Hiding ourselves from community detection through genetic algorithms.

Sofia Almirante Castillo, 1527718
Rut Blanco Prieto, 1490761
Sergi Picó Cabiró, 1531767
Pau Reig Llunell, 1527816

January 2023

1 Community detection algotithms

These days, many companies are interested in detecting the social networks that are established between persons. To achieve this, the community detection method is used, which is based on the network topology and assumes that the community structure conforms to the network topology characteristics of specific connections with relatively dense nodes within the community and relatively dispersed connections between communities. Being aware of the groups that are formed in a network can be useful for them for the following reasons:

- Understanding how information flows through the network.
- It is easier to know the interests of the individuals that form the network and can help the business to focus their advertising targets
- People that influence others can be detected and making bigger market efforts on them can be very beneficial.
- It allows to know the relationships in a real network by analysing the internal hierarchies formed as well as the existence of internal organisation at a higher level of precision.

However, some people are worrying about their privacy and security because of the development of community detection techniques. Some information might be overexposed. For example, one can have their identity, interests or occupations invisible but, if this person is detected in a community, all this information can be made visible. That is why developing hiding community algorithms is day after day more demanded.

Community detection is the process of identifying groups of nodes in a graph that are more densely connected to each other than to the rest of the graph. There are many algorithms and approaches for detecting communities in a graph, and some of these approaches may be susceptible to being ”gamed” by nodes that are trying to hide from the community detection process [3].

One way that a node could try to hide from community detection is by reducing its connectivity to the rest of the community. For example, a node might remove its connections to other nodes in the community, or it might reduce the weight of these connections. This can make it harder for the community detection algorithm to identify the node as being part of the community.

Another way that a node could try to hide from community detection is by increasing its connectivity to nodes outside the community. This can make it appear as if the node is more closely connected to nodes outside the community than to those within the community, which could cause the community detection algorithm to classify the node as being part of a different community.

It's worth noting that these strategies may not always be successful in hiding a node from community detection, as the algorithm may still be able to identify the node as being part of the community based on other factors, such as the overall structure of the graph or the patterns of connectivity among the nodes.

1.1 Girvan-Newman algorithm

One example of community detetion algorithm is the Girvan-Newman algorithm. It is a topology-based algorithm that computes the betweenness centrality of the edges to break the graph and isolate communities.
The betweenness centrality of an edge is defined as the number of shortest paths between every pair of nodes of the graph.
With this in mind, the algorithm steps are:

- 1. The betweenness of all existing edges in the network is calculated.
- 2. The edge(s) with the highest betweenness are removed.
- 3. The betweenness of all edges affected by the removal is recalculated.
- 4. Steps 2 and 3 are repeated until no edges remain.

As a result of this algorithm, we obtain a different graph at each iteration. At some iteration, the graph starts to break in smaller graphs, we can identify these sub-graphs as the different communities, and depending on the number of iterations we compute we will go further (in the number of the communities). This is dangerous because at the end, for large enough iterations, the algorithm ends with every single node as a sub-graph, which does not give any information of the community structure of the graph. This is why normally the result of this algorithm is represented as a dendrogram.

2 Community hiding method

2.1. Genetic algorithm

A genetic algorithm is a random-based classical evolutionary algorithm. In the genetic algorithm random changes are applied to the current solutions to generate new ones. This kinds of algorithms are based on Darwin's theory evolution, slow gradual process that makes slight changes to its solutions until getting the best solution [2].

The genetic algorithm works on a population consisting of some solutions where the population size is the number of solutions. Each solution is called individual and each individual solution has a chromosome and a fitness value. The chromosome is represented as a set of features that defines the individual composed of a set of genes [Fig. 1].

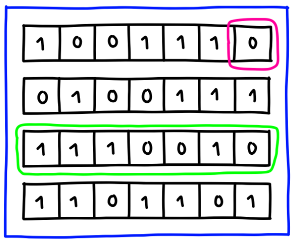


Figure 1: Composition of the population in the genetic algorithms. In blue is marked the whole population, in green the chromosome representing each individual solution, and in pink the genes. Binary example.

Every two parents can produce two offspring. If only high quality individuals are mated, it is expected that better quality offspring will be obtained than their parents, which will prevent bad individuals from generating more bad individuals.

But the offspring currently generated using the selected parents only have the characteristics of their parents and therefore the same drawbacks will actually exist in the new offspring. To overcome this problem, some changes will be applied to each offspring to create new individuals. The set of all newly generated individuals will be the new population that will replace the old population used previously. Each population created is called a generation, and this process of replacing the old population with the new one is called replacement.

These changes applied to the parents can be summarised as crossover and mutation. Crossover generates new generation the same as natural mutation, the new generation offspring comes by carrying genes from both parents, knowing that the amount of genes carried from each parent is random. In mutation, for each offspring select some genes and change its value. The fitness value of each chromosome is calculated according to the fitness function and the chromosome with the highest fitness value is selected [Fig. 2]. After selection, crossover and mutation of several generations, the optimal attack strategy was obtained.

