# ComplexCi

* **Introduction**

This project (ComplexCi) 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 importanceand the 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, which will break down the network into many disconnected pieces. 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 . Considered that CI algorithms are able to be used in so many scientific fields , I implement this algorithm in modern C++ style code. 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 , maintain 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 when starting re-inserting algorithm which complex network collapses to
   3. The user can determine whether they need re-insert operation in the CI algorithm to optimize the result
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 is more efficient and spends much less time on some datasets than the traditional c program. See in the Benchmark Section
4. The traditional 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 and syntax 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 and it doesn’t rely on the other extra library

* + - * + **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 syntax errors.

* + - * + **Windows**

If you have the Visual Studio 2013 and higher version of Visual Studio, the user can directly open the ComplexCi.sln and compile the code in the IDE. The binary file will be generated in the x64/Release/ComplexCi.exe . 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**

The data of network can be written in one txt/csv files and the network is considered as undirected network. Each row contains 2 node IDs divided by one comma, which means there is a connection between these 2 nodes. The node IDs should be integer and started from 0. For example:

0,1

1,2

1,0

2,1

....

The complete Example is at data/test/karate.csv

* + - **Easy Start to use Script**

Using Script in the project is a quick start to utilize Collective Influence (CI) algorithm. There are 3 scripts the users can execute in the project.

1. traditionalCollectiveInfluence: The user can use this script to utilize traditional c style code of <http://www-levich.engr.ccny.cuny.edu/~hernanlab/uploads/CI_HEAP.c> . I merge it into this repository and set the compatible interface layer to call the c code in c++ .

Script “traditionalCollectiveInfluence” accepts 3 parameters:

./traditionalCollectiveInfluence.sh <networkPath> <ballRadius> <isPrintMinPointCausingMinComponent>

<networkPath> is the file path. The file format is described in the section Network Input File Format

<ballRadius> is the Radius parameter defined in the Collective Influence Algorithm

When the ballRadius is zero, pls notice that Collective Influence Algorithm will degenerate into HDA (high degree adaptive) algorithm. a.k.a. CI value of each node will be equal with degree of the node.

<isPrintMinPointCausingMinComponent> whether the output file contains limited point leading to 0.01 of giant component ratio or all points. If it is set to 0, the program will output all nodes. Otherwise the program will output partial points, which will make the giant component ratio to 0.01 in the deleting nodes process

e.g.

./traditionalCollectiveInfluence.sh /home/network/model1.csv 3 1  
  
 It means the users are using the traditionalCollectiveInfluence stragety for the input file model1.csv with parameter ballRadius 3. The output file only contain the partial points, which will make the giant component ratio to 0.01 in the deleting nodes process

1. cppCollectiveInfluence: The user can use this script to utilize new c++ implementation of Collective Influence (CI) algorithm.   
     
   ./cppCollectiveInfluence.sh <networkPath> <ballRadius> <updateBatch> <isPrintMinPointCausingMinComponent>

<networkPath> is the file path. The file format is described in the section Network Input File Format

<ballRadius> is the Radius parameter defined in the Collective Influence Algorithm

When the ballRadius is zero, pls notice that Collective Influence Algorithm will degenerate into HDA (high degree adaptive) algorithm. a.k.a. CI value of each node will be equal with degree of the node.

<updateBatch> batch size of deleting nodes in Network per updating CI values when the Complex Network collapses

<isPrintMinPointCausingMinComponent> whether the output file contains limited point leading to 0 of giant component ratio or all points. If it is set to 0, the program will output all nodes. Otherwise the program will output partial points, which will make the giant component ratio to 0 in the deleting nodes process. i.e. There is no edge but still left point in the network. Pls notice the different behaviour compared with traditionalCollectiveInfluence

e.g.

./cppCollectiveInfluence.sh /home/network/model1.csv 2 500 0

It means the users are using the cppCollectiveInfluence stragety for the input file model1.csv with parameter ballRadius 2. The output file only contain all nodes and 500 nodes will be deleted in a batch per updating CI values when the Complex Network collapses.

1. newReinsertCollectiveInfluence: This script involves the **new re-insert algorithm** of collective influence. After verified on the 8 datasets, this new algorithm can achieve better performance in the metric of Robustness than **original re-insert algorithm** in collective influence. See in the benchmark sections.  
     
   ./newReinsertCollectiveInfluence.sh <networkPath> <ballRadius> <updateBatch> <isPrintMinPointCausingMinComponent>

<networkPath> is the file path. The file format is described in the section Network Input File Format

<ballRadius> is the Radius parameter defined in the Collective Influence Algorithm

When the ballRadius is zero, pls notice that Collective Influence Algorithm will degenerate into HDA (high degree adaptive) algorithm. a.k.a. CI value of each node will be equal with degree of the node.

<updateBatch> batch size of deleting nodes in Network per updating CI values when the Complex Network collapses

<isPrintMinPointCausingMinComponent> whether the output file contains limited point leading to 0 of giant component ratio or all points. If it is set to 0, the program will output all nodes. Otherwise the program will output partial points, which will make the giant component ratio to 0 in the deleting nodes process. i.e. There is no edge but still left point in the network. Pls notice the different behaviour compared with traditionalCollectiveInfluence

e.g.

