University of Groningen

Master's Thesis Artifical Intelligence

The Web-Graph: Clustering, Collecting & Classifying

Author:
Ivo de Jong *(s3174034)*

Supervisor:
dr. M.A. Wiering (Bernoulli institute)

External supervisor:
B. Zijlema, MSc (Dataprovider.com)

Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence
University of Groningen, The Netherlands

Dataprovider.com, The Netherlands

May 12, 2021

Abstract

The Web-Graph is an intriguing structure about the internet that arises from the hyperlinks between websites. While it has been studied to a practical extent for various purposes - and has also been effectively applied as important drivers in how the internet is used - specific research dedicated to community detection on the Web-Graph has been missing.

Community detection (i.e. graph clustering) has been studied from several theoretical perspectives and methodological frameworks. A sparse segment in this research is Genetic Algorithm based Modularity maximization. The first chapter of this research explores variations on the state of the art in this domain, and finds an improvement that may be universally relevant when designing Genetic Algorithms. Ultimately however, statistical inference methods for community detection are found to vastly outperform Genetic Algorithm methods. While this is only shown for the Web-Graph, it may hold universally.

Statistical community detection methods are subsequently used for a novel approach to improve an existing website Trust Score regression model. By predicting the Trust Score of a site using some nearby sites (either from BFS or clusters), two models with some error distributions can create a joint probability distribution of where the true Trust Score should lie. This chapter did not find any benefit to the clusters beyond the Web-Graph, but it does propose a novel method for improving an existing regression model without a ground-truth dataset.

Lastly, this research investigates the use of statistical community detection for Fake News site classification. By taking a BFS graph sample from a labeled training set, candidate websites are collected and clustered. A classifier using cluster indices as features outperforms one based on extracted keyphrases on the testing set, demonstrating the effectiveness of Web-Graph community detection. Unfortunately, neither classifier generalizes beyond the constructed dataset, indicating a problematic bias in that dataset. Nonetheless, this does not indicate any problems with the designed collection & classification method.

The research overall concludes that community detection on the Web-Graph could provide valuable information for some website classification or regression tasks. However, it should be noted that this is computationally costly, even when using efficient statistical inference methods. For tasks where connectivity is particularly relevant, and other features are missing or inadequate, community detection systems may provide a solution.

Acknowledgments

During the writing and research of this thesis I have been thankful for the support of those around me.
Firstly, I would like to extend my gratitude to my supervisor Dr. Marco Wiering for finding the time and space to supervise me and for keeping me motivated.
I would also like to thank my second supervisor Bastiaan Zijlema, MSc. for his practical and effective day-to-day guidance and collaboration.
Additional thanks goes out to all my colleagues at Dataprovider.com for inspiring me and thinking along with me during this research. I would specifically like to thank Edine and Marjolein for going through the arduous task of labeling Fake News sites.
Lastly I would like to thank my parents for providing me the comfort from which I've been able to do this research, and my partner for keeping me grounded throughout this process.

Contents

Chapter 1

Introduction

ARPANET, initially demonstrated in 1972 at the ICCC can be considered as the forefather of the internet today [36]. This was also the year that e-mail was first introduced. This fundament of *people-to-people* traffic persists through into the architecture that is the internet today. By design, anyone with an internet connection can send and receive