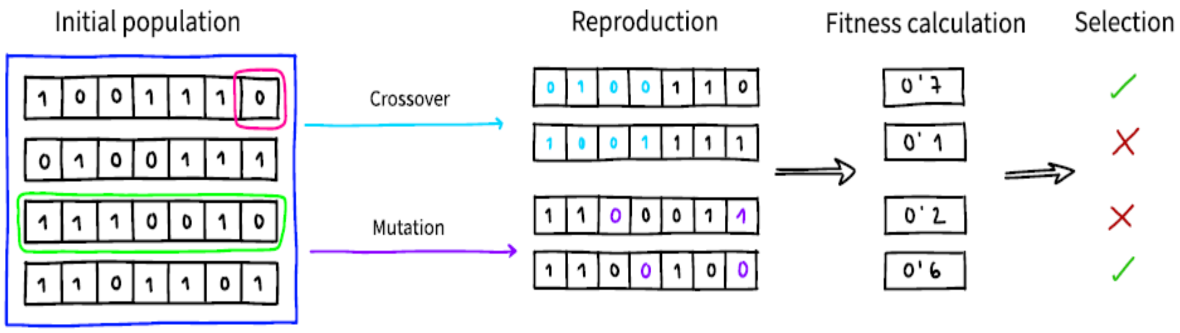


Figure 2: Mating pool and variation operators of the chromosomes. Example in binary

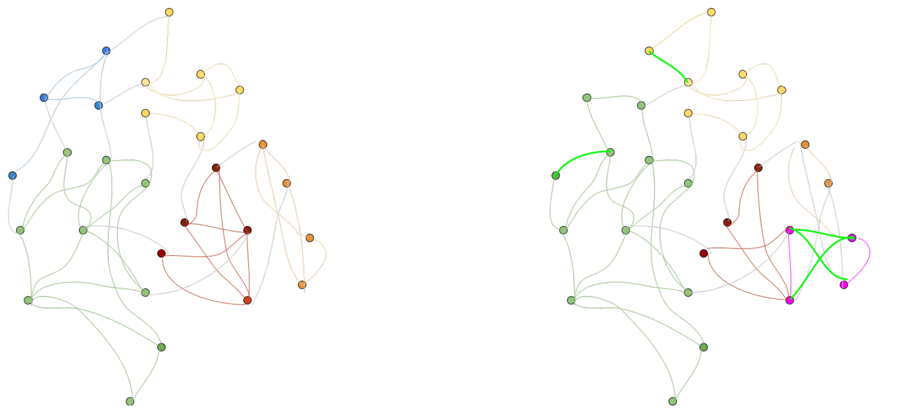
2.1.1. Genetic algorithm for community hiding

Genetic algorithms are chosen to be used for this class of problems because they have already given good results when used to solve optimisation problems with complex networks and, in addition, graphs related to social networks are usually formed with discrete values, which fits with the data that genetic algorithms work with.

The community detection algorithms and the hiding community detection algorithms start from a model of the network based on graph theory. One can represent a set of people as the nodes of an undirected graph, and the relations that are established between them as edges.

The fundamental strategy of hiding community algorithms is to add or remove edges from the original graph. This procedure will be called "attack", and will give as a result a new graph and a community detection algorithm might not identify the same communities for the new graph as before. The more difference between the graphs before and after the attack, the more efficient it will have been (difference in the sense of detecting communities).

In fact, this method represents an optimization problem, since our real goal is to introduce or delete the smallest number of edges to the initial graph so that the community detection algorithm identifies the least number of real communities that exist in the network. In this context, we will be using the word "distance" when referring to the difference in community detection between two graphs. Let us introduce now the notation that we will be using throughout the project:



(a) Initial graph. Before the attack

(b) Resulting graph. After the attack

Figure 3: Different communities detected because of the attack to the initial graph

The initial graph [Fig. 3] that represents a set of people will be denoted as G which is characterized by the number of nodes V and edges E. A community detection algorithm will extract K communities, which will be K "subgraphs" of G. They will be defined as the sets G_i, where $i = 1, 2, ..., K$. The resulting community corresponding to the i-th element will be represented by $\bar{C}_i$.

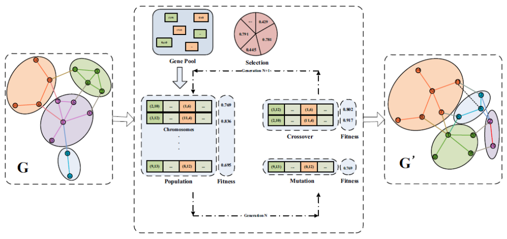
2.2. Frame work

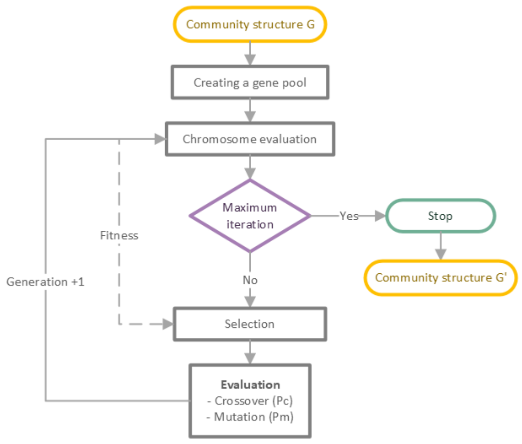
There are three main steps to achieving community hiding:

1. Community detection with an algorithm for detecting the graph
2. Rewiring the edges in the graph
3. Community re-detection and results comparison

The final objective is to find the smallest set of edges that can be removed o added to have the greatest impact on the structure of the graph through a genetic algorithm.

