## Summary

The project is about a new hypothesis: soft graph colouring can be done by finding communities in the complement graph. The aim of this project is to test the hypothesis on unweighted graphs. Various community detection algorithms will be tested on the complement graph, and evaluated according to how well the colouring constrains in the original graph are preserved.

Main contributions and achievements:

- I created six graphs and coloured them, see pages 6-10.
- I wrote a program to transform the original graphs to the complement graphs, see pages 11-14.
- I found a large network with 964 vertices and 18707 edges, see page 15.
- I chose five community detection algorithms to test the six graphs, see pages 16-46.
- I wrote a program to find and count the conflict edges for each community, see page16-50.
- I chose four community detection algorithms to test the large network, see pages 47-50.
- I evaluated those results and compared them with soft graph colouring algorithm, see pages 51-55.

# Table of contents

# 1. Introduction

Graph colouring algorithms are widely used for solving scheduling and resource allocation problems. Hard graph colouring is the traditional graph colouring and colour all the vertices in a graph with no two adjacent vertices have the same colour. Using the method of cliques-finding algorithm can approximate associate the problem of hard graph colouring, which is a NP hard. In soft graph colouring, a pair of vertices connected by an edge may be given the same colour, but the number of such edges is minimized.

As is well known to all, simple graph colouring is equivalent to finding cliques in the complement graph. We believe that soft graph colouring can be done by finding communities in the complement graph. Detecting communities in graphs is to identify the modules and the hierarchical organization, by only using the information encoded in the graph topology. The problem has a long tradition and there are hundreds of community detection (CD) algorithms. [1]

The aim of this project is to test the hypothesis on unweighted graphs. Various community detection algorithms will be tested on the complement graph, and evaluated according to how well the colouring constrains in the original graph are preserved. The process of the project has four steps. First of all, renew the original network to the complement graph. Secondly, find communities by community detection algorithms in the complement graph. Next, transform the communities to the colours. Finally, evaluate the result according to the conflict edges.

In this report I try to cover in some detail the work done in the project. The report can be divided into six parts. Part two show all the networks in this project, with descriptions in detail, illustrating the original graph and complement. Next, experiments for small graphs are shown in part three, while part four analyzes the experiments for the large network. Those two parts have an explanation for the community detection algorithms, and show examples and results of each experiment. Evaluation of the results is in part five. The final part is conclusion for the whole project.

# 2. Networks

Before I begin the experiments, it is important to find some example problems, and then start using the proposed method to solve them. Those example problems are some networks that need to be coloured using soft graph colouring.

## 2.1 Six graphs

Those might be for some scheduling problem like scheduling exams, in which case the vertices represent exams and the edges represent the fact that two exams are taken by the same students. That is the reason I created six graphs (graph 0 to graph 5) based on the 2010 MSc timetable of computer science department [4]. Those small graphs are good to test CD algorithms at first.

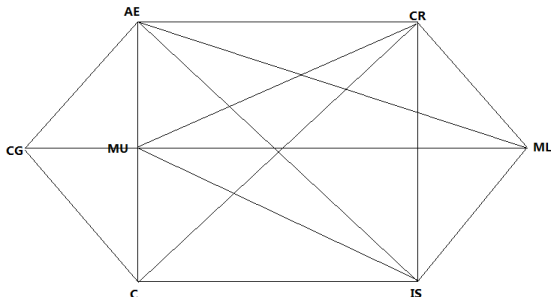


Figure 1: Graph 5

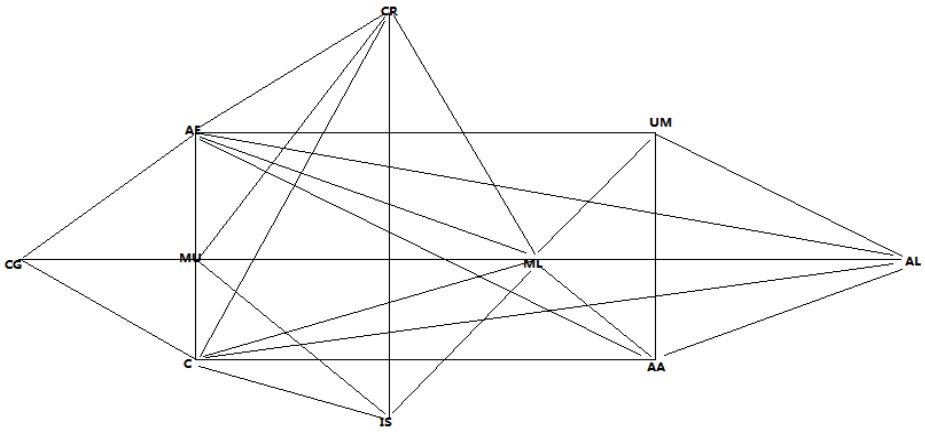


Figure 2: Graph 4

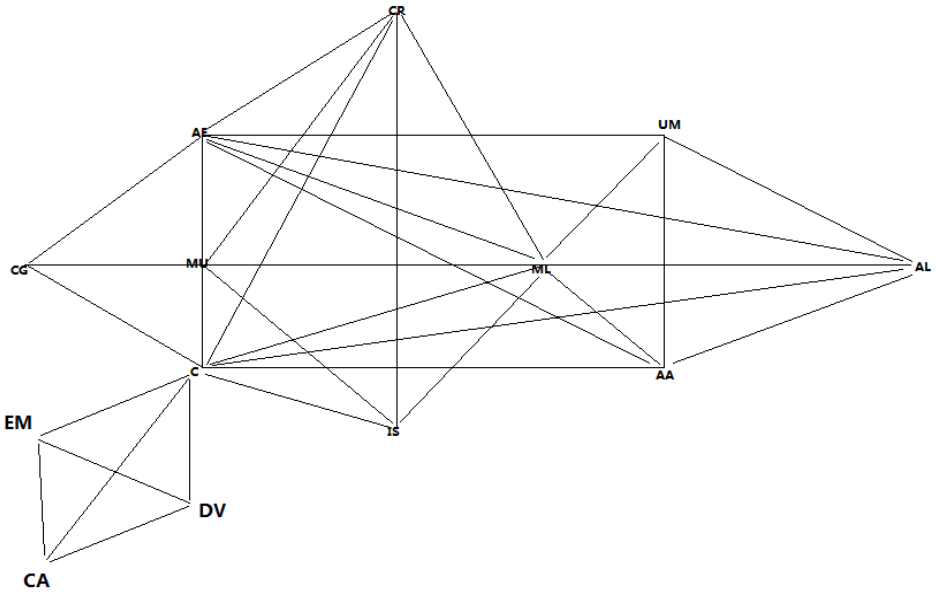


Figure 3: Graph 3

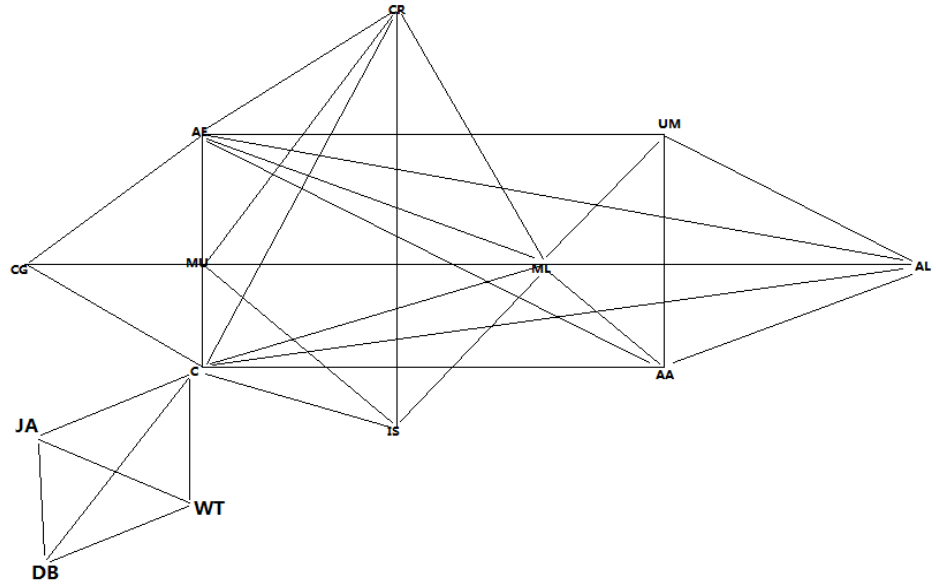


Figure 4: Graph 2

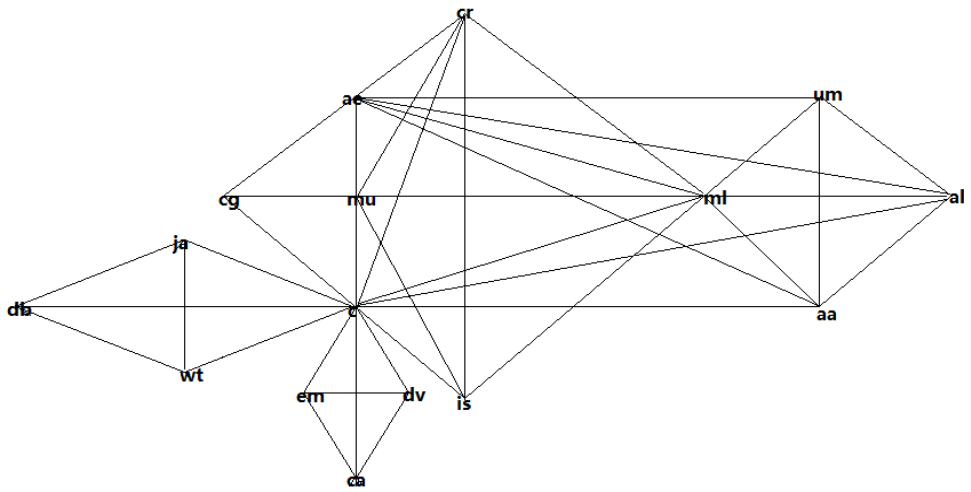


Figure 5: Graph 1

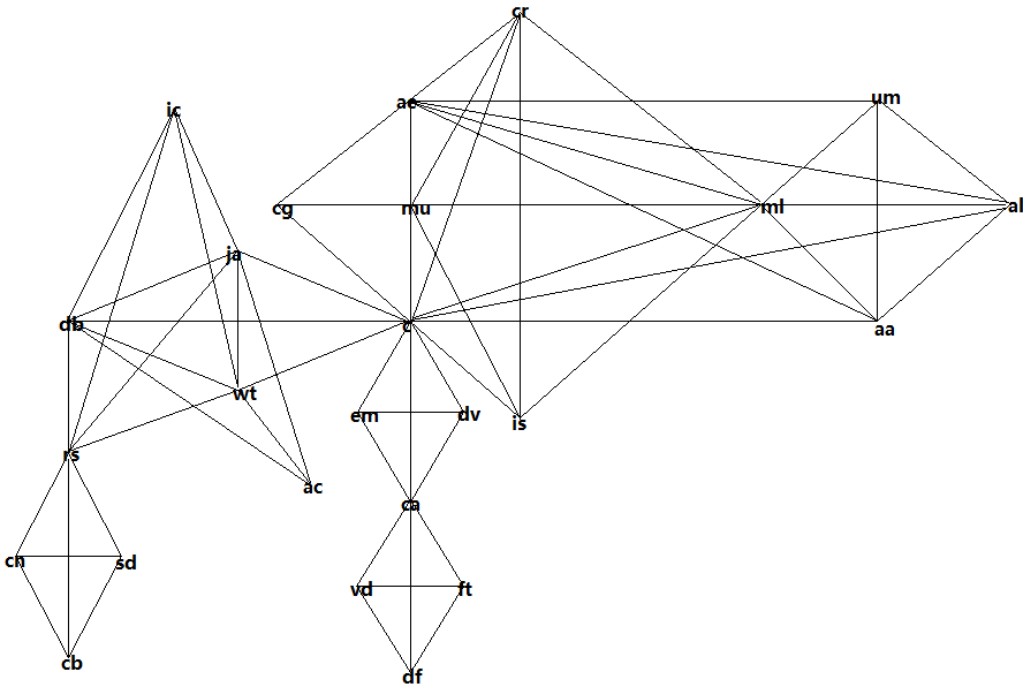


Figure 6: Graph 0

Next, it is good to know the number of hard graph colouring for each graph. In hard graph colouring (HGC), two vertices shared one edge must be given different colours. For the whole graph, make the number of colours as minimized as possible. All the six graphs can be HGC by 5 colours. I believe that five is the minimum colours of HGC for each graph, but the figures under show only one case of five colours in the original graphs.

