how to obtain inferred chemical graphs by using the program of the GNS procedure.

Structure of Files and Folders

```
├── LICENSE
├── doc
│   └── Manual_Grid_neighbor_search_en.pdf
├── instances_used_in_GNS_exps
│   └── instances.zip
└── GNS
    ├── infer_2LMM_GNS.py
    ├── read_instance_2layer_2LMM_L.py
    ├── twolayered_MILP_2LMM_L.py
    ├── lr_inverter.py
    ├── ann_inverter.py
    ├── pulp_modified
    └── 2LMM_v018
        ├── compute_fc.hpp
        ├── fv_2LMM.cpp
        ├── fv_common.h
        ├── fv_common.hpp
        ├── fv_def.h
        ├── FV_2LMM_V018
        └── fringe
            ├── ChemicalGraph.hpp
            └── commonused.hpp
```

```
        ├── cross_timer.hpp
        ├── RootedGraph.hpp
        └── TopologyGraph.hpp
    ├── sample_instance
        ├── Sl_Hdef_eli.sdf
        ├── output_fringe.txt
        ├── output_desc.csv
        └── output_desc_norm.csv
├── sample_instance
    ├── SfT_biases.txt
    ├── SfT_desc.csv
    ├── SfT_fringe.txt
    ├── SfT_selected_desc.csv
    ├── SfT_selected_desc_norm.csv
    ├── SfT_values.txt
    ├── SfT_weights.txt
    ├── SfT.sdf
    ├── instance_c.txt
    ├── instance_c_fringe.txt
    ├── p_max_delta_r_1e-1.txt
        ├── Lp
            ├── Lp_desc.csv
            ├── Lp_desc_norm.csv
            ├── Lp_fringe.txt
            └── Lp_linreg.txt
        └── Sl
            ├── Sl_desc.csv
            ├── Sl_desc_norm.csv
            ├── Sl_fringe.txt
            └── Sl_linreg.txt
├── sample_results
    ├── SfT_c_44.0_45.0.sdf
    ├── SfT_c_44.0_45.0_partition.txt
    ├── SfT_c_44.0_45.0_test_tmp_desc.csv
    ├── SfT_c_44.0_45.0_test_tmp_desc_norm.csv
    ├── SfT_c_44.0_45.0_i.sdf
    ├── SfT_c_44.0_45.0_i_partition.txt
    ├── SfT_c_44.0_45.0_i_test_tmp_desc.csv
    ├── SfT_c_44.0_45.0_i_test_tmp_desc_norm.csv
    └── SfT_c_44.0_45.0_GS_log.csv
```

Contents

Chapter 1

Grid Neighbor Search

1.1 Outline

This chapter explains how to use an implementation of the grid neighbor search procedure that can infer many chemical graphs with a desired property value by using a mixed-integer linear programming (MILP) formulation and the weights and biases of a prediction function obtained by an artificial neural network (ANN).

The MILP is implemented in Python, using the PuLP modeling module of the COIN-OR package [2, 3, 4, 5].

The chapter is organized as follows. Chapter 1.2 explains the input and output data of the program, and Chapter 1.3 explains how to invoke the program.

1.2 The Program's Input and Output

This section explains the format of the input and the output of the program. Chapter 1.2.1 illustrates an example of the program's input format. Following, Chapter 1.2.2 illustrates an example of the program's output format, and Chapter 1.2.3 gives a concrete computational example.

1.2.1 Program Input

This section gives an explanation of the input to the program.

The input requires the following files containing:

- The two feature vector files with normalized and non-normalized form of descriptors, in csv format as explained in Module 1 of the model 2LMM-LLR [8];
- The files with weights and biases of a trained ANN in textual format as explained in Module 2 of 2L-model [9];
- The property values file in textual format as explained in Module 1 of the model 2LMM-LLR, that contains non-normalized property value of each chemical graph in the underlying data set;
- The fringe tree file in textual format as explained in Module 1 of the model 2LMM-LLR;

- The file containing all chemical graphs in the underlying data set in SDF (structure data file) format.
  For a common prefix `TT` which the program accepts as a command-line parameter, these files must be saved with file names `TT_desc.csv` and `TT_desc_norm.csv` for the files containing the descriptor names with non-normalized and normalized values, resp., `TT_weights.txt` and `TT_biases.txt` for the files containing the weights and the biases, resp., `TT_fringe.txt` for the file containing the list of non-isomorphic fringe tree for the underlying data set, and `TT.sdf` for the file containing all the chemical graphs in the underlying data set;