At the beginning of the algorithm, the chromosome groups representing individuals with different characteristics are initialized, and then descendants are generated according to the designed crossover, mutation and other genetic operators. Better-adapted individuals have a greater chance of surviving and reproducing, so populations can evolve. Therefore, we further designed a more feasible hiding strategy that can implement the attacker's behavior more effectively.


(a) Overview of the strategy.



(b) Flow-chat for community hiding

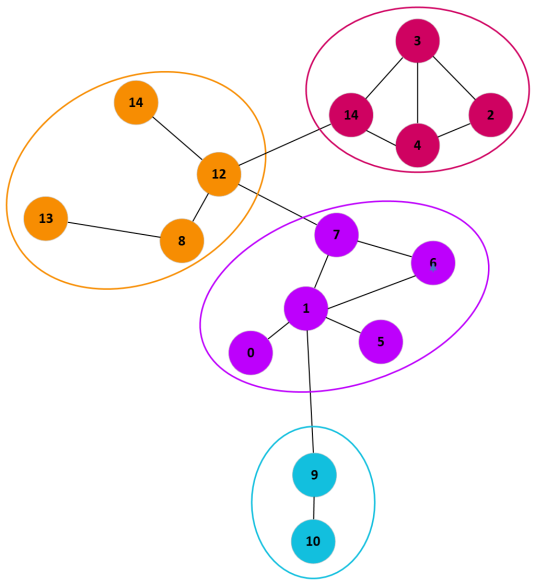
Figure 4: Overview of the Genetic Algorithm

More specifically, as can be seen in the Figure 4, the detection algorithm discovers a community structure represented in graph G. From this structure, a gene pool is created from which the genes that will form the different chromosomes (individuals) that will make up the population are obtained. After crosses and mutations to give several generations, the most optimal attack strategy is obtained. In this last generation (G'), the community detection algorithm is used again, giving a graph with a different structure to that of G.

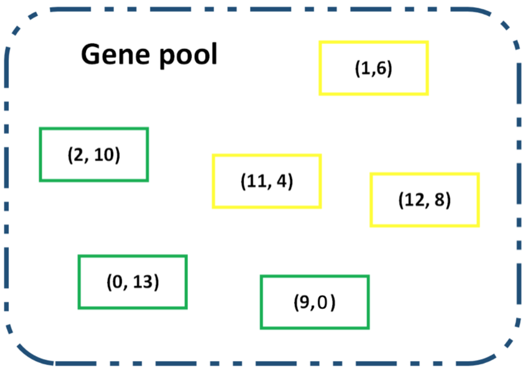
2.2.1. Important concepts

In order to be able to carry out this process, some important concepts need to be put forward:

- Gene pool: serves to reduce the size of the knowledge space and improve the efficiency of the algorithm. It contain prior information according to the community structure discovered by the community detection algorithm, edges within the community (edges that can be removed) and edges between communities (edges that can be added) are mixed into a set [Fig. 5b]. Each edge of the set is represented as a gene. Genetic markers formed by edges within the community are removed, and genetic markers formed by edges between communities are mixed into a set formed by the edges between the communities are added. From this gene pool, the edges that form the chromosomes are randomly extracted, whether they are added or extracted edges.



(a) Example of a community structure.



(b) Example of a gene pool. In green are the edges between communities and in yellow the edges within the community.

Figure 5: Example of a gene pool

- Encoding: under budget constraints (β), a chromosome is made up of randomly selected genes that are not put back into the gene pool. The length of the chromosome is the budget value. According to the community structure in the original network, edges inside the community were selected as edges to be deleted and edges outside the community were selected as edges to be added to form a gene pool. According to the budget value, the edges in the gene pool are selected to form chromosomes. The encoding process can be seen in the Figure 6

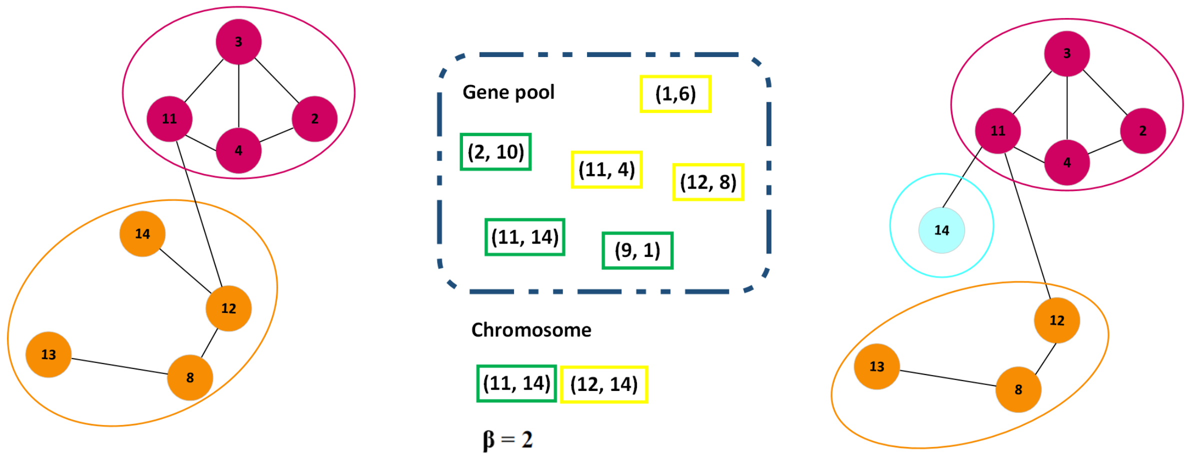


Figure 6: Example of the encoding process

- The fitness function f is defined as:

$$f = 1 - NMI$$

where NMI stands for *Normalized Mutual Information*. It takes values between 0 and 1 and it represents the degree of difference between two networks. So, the objective is to maximize this value because the bigger it is, the better performance the attack has shown (the two networks are more different).

- Population. The set of chromosomes in which the number of individuals is called the population size [Fig. 7]. Each chromosome in the population can crossover and mutate, and the optimal chromosome is the attack strategy with the maximum value of the fitness function to be adopted. In turn, a chromosome is composed of several edges that are marked as deleted or added. Each chromosome will get its own f value.

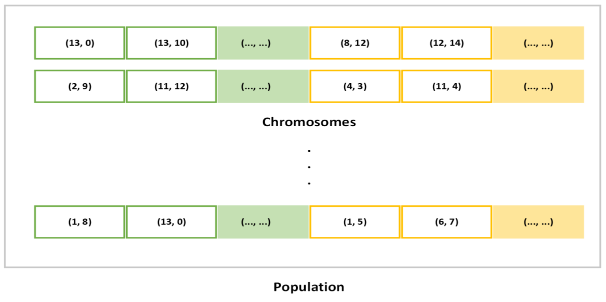


Figure 7: Example of different chromosomes

- Crossover: two chromosomes are cut at the same place and the two strands are crossed to form two new chromosomes [Fig. 8]. Here, a random generation method of an adopted cut point is adopted to switch gene fragments between the parents. The probability that a pair of individuals will produce offspring by crossover operations is represented by P_c .
- Mutation: it is possible to make some kind of replication error, mutate into new chromosomes and show new traits [Fig 8]. This is reflected in the genetic algorithm to avoid falling into the local optimal solution. This is done by selecting a random gene from the gene pool to replace a random gene on the chromosome. The mutation probability is expressed by P_m.
- Selection: in nature, the more adaptable individuals are, the more likely they are to reproduce. However, the fittest does not necessarily have to have the most offspring. The fitness values of the chromosomes in a population form a roulette wheel, and the higher the fitness value, the larger the area of the roulette wheel occupied by the chromosome. In random selection in the roulette, the chromosome has a higher probability of being selected [Fig. 8]. The probability of being selected is based on: $P_i = \frac{f(i)}{\sum_j f(j)}$.

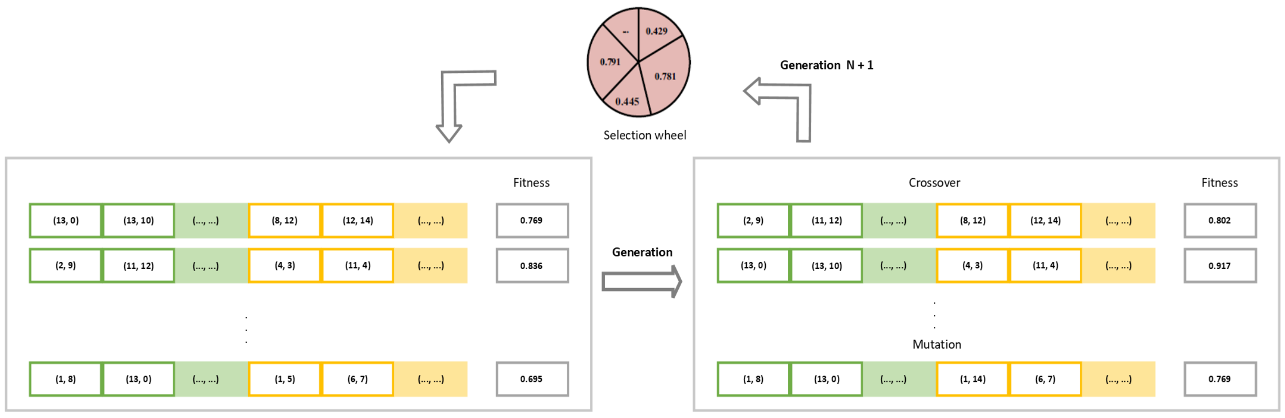


Figure 8: Crossover and mutation of the chromosomes, and formation of the N + 1 generation by means of the selection wheel and the fitness values.

- Termination conditions: the evolution generation is set to be a constant, and when this condition is met, the algorithm stops.