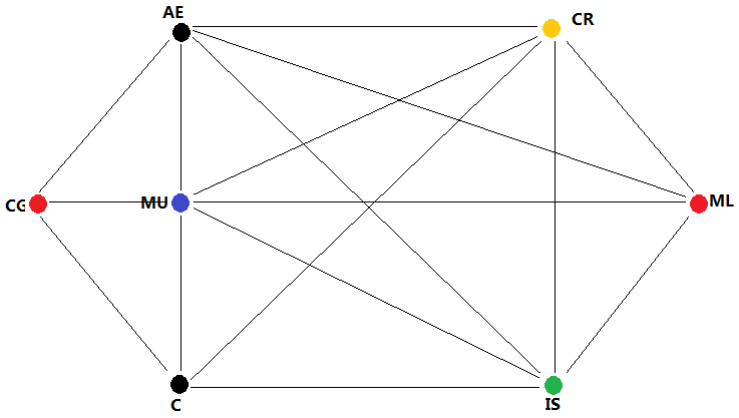


Figure 7: HGC for graph 5

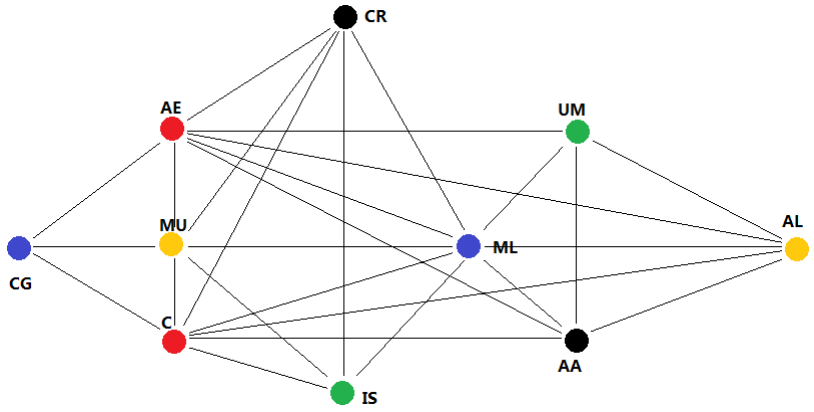


Figure 8: HGC for graph 4

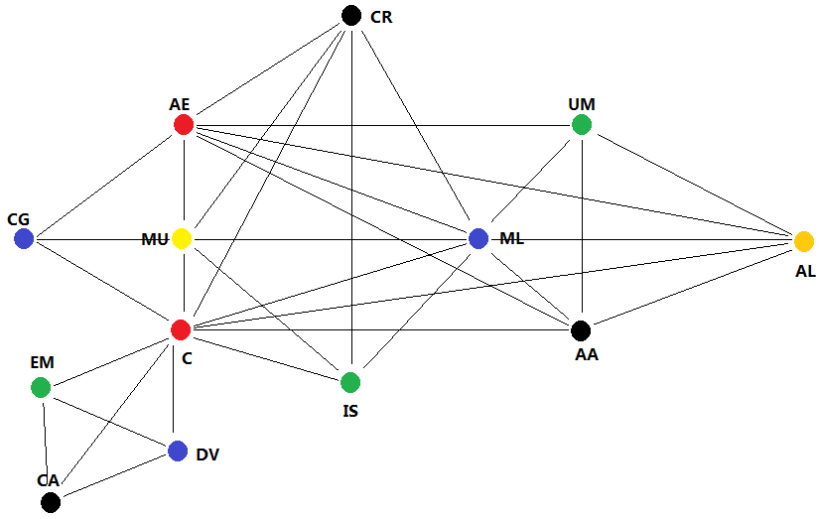


Figure 9: HGC for graph 3

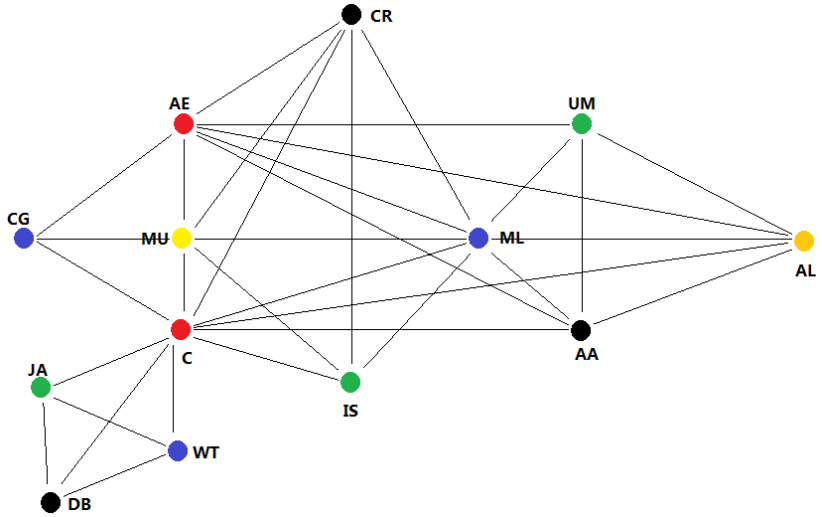


Figure 10: HGC for graph 2

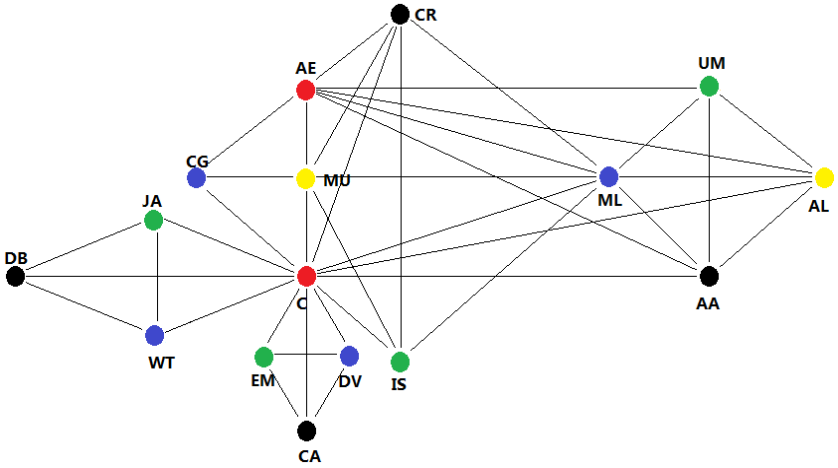


Figure 11: HGC for graph 1

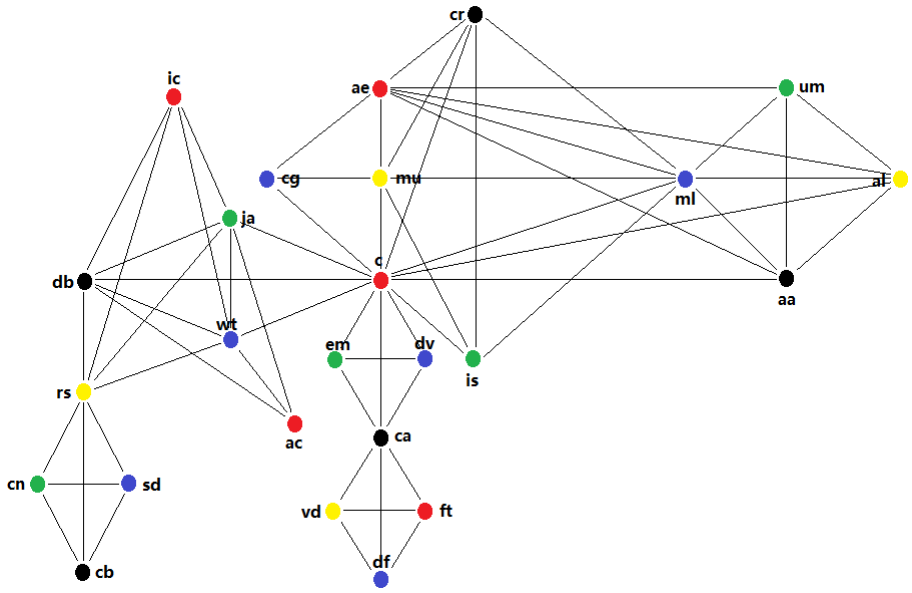

Figure 12: HGC for graph 0

To transform the original graphs to the complement graphs, I wrote a simply program to do so. It shows all the edges in the complement graphs. The format of output file is: vertex1 tab vertex2

The complement of graph 5:

ae c
cr cg
ml c
ml cg
is cg

The complement of graph 4:

cr um
cr al
cr aa
cr cg
um cg
um mu
um c
um is
al cg
al mu
al is
aa cg
aa mu
aa is
ml cg
is ae
is cg
c ae

The complement of graph 3:

c um
c ae
cg al
cg ml
cg aa
cg is
cg um
cg cr
cg em
cg ca
cg dv
mu um
mu al
mu aa
ae is
al is
al cr
aa is
aa cr
is um
um cr
em cr
em um
em is
em aa
em ml
em al
em ae
em mu
em cg
ca cr
ca um
ca is
ca aa
ca ml
ca al
ca ae
ca mu
ca cg
dv cr
dv um
dv is
dv aa
dv ml
dv al
dv ae
dv mu
dv cg

The complement of graph 2:

c um
c ae
cg al
cg ml
cg aa
cg is
cg um
cg cr
cg db
cg ja
cg wt
mu um
mu al
mu aa
ae is
al is
al cr
aa is
aa cr
is um
um cr
wt cr
wt um
wt is
wt aa
wt ml
wt al
wt ae
wt mu
wt cg
ja cr
ja um
ja is
ja aa
ja ml
ja al
ja ae
ja mu
ja cg
db cr
db um
db is
db aa
db ml
db al
db ae
db mu
db cg

The complement of graph 1:

c um
c ae
em wt
em ja
em db
em cr
em um
em is
em aa
em ml
em al
em ae
em mu
em cg
ca wt
ca ja
ca db
ca cr
ca um
ca is
ca aa
ca ml
ca al
ca ae
ca mu
ca cg
dv wt
dv ja
dv db
dv cr
dv um
dv is
dv aa
dv ml
dv al
dv ae
dv mu
dv cg
cg al
cg ml
cg aa
cg is
cg um
cg cr
cg db
cg ja
cg wt
mu um
mu al
mu aa
ae is
al is
al cr
aa is
aa cr
is um
um cr
wt em
wt ca
wt dv
wt cr
wt um
wt is
wt aa
wt ml
wt al
wt ae
wt mu
wt cg
ja em
ja ca
ja dv
ja cr
ja um
ja is

ja aa
ja ml
ja al
ja ae
ja mu
ja cg

db em
db ca
db dv
db cr
db um
db is

db aa
db ml
db al
db ae
db mu
db cg

The complement of graph 0:

c um
c ae
c vd
c ft
c df
c ic
c ac
c rs
c cn
c sd
c cb
em wt
em ja
em db
em cr
em um
em is
em aa
em ml
em al
em ae
em mu
em cg
em vd
em ft
em df
em ac
em ic
em rs
em cn
em sd
em cb
ca wt
ca ja
ca db
ca cr

ca um
ca is
ca aa
ca ml
ca al
ca ae
ca mu
ca cg
ca ic
ca ac
ca rs
ca cn
ca sd
ca cb
dv wt
dv ja
dv db
dv cr
dv um
dv is
dv aa
dv ml
dv al
dv ae
dv mu
dv cg
dv vd
dv df
dv ft
dv ic
dv ac
dv rs
dv cn
dv sd
dv cb
cg al

cg ml
cg aa
cg is
cg um
cg cr
cg db
cg ja
cg wt
cg ic
cg ac
cg rs
cg cn
cg sd
cg cb
cg vd
cg ft
cg df
mu um
mu al
mu aa
mu ic
mu ac
mu rs
mu cn
mu sd
mu cb
mu vd
mu ft
mu df
ae is
ae ic
ae ac
ae rs
ae cn
ae sd
ae cb

ae vd
ae ft
ae df
al is
al cr
al ic
al ac
al rs
al cn
al sd
al cb
al vd
al ft
al df
aa is
aa cr
aa ic
aa ac
aa rs
aa cn
aa sd
aa cb
aa vd
aa ft
aa df
is um
is ic
is ac
is rs
is cn
is sd
is cb
is vd
is ft
is df
um cr
um ic
um ac
um rs
um cn
um sd
um cb
um vd
um ft
um df
ml ic
ml ac
ml rs
ml cn
ml sd
ml cb
ml vd
ml ft
ml df
cr ic
cr ac
cr rs
cr cn
cr sd
cr cb
cr vd
cr ft
cr df
wt em
wt ca
wt dv
wt cr
wt um
wt is
wt aa
wt ml
wt al
wt ae
wt mu
wt cg
wt sd
wt cn
wt cb
ja em
ja ca
ja dv
ja cr
ja um
ja is
ja aa
ja ml
ja al
ja ae
ja mu
ja cg
ja sd
ja cn
ja cb
db em
db ca
db dv
db cr
db um
db is
db aa
db ml
db al
db ae
db mu
db cg
db sd
db cn
db cb

The table below shows the information of those graphs.

| ID | Vertices | Edges (Original) | Edges (Complement) | Hard graph colouring (HGC) |
|---|---|---|---|---|
| 0 | 25 | 61 | 216 | 5 |
| 1 | 16 | 39 | 93 | 5 |
| 2 | 13 | 33 | 48 | 5 |
| 3 | 13 | 33 | 48 | 5 |
| 4 | 10 | 27 | 18 | 5 |

| | | | | |
|---|---|---|---|---|
| 5 | 7 | 16 | 5 | 5 |

Table 1: Six graphs. The first column is ID number of each graph. Column two and three show the amount of vertices and edges in the original graphs, respectively. The fourth column means the number of edges in the complement graphs. Column five represents colours used for hard graph colouring.

Another simple program is written to count conflict edges. If two vertices are given the same colour, in other words, the two vertices are both in one community, and then the edge between them is a conflict edge. As can be seen in figure 7 to 12, there are not any conflict edges in hard graph colouring.

## 2.2 Large network

It is easier to test CD algorithms using small examples like those. However, I need a large one. It should be a real network that needs colouring. E.g., a scheduling network or a register interference graph. It might be worth looking at a register allocation example. Soft graph colouring could be useful there, because there is only a fixed number of registers (colours) available.

I found a large network named “inithx.i.1” from Ref. [5]. The network is based on register allocation for variables in real codes. It has 864 vertices and 18707 edges. The chromatic number is 54.

# 3. Experiments for small examples

The main point of this project is experiments. To see if community detection algorithms can be used for soft graph colouring, and check how good the graph colouring results are. However, there are hundreds of algorithms. For my purposes, I do not need a fast one yet but it is good to have one that is tunable, because I want to be able to control the number of communities. The reason for choosing those algorithms below is they are tunable.

## 3.1 GN algorithm experiments

3.1.1 GN algorithm

The algorithm of Girvan and Newman (GN) is a popular way to detect communities in graphs by finding the edges that connect vertices of different communities and eliminate them. That makes the clusters get separated from each other. One important point is to focus on the attributes of those edges and easy to detection. [1]

This is a good example to show intercommunity edge.

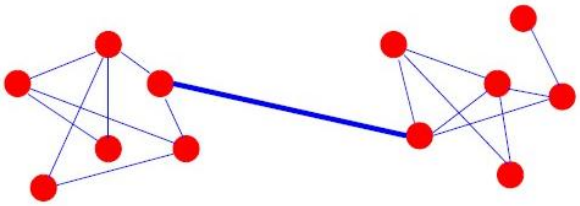

Figure 13: GN algorithm. The picture of community detection in graphs of Santo Fortunato is copied from Ref. [1].

The steps of GN algorithm can be listed as follow:

1. Estimation of the importance of all edges by their attributes;
2. Eliminate the most important edge;
3. Recomputation of the importance for the rest edges;
4. Repeat step 2. [1]

3.1.2 GN algorithm for soft graph colouring

I found a program named “CONGA/CONGO/Peacock Software” from Ref. [6], and tested the six graphs (I mentioned before) by GN algorithm.

The steps of the experiment are:

Step 1: run the software and find four communities in the complement graph

Step 2: identify conflict edges in each community

Step 3: count the number of conflict edges

I did the three steps for each graph by GN algorithm.

3.1.2.1 Example one

Step 1: the communities are:

cg cr is

c ml
ae
mu

Step 2: the conflict edge is CR-IS.

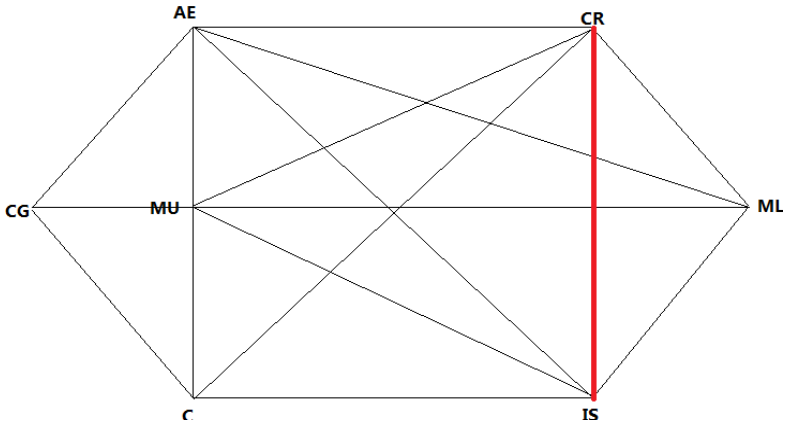


Figure 14: Conflict edge in graph 5. As can be seen in this picture, the red edge is conflict edge.

Step 3: the number of conflict edges is 1.

3.1.2.2 Example two

Step 1: the communities are:

aa al cg cr is um
ae c
mu
ml

Step 2: the conflict edges are:

CR-IS, UM-AL, AL-AA, AA-UM

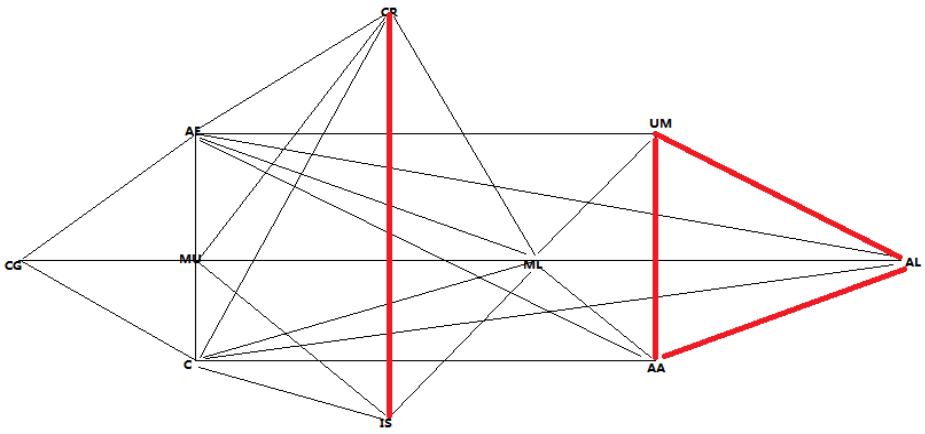


Figure 15: Conflict edges in graph 4. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 4.

