JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 27, Number 3, 2020

Pp. 436–439
DOI: 10.1089/cmb.2019.0510

PyGTED: Python Application for Computing Graph Traversal Edit Distance

ALI EBRAHIMPOUR BOROOJENY,[1] AKASH SHRESTHA,[1]
ALI SHARIFI-ZARCHI,[2] SUZANNE RENICK GALLAGHER,[1]
SÜLEYMAN CENK SAHINALP,[3] and HAMIDREZA CHITSAZ[1]

ABSTRACT

Graph Traversal Edit Distance (GTED) is a measure of distance (or dissimilarity) between two graphs introduced. This measure is based on the minimum edit distance between two strings formed by the edge labels of respective Eulerian traversals of the two graphs. GTED was motivated by and provides the first mathematical formalism for sequence coassembly and de novo variation detection in bioinformatics. Many problems in applied machine learning deal with graphs (also called networks), including social networks, security, web data mining, protein function prediction, and genome informatics. The kernel paradigm beautifully decouples the learning algorithm from the underlying geometric space, which renders graph kernels important for the aforementioned applications. In this article, we introduce a tool, PyGTED to compute GTED. It implements the algorithm based on the polynomial time algorithm devised for it by the authors. Informally, the GTED is the minimum edit distance between two strings formed by the edge labels of respective Eulerian traversals of the two graphs.

Keywords: clustering genera, coassembly, de novo variation detaction, graph comparison, graph kernel, linear programming.

1. INTRODUCTION

Networks, or graphs as they are called in mathematics, have become a common tool in modern biology. Biological information from DNA sequences to protein interaction to metabolic data to the shapes of important biological chemicals is often encoded in networks.

One goal in studying these networks is to compare them. We might want to know whether two DNA assembly graphs produce the same final sequences or how close the protein interaction networks of two related species are. *Graph Traversal Edit Distance* (*GTED*), is a new method of comparing two networks. Informally, GTED gives a measure of similarity between two directed Eulerian graphs with labeled edges

[1]Department of Computer Science, Colorado State University, Fort Collins, Colorado.
[2]Department of Computer Engineering, Sharif University of Technology, Tehran, Iran.
[3]National Cancer Institute, NIH, Bethesda, Maryland.

by looking at the smallest edit distance that can be obtained between strings from each graph via an Eulerian traversal. GTED was inspired by the problem of *differential genome assembly*, determining if two DNA assembly graphs will assemble to the same string.

In addition to comparing assembly graphs, GTED can also be used to compare other types of networks. GTED yields a (pseudo-)metric for general graphs because it is based on the edit distance metric. Hence, it can be used as a graph kernel for a number of classification problems. GTED is the first mathematical formalism in which global traversals play a direct role in the graph metric. In this application note, we present our tool, PyGTED, that has a full implementation of all the flavors of GTED and can be used for computing the distance between any two graphs. The ability to make a graph kernel on a set of graphs has been also added to this tool.

2. WORKFLOW AND USAGE

In this section, we explain the steps that should be considered to compute GTED for a number of graphs. This might also include the preprocessing of the graphs, for example, making the graphs Eulerian; depending on the initial data set or results, of interest, some of the steps might be discarded. In this tool, we have assumed that our graphs are in either GraphML format or in, what we refer to as, Simple Graph format (SGF). In the SGF, the file contains the number of nodes and edges of the graph in the first line, and each of the other lines represents one edge of the graph by showing its source node, destination node, weight, and label. The values on each line are assumed to be separated by white space.

2.1. Generalizing GTED

Since GTED is defined based on the Eulerian cycles of the two graphs, the assumption is that the graphs are Eulerian; otherwise, the constraints of the linear programming (LP) for finding an Eulerian traversal while keeping the projections of all the edges of the input graphs equal to their weights cannot be satisfied and will lead to an LP for which no solution exists. However, being Eulerian is a very strong constraint, and even on the de Bruijn graphs, this is not usually the case. Two general approaches have been introduced that enable us to generalize GTED to all (strongly) connected graphs according to Boroojeny et al. (2018).

One approach uses the Chinese Postman algorithm to make the graphs Eulerian. Chinese Postman Problem or Route Inspection Problem is the problem of finding the minimum number of edges that have to be duplicated in a graph such that the resulting graph contains a closed path that visits every edge of the graph exactly once, that is, the graph becomes Eulerian. This can similarly be applied to weighted graphs (the ones with discrete weights so that they can be scaled to the integer ones) by assuming that each integer weight is the number of copies of a single edge. Although the original problem considers the case of undirected graphs, the same definition can be used for the directed ones.

To facilitate this preprocessing step, we have provided the possibility of making a number of graphs Eulerian using the Chinese Postman algorithm by the command below:

```
> python convert.py <Input Dir> <Ouput Dir> [options]
```

This command gets all the graphs in the input directory, makes them Eulerian, and writes the new ones to the output directory. By default, it is assumed that the graphs are in the SGF format we defined earlier. If they are in GraphML format, `--graphml` option has to be used. Other options include `--log`, which keeps a log of the number of edges added to each of the graphs, and `--summary`, which only gives an overall summary (mean, standard deviation, etc.) of the number of edges that were added.

Another approach to deal with non-Eulerian graphs is to relax the conditions of the LP that is generated, as explained in Boroojeny et al. (2018) and Ebrahimpour Boroojeny (2019). In that case, there is nothing to be done here because we do not need to change the graphs. The graphs are checked, automatically by the program, if they are Eulerian; if not, the relaxation of the LP conditions will be performed.