2.3. Metrics

In order to be able to apply the algorithm to real data, it has to be done with certain metrics:

- **Normalized Mutual Information (NMI):** This magnitude represents the mutual dependence between the graph that we obtain and the original graph. Its origin comes from the information theory and it takes values from 0 to 1. [1] The smaller it is, the more different the attacked network is from the original one. The general formula is given by:

$$NMI(X;Y) = 2\frac{I(X,Y)}{H(X)+H(Y)}, \qquad \text{where} \begin{cases} I(X,Y) = H(X) - H(X|Y) \\ H(X) = -\sum_i p(x_i)\log p(x_i) \end{cases}$$

  Here X and Y represent random variables which in our case will represent the x_i-th community for the case of the X random variable and the y-th network, which can only take values from 1 to 2 since we are always comparing two graphs. The probabilities that we show on the formulas stand for the probability that a node is found in the community x_i (for the case of $p(x_i)$) and the probability that a node belongs to the $y-th$ network (for the case of $p(y_i)$). Here we show a little example of two different graphs [Fig. 14]:

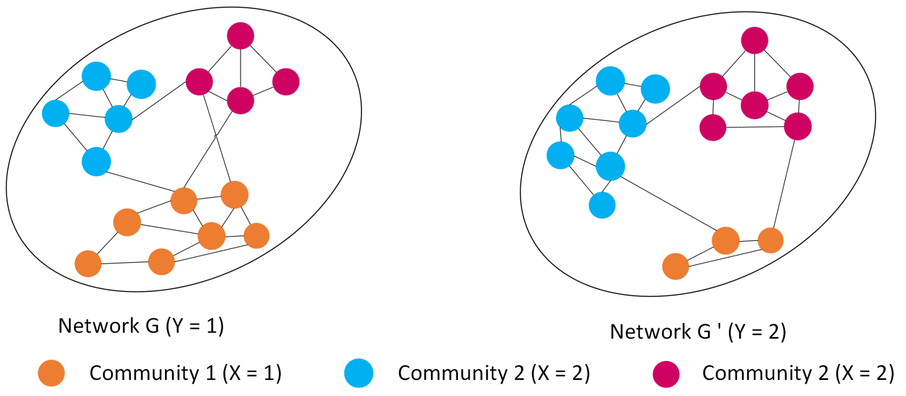


Figure 9: Image of two networks ($Y = 1$ and $Y = 2$) before and after a attack containing three different communities $X = 1$, $X = 2$ and $X = 3$.

  Where it is clarified the meaning of these random variables X representing communities and Y representing the network.

- **β times NMI (BN):** Shows the cost of an attack. This attack cost is a measure of how many edges are modified while lowering the NMI value. This value will be greater then 1 and smaller than β, which means that the attack was successful despite the cost of the comeback, and the smaller the value the more successful the attack has been. Therefore, BN is defined as: $BN = \beta * NMI$

- **Modularity (Q):** It is a method used to measure the structural strength of network communities. The value of modularity mainly depends on the community allocation of nodes in the network, that is, the community division of the network. It can be used to quantitatively measure the quality of network community division. The closer the value is to 1, the stronger the community structure divided by the network, and the better the quality of network division. The modularity is defined as. $Q = \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})\delta(i,j)$. Where m represents the number of edges in the network, A_{ij} represents the number of edges between node i and node j, and k represents the degree of the node $\delta(i,j) = 0$ means that node i and node j are not in the same community and $\delta(i,j) = 1$ means that node i and node j are in the same community.

3. Simplified example

For this part we consider a simple example of a population represented in the following graph [Fig. 10], where the different colors of the vertex indicate the different communities.

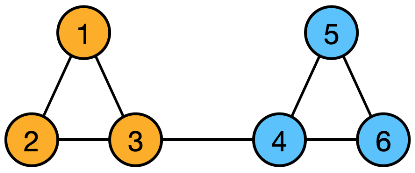


Figure 10: Simple example of a population represented by a graph.

First we are going to use the Girvan-Newman topology-based algorithm to prove that the communities are well defined in the graph.

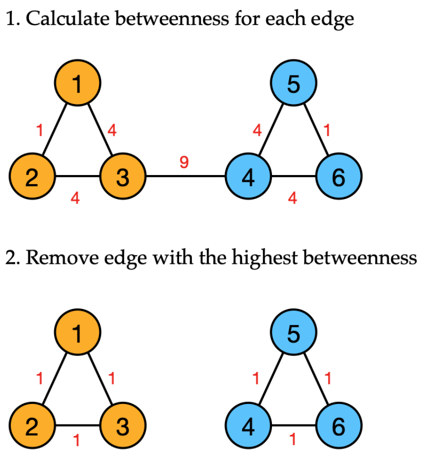

1. Calculate betweenness for each edge
1
5
1
4
4
1
9
2
3
4
6
4
4
2. Remove edge with the highest betweenness
1
5
1
1
1
1
2
3
4
6
1
1

Figure 11: First (and in this case, the last) iteration of the Girvan-Newman algorithm to detect communities.

Once we have the initial graph with its known communities, we are going to start the attack.