./newReinsertCollectiveInfluence.sh /home/network/model1.csv 2 500 0

It means the users are using the newReinsertCollectiveInfluence stragety for the input file model1.csv with parameter ballRadius 2. The output file only contain all nodes and 500 nodes will be deleted in a batch per updating CI values when the Complex Network collapses.

The project contains both bash scripts for linux and cmd scripts for windows. Bash for linux can be found in the dailyUse\linux and cmd for windows are in the dailyUse\windows

* + - **Directly use with Binary ComplexCi**If the user want to control the more flexible behaviour in the Collective Influence algorithm, binary program ComplexCi is able to achieve the goal.  
        
      ./ComplexCi [path] [ballRadius] [updateBatch] [outputNumBatch] [method] [biggestComponentEndThreshold] [isPrintMinPointCausingMinComponent]

<path> is the file path. The file format is described in the section Network Input File Format

<ballRadius> is the Radius parameter defined in the Collective Influence Algorithm

When the ballRadius is zero, pls notice that Collective Influence Algorithm will degenerate into HDA (high degree adaptive) algorithm. a.k.a. CI value of each node will be equal with degree of the node.

<updateBatch> batch size of deleted points per updating Collective Influence value (traditional ci will be fixed to 1). The user can determine the batch size of deleting nodes in ComplexCi per updating CI values when the Complex Network collapses

<outputNumBatch> output number of point each line in the final result file

<method> method of several algorithms in collective influence. Here the user can input the integer represented as method

1. 0 means the new c++ implementation of Collective Influence (CI) algorithm **without** re-insert process at the end
2. 1 means the traditional c style implementation (<http://www-levich.engr.ccny.cuny.edu/~hernanlab/uploads/CI_HEAP.c>) of Collective Influence (CI) algorithm **with original** re-insert process at the end. This method is also called by the script **traditionalCollectiveInfluence** mentioned above
3. 2 means the traditional c style implementation (<http://www-levich.engr.ccny.cuny.edu/~hernanlab/uploads/CI_HEAP.c>) of Collective Influence (CI) algorithm **without** re-insert process at the end
4. 3 is not used here.
5. (Experimental) 4 means c++ **concurrent** implementation of Collective Influence (CI) algorithm **without** re-insert process at the end (method 1). However , the algorithm is not mature and even sometimes it will be slower than single thread for the small dataset. Because forking thread is an expensive operation on the operation system level and there is no internal thread pool in c++.
6. 5 means the new c++ implementation of Collective Influence (CI) algorithm **with original** re-insert process at the end using **more strict** internal parameters. In detail , there are 2 internal parameters changed here.
   1. computeComponentInterval: It means the Interval to calculate Giant Component when Complex Network Collapses. The original is 1 Percent of the size of complete network. For the more strict parameter , the value is re-scaled to 200 if the original is larger than 200.
   2. reinsertEachStep: It means the batch size of nodes reinserted into the network per updating the graph. The original is 1 Per mille of the size of complete network. For the more strict parameter , the value is re-scaled to 20 if the original is larger than 20.
7. (Experimental) 6 means c++ **concurrent** implementation of Collective Influence (CI) algorithm **with original** re-insert process at the end in **more strict** internal parameters (method 5). However , the algorithm is not mature and even sometimes it will be slower than single thread for the small dataset. Because forking thread is an expensive operation on the operation system level and there is no internal thread pool in c++.
8. 7 means the new c++ implementation of Collective Influence (CI) algorithm **with new** re-insert process at the end using **more strict** internal parameters. In detail , there are 2 internal parameters changed here.
   1. computeComponentInterval: It means the Interval to calculate Giant Component when Complex Network Collapses. The original is 1 Percent of the size of complete network. For the more strict parameter , the value is re-scaled to 200 if the original is larger than 200.
   2. reinsertEachStep: It means the batch size of nodes reinserted into the network per updating the graph. The original is 1 Per mille of the size of complete network. For the more strict parameter , the value is re-scaled to 20 if the original is larger than 20.
9. (Experimental) 8 means c++ **concurrent** implementation of Collective Influence (CI) algorithm **with new** re-insert process at the end in **more strict** internal parameters (method 7). However , the algorithm is not mature and even sometimes it will be slower than single thread for the small dataset. Because forking thread is an expensive operation on the operation system level and there is no internal thread pool in c++.
10. 9 means the new c++ implementation of Collective Influence (CI) algorithm **with original** re-insert process at the end using **the same** internal parameters. This method is also called by the script **cppCollectiveInfluence** mentioned above. In detail , there are 2 same internal parameters here.
    1. computeComponentInterval: It means the Interval to calculate Giant Component when Complex Network Collapses. The original is 1 Percent of the size of complete network.
    2. reinsertEachStep: It means the batch size of nodes reinserted into the network per updating the graph. The original is 1 Per mille of the size of complete network.
11. 10 means the new c++ implementation of Collective Influence (CI) algorithm **with new** re-insert process at the end using **the same** internal parameters. This method is also called by the script **newReinsertCollectiveInfluence** mentioned above. In detail , there are 2 same internal parameters here.
    1. computeComponentInterval: It means the Interval to calculate Giant Component when Complex Network Collapses. The original is 1 Percent of the size of complete network.
    2. reinsertEachStep: It means the batch size of nodes reinserted into the network per updating the graph. The original is 1 Per mille of the size of complete network.
12. (Experimental) 11 is based on method 6 . Another enhanced re-insert algorithm is applied on it ,which will choose rank **behind** in the sequence of node importance with high priority. Method 6 will just ignore the rank information. This method is also the **concurrent** implementation of the method 15.
13. (Experimental) 12 is based on method 8 . Another enhanced re-insert algorithm is applied on it ,which will choose rank **behind** in the sequence of node importance with high priority. Method 8 will just ignore the rank information. This method is also the **concurrent** implementation of the method 16.
14. (Experimental) 13 is based on method 8. Reinsert algorithm will use the **multiple** kernel when combining giant component infomation . This method is also the **concurrent** implementation of the method 17.
15. (Experimental) 14 is based on method 13. Reinsert algorithm will choose rank **behind** in the sequence of node importance with high priority combined with **multiple** kernel. This method is also the **concurrent** implementation of the method 18.
16. (Experimental) 15 is based on method 6 . Another enhanced re-insert algorithm is applied on it ,which will choose rank **behind** in the sequence of node importance with high priority. Method 6 will just ignore the rank information. This method is also the **single** thread implementation of the method 11.
17. (Experimental) 16 is based on method 8 . Another enhanced re-insert algorithm is applied on it ,which will choose rank **behind** in the sequence of node importance with high priority. Method 8 will just ignore the rank information. This method is also the **single** thread implementation of the method 12.
18. (Experimental) 17 is based on method 8. Reinsert algorithm will use the **multiple** kernel when combining giant component infomation . This method is also the **single** implementation of the method 13.
19. (Experimental) 18 is based on method 17. Reinsert algorithm will choose rank **behind** in the sequence of node importance with high priority combined with **multiple** kernel. This method is also the **single** thread implementation of the method 14.

