# ComplexCi

* **Introduction**

This project mainly focuses on the c++ implementation of Collective Influence (CI) algorithm, which is designed to find the most important node (or influencers) in the Complex Network via optimal percolation developed by

Morone F, Makse H A. Influence maximization in complex networks through optimal percolation[J]. Nature, 2015, 524(7563): 65-U122.

Morone F, Min B, Bo L, et al. Collective Influence Algorithm to find influencers via optimal percolation in massively large social media[J]. Scientific reports, 2016, 6.

Overall, the target of CI algorithm is to give a ranking list of nodes according to their importance. Top-ranked nodes will have more importance. We can remove the nodes from the top-ranked ones in the ranking list generated by CI algorithm and calculate the size of giant component after each removal. The ratio of giant component will reach zero with the one-by-one removal operation finally. Therefore, the better algorithm, the sooner the network will collapse to the zero giant component with smaller count of provided nodes.

The users can get the minimal set of influencers of the Complex Network by the C++ program in this repository . Compared with the original c code <http://www-levich.engr.ccny.cuny.edu/~hernanlab/uploads/CI_HEAP.c> mentioned on the http://www-levich.engr.ccny.cuny.edu/webpage/hmakse/software-and-data/ with above paper , the ComplexCi has the following features :

1. This project is using the c++ code style of modern oriented object programming with Standard Template Library (STL), which is easier for the user to modify than the original c code. After reviewing the original c language code CI\_HEAP.c, I believe it is difficult for the user to read and implement it if they have their own idea, especially CI\_HEAP.c contains lots of simple variable abbreviation, multilevel pointer, unsafe memory management and unfriendly c style code.
2. ComplexCi accepts more input parameters and they can be used in more flexible behaviour of operating nodes in the provided network.
   1. The user can determine the batch size of deleting nodes in ComplexCi per updating CI values when the Complex Network collapses
   2. The user can determine the certain giant component ratio of starting re-inserting algorithm where complex network collapses to
3. Verified that the result of traditional c implementation and new c++ can both achieve the nearly same result in the metric of Robustness, even the new c++ implementation spends much less time on some datasets than the traditional c program. See in the Benchmark Section
4. The CI\_HEAP.c is merged into this project as well. The user can switch the option to use the new c++ designer or the traditional c style code depending on their taste.

* **Usage**

This section describes the Usage of the ComplexCi and its corresponding scripts

* + **Get Repository**
    - **Download Release**
    - **Fetch from Github Source**

The user can directly clone the repository by the git command or just download the zip archiver on the webpage

git clone https://github.com/zhfkt/ComplexCi.git

* + - * **Compile**

There are lots of C++11 features in the code so that the C++ Compiler needs to support C++11 Standard . In fact, there is only one cpp file ComplexCi/ComplexCi.cpp need to be compiled

* + - * + **Linux**

The user can enter into the “bin/” under root project folder and execute

g++ -pthread ../ComplexCi/ComplexCi.cpp -o ComplexCi -O3 -std=c++11 ;

or just execute

./make.sh

to generate binary bin/ComplexCi . Pls notice that the version of g++ needs to support c++11. For my own dev, the compilation is passed under g++ 5.4.0 on Ubuntu 16.04. Otherwise , you will get failure of several incorrect errors.

* + - * + **Windows**

If you have the Visual Studio 2013, the user can directly click the ComplexCi.sln and compile the code in the IDE. I believe the user can also pick up any other IDE or Compiler supporting C++11 to compile the file ComplexCi/ComplexCi.cpp

* + **Run**
    - **Network Input File Format**
    - **Easy to use with Script**
      * **Input Parameters**
      * **Windows**
      * **Linux**
    - **Directly use with Binary ComplexCi**
      * **Input Parameters**
        + **Long Explanation on method**
      * **Windows**
      * **Linux**
    - **Output file**
  + **Benchmark**
    - **Robustness value**
  + **Folder/Files**
  + **DataCastle Competition**
    - **Introduction**http://www.pkbigdata.com/common/cmpt/%E5%A4%A7%E5%B8%88%E8%B5%9B\_%E7%AB%9E%E8%B5%9B%E4%BF%A1%E6%81%AF.html?lang=en\_US
    - **Benchmark**
      * **Quick Result**
      * **Best Result**
    - **Details will be in the experiment paper**