3.1. Parameters

In this example, we are going to set the following parameters:

- $\beta = 3$
- $SelectionSize = 3$
- $P_c = 0.6$
- $P_m = 0.01$

3.2. Gene pool

In red, the edges of the same community (will be removed) and in green, the edges connecting people from different communities (will be added).

$$\text{Gene Pool} = \{(1,3),(1,2),(3,2),(4,5),(5,6),(1,4),(2,4),(1,5),(2,5),(3,5),(1,6),(2,6),(3,6)\}$$

3.3. Chromosomes

With the boundary of $\beta = 3$ we can randomly generate 5 chromosomes (repeating just one gene).

$$\text{Chromosome 1} = \{(1,6),(1,3),(3,2)\} \quad \text{Chromosome 4} = \{(2,5),(4,6),(5,6)\}$$

$$\text{Chromosome 2} = \{(1,4),(2,6),(1,2)\} \quad \text{Chromosome 5} = \{(3,5),(2,6),(4,5)\}$$

$$\text{Chromosome 3} = \{(2,4),(3,6),(1,5)\}$$

3.4. Compute f for each chromosome

To be able to automatize this part, we have build a code[1] that, given the graph shown in Fig.10 and a chromosome, it computes the f associated to it.
To do it, it applies the change described by the chromosome to the original graph and then, using the Girvan-Newman algorithm, it decomposes the new graph into its "detected" communities. With this decomposition we can compute f using Eq.2
In the article, they compute f using the NMI between the two graphs (attacked vs original). We have not found a way to compute this NMI between graphs, so we have used the NMI between the communities and the graphs in the following way:

$$\text{NMI}(Y, C) = \frac{2I(Y;C)}{[H(Y) + H(C)]} \tag{1}$$

where Y are the different communities and C the graphs.
With this idea, due to the fact that two identical graphs have $\text{NMI}(Y, C) = 0$, we are going to maximize this NMI. Therfore, in this example f =NMI.
This is not very rigorous, but this article is not about calculating NMI, so we will assume that our calculation makes sense.
Using this equation and the code provided in the delivery, we obtain:

Chromosome	f
1	0.08
2	0.14
3	0
4	0.08
5	0.17

Table 1: Table showing the corresponding f to each chromosome.

3.5. Selection

The probability of being selected, as we have seen, can be computed as

$$P_i = \frac{f(i)}{\sum_j f(j)}.$$

[1]See apendix

In our case:

Chromosome	P_i
1	0.17
2	0.30
3	0
4	0.17
5	0.36

Table 2: Table showing the corresponding probability of being selected (P_i) to each chromosome (i).

$$\text{Selection} = \{\text{Chromosome } 4, \ \text{Chromosome } 5, \ \text{Chromosome } 2\}$$

3.6. Crossover and mutation

As we have fixed, the probability of two chromosomes being crossed is $P_c = 0.6$.

$$\text{Chromosomes crossed} = \{\text{Chromosome } 4 - \text{Chromosome } 5, \text{Chromosome } 4 - \text{Chromosome } 2\}$$

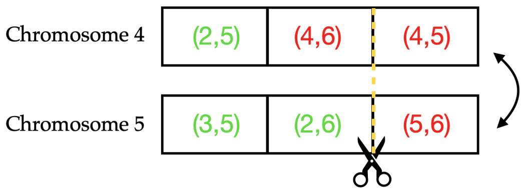


Figure 12: Crossover between chromosome 4 and 5.

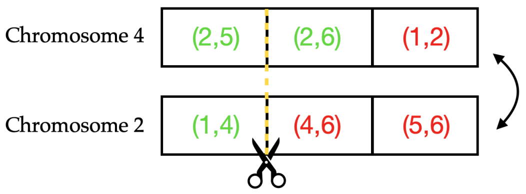


Figure 13: Crossover between chromosome 4 and 5.

And we will also add a mutation to the crossover of the cromosomes 4-5 such that:

$$\text{Chromosome } 1^* = \{(3, 6), (2, 6)(5, 6)\}$$

With all of this, we now have the new 5 chromosomes that form the next generation:

$$\text{Chromosome } 1^* = \{(3, 6), (2, 6), (5, 6)\} \quad \text{Chromosome } 3^* = \{(2, 5), (4, 6), (4, 5)\}$$

$$\text{Chromosome } 2^* = \{(2, 5), (2, 6), (1, 2)\} \quad \text{Chromosome } 4^* = \{(1, 4), (4, 6), (5, 6)\}$$

$$\text{Chromosome } 5^* = \{(3, 5), (2, 6)(5, 6)\}$$

We should repeat the process until the maximum iteration (that we choose) is reached.
In the generation 3, the chromosome with higher f ($f = 0.22$) correspond to:

$$\text{Chromosome} * = \{(1, 4), (2, 6), (5, 6)\},$$

Which corresponds to the following final graph after the attack:

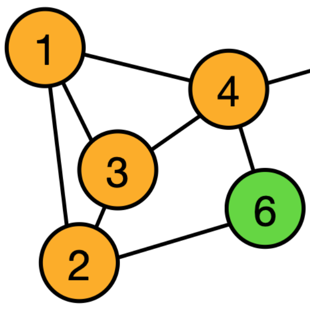


Figure 14: Modified graph after the attack.