I believe the user can just use the method without Experimental tag for their daily use becasue these methods with Experimental tag do not take effect obviously for most Complex Network datasets. There is no need for the user to get the knowledge of these method in most cases. Pls ignore them. See in the DataCastle Comptition Section.

<biggestComponentEndThreshold> complex network collapses to the certain giant component ratio where the reinsert algorithm starts for cpp ci (traditional ci will be fixed to 0.01)

<isPrintMinPointCausingMinComponent> whether output limited point leading to 0(new cpp ci)/0.01(traditional ci) of giant component ratio or all points. If it is set to 0, the program will output all nodes. Otherwise the program will output partial nodes. This input parameter contains the same meaning with the <isPrintMinPointCausingMinComponent> in the script traditionalCollectiveInfluence, cppCollectiveInfluence and newReinsertCollectiveInfluence.

If the user compiles correctly , the binary program will be in the bin/ComplexCi on the linux platform and x64/Release/ComplexCi.exe on the windows platform.

* + - **Output file**

When the algorithm finishes, the output file will be generated in the same folder of the original file with suffix “\_out”. Each row in the output file contains nodes, whose count is a group of <outputNumBatch> defined in runing ComplexCi binary. If the user runs the script, the value <outputNumBatch> will be fixed to 500, for instance：

model1,4,5,2,33,…,2214

model1,2432,4554,3222,1123,…,2233

…

From the above instance, the nodes of networks model1 will be removed following such order: 4, 5, 2, 33, …, 2214, 2432, 4554, 3222, 1123, …, 2233… . If nodes count % outputNumBatch !=0, some comma will be at the last line of the output.

* + **Benchmark**

In order to demonstrate the performance of 3 main algorithms mentioned in the scripts traditionalCollectiveInfluence, cppCollectiveInfluence and newReinsertCollectiveInfluence, I provide 8 test datasets downloaded from DataCastle Competition at GITHUB LINK . Here the metric of Robustness is another measure to quantify the performance of ranking methods introduced by the paper

Schneider C M, Moreira A A, Andrade J S, et al. Mitigation of malicious attacks on networks[J]. Proceedings of the National Academy of Sciences, 2011, 108(10): 3838-3841.

The smaller value is, the better the algorithm is.

BenchMark TABLE

From the benchmark ,we can see that the result of traditional c implementation traditionalCollectiveInfluence and new c++ cppCollectiveInfluence can both achieve nearly the same result in the metric of Robustness, even the new c++ implementation is more efficient and spends much less time on some datasets than the traditional c program. I am using the data structure of disjoint-set in the reinsertion in the new c++ implementation ComplexCi and it can boost a lot. The traditionalCollectiveInfluence didn’t use this data structure and I think that’s the reason why the traditional c program was slow.

We can also see that newReinsertCollectiveInfluence can achieve the better Robust

* + **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**