3.1.2.3 Example three

Step 1: the communities are:

aa al ca cg cr dv em is mu um

c

ml

ae

Step 2: the conflict edges are:

CR-IS, UM-AL, AL-AA, AA-UM, CG-MU, MU-CR, IS-MU, CA-EM, EM-DV, DV-CA

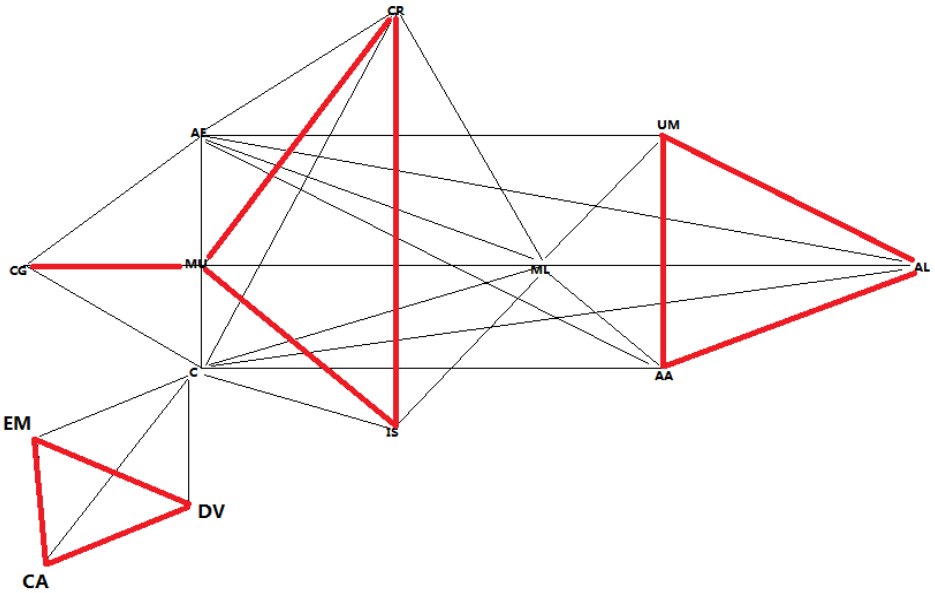


Figure 16: Conflict edges in graph 3. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 10.

3.1.2.4 Example four

Step 1: the communities are:

aa al cg cr db is ja mu um wt

c

ml

ae

Step 2: the conflict edges are:

CR-IS, UM-AL, AL-AA, AA-UM, CG-MU, MU-CR, IS-MU, DB-JA, JA-WT, WT-DB

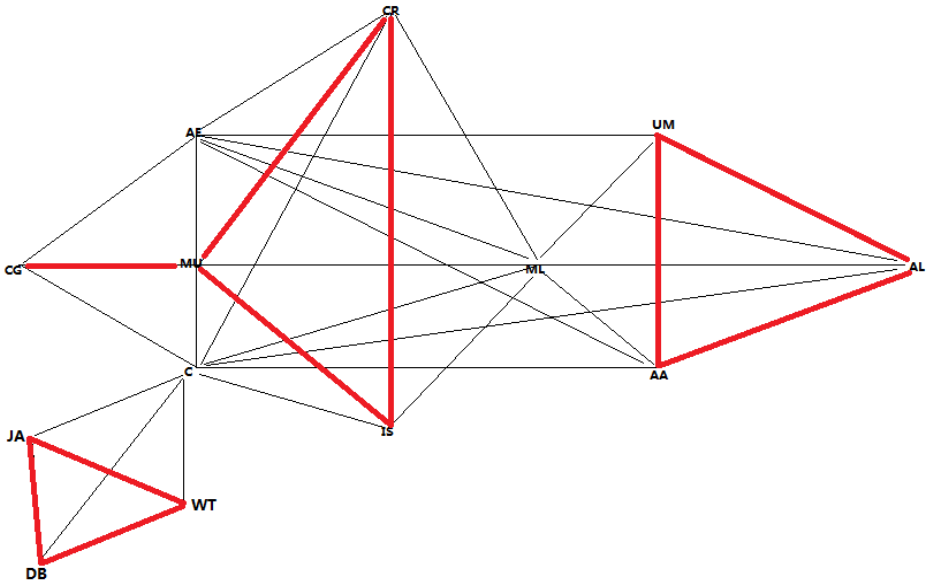


Figure 17: Conflict edges in graph 2. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 10.

3.1.2.5 Example five

Step 1: the communities are:

aa al ca cg cr db dv em is ja um wt mu

c

ml

ae

Step 2: the conflict edges are:

CR-IS, UM-AL, AL-AA, AA-UM, CG-MU, MU-CR, IS-MU, CA-EM, EM-DV, DV-CA, DB-JA, JA-WT, WT-DB

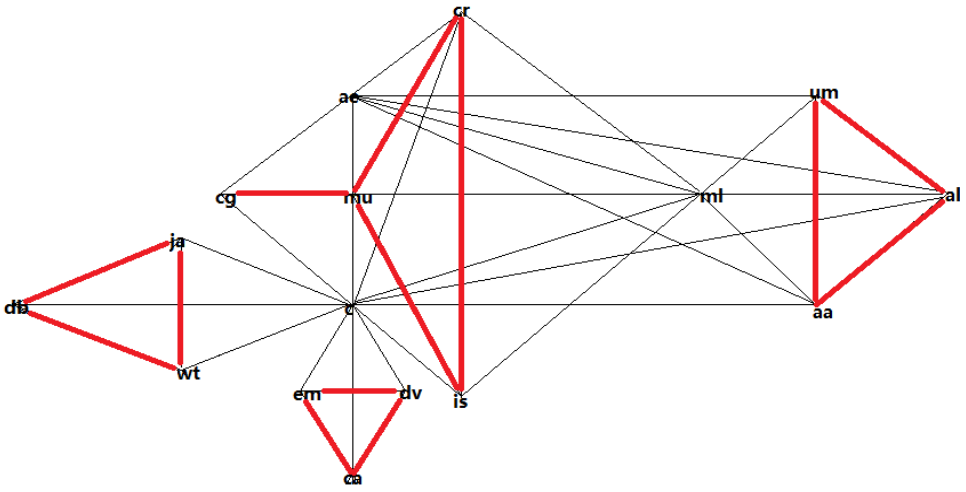


Figure 18: Conflict edges in graph 1. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 13.

3.1.2.6 Example six

Step 1: the communities are:

aa ac ae al ca cb cg cn cr db df dv em ic is ja ml mu rs sd um wt

vd

ft

c

Step 2: the conflict edges are:

CN-CB, DB-IC, DB-AC, EM-DV, DV-CA, CA-EM, CA-DF, CR-AE, CR-IS, AE-AL, AE-AA AA-AL, UM-ML, ML-MU, JA-WT, JA-RS, RS-WT, RS-SD, UM-AL,UM-AA, UM-AE, ML-AL, ML-AA, ML-CR, ML-AE, ML-IS, MU-CR, AE-MU, IS-MU, IC-JA, WT-IC, IC-RS, DB-JA, RS-DB, WT-DB, AC-WT, JA-AC, CN-RS, CB-SD, CB-RS, CN-SD, CG-AE, MU-CG

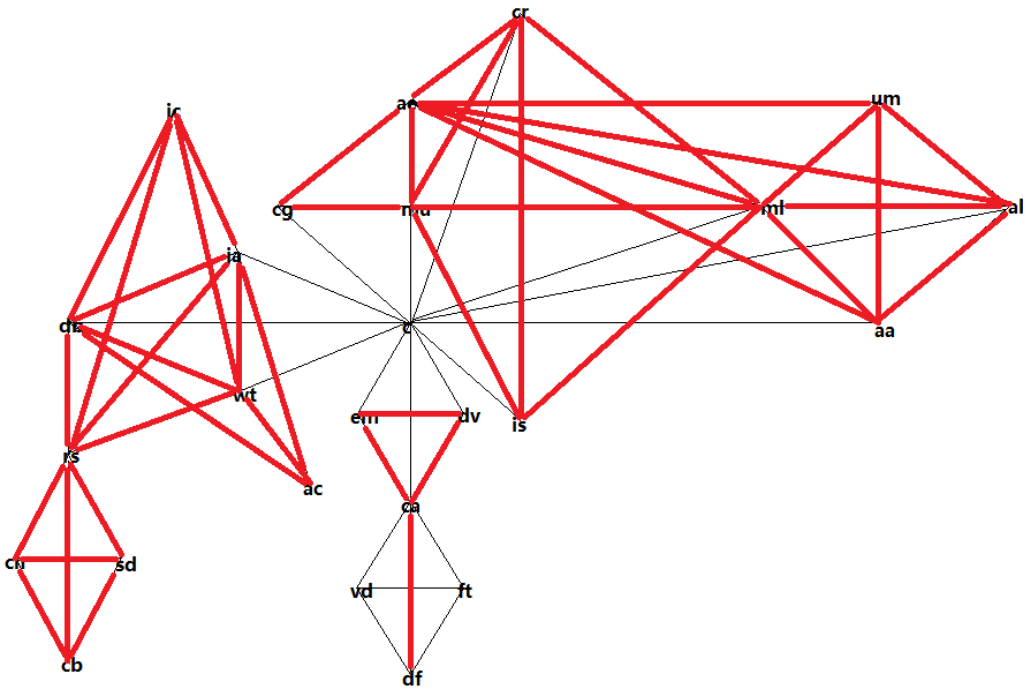


Figure 19: Conflict edges in graph 0. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 43.

3.1.3 Summary for GN algorithm

All the results of GN algorithm can be listed as follow:

| ID | Vertices | Edges | Hard graph colouring (HGC) | Conflict edges |
|---|---|---|---|---|
| 0 | 25 | 61 | 5 | 43 |
| 1 | 16 | 39 | 5 | 13 |
| 2 | 13 | 33 | 5 | 10 |
| 3 | 13 | 33 | 5 | 10 |

| | | | | |
|---|---|---|---|---|
| 4 | 10 | 27 | 5 | 4 |
| 5 | 7 | 16 | 5 | 1 |

Table 2: Results of GN algorithm. The first column is ID number of each graph. Column two and three show the amount of vertices and edges in the graphs, respectively. The fourth column means the number of colours used for hard graph colouring. Column five shows the amount of conflict edges when I use 4 colours.

As can be seen from this table, the number of conflict edges is increased by vertices addition. In graph 0, there are 43 conflicts, which is 70% of total edges. It seems that GN algorithm is not good at soft graph colouring.

## 3.2 CM+BK algorithm experiments

3.2.1 CM+BK algorithm

The main idea of CM+BK algorithm is cliques merging and Bron Kerbosch (BK) algorithm.

The steps of CM+BK algorithm can be listed as follow:

1. Expand candidate cliques
2. Output result if the algorithm find a clique
3. Delete all vertices of the clique in data structures
4. Repeat step 1 until the cliques are maximal [2]

3.2.2 CM+BK algorithm for soft graph colouring

I found a program named “CliqueMod Software” from Ref. [7], and tested the six graphs (I mentioned before) by CM+BK algorithm.

The steps of the experiment are:

Step 1: run the software and find four communities in the complement graph

Step 2: identify conflict edges in each community

Step 3: count the number of conflict edges

I did the three steps for each graph by CM+BK algorithm.

3.2.2.1 Example one

Step 1: the communities are:

cr is cg

c ae

ml

mu

Step 2: the conflict edge is CR-IS.

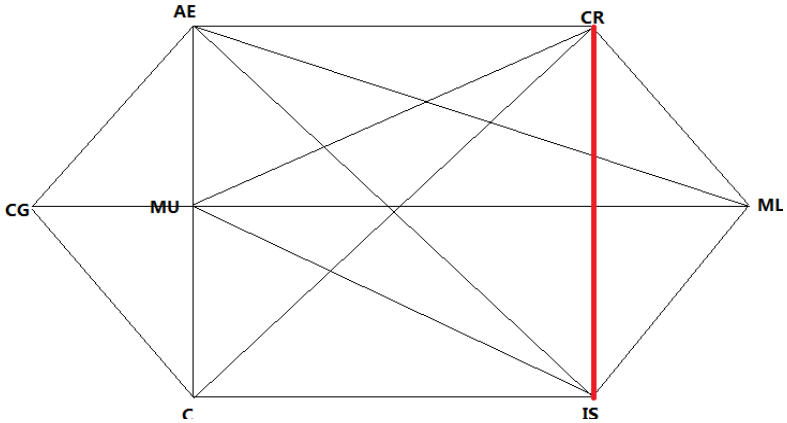

Figure 20: Conflict edge in graph 5. As can be seen in this picture, the red edge is conflict edge.

Step 3: the number of conflict edges is 1.

3.2.2.2 Example two

Step 1: the communities are:

cr al cg ml

is

c um ae

mu aa

Step 2: the conflict edges are:

CR-ML, ML-AL, UM-AE

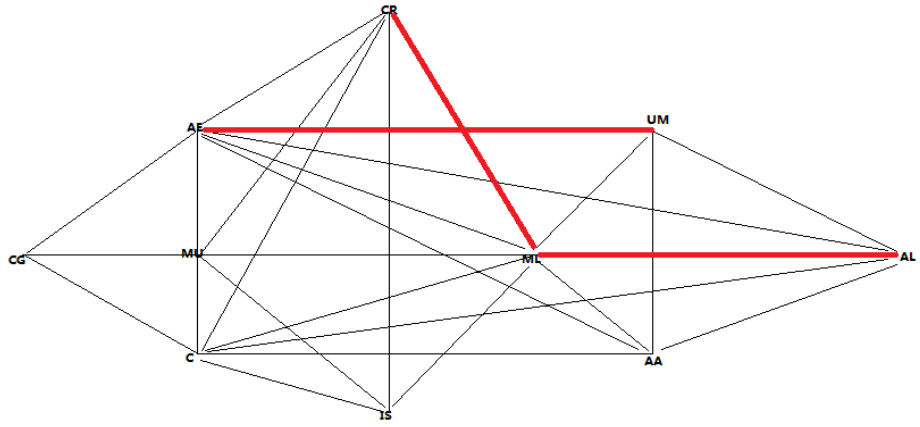

Figure 21: Conflict edges in graph 4. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 3.