4. Results

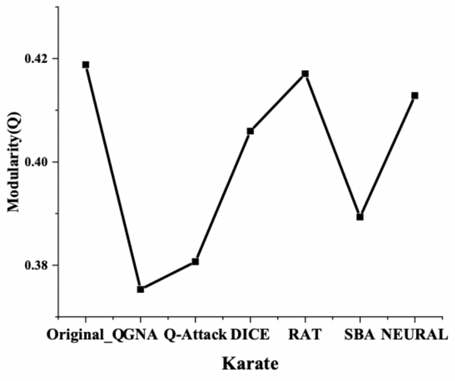
The team carried an experimental evaluation of the CGN algorithm applied to four different populations given by real datasets[2], comparing the community (detected by four different community detection algorithms[3]) before and after the attack. They also compared the differences between the communities when attacking it with other five algorithms[4]. The parameters used were $P_c = 0.7$, $P_m = 0.01$ and β had a variable value.

As shown [Fig. 15], the CGN algorithm was the one that obtained a lower Modularity Q in all databases.

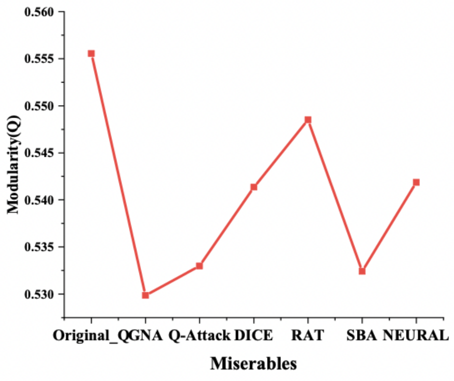
[2]The Zachary Network (Karate club), the characters in the novel Les Miserables, authors in the Digital Bibliography Project (DBLP) and medical citations in PubMed.

[3]Multilevel, Fastgreedy, Infomap and Label Propagation

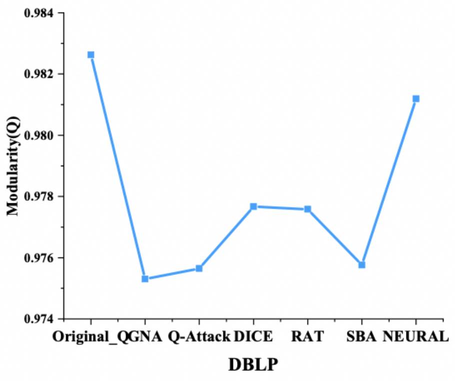
[4]Q-Attack, DICE, RAT, SBA and NEURAL



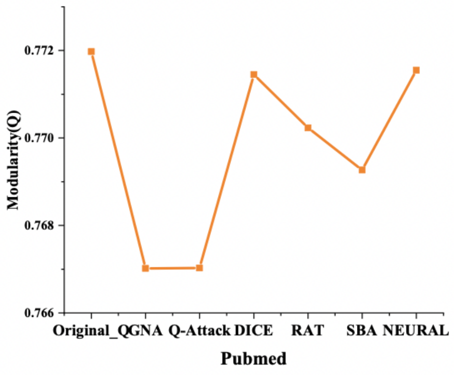
(a) Based on Multilevel algorithm



(b) Based on Multilevel algorithm



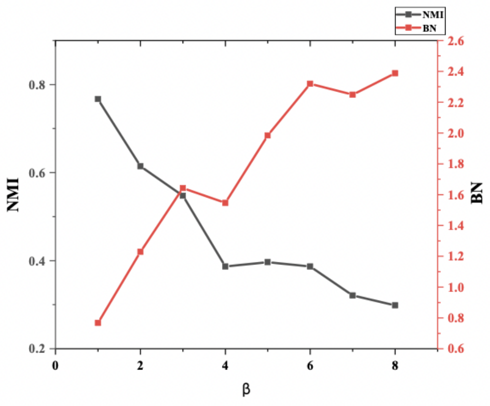
(c) Based on Multilevel algorithm



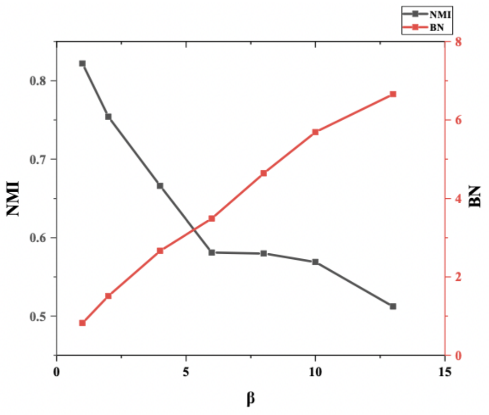
(d) Based on Multilevel algorithm

Figure 15: Caption

After the study of the performance of NMI and BN under different budgets β shown in [Fig. 16], it was confirmed the hypothesis that the higher the β the better the attack would be. In order to add an invisible disturbance to the network, they imposed a maximum value of *beta* corresponding to the 5% of edges of the original community.



(a) Karate



(b) Miserables

Figure 16: Mean values of MNI and BN in function of β.

It can also be observed that good attack effects are shown even for $\beta = 1$. When adding only one edge to the graph, it has been found that the community in which the nodes altered reside does not change much, but it changes dramatically the affiliations for further away communities, as shown in [Fig. 17].