- Next, comes the lower and upper bounds for the target value for which we wish to infer a chemical graph based on the prediction function obtained by an ANN;

- Following is a chemical specification given in a textual file, as described in [1];

- A list of fringe trees given in a textual file as explained in Module 3 of the model 2LMM-LLR;

- A filename prefix for the output files;

- A file in text format containing the information of parameters related to GNS such as width vector of a subspace and radius vector. An example of such file is given below, where the variables p_max, delta, and r represent the number of linear functions, width vector, and radius vector as defined in [7], and their user defined values are written in the subsequent lines:

File GNS parameters

```
#p_max
2
#delta
1e-1 1e-1
#r
3 3
```

- Files with weights in textual format for the linear functions. Our implementation supports two ways of providing the weights files:

  (i) weights files obtained by Lasso linear regression (LLR) for some properties as explained in Module 2 of the model 2LMM-LLR. In this case for each property `P`, the input requires the files `P_desc.csv` and `P_desc_norm.csv` containing the descriptor names with non-normalized and normalized values, resp., `P_linreg.txt` the file containing the weights, and the bias, to be used for linear functions, and `P_fringe.txt` for the file containing the list of non-isomorphic fringe trees for the underlying data set for the the property `P`. The format of each of these file is explained in Modules 1 and 2 of the model 2LMM-LLR;

  (ii) weights files manually prepared by the users. In this case for each linear function, only one text file in the format of `P_linreg.txt` is required that consists of weights and bias.

- Finally, comes a choice of MILP solver program to be used. In `infer_2LMM_GNS.py`, we can choose:

  (1) CPLEX, a commercial MILP solver [6], free for academic use.
  (Note, in this case the parameter `CPLEX_PATH` in the file `infer_2LMM_GNS.py` must be set to the correct path of the CPLEX program executable file.)

  (2) CBC, a free and open-source MILP solver. It comes together with the PuLP package for Python [2].

1.2.2 Program Output

The aim of the GNS procedure is to infer many chemical graphs with property value in some given interval. This procedure solves an original MILP where no linear functions are used. If this MILP is feasible, then the procedure will solve augmented MILPs where linear functions are used for each subspace in a specified region as explained in [7]. These subspaces are indexed in a typical way as explained in [7]. Whenever an MILP is feasible, the procedure outputs a chemical graph with the desired property value. The output format and a computational example are discussed below.

1.2.3 Output Data Format

This section describes the output data of the program as obtained on a personal computer.

Once invoked, the program will print some messages to the standard error stream, which appear on the terminal. Once the solver completes the computation for the original MILP or the augmented MILP, the status of the computation is printed on the terminal.

Text output on the terminal

```
Initializing Time: 0.511          # Written to stdout
Number of variables: 7157         # The number of variables in the constructed model
- Integer: 6800                   # Written to stdout
- Binary: 5447                    # Written to stdout
Number of constraints: 5633       # The number of constraints in the constructed model
Status: Feasible                  # Solution status
Solving Time: 2.712               # Time taken by the MILP solver
MILP y*: 1913.633                 # Calculated target value in the MILP
```

If there exists a feasible solution the program writes to disk four files. Recall that as a part of the input the program requires a filename used for the output files. Assume that the supplied parameter is `filename`. Then for the original MILP, the name and explanation of the resulting files are given below:

- `filename.sdf` file contains information on the inferred chemical graph in the SDF (Structure Data File) format as explained in Module 3 of the model 2LMM-LLR;

- `filename_partition.txt` contains a decomposition of the graph as explained in Module 3 of the model 2LMM-LLR;

- `filename_test_tmp_desc.csv` contains feature vector without normalized descriptors of the inferred graph as explained in Module 1 of the model 2LMM-LLR; and

- `filename_test_tmp_desc_norm.csv` contains feature vector with normalized descriptors of the inferred graph as explained in Module 1 of the model 2LMM-LLR.

Similarly, for the augmented MILP corresponding to the i-th subspace, the resulting files are named

- `filename_i.sdf`;

- `filename_i_partition.txt`;

- `filename_i_test_tmp_desc.csv`; and

- `filename_i_test_tmp_desc_norm.csv`.