3.2.2.3 Example three

Step 1: the communities are:

dv is cg um ml

al ca mu

cr aa em
c ae

Step 2: the conflict edges are:
ML-UM, IS-ML

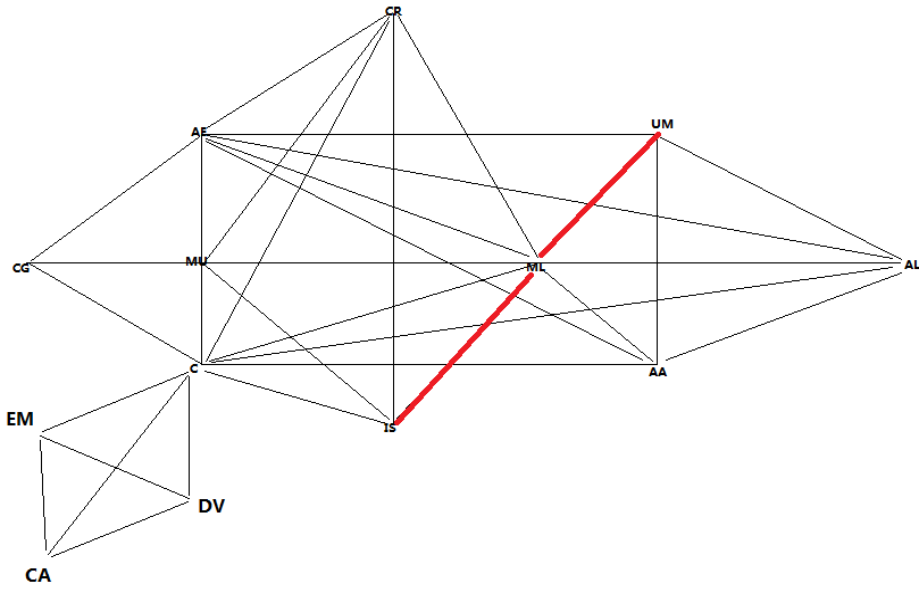


Figure 22: Conflict edges in graph 3. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 2.

3.2.2.4 Example four
Step 1: the communities are:
db is cg um ml
al mu wt
cr aa ja
c ae

Step 2: the conflict edges are:
ML-UM, IS-ML

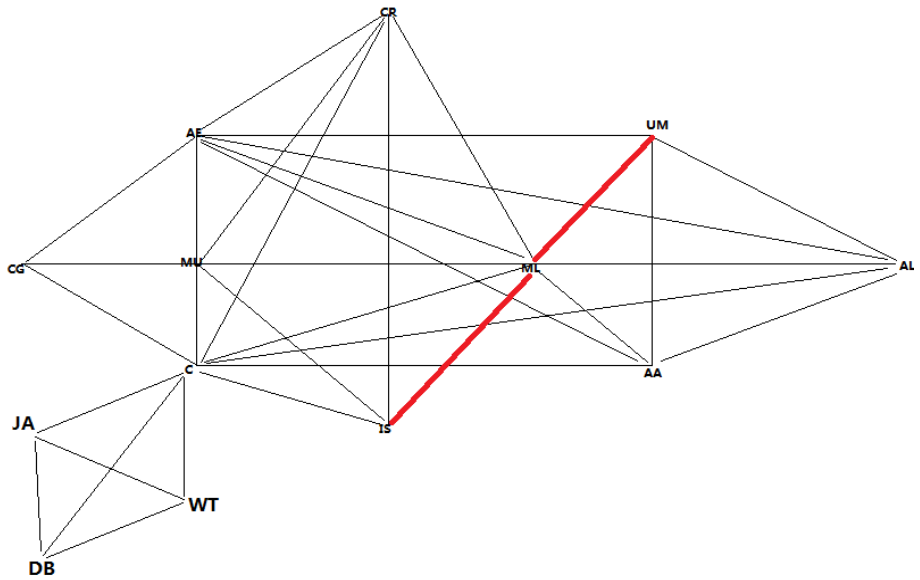


Figure 23: Conflict edges in graph 2. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 2.

3.2.2.5 Example five

Step 1: the communities are:

db is dv cg um ml

al ca mu ja

cr aa em wt

c ae

Step 2: the conflict edges are:

ML-UM, IS-ML

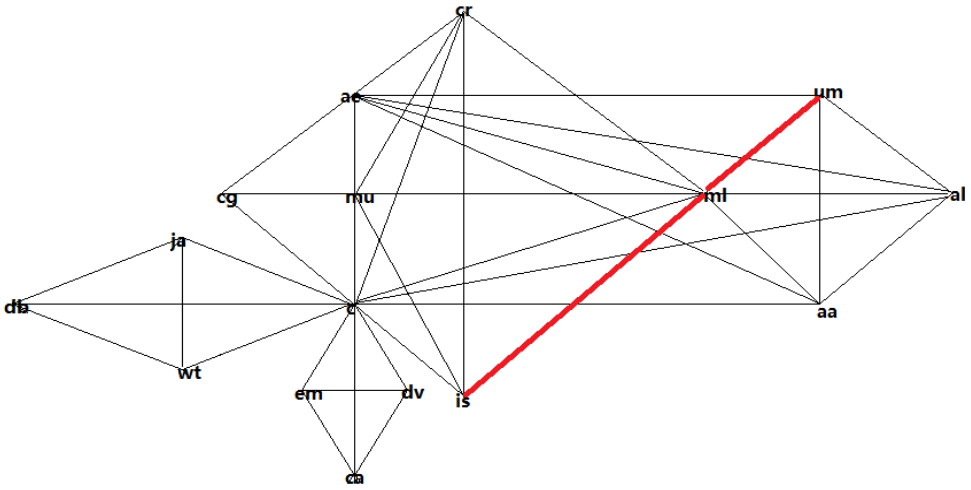


Figure 24: Conflict edges in graph 1. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 2.

3.2.2.6 Example six

Step 1: the communities are:

db is dv cb sd cg um wt ml

al df cn mu em ja

cr ca ac rs aa

ic c vd ft ae

Step 2: the conflict edges are:

Group 1: CB-SD, DB-WT, ML-UM, IS-ML

Group 2: VD-FT

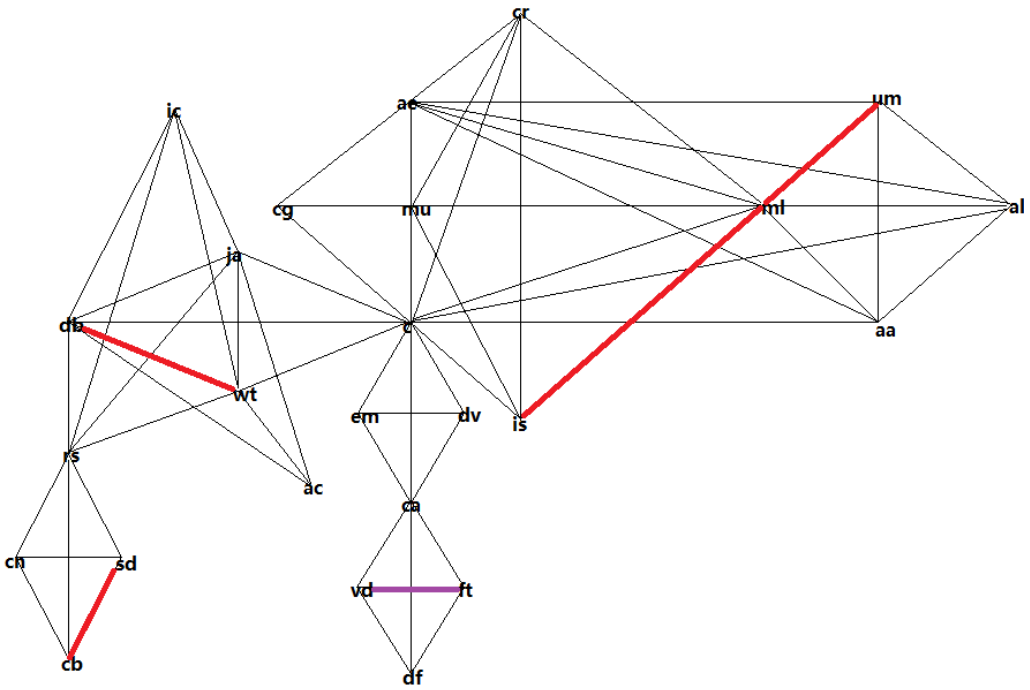


Figure 25: Conflict edges in graph 0. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 3: the number of conflict edges is 5.

3.2.3 Summary for CM+BK algorithm

All the results of CM+BK algorithm can be listed as follow:

| ID | Vertices | Edges | Hard graph colouring (HGC) | Conflict edges |
|---|---|---|---|---|
| 0 | 25 | 61 | 5 | 5 |
| 1 | 16 | 39 | 5 | 2 |
| 2 | 13 | 33 | 5 | 2 |
| 3 | 13 | 33 | 5 | 2 |
| 4 | 10 | 27 | 5 | 3 |
| 5 | 7 | 16 | 5 | 1 |

Table 3: Results of CM+BK algorithm. The first column is ID number of each graph.

Column two and three show the amount of vertices and edges in the graphs, respectively. The fourth column means the number of colours used for hard graph colouring. Column five shows the amount of conflict edges when I use 4 colours.

It can be seen in the table, the number of conflict edges is fluctuated while the number of vertices is increased. There are only 5 conflicts in graph 0, which has 61 edges in all. CM+BK algorithm is much better than GN algorithm.

## 3.3 CM+KJ algorithm experiments

3.3.1 CM+KJ algorithm

The main idea of CM+KJ algorithm is cliques merging and Konc Janezic (KJ) algorithm. The change of KJ algorithm is about upper bound. When invoking KJ algorithm again, CM+KJ algorithm use the size of the last clique found as an upper bound on the clique size. [2]

3.3.2 CM+KJ algorithm for soft graph colouring

I found a program named “CliqueMod Software” from Ref. [7], and tested the six graphs (I mentioned before) by CM+KJ algorithm.

The steps of the experiment are:
Step 1: run the software and find four communities in the complement graph
Step 2: identify conflict edges in each community
Step 3: count the number of conflict edges
I did the three steps for each graph by CM+KJ algorithm.

3.3.2.1 Example one

Step 1: the communities are:

cr is cg
c ae
ml
mu

Step 2: the conflict edge is CR-IS.

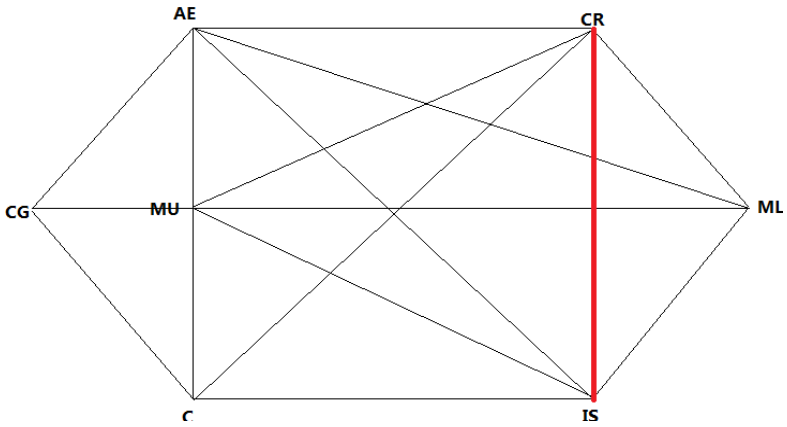

Figure 26: Conflict edge in graph 5. As can be seen in this picture, the red edge is conflict edge.

Step 3: the number of conflict edges is 1.

3.3.2.2 Example two

Step 1: the communities are:

cr al cg ml

is

c um ae

mu aa

Step 2: the conflict edges are:

CR-ML, ML-AL, UM-AE

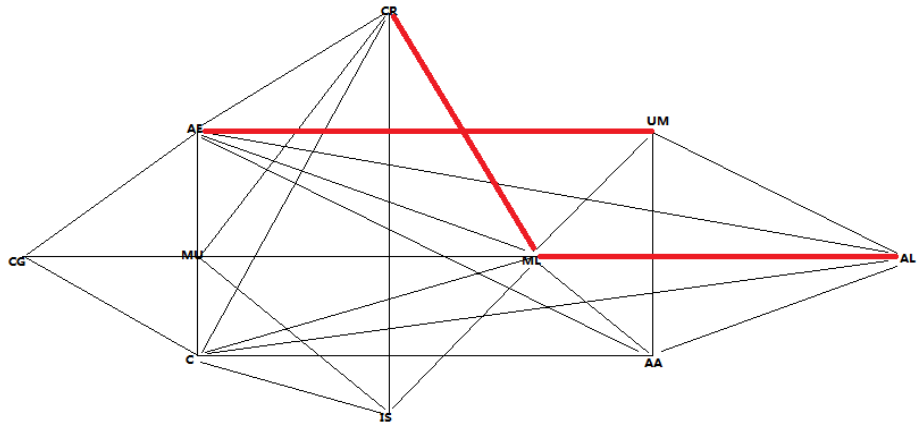


Figure 27: Conflict edges in graph 4. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 3.

3.3.2.3 Example three

Step 1: the communities are:

cr al dv cg

ca mu aa

is c em um ae

ml

Step 2: the conflict edges are:

EM-C, C-IS, UM-AE

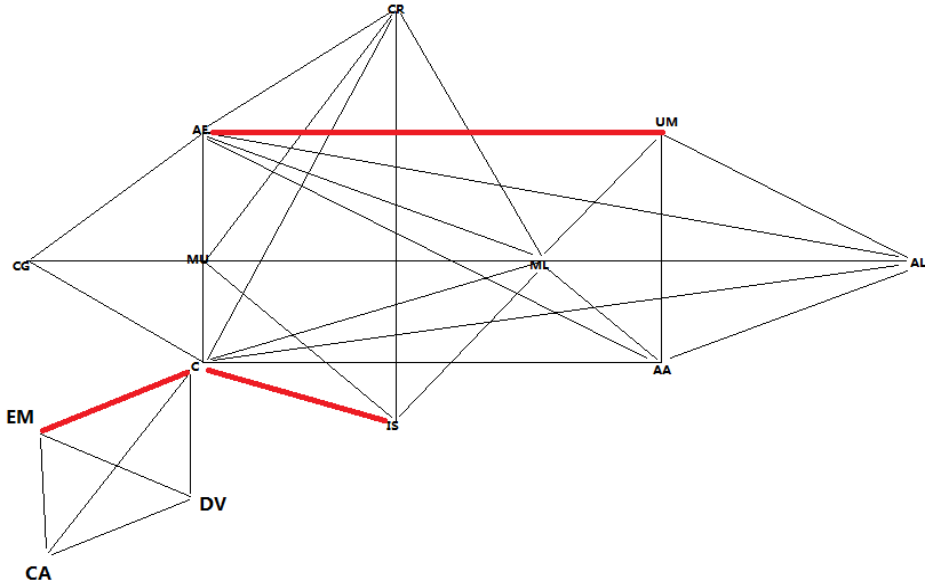

Figure 28: Conflict edges in graph 3. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 3.

3.3.2.4 Example four

Step 1: the communities are:

cr al db cg
mu aa wt
is c um ja ae
ml

Step 2: the conflict edges are:

IS-C, C-JA, UM-AE

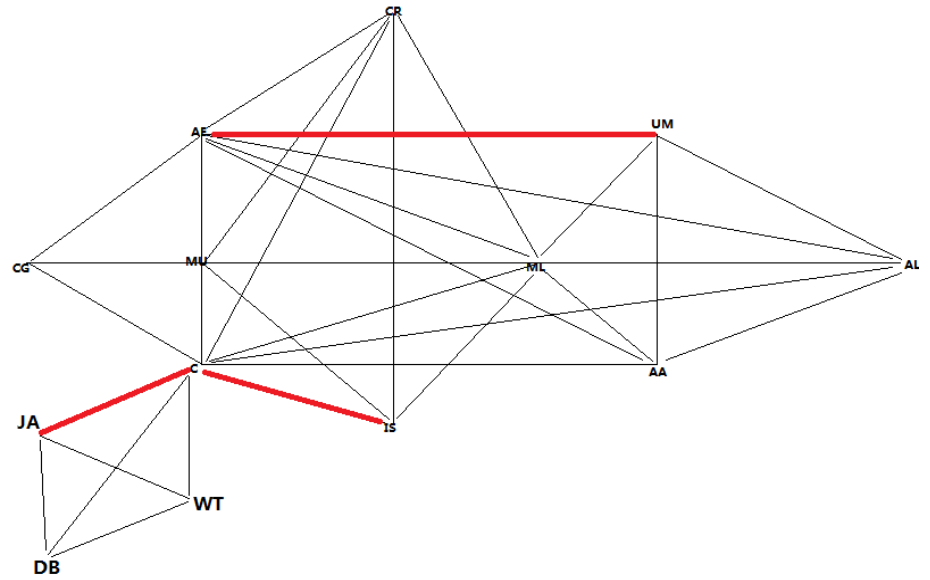

Figure 29: Conflict edges in graph 2. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 3.