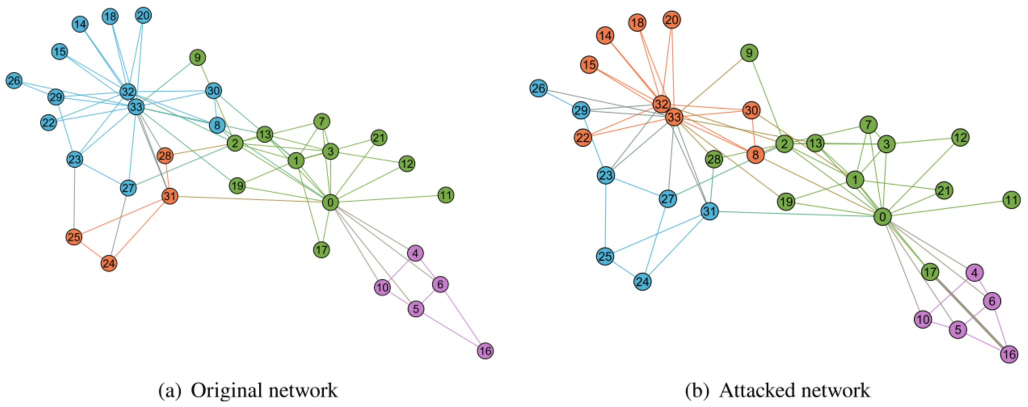


Figure 17: *Karate* network before and after the CGN attack, $\beta = 1$.

They found that the algorithm using NMI as fitness function has a faster convergence speed and higher efficiency than using Q. The performance time of the algorithm also decreased drastically when using a gene pool with prior information than using a normal gene pool. This can be observed in [Fig. 18(a)] and [Fig. 18(b)] respectively. The introduction of the gene pool with prior information also results in a better attack effect because it guarantees that the tightness within the community is destroyed.

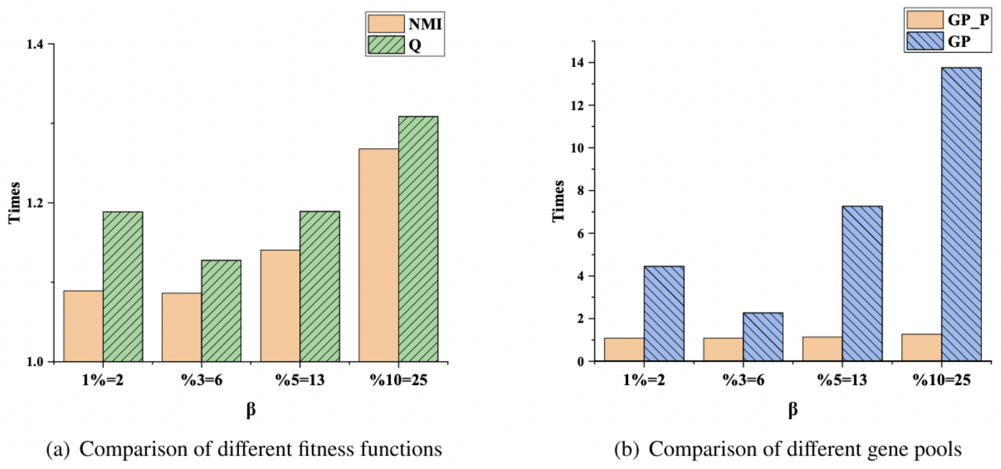


Figure 18: Caption

They also concluded that the CGN algorithm has the same effect in both white-box attacks and black-box attacks, which is adding imperceptible disturbances to the original community in order to destroy the structure.

Overall, the paper proposed a community hiding algorithm based on a genetic algorithm with NMI as the fitness function, since its minimum value indicates a bigger difference in the structure of the network before and after the attack. Compared to other advanced baseline algorithms and attacking different community detection algorithms, the algorithm showed an excellence performance. It achieved optimum results using different gene pools and fitness functions, and showed a good performance in both white box and black box attacks. For these reasons, we conclude it is possible to use the CGN algorithm in order to protect people's privacy from community detection algorithms in real social networks.

5. Apendix

The code is quite long, we will load it to the campus virtual and here we will explain the execution instructions.
The code reads a graph in the following format:

```
gorder #number of nodes
gsize  #number of edges
i j    #edge between node i and j
```

Even though the procedure to find the graph decomposition is general for any graph (just by changing the definition of N as the number of nodes/people), the calculation of f is specific for the example graph shown.
Compile and execution line:

```
gcc -o Ex Example.c -lm -Wall #compile
./Ex GraphEX.txt #execute (GraphEX.txt is the graph written in the format mentioned)
```

When the code runs, it will ask you to enter some numbers:

- First it will ask you for the chromosome that you want to study, they are defined and labeled on the code.
- Then it will ask you for the number of iterations for the Girvan-Newman algorithm.
- Finally, when the classes are shown to the screen, one has to enter the number of elements in each class starting with the two classes that are present in the original graph before the attack. The numbers have to be written separated by blank spaces.

It is a code that works (in principle), although it has been done fast without time for thinking in its efficiency and cleanness.

References

[1] , . Normalized mutual information: estimating clustering quality. URL: https://course.ccs.neu.edu/cs6140sp15/7_locality_cluster/Assignment-6/NMI.pdf.

[2] Gad, A., 2018. Introduction to optimization with genetic algorithm. URL: https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b.

[3] Liu, D., Chang, Z., Yang, G., Chen, E., 2022. Hiding ourselves from community detection through genetic algorithms. Information Sciences 614, 123–137.