Finally, the program outputs a csv file named `filename_GS_log.csv` that contains a summary of all results obtained for the given inputs.

1.3 Invoking the Program

This section explains how to invoke the program. Following is an example of invoking the program `infer_2LMM_LLR_GNS.py`.

Before using the program `infer_2LMM_LLR_GNS.py`, compile the feature vector generator available in the folder `GNS/2LMM_v018` in the following way after directing the terminal to the folder `2LMM_v018`.

```
$ g++ -o FV_2LMM_V018 fv_2LMM.cpp -O3 -std=c++11
```

This step is exactly the same as that of Module 1 of the model 2LMM-LLR.
Now make sure that the terminal is correctly directed to the location of the `GNS` folder.
There are the following two ways to execute the program:

(i) Weights for linear functions are obtained by using LLR for some properties as explained in Module 2 of the model 2LMM-LLR. In this case, the program is executed as follows

```
$ python infer_2LMM_LLR_GNS.py prefix_for_the_property
lower_bound_for_predicted_value upper_bound_for_predicted_value
instance_file.txt fringe_tree_file.txt prefix_for_output
list_of_prefix_for_the_properties_used_for_LLR
```

(ii) Weights of linear functions are manually prepared by the users. In this case, the program is executed as follows

```
$ python infer_2LMM_LLR_GNS.py prefix_for_the_property
lower_bound_for_predicted_value upper_bound_for_predicted_value
instance_file.txt fringe_tree_file.txt prefix_for_output
list_of_text_weight_files_for_linear_functions
```

As an example we use the target property Surface Tension, that is, the files from the process of constructing prediction function by using ANN with filename prefix `SfT`, target value lower and upper bounds 44.0 and 45.0, resp., the file `instance_c.txt` for a chemical specification, the file `instance_c_fringe.txt` for a list of fringe trees, `SfT_c_44.0_45.0` is the prefix for the output files, the file `p_max_delta_r_1e-1.txt` containing the parameters for GNS, and the prefix `Sl` and `Lp` of the properties Solubility and Lipophilicity, resp., used to obtain weights by using LLR.

```
   $ python infer_2LMM_LLR_GNS.py ./sample_instance/Sft 44.0 45.0
./sample_instance/instance_c.txt ./sample_instance/instance_c_fringe.txt
./sample_results/SfT_c_44.0_45.0 ./sample_instance/p_max_delta_r_1e-1.txt
./sample_instance/Sl/Sl ./sample_instance/Lp/Lp
```

By executing the above command (without a line break), the following following files should appear in the folder `sample_results` for the original MILP:
- `SfT_c_44.0_45.0.sdf`;
- `SfT_c_44.0_45.0_partition.txt`;
- `SfT_c_44.0_45.0_test_tmp_desc.csv`;
- `SfT_c_44.0_45.0_test_tmp_desc_norm.csv`; and

for the augmented MILP corresponding to the i-th subspace for which a feasible solution is attained:
- `SfT_c_44.0_45.0_i.sdf`;
- `SfT_c_44.0_45.0_i_partition.txt`;
- `SfT_c_44.0_45.0_i_test_tmp_desc.csv`;
- `SfT_c_44.0_45.0_i_test_tmp_desc_norm.csv`; and

a summary file `SfT_c_44.0_45.0_GS_log.csv`.

Bibliography

[1] J. Zhu, N. A. Azam, K. Haraguchi, L. Zhao, H. Nagamochi, and T. Akutsu. A Method for Molecular Design Based on Linear Regression and Integer Programming, Arxiv preprint, arXiv:2107.02381v2.

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

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

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

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

[6] IBM Cplex Optimizer `https://www.ibm.com/analytics/cplex-optimizer`.

[7] N. A. Azam, J. Zhu, K. Haraguchi, L. Zhao, H. Nagamochi and T. Akutsu. Molecular Design Based on Artificial Neural Networks, Integer Programming and Grid Neighbor Search, IEEE BIBM, 2021. [Accepted]

[8] `https://github.com/ku-dml/mol-infer/tree/master/2LMM-LLR`.

[9] `https://github.com/ku-dml/mol-infer/tree/master/2L-model`.