3.3.2.5 Example five

Step 1: the communities are:

al cr db dv cg

ca mu aa wt

is c em um ja ae

ml

Step 2: the conflict edges are:

C-EM, C-JA, UM-AE, C-IS

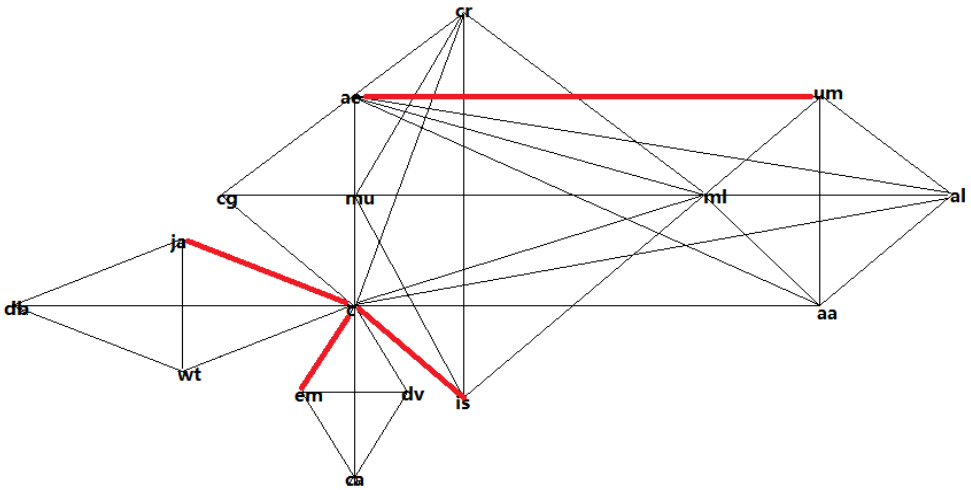


Figure 30: Conflict edges in graph 1. As can be seen in this picture, the red edges are conflict edges.

Step 3: the number of conflict edges is 4.

3.3.2.6 Example six

Step 1: the communities are:

db cb dv is ca sd mu um wt cr al cn rs aa cg em ja

c ic df ft ae

vd ml

ac

Step 2: the conflict edges are:

Group 1: UM-AL, AL-AA, AA-UM, CR-IS, IS-MU, MU-CR, MU-CG, EM-DV, DV-CA, CA-EM, JA-DB, JA-WT, JA-RS, DB-WT, DB-RS, RS-WT, RS-SD, SD-CB, SD-CN, RS-CB, RS-CN, CN-CB

Group 2: FT-DF

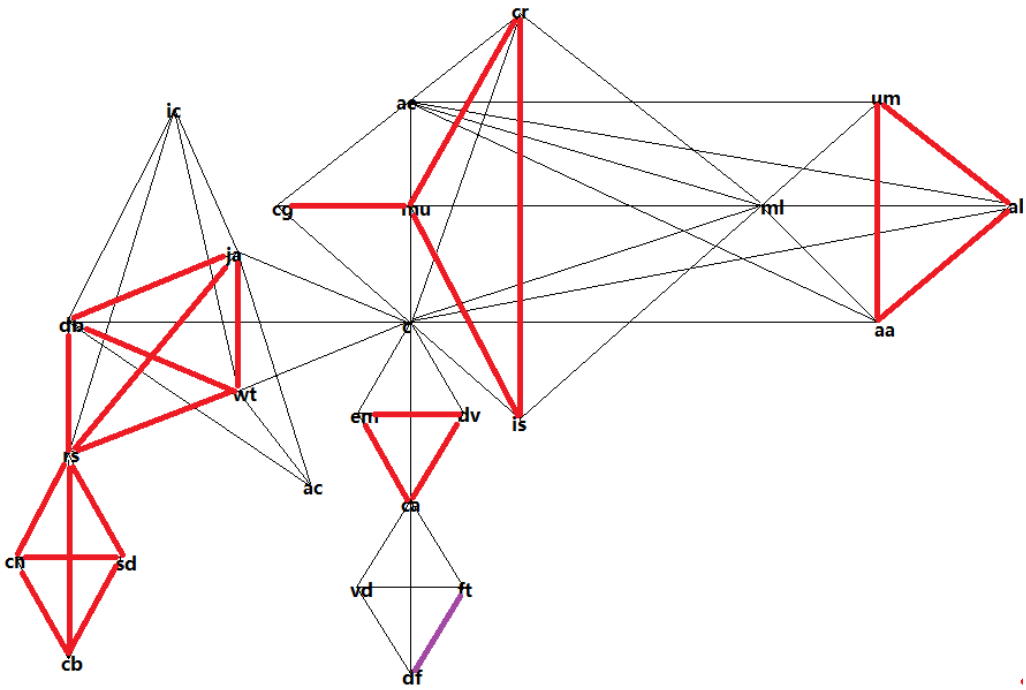


Figure 31: Conflict edges in graph 0. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 3: the number of conflict edges is 23.

3.3.3 Summary for CM+KJ algorithm

All the results of CM+KJ algorithm can be listed as follow:

| ID | Vertices | Edges | Hard graph colouring (HGC) | Conflict edges |
|---|---|---|---|---|
| 0 | 25 | 61 | 5 | 23 |
| 1 | 16 | 39 | 5 | 4 |
| 2 | 13 | 33 | 5 | 3 |
| 3 | 13 | 33 | 5 | 3 |
| 4 | 10 | 27 | 5 | 3 |
| 5 | 7 | 16 | 5 | 1 |

Table 4: Results of CM+KJ algorithm. The first column is ID number of each graph. Column two and three show the amount of vertices and edges in the graphs, respectively. The fourth column means the number of colours used for hard graph colouring. Column five shows the amount of conflict edges when I use 4 colours.

As can be seen from this table, the number of conflict edges is increased by vertices addition. In graph 0, there are 23 conflicts, which are fewer than 43 conflict edges of GN algorithm. However, it is worse than CM+BK algorithm.

## 3.4 CNM algorithm experiments

3.4.1 CNM algorithm

Newman and Girvan essay to estimate the value of graph clustering by means of modularity. CNM (Clauset, Newman, and Moore) algorithm is a bottom-up agglomerative clustering which continuously identifies and merges pairs of clusters attempting to maximize modularity of the community structure in a greedy manner. [3]

3.4.2 CNM algorithm for soft graph colouring

I found a program named “Fast Modularity” from Ref. [8], and tested the six graphs (I mentioned before) by CNM algorithm.

The steps of the experiment are:

Step 1: run the software and find four communities in the complement graph

Step 2: transform the numeric vertices to the original vertices.

Step 3: identify conflict edges in each community

Step 4: count the number of conflict edges

I did the four steps for each graph by CNM algorithm.

3.4.2.1 Example one

Step 1: the communities are:

GROUP[ 1 ][ 3 ]
1
6
5
GROUP[ 3 ][ 1 ]
3
GROUP[ 4 ][ 2 ]
4
2

Step 2: the original vertices are:

GROUP[ cg ][ 3 ]
cg
is
cr
GROUP[ ml ][ 1 ]
ml
GROUP[ c ][ 2 ]
c
ae

Step 3: the conflict edge is CR-IS.

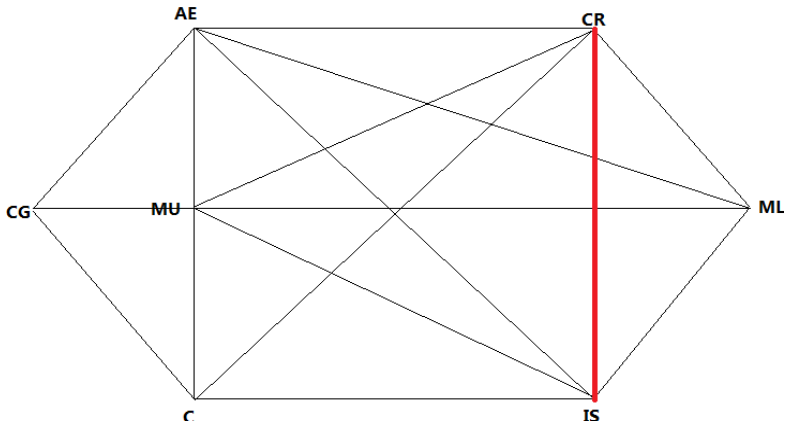

Figure 32: Conflict edge in graph 5. As can be seen in this picture, the red edge is conflict edge.

Step 4: the number of conflict edges is 1.

3.4.2.2 Example two

Step 1: the communities are:

GROUP[ 2 ][ 4 ]
2
6
1
5
GROUP[ 3 ][ 1 ]
3
GROUP[ 7 ][ 2 ]
7
9
GROUP[ 8 ][ 3 ]
8
4
10

Step 2: the original vertices are:

GROUP[ cr ][ 4 ]
cr
aa
al
mu
GROUP[ is ][ 1 ]
is
GROUP[ cg ][ 2 ]
cg
ml
GROUP[ um ][ 3 ]

um
c
ae

Step 3: the conflict edges are:
Group 1: UM-AE
Group 2: CR-ML, ML-AL

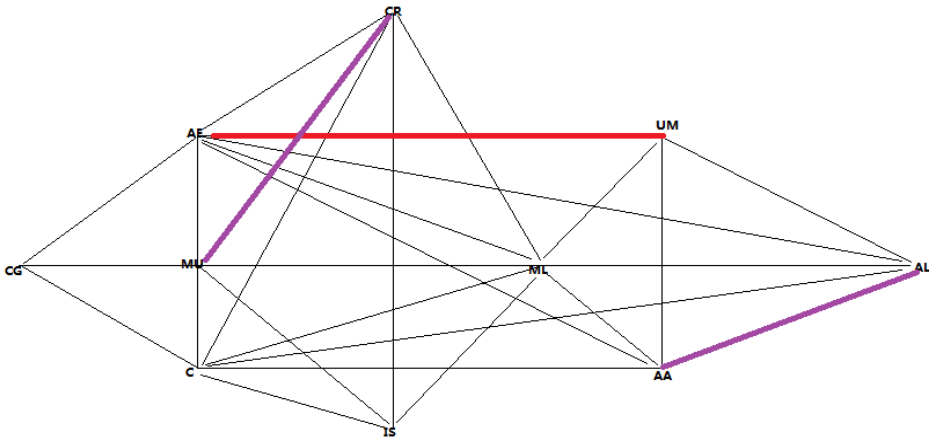


Figure 33: Conflict edges in graph 4. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 3.

3.4.2.3 Example three
Step 1: the communities are:
GROUP[ 8 ][ 4 ]
8
10
4
3
GROUP[ 11 ][ 3 ]
11
1
5
GROUP[ 12 ][ 3 ]
12
7
9
GROUP[ 13 ][ 3 ]
13
6
2

Step 2: the original vertices are:

GROUP[ um ][ 4 ]
um
ae
c
is
GROUP[ em ][ 3 ]
em
al
mu
GROUP[ dv ][ 3 ]
dv
cg
ml
GROUP[ ca ][ 3 ]
ca
aa
cr

Step 3: the conflict edges are:
C-IS, UM-AE

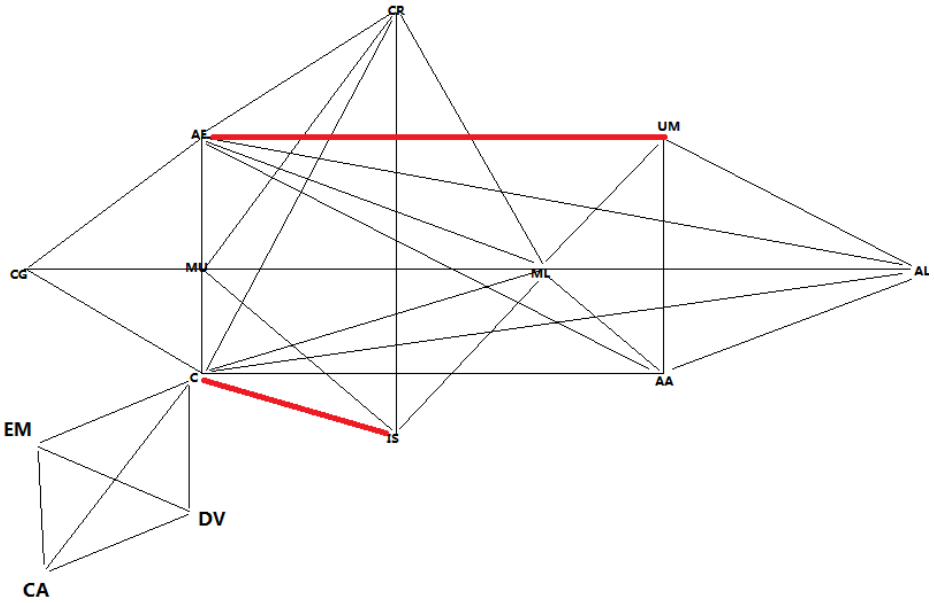


Figure 34: Conflict edges in graph 3. As can be seen in this picture, the red edges are conflict edges.

Step 4: the number of conflict edges is 2.

3.4.2.4 Example four
Step 1: the communities are:
GROUP[ 8 ][ 4 ]
8

3
10
4
GROUP[ 17 ][ 3 ]
17
7
9
GROUP[ 18 ][ 3 ]
18
1
5
GROUP[ 19 ][ 3 ]
19
6
2

Step 2: the original vertices are:
GROUP[ um ][ 4 ]
um
is
ae
c
GROUP[ ja ][ 3 ]
ja
cg
ml
GROUP[ wt ][ 3 ]
wt
al
mu
GROUP[ db ][ 3 ]
db
aa
cr

Step 3: the conflict edges are:
C-IS, UM-AE

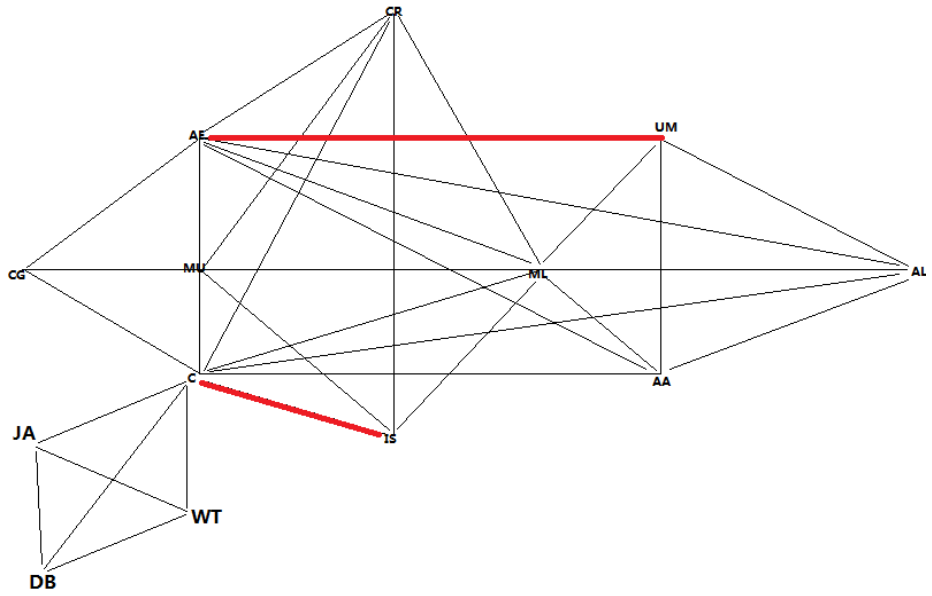


Figure 35: Conflict edges in graph 2. As can be seen in this picture, the red edges are conflict edges.

Step 4: the number of conflict edges is 2.

3.4.2.5 Example five

Step 1: the communities are:

GROUP[ 3 ][ 1 ]
3
GROUP[ 10 ][ 3 ]
10
4
8
GROUP[ 11 ][ 4 ]
11
1
5
18
GROUP[ 13 ][ 8 ]
13
12
7
9
17
2
6
19

Step 2: the original vertices are:

GROUP[ is ][ 1 ]
is
GROUP[ ae ][ 3 ]
ae
c
um
GROUP[ em ][ 4 ]
em
al
mu
wt
GROUP[ ca ][ 8 ]
ca
dv
cg
ml
ja
cr
aa
db

Step 3: the conflict edges are:

Group 1: UM-AE

Group 2: CR-ML, ML-AA, CA-DV, DB-JA

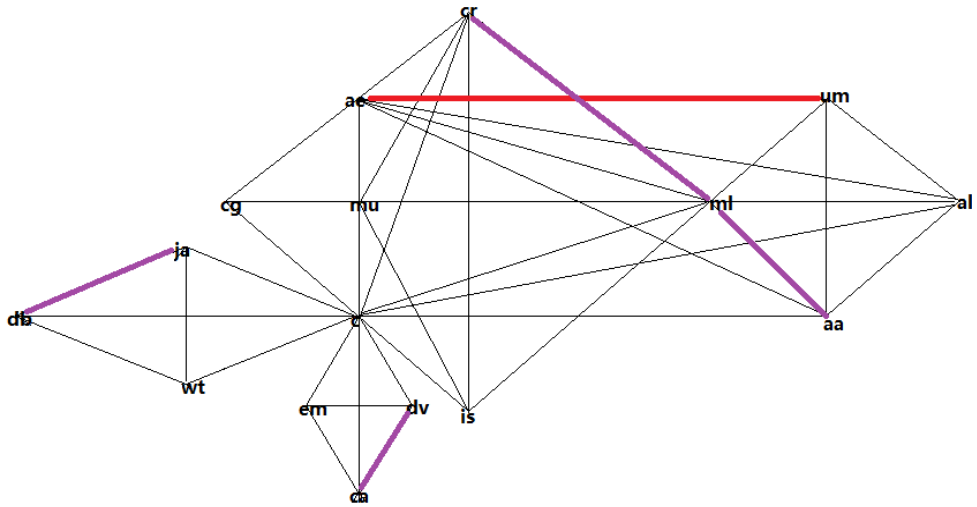


Figure 36: Conflict edges in graph 1. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 5.

3.4.2.6 Example six

Step 1: the communities are:

GROUP[ 2 ][ 9 ]
2
13
23
18
5
1
19
24
17
GROUP[ 6 ][ 1 ]
6
GROUP[ 9 ][ 14 ]
9
16
7
12
21
14
4
10
15
3
22
8
20
11
GROUP[ 25 ][ 1 ]
25

Step 2: the original vertices are:
GROUP[ cr ][ 9 ]
cr
ca
cn
wt
mu
al
db
sd
ja
GROUP[ aa ][ 1 ]
aa
GROUP[ ml ][ 14 ]

ml
df
cg
dv
ic
vd
c
ae
ft
is
ac
um
rs
em
GROUP[ cb ][ 1 ]
cb

Step 3: the conflict edges are:

Group 1: CR-MU, WT-JA, JA-DB, DB-WT, CN-SD

Group 2: IC-RS, C-EM, EM-DV, DV-C, VD-FT, FT-DF, DF-VD, C-IS, IS-ML, ML-UM, AE-UM, AE-ML, AE-CG, CG-C, C-ML

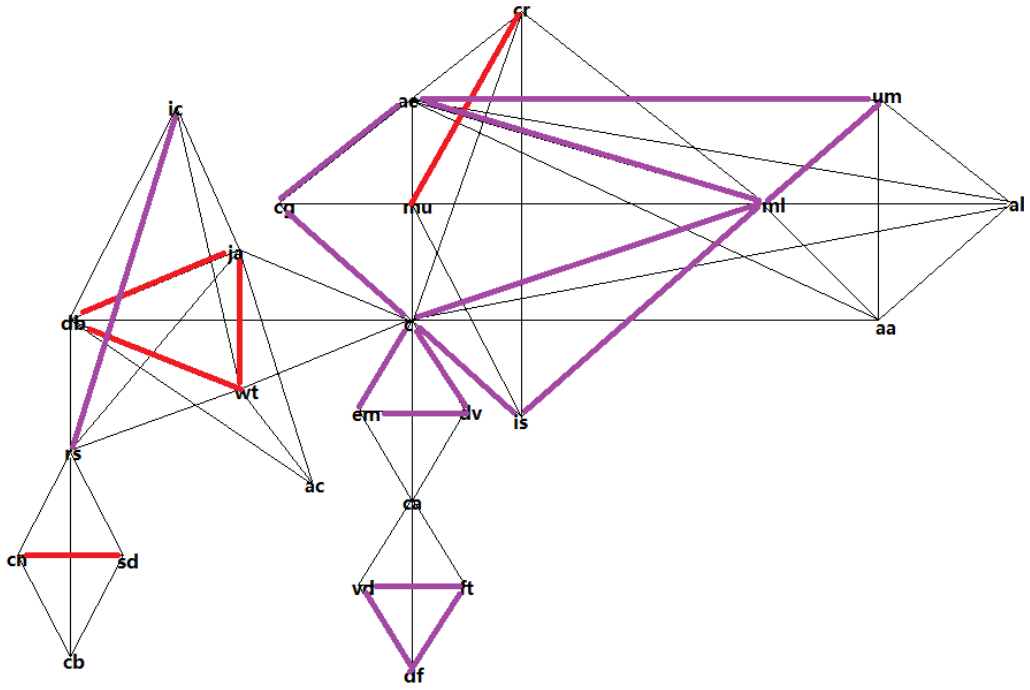


Figure 37: Conflict edges in graph 0. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 20.

3.4.3 Summary for CNM algorithm

All the results of CNM algorithm can be listed as follow:

| ID | Vertices | Edges | Hard graph colouring (HGC) | Conflict edges |
|---|---|---|---|---|
| 0 | 25 | 61 | 5 | 20 |
| 1 | 16 | 39 | 5 | 5 |
| 2 | 13 | 33 | 5 | 2 |
| 3 | 13 | 33 | 5 | 2 |
| 4 | 10 | 27 | 5 | 3 |
| 5 | 7 | 16 | 5 | 1 |

Table 5: Results of CNM algorithm. The first column is ID number of each graph. Column two and three show the amount of vertices and edges in the graphs, respectively. The fourth column means the number of colours used for hard graph colouring. Column five shows the amount of conflict edges when I use 4 colours.

It can be seen in the table, the number of conflict edges is fluctuated while the number of vertices is increased. All in all, CNM algorithm is better than GN and CM+KJ algorithm, but worse than CM+BK algorithm.

## 3.5 Walktrap algorithm experiments

3.5.1 Walktrap algorithm

The idea of walktrap algorithm is that a random walker tends to be trapped in the same community. It tries to find communities in a graph by random walks.

3.5.2 Walktrap algorithm for soft graph colouring

I found a program named “Walktrap v0.2” from Steve, and tested the six graphs (I mentioned before) by Walktrap algorithm.

The steps of the experiment are:
Step 1: run the software and find four communities in the complement graph
Step 2: transform the numeric vertices to the original vertices.
Step 3: identify conflict edges in each community
Step 4: count the number of conflict edges
I did the four steps for each graph by Walktrap algorithm.

3.5.2.1 Example one

Step 1: the communities are:

community 2 = {2}
community 7 = {4, 0, 5}
community 8 = {1, 3}
community 9 = {6}

Step 2: the original vertices are:

community 2 = {ml}

community 7 = {cr, cg, is}
community 8 = {ae, c}
community 9 = {mu}

Step 3: the conflict edge is CR-IS.

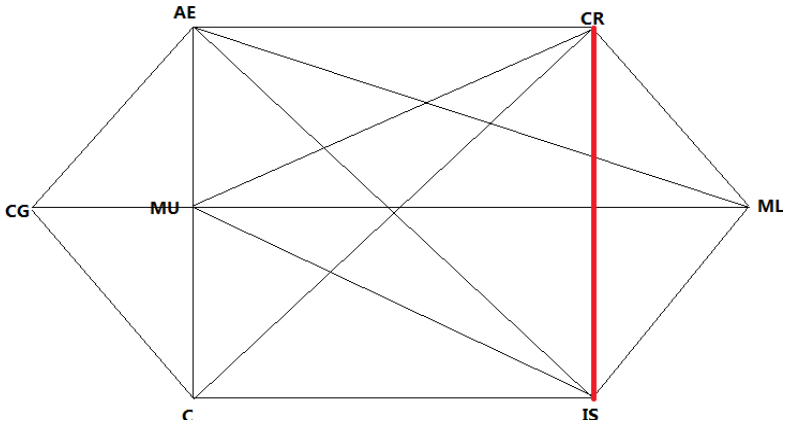


Figure 38: Conflict edge in graph 5. As can be seen in this picture, the red edge is conflict edge.

Step 4: the number of conflict edges is 1.

3.5.2.2 Example two

Step 1: the communities are:

community 6 = {6}
community 10 = {1, 3}
community 13 = {5, 7}
community 15 = {0, 2, 9, 4, 8}

Step 2: the original vertices are:

community 6 = {ml}
community 10 = {ae, c}
community 13 = {is, um}
community 15 = {cg, mu, al, cr, aa}

Step 3: the conflict edges are:

CG-MU, MU-CR, AL-AA

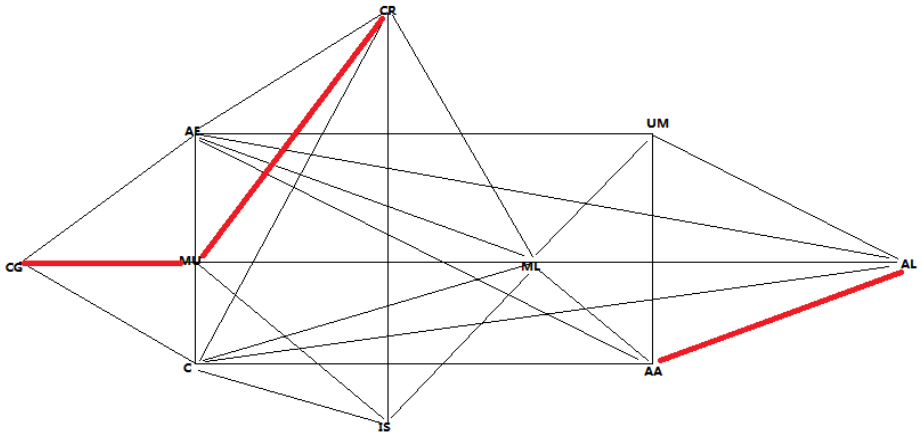


Figure 39: Conflict edges in graph 4. As can be seen in this picture, the red edges are conflict edges.

Step 4: the number of conflict edges is 3.

3.5.2.3 Example three

Step 1: the communities are:

community 1 = {1}
community 3 = {3}
community 20 = {6, 2, 9, 8, 0, 4}
community 21 = {7, 10, 12, 5, 11}

Step 2: the original vertices are:

community 1 = {ae}
community 3 = {c}
community 20 = {ml, mu, al, aa, cg, cr}
community 21 = {um, em, ca, is, dv}

Step 3: the conflict edges are:

Group 1: EM-DV, DV-CA, CA-EM
Group 2: CG-MU, MU-CR, MU-ML, AA-AL

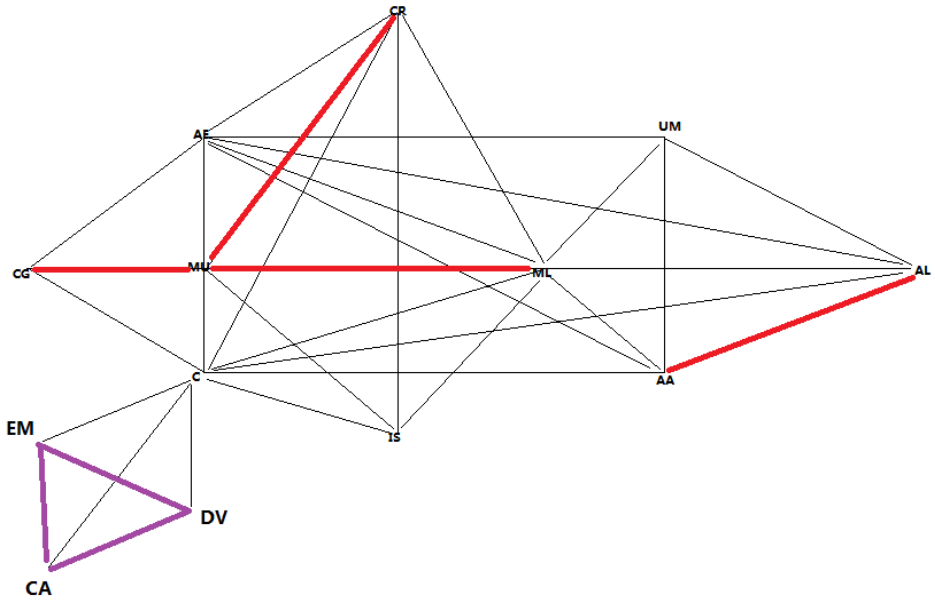


Figure 40: Conflict edges in graph 3. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 7.

3.5.2.4 Example four

Step 1: the communities are:

community 1 = {1}
community 3 = {3}
community 20 = {6, 2, 9, 8, 0, 4}
community 21 = {7, 11, 10, 5, 12}

Step 2: the original vertices are:

community 1 = {ae}
community 3 = {c}
community 20 = {ml, mu, al, aa, cg, cr}
community 21 = {um, ja, db, wt, is}

Step 3: the conflict edges are:

Group 1: JA-WT, WT-DB, DB-JA
Group 2: CG-MU, MU-CR, MU-ML, AA-AL

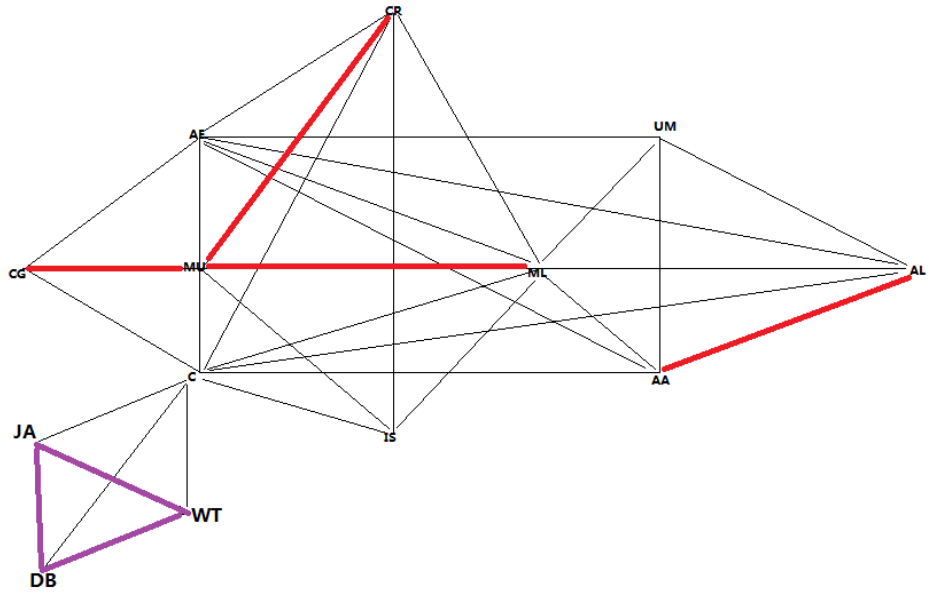


Figure 41: Conflict edges in graph 2. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 7.

3.5.2.5 Example five

Step 1: the communities are:

community 1 = {1}
community 3 = {3}
community 7 = {7}
community 27 = {15, 14, 5, 13, 2, 8, 4, 9, 6, 12, 11, 0, 10}

Step 2: the original vertices are:

community 1 = {ae}
community 3 = {c}
community 7 = {um}
community 27 = {wt, db, is, ja, mu, aa, cr, al, ml, ca, dv, cg, em}

Step 3: the conflict edges are:

JA-WT, WT-DB, DB-JA, EM-DV, DV-CA, CA-EM, CG-MU, MU-CR, CR-IS, IS-MU, MU-ML, CR-ML, IS-ML, ML-AL, AA-AL, AA-ML

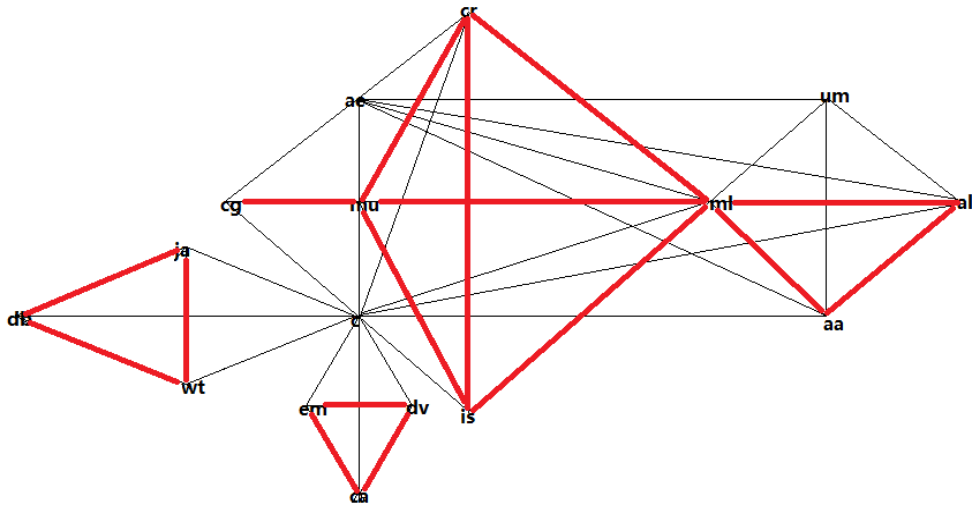


Figure 42: Conflict edges in graph 1. As can be seen in this picture, the red edges are conflict edges.

Step 4: the number of conflict edges is 16.

3.5.2.6 Example six

Step 1: the communities are:

community 23 = {23}
community 24 = {24}
community 34 = {19, 13, 21, 15, 14, 20}
community 45 = {22, 17, 16, 18, 1, 3, 12, 6, 2, 9, 4, 8, 11, 10, 0, 5, 7}

Step 2: the original vertices are:

community 23 = {ft}
community 24 = {df}
community 34 = {cn, ja, cb, wt, db, sd}
community 45 = {vd, ic, rs, ac, ae, c, ca, ml, mu, al, cr, aa, dv, em, cg, is, um}

Step 3: the conflict edges are:

Group 1: JA-WT, WT-DB, DB-JA, CN-SD, SD-CB, CB-CN
Group 2: IC-RS, CA-VD, EM-CA, EM-DV, DV-CA, CA-C, C-EM, C-DV, CG-AE, CG-MU, CG-C, MU-AE, MU-C, MU-CR, MU-ML, MU-IS, C-IS, C-CR, AE-CR, CR-IS, CR-ML, AE-ML, MU-ML, C-ML, IS-ML, AE-UM, ML-UM, AA-UM, ML-AA, AE-AA, C-AA, UM-AL, AA-AL, AE-AL, C-AL

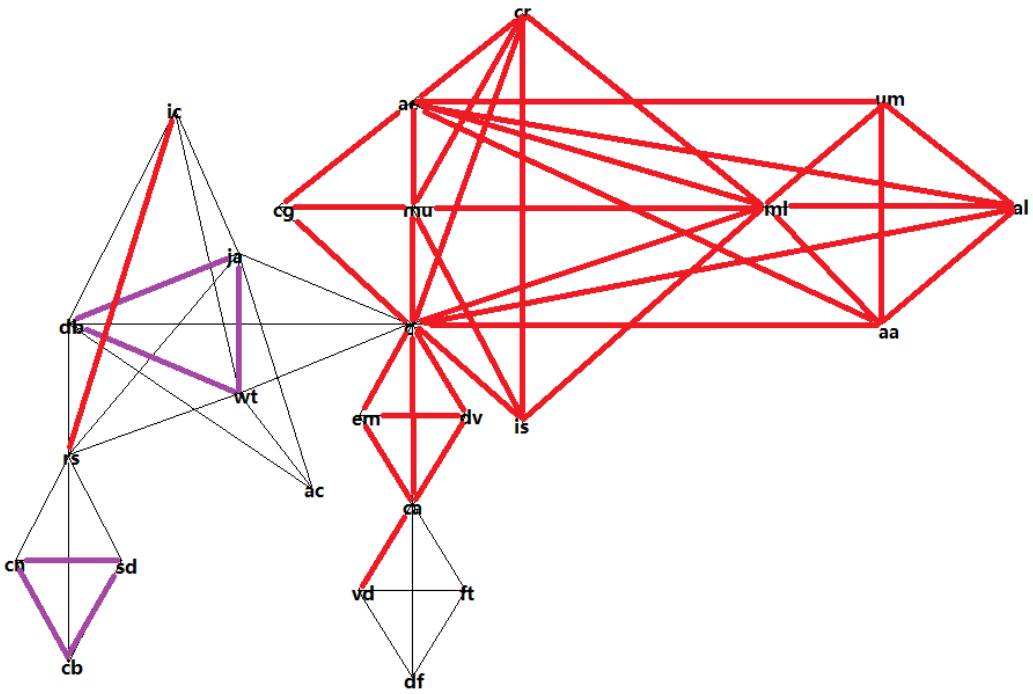


Figure 43: Conflict edges in graph 0. As can be seen in this picture, the red edges are conflict edges of group 1, while conflict edges of group 2 are the purple ones.

Step 4: the number of conflict edges is 41.

3.5.3 Summary for Walktrap algorithm

All the results of Walktrap algorithm can be listed as follow:

| ID | Vertices | Edges | Hard graph colouring (HGC) | Conflict edges |
|---|---|---|---|---|
| 0 | 25 | 61 | 5 | 41 |
| 1 | 16 | 39 | 5 | 16 |
| 2 | 13 | 33 | 5 | 7 |
| 3 | 13 | 33 | 5 | 7 |
| 4 | 10 | 27 | 5 | 3 |
| 5 | 7 | 16 | 5 | 1 |

Table 6: Results of Walktrap algorithm. The first column is ID number of each graph. Column two and three show the amount of vertices and edges in the graphs, respectively. The fourth column means the number of colours used for hard graph colouring. Column five shows the amount of conflict edges when I use 4 colours.

As can be seen in this table, the number of conflict edges is increased by vertices addition. In graph 0, there are 41 conflicts, which is over half of the edges in the graph. Walktrap algorithm is worse than CM+BK, CM+KJ, and CNM algorithm.

# 4. Experiment for a large network

After testing six graphs, I try the large network by those algorithms. The large network has 864 vertices and 18707 edges. After transforming it to the complement graph, it has 354109 edges. For each algorithm, I randomly pick some integers as communities, and test the conflict edges with those communities. Unfortunately, GN algorithm is very slow to run the large graph, and I have to move on to others.

## 4.1CM+BK algorithm experiments

The data of CM+BK algorithm can be listed as follow:

| Communities | 2 | 3 | 4 | 5 | 10 | 15 | 25 | 32 | 53 | 54 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Conflicts | 4566 | 4540 | 4520 | 4488 | 4406 | 4346 | 4246 | 4198 | 4121 | 4120 | 4094 |

Table 7: Results of BK algorithm for the large network. The first row is the number of communities input in the program. Row two shows the amount of conflict edges with those communities.

I use the results to generate a curve:

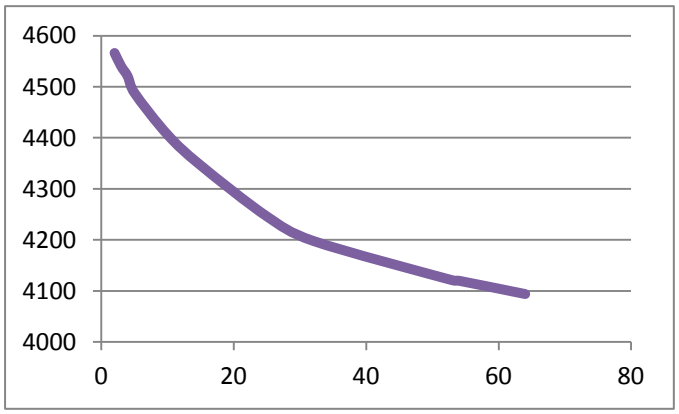


Figure 44: The figure is based on the results of table 6. The x axis is the number of communities, while the y axis is the number of conflict edges.

From the figure, it can be shown that the curve reduced gradually. The conflicts are over 4000 from communities 2 to communities 64.

## 4.2CM+KJ algorithm experiments

The data of CM+KJ algorithm can be listed as follow:

| Communities | Conflicts |
|---|---|
| 2 | 5620 |
| 3 | 5613 |
| 4 | 5472 |
| 5 | 5466 |
| 10 | 5442 |
| 15 | 5424 |

| | |
|---|---|
| 25 | 5406 |
| 32 | 4865 |
| 40 | 3870 |
| 46 | 3197 |
| 53 | 2338 |
| 54 | 2165 |
| 60 | 1218 |
| 62 | 894 |
| 64 | 560 |

Table 8: Results of KJ algorithm for the large network. The first column is the number of communities input in the program. Column two shows the amount of conflict edges with those communities.

I use the results to generate a curve:

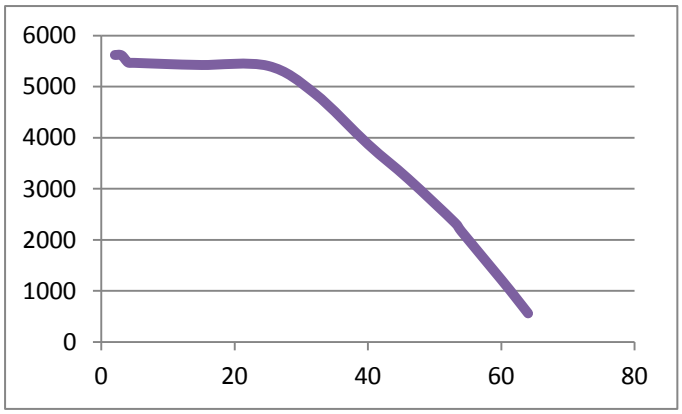


Figure 45: The figure is based on the results of table 7. The x axis is the number of communities, while the y axis is the number of conflict edges.

As can be seen from this figure, the curve leveled off at first, and then declined steadily. The whole curve is better than CM+BK algorithm, because the numbers of conflicts are fewer than those.

## 4.3Walktrap algorithm experiments

The data of Walktrap algorithm can be listed as follow:

| Communities | Conflicts |
|---|---|
| 2 | 7636 |
| 3 | 5071 |
| 4 | 2794 |
| 5 | 2787 |
| 10 | 2757 |
| 15 | 2355 |
| 17 | 2112 |
| 20 | 1767 |
| 25 | 1617 |

| | |
|---|---|
| 32 | 1575 |
| 40 | 1557 |
| 45 | 1517 |
| 46 | 1511 |
| 47 | 862 |
| 50 | 845 |
| 54 | 830 |
| 60 | 823 |
| 62 | 784 |
| 64 | 475 |

Table 9: Results of Walktrap algorithm for the large network. The first column is the number of communities input in the program. Column two shows the amount of conflict edges with those communities.

I use the results to generate a curve:

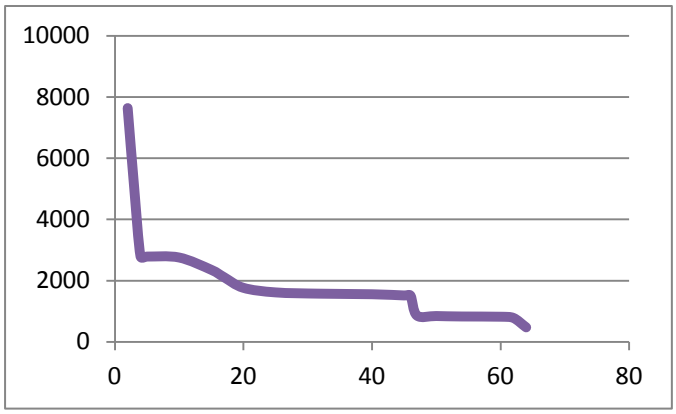


Figure 46: The figure is based on the results of table 8. The x axis is the number of communities, while the y axis is the number of conflict edges.

There is a dramatic reduction of Walktrap curve from communities 2 to communities 4. The rest of the curve reduced steadily, and is better than CM+BK and CM+KJ algorithms.

## 4.4CNM algorithm experiments

The data of CNM algorithm can be listed as follow:

| Communities | 2 | 3 | 4 | 5 | 10 | 15 | 25 | 32 | 40 | 54 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Conflicts | 1695 | 1677 | 1671 | 1658 | 1625 | 1606 | 1566 | 1544 | 1515 | 1468 | 1428 |

Table 10: Results of CNM algorithm for the large network. The first row is the number of communities input in the program. Row two shows the amount of conflict edges with those communities.

I use the results to generate a curve:

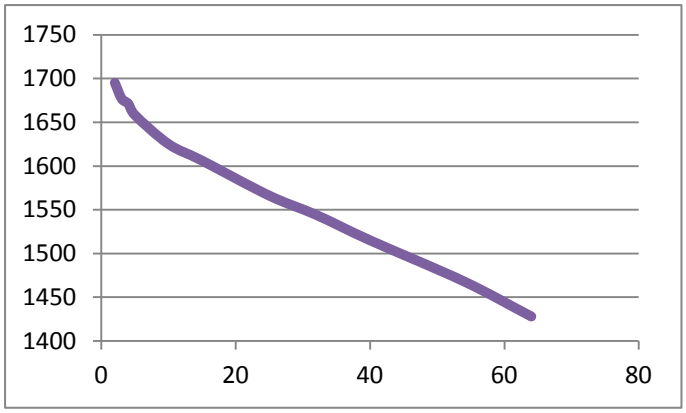

Figure 47: The figure is based on the results of table 9. The x axis is the number of communities, while the y axis is the number of conflict edges.

From the figure, it may be seen that the curve decreased gradually. The conflicts are over 1400 from communities 2 to communities 64. The result is worse than the curve of Walktrap algorithm.

# 5. Evaluation

The final part of the process is to evaluate the results according to conflict edges. It is important to know that which CD algorithm is the best. Moreover, does the best one work better than soft graph colouring algorithm?

## 5.1 CD algorithms

Although CM+BK algorithm gets good results for the six graphs, it does not work well for the large network. However, it is very useful to know which algorithm is best.

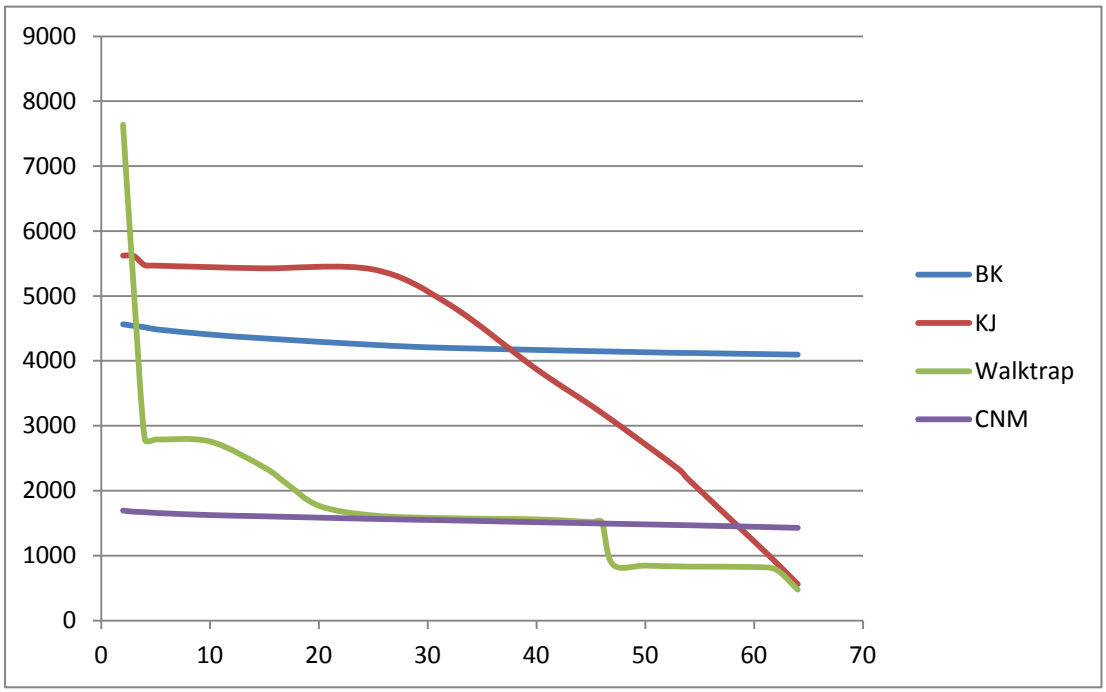


Figure 48: The figure is based on the results BK, KJ, Walktrap and CNM. The x axis is the number of communities, while the y axis is the number of conflict edges.

As can be shown from this figure, both Walktrap and CNM algorithm work well and get fewer conflicts.

## 5.2 Soft graph colouring algorithm

Another important thing is to find a soft graph colouring algorithm and compare them. There is an algorithm for soft graph colouring from Steve: To colour the network using c colours, use colours 1, … , c-1 for the c-1 for the c-1 largest cliques, and use colour c for the remaining (54-c+1) smallest cliques.

The steps of the algorithm can be listed as follow:
Step 1: compute complement network of the large graph.
Step 2: find all maximal cliques in complement graph, in order of size.
Step 3: Reduce the number of cliques by replacing the smallest cliques by one

“community” that contains all of them.
Step 4: Evaluate the list of communities (actually one “community” and many cliques) in the usual way: count the number of conflicts.

The smallest cliques have only one vertex, and they are:
21
22
23
24
25
26
27
28
29
30
31
32
95
102
108
109
209
238
334

I take those smallest cliques and combine them into one set. That gives 36 sets: 35 cliques and one set of 19 vertices. Then count the conflicts in this. It is 85 conflict edges caused by grouping those 19 singletons into one set. Next, repeat step 2 to merge the last N cliques into one set, and count conflicts again.

The results of this algorithm are:

| Colour solution | Conflicts |
|---|---|
| 36 | 85 |
| 35 | 98 |
| 34 | 112 |
| 33 | 127 |
| 32 | 143 |
| 31 | 159 |
| 30 | 175 |
| 29 | 193 |
| 28 | 211 |
| 27 | 231 |
| 26 | 251 |
| 25 | 273 |

| | |
|---|---|
| 24 | 297 |
| 23 | 326 |
| 22 | 356 |
| 21 | 387 |
| 20 | 421 |
| 19 | 471 |
| 18 | 524 |
| 17 | 578 |
| 16 | 678 |
| 15 | 780 |
| 14 | 884 |
| 13 | 989 |
| 12 | 1119 |
| 11 | 1250 |
| 10 | 1402 |
| 9 | 1558 |
| 8 | 1741 |
| 7 | 1933 |
| 6 | 2178 |
| 5 | 2455 |
| 4 | 2942 |
| 3 | 3667 |
| 2 | 5311 |

Table 11: Results of soft graph colouring algorithm for the large network. The first column is the number of colours, in other words, the set after merging. Column two shows the amount of conflict edges with those colours.

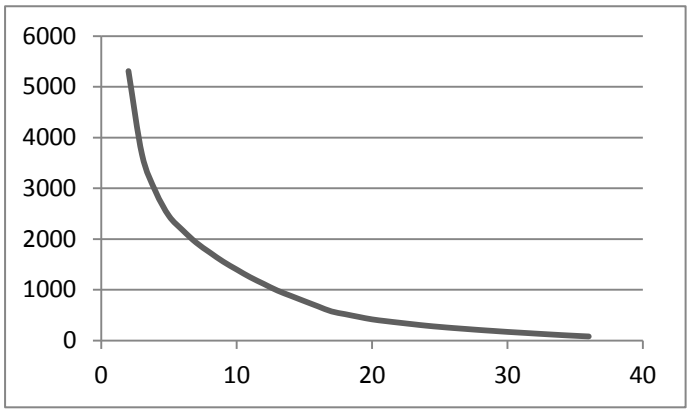


Figure 49: The figure is based on the results of table 10. The x axis is the number of communities, while the y axis is the number of conflict edges.

## 5.3 Evaluation

5.3.1 Walktrap and SGC

The figure below shows the curves of Walktrap algorithm and soft graph colouring algorithm.

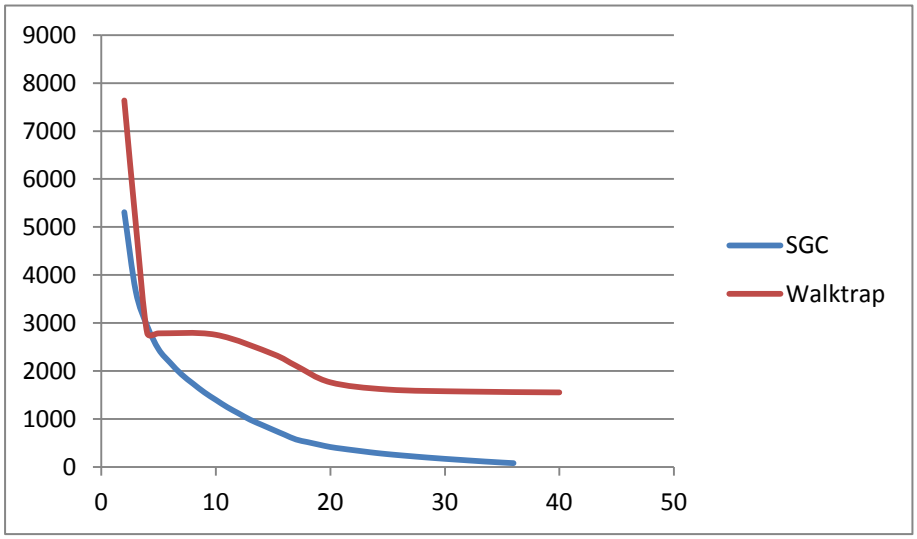


Figure 50: The figure is based on the results SGC and Walktrap algorithm. The x axis is the number of communities, while the y axis is the number of conflict edges.

From the figure, it may be seen that the conflicts of Walktrap algorithm are more than the conflicts of SGC algorithm. Walktrap algorithm does not work well than SGC algorithm.

5.3.2 CNM and SGC

The figure below shows the curves of CNM algorithm and soft graph colouring algorithm.

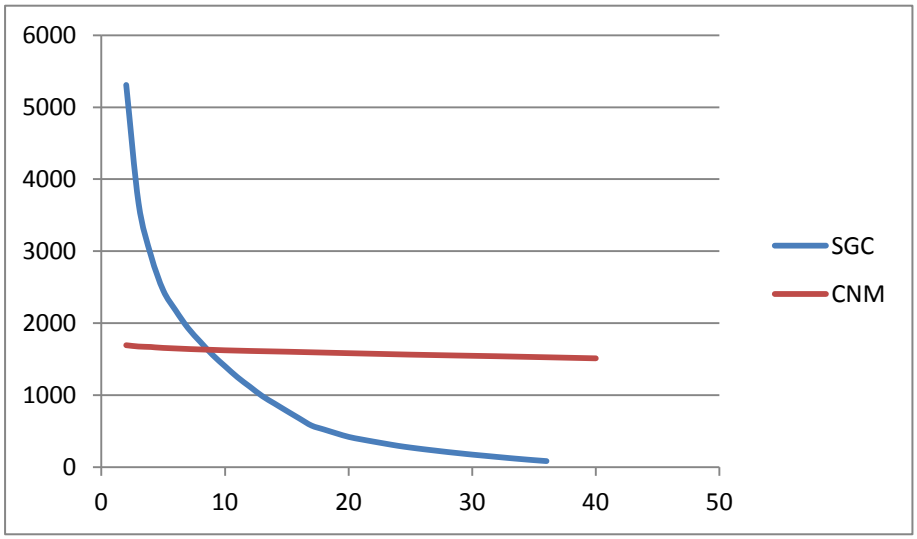


Figure 51: The figure is based on the results SGC and CNM algorithm. The x axis is the number of communities, while the y axis is the number of conflict edges.

As can be seen from the figure, the curve of CNM keeps straight, while the curve of soft graph colouring algorithm decreased gradually. It looks like the CNM algorithm is better than SGC algorithm in the front of those curves.

All in all, the experiments test five community detection algorithms, and compare them with soft graph colouring algorithm. CNM algorithm is the best one in those five algorithms, and is better than soft graph colouring algorithm in some case.

# 6. Outlook

Outlook part summarizes the whole project, and future work of it.

## 6.1 Conclusion

It is generally accepted that simple graph colouring is equivalent to finding cliques in the complement graph. This project tested a new hypothesis, which is soft graph colouring can be done by finding communities in the complement graphs.

GN, CM+BK, CM+KJ, CNM, and Walktrap, five community detection algorithms are chosen to run experiments, because they are easy to control the number of communities. Seven graphs are tested in this project, and CNM algorithm works well in the large network.

After comparing CNM algorithm with soft graph colouring algorithm, it can be concluded that CNM algorithm is better than soft graph colouring algorithm in some case.

## 6.2 Future work

The project can be extended in future. First and foremost, it can be tested in other unweighted large networks. Secondly, there are hundreds of community detection algorithms, and it is possible that other CD algorithm can do a better work than CNM algorithm. What is more, find other soft graph colouring algorithm, and compare them. Finally, the project can be further extended to weighted graphs. It is more important to use different colours for two vertices if they are connected by a high weight edge.

# Bibliography

[1] Fortunato, S. Community detection in graphs. Complex Networks and Systems Lagrange Laboratory, ISI Foundation, Viale S. Severo 65, 10133, Torino.

[2] Bowen, Y and Steve, G. Detecting Communities in Networks by Merging Cliques. University of Bristol, Bristol, UK.

[3] Ken, W and Toshiyuki, T. Finding Community Structure in Mega-scale Social Networks. Tokyo Institute of Technology, Tokyo, Japan. http://www2007.org/htmlposters/poster950

[4] http://www.cs.bris.ac.uk/admissions/pg/units.jsp

[5] http://mat.tepper.cmu.edu/ROIS/instances/graph/display_graph.php

[6] http://www.cs.bris.ac.uk/~steve/networks/software/conga.html

[7] http://www.cs.bris.ac.uk/~steve/networks/software/cliquemod.html

[8] http://www.cs.unm.edu/~aaron/research/fastmodularity.htm

[9] Aaron C, M.E.J. Newman, and Cristopher M. Finding community structure in very large networks. Aug 2004.

[10] Kestrel Institute. e-Merge-ANT: November 2000. Stephen Fitzpatrick, Cordell Green & Lambert Meertens. http://ants.kestrel.edu/. ANTs PI Meeting, Charleston, SC, 28-30 November 2000.

[11] Amin, C and Anusch, T. Exact and Approximative algorithms for Coloring G(n,p). Unter den Linden 6, Berlin, Germany. Oct. 2003.

[12] Kazunori, M and Seiichi, N. Constructive generation of very hard 3-colorability instances. July 2006.

[13] Gregory, C. Register Allocation and Spilling via Graph Coloring. IBM T. J. Watson Research Center.

[14] Stefano, G and Federico, M. Exact Solution of Graph Coloring Problems via Constraint Programming and Column Generation.

[15] Ramarathnam, V and Leonid, A. L. Random Instances of a Graph Coloring Problem are Hard. Boston University.

# Appendix A

The software for CNM algorithm uses numeric vertices as the input file.
This is the encode table for CNM algorithm in graph 5

| Vertex (original graph) | Numeric Vertex (algorithm) |
|---|---|
| cg | 1 |
| ae | 2 |
| ml | 3 |
| c | 4 |
| cr | 5 |
| is | 6 |

This is the encode table for CNM algorithm in graph 0-4

| Vertex (original graph) | Numeric Vertex (algorithm) |
|---|---|
| al | 1 |
| cr | 2 |
| is | 3 |
| c | 4 |
| mu | 5 |
| aa | 6 |
| cg | 7 |
| um | 8 |
| ml | 9 |
| ae | 10 |
| em | 11 |
| dv | 12 |
| ca | 13 |
| vd | 14 |
| ft | 15 |
| df | 16 |
| ja | 17 |
| wt | 18 |
| db | 19 |
| rs | 20 |
| ic | 21 |
| ac | 22 |
| cn | 23 |
| sd | 24 |
| cb | 25 |

# Appendix B

The software for Walktrap algorithm uses numeric vertices as the input file.
This is the encode table for Walktrap algorithm in graph 5

| Vertex (original graph) | Numeric Vertex (algorithm) |
|---|---|
| cg | 0 |
| ae | 1 |
| ml | 2 |
| c | 3 |
| cr | 4 |
| is | 5 |

This is the encode table for Walktrap algorithm in graph 0-4

| Vertex (original graph) | Numeric Vertex (algorithm) |
|---|---|
| cg | 0 |
| ae | 1 |
| mu | 2 |
| c | 3 |
| cr | 4 |
| is | 5 |
| ml | 6 |
| um | 7 |
| aa | 8 |
| al | 9 |
| em | 10 |
| dv | 11 |
| ca | 12 |
| ja | 13 |
| db | 14 |
| wt | 15 |
| rs | 16 |
| ic | 17 |
| ac | 18 |
| cn | 19 |
| sd | 20 |
| cb | 21 |
| vd | 22 |
| ft | 23 |
| df | 24 |