2.2. Computing GTED

To compute GTED for a pair of graphs that are in SGF notation, we have to run the command below:

```
> python computeGTED.py <graph 1> <graph 2> [options]
```

Different options can be used to accommodate different needs. If the graphs are in GraphML format, `--graphml` has to be used in the command. Match reward has a default value of 1, but different values

can be set using the `--match_reward` option. If the graphs are disconnected, the flag `--SCC` has to be used; this will tell the program to find the strongly connected components of each graph and make an LP for each pair of them. Since the number of the connected components might get very large and many of them might not be important or big enough to consider, the options `--num_SCC_1` and `--num_SCC_2` can be used to specify the number of the largest components that have to be considered for generating the LPs. Some other options related to other flavors of GTED are explained in the succeeding sections.

When the purpose is to compare two sets of reads generated from sequencing of two different genomes, rather than making an assembly graph of each, HyDA, presented in Shariat Razavi et al. (2014), provides the ability to generate a joint (colored) de Bruijn graph, which provides us with one assembly graph whose edges are colored with two different colors. Therefore, our tool provides the capability of parsing this colored de Bruijn graph to extract the two assembly graphs that are embedded in that. To generate the colored de Bruijn grpah without knowing how HyDA works, we have provided a script that can be used as below:

```
> ./coloredAssembly.sh <HyDA directory> <read file 1> <read file 2> <k>
```

Read files have to be in FASTA or FASTQ format, and k determines the size of the k-mers. Once the colored de Bruijn graph is created, the command below can be used to extract the graphs:

```
> python parseHyDA.py <name of the colored assembly graph> [options]
```

If run, by default, it will create the original graphs that might not be connected. The `--SCC` option, when used, finds all the strongly connected components of the graph and will generate a file in the SGF format for each of them. The options `--num_SCC_1` and `--num_SCC_2` can be used to specify the number of the largest components that have to be considered when generating the graphs.

2.2.1. GTED as a graph kernel. As mentioned earlier, GTED can be used to make a graph kernel from a set of graphs. This kernel can be used for classification or clustering problems in different domains. To compute a graph kernel from a number of graphs present in a directory, the command below can be used:

```
> python graphKernel.py <input directory> <output file> [options]
```

Options that can be used with this command are similar to those of the regular computeGTED.py command.

2.3. Extentions

Analogous to the problem of finding optimal alignments between two strings where the notions of local alignment, presented in Smith et al. (1981), and fit alignment are derived from the original one to deal with different problems, for the alignment of two graphs, extensions of GTED have been defined to answer different questions (Ebrahimpour Boroojeny, 2019; 2020). By using the flag `--local`, the command will compute the Local GTED and using the flag `--fit`, Fit Gted score of the two graphs will be computed where the first graph is the larger one that might be used only partially without being penalized for some nonused sections.

2.4. Fast GTED

A way to make the computation faster by making the alignment graph smaller using the common nodes of the two graphs has been introduced in Ebrahimpour Boroojeny (2019). When using this option, all the nodes that have the same name will be considered common nodes. To run the algorithm with this optimization, the flag `--fast` has to be used.

3. CONCLUSION

In this application note, we have introduced PyGTED, a tool for computing the GTED value, which is a measure for computing the distance (dissimilarity) of two graphs based on a traversal of their edge labels. This tool provides all the capabilities required for preprocessing steps, such as computing the strongly connected components of the graphs and making the graphs Eulerian, and has a complete implementation of all the variants of the method, Local GTED, Fit GTED, and Fast GTED. The details for using this tool on genomic data as well as using it as a graph kernel are

4. AVAILABILITY

Our source code is freely available at https://gitub.com/Ali-E/GTED.

AUTHOR DISCLOSURE STATEMENT

The authors declare they have no conflicting financial interests.

FUNDING INFORMATION

The authors received no funding for this project.

REFERENCES

Boroojeny, A.E., Shrestha, A., Sharifi-Zarchi, A., et al. 2018. Gted: Graph traversal edit distance, 37–53. In *International Conference on Research in Computational Molecular Biology*. Springer, Paris, France.

Ebrahimpour Boroojeny, A. 2019. *Theory of Graph Traversal Edit Distance, Extensions, and Applications*. Ph.D. Thesis. Available from ProQuest Dissertations & Theses Global database. (Accession Order No. AAT 13812734).

Ebrahimpour Boroojeny, A., Shrestha, A., Sharifi-Zarchi, A., et al. 2020. Graph traversal edit distance and extensions. *J. Comp. Biol.* 27 (this issue).

Shariat Razavi, S.B., Movahedi Tabrizi, N.S., Chitsaz, H., et al. 2014. HyDA-Vista: Towards optimal guided selection of *k*-mer size for sequence assembly. *BMC Genomics* 15(Suppl 10), S9. Also GIW/ISCB-Asia proceedings.

Smith, T.F., and Waterman, M.S. 1981. Identification of common molecular subsequences. *J Mol Biol* 147, 195–197.

Address correspondence to:
Dr. Hamidreza Chitsaz
Department of Computer Science
Colorado State University
279 Computer Science Building
1873 Campus Delivery
Fort Collins, CO 80523-1873

E-mail: hamidreza.chitsaz@colostate.edu