

Capstone Project 2016

User Manual

Paul Ziarko, Jack Redmond

Introduction

The problem of maximal clique enumeration in undirected graphs, both static and dynamic, is a problem that many fields are facing. These include fields such as bio-informatics, sociology, and even political science. Several people have created algorithms that potentially solve some of the problems associated with maximal clique enumeration. The functions included in this suite of programs include implementations of some of these algorithms in c++. These include an implementation of Bron-Kerbosch algorithm, an implementation of the rc-procedure, and an implementation of the ac-procedure. The Bron-kerbosch algorithm is a classic maximal clique enumeration algorithm used by many to calculate the number of maximal cliques in static graphs. The rc-procedure presented in the “Reconstructability Analysis of Multi-Dimensional Relations: A Theoretical Basis for Computer-Aided Determination of Acceptable Systems Models” paper by Roger Cavallo and George Klir is used to find maximal cliques in a graph after removing an edge from the graph. The ac-procedure, also presented in Cavallo and Klir’s paper, is used to find maximal cliques in a graph after adding an edge to a graph. These implementations are explained in greater detail in later sections of this manual.

This manual contains the following: instructions on how to download the files needed, a list of files included, instructions on how to run the program, and an in depth look at the files containing the implementations of the algorithms listed above.

Download

If you don’t already have the files needed to run the program, you can download them from: https://github.com/redmonjp/ac_procedure

Download Instructions

LINUX/UNIX Command Line

- 1.) Open a terminal and change to the directory you want to download the files in
- 2.) On the command line enter:

```
user@home:~$ git clone https://github.com/redmonjp/ac_procedure.git
```

Alternatively, you could just download the zip file from:

https://github.com/redmonjp/ac_procedur

To Download Zip

An alternative to the cloning methods above would be to simply download the zip file from github:

- 1.) Go to: https://github.com/redmonjp/ac_procedure
- 2.) Click on Download Zip
- 3.) Extract the files to the directory of your choosing:

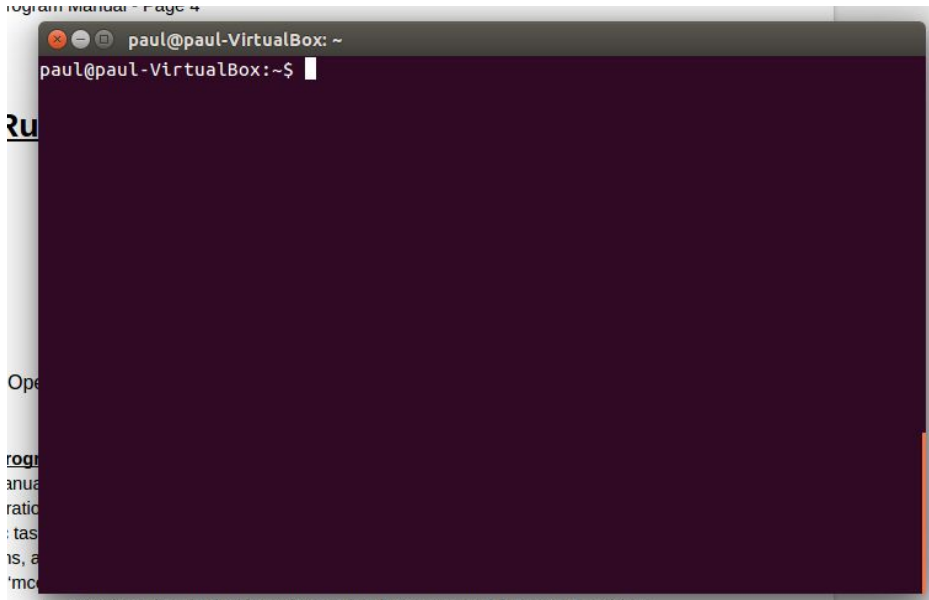
Files Included

As of right now the files in the program repository consist of:

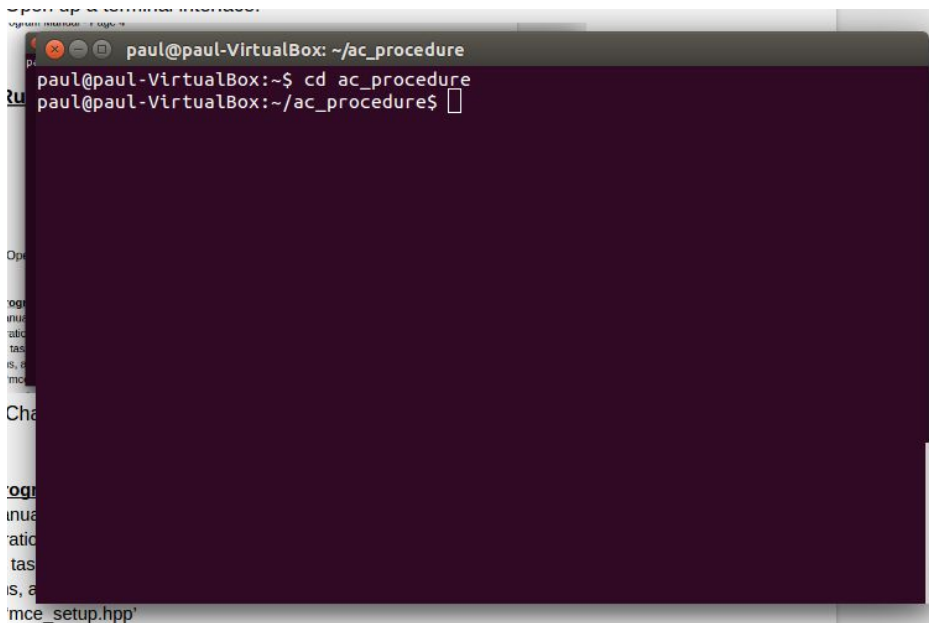
- README.md- Brief description of what our program does (example graph input)
- graph_data.txt - simple graph data file (used when running program)
- main.cpp - main function for program, links all other files together
- Manual - document explaining how to run program and what program does (this document)
- Headers- directory holding implementation files, includes
 - mce_setup.hpp - header file containing functions used to prepare data and print results
 - bk.hpp - header file containing implementation of Bron-kerbosch algorithm
 - rc.hpp - header file containing implementation of rc-procedure
 - ac.hpp - header file containing implementation of ac-procedure

To Run (LINUX/UNIX)

1.) Open up a terminal interface:



2.) Change to the you saved the program files in:



3.) Compile the program by typing:

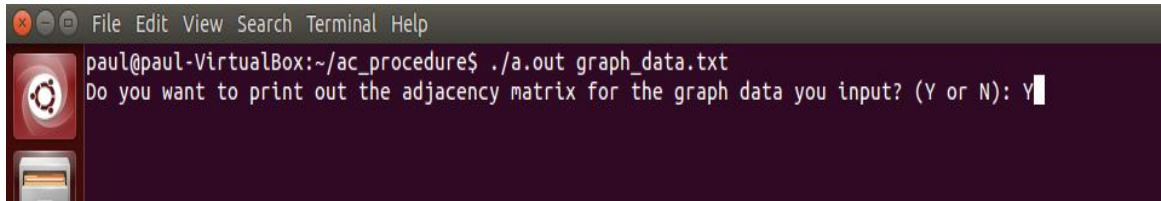
```
paul@paul-VirtualBox:~/ac_procedure$ g++ main.cpp
```

4.) Run the program by typing:

```
paul@paul-VirtualBox:~/ac_procedure$ ./a.out <graph_file>
```

Where <graph_file> is the input file containing the graph information for the graph you want to analyze.

5.) When prompted by the program, enter either Y for yes or N for no.



```

File Edit View Search Terminal Help
paul@paul-VirtualBox:~/ac_procedure$ ./a.out graph_data.txt
Do you want to print out the adjacency matrix for the graph data you input? (Y or N): Y

```

MCE program and file descriptions

This portion of the manual describes how to make use of the utilities included in this suite for Maximal Clique Enumeration problems. The suite consists of four separate header files that each deal with a specific task involved in these problems, a main program file demonstrating the uses of these functions, and some test graph files.

- 'main.cpp'
 - Calls the functions that run the mce algorithm implementation
- 'mce_setup.hpp'
 - Functions that deal with setting up the data structures for a MCE problem
- 'bk.hpp'
 - Functions that deal with an implementation of the Bron-Kerbosh Algorithm
- 'ac.hpp'
 - Functions that deal with an implementation of the AC-procedure as described by Cavallo and Klir in "Reconstructability Analysis of Multi-Dimensional Relations."
- 'rc.hpp'
 - Functions that deal with an implementation of the RC-procedure as described by Cavallo and Klir in "Reconstructability Analysis of Multi-Dimensional Relations."
- 'Graph_data.txt'
 - Sample graph data file

Layout and Usage:

As the functions have been laid out into header files each of the four must be included into a '.cpp' file and then call the functions from there. As a whole the program (.cpp) takes one parameter that is a .txt file describing the graph, this is the graph description file. The file may be named anything but must be the first and only argument given to the program. The graph description file is laid out in a manner that the first line is the number of vertices in the graph and every line following represents an edge in the graph. The lines that represent the edges must enclose the elements in a comma separated set of brackets. An example layout of this file looks something like this:

```

10
{1,2,4}
{2,5}
{3}
{5,7}
{6,8}
{7,10}
{8,2}
{9}

```

The first line - '10' tells the program that there are ten vertices in the graph. The second line - '{1,2,4}' tells program that {1,2,4} is a valid edge in this graph.

MCE-Setup

MCE-Setup is a header file (mce_setup.hpp) that includes functions for setting up data structures for a program involving Maximal Clique Problems. The main function that is included in this header is called 'read_file' and accepts six arguments. The first arguments is a pointer to the membership array followed by a pointer to the edge vector, a pointer to the primal edge vector, a pointer to the vector holding the vertices, argc and argv. The function accepts pointers to these data structures so they must be declared in the .cpp file from which the function is being called. Below is a description of these data structures.

Data structures for 'mce_setup' -

- 2d vector of ints that will hold the edges of the graph -
 - `vector<vector<int>>edge_vector;`
- 2d vector of pairs that are of type (int,int)
 - `vector<pair<int, int>>primal_edge_vector;`
- 2d vector of type bool that will hold the membership array for the graph
 - `vector<vector<bool>> membership_array;`
- Vector of ints that holds the vertices in the graph
 - `vector<int>vertices`
- argc
- argv - the graph description file should be in argv[1]

When this function returns each data structure is packed with the corresponding data from the graph description file that was included as an argument and leaves the user in a state that they can then utilize other functions in this suite like 'bk.hpp', 'ac.hpp' and 'rc.hpp'.

Print functions in 'mce_setup'

Included as utilities in this header are functions that print each one of the data structures described above. These functions include 'print_matrix', 'print_edge_vector', 'print_primal_edge_vector', 'print_maximal_cliques' and 'print_clique'

- 'print_matrix'

This function takes one parameter that is a 2d vector of bools. When printed this array is the row-column representation of edges in the graph where '1' is a valid edge and '0' is an edge that does not exist

- 'print_edge_vector'

This function takes one parameter that is a 2d vector of ints. When this vector is printed each edge included is printed on a new line in set notation

- {1,2,3}

- 'print_primal_edge_vector'

This function takes one parameter that is a 2d vector of pairs where both elements are of type int. When this vector is printed each edge included is printed on a new line in set notation

- {1,2}

- 'print_maximal_cliques'

This function takes one parameter that is a 2d vector of ints. When this vector is printed each clique included is printed on a new line in set notation

- {1,2,3,4}

- 'print_clique'

This function takes one parameter that is a vector of ints. When this vector is printed the clique is printed on the current line in set notation.

BK-algorithm

Our implementation of the Bron-Kerbosch algorithm can be found in the bk.hpp file. This file includes a basic implementation of the bk-algorithm with pivots. This implementation of the Bk-algorithm takes as input an undirected graph, as well as three sets:

- R: A set possibly containing a clique in the graph
- P: Holds vertices that are adjacent to every vertex in R.
- X: Contains nodes already in some clique or already processed (used to remove duplicate cliques).

It then outputs the set of maximal cliques for the undirected graph that was given as input. After the bk-algorithm outputs all maximal cliques, the function check_cstructure() will be called. This function takes as input the possible c-structure that was given in the input file when the program started running and the c-structure that was found as a result of running the bk-algorithm. It then

compares these two sets to see if they are the same. If they are the same then a value of true is returned and the c-structure used for the rest of the program is the c-structure in the file given by the user. If the sets are different however, then a value of false is returned and a message stating that they are not the same is output (in case the user wants to change the input file to reflect the c-structure). If they are not the same then the program will use the c-structure output by the bk-algorithm as input for any function that require the c-structure of the graph.

Functions List

- CheckCStructure- Takes as input a set of maximal cliques for a graph and another set of cliques, given by the user, for the same graph which may or may not be a set of maximal cliques. It sorts and compares these two sets of cliques to see if the set input by the user is the set of maximal cliques for the graph. It will output false if the user input set of cliques does not consist of only and all maximal cliques for the graph and true if it does.
- GetPivot- Function called in GetAllMaxCliques. Used to decide which vertex in the graph should be used as the pivot vertex for a particular call of the Bk-algorithm.
- GetNeighbors- Function called in GetAllMaxCliques. Takes as input a graph and a particular vertex, and outputs a set containing all neighbors of the vertex used as input.
- GetAllMaxCliques- Function implementing the Bk-algorithm. Input described above. Outputs set of all maximal cliques(C-structure) for a given graph.
- CreateSetofVertices- Takes the number of vertices from the file given by the user and uses the number of vertices to create a set of vertices which can be used by the GetAllMaxCliques function.

AC-algorithm

The AC-algorithm header file (ac.hpp) contains functions that aid in implementing the AC-procedure as described by Cavallo and Klir in “Reconstructability Analysis of Multi-Dimensional Relations.” This procedure deals with adding an edge to an existing C-structure. When the edge is added it may change the set of Maximal Cliques in the graph. One way to solve this would be to add the edge and then recalculate the set of Maximal Cliques using the BK algorithm as described above, however that is a fairly inefficient way to find the change, especially if add the edge only changes the set of Maximal Cliques slightly. The approach to the AC-procedure aims to only recalculate the edges that have been affected by the addition in some manner. This header is made up of one function, ‘ac_procedure’ and takes as input the set of all Maximal Cliques for a graph and the membership matrix of the graph. Below is a description of these data structures.

Data structures for 'ac_procedure' -

- 2d vector of ints that holds the set of all Maximal Cliques in the graph
 - `vector< vector <int> > maximal_cliques`
- 2d vector of type bool that will hold the membership array for the graph
 - `vector<vector<bool> > membership_array;`

This function accepts the parameters as copies so the originals, located in the '.cpp' file, are unchanged when the function is complete. As the procedure progresses the user is given output as to what the immediate aggregate of the graph is by adding in the specified edge and when the function is finished the user is displayed a list of the immediate aggregates for adding every possible valid edge to the graph.

RC-algorithm

Our implementation of the Rc-algorithm (Cavallo and Klir) is found in the file rc.hpp. It takes as input a C-structure and the corresponding primal graph for that C-structure. It then removes an edge from the graph and using the process outlined in Cavallo and Klir's "Reconstructability Analysis" paper it finds the C-structure for the new graph with the edge removed. The edge that was taken out is then replaced and the process is repeated until every immediate refinement of the C-structure given as input is found. The function will then output the set of all immediate refinement for the given C-structure.

Function List

- GetRefinement- Implementation of Rc-algorithm. Takes as input a C-structure and its corresponding primal graph. It outputs the set of all immediate refinements of the C-structure that was input by the user.
- CheckNewClique- Function called in GetRefinement. It takes as input a possible maximal clique in the new C-structure and outputs a boolean value (True or False). If the possible maximal clique is in the new C-structure it outputs true, however, if the possible maximal clique is not in the new C-structure, a value of false is output.

References

R.E. Cavallo and G. J. Klir. "Reconstructability Analysis of Multi-Dimensional Relations: A Theoretical Basis for Computer-Aided Determination of Acceptable Systems Models." Int. J. General Systems. 1979 Vol 5, pp. 143-171.

C. Bron and J. Kerbosch. "Finding All Cliques of an Undirected Graph."
Communications of the ACM. Sept. 1973 Vol. 16 (9), pp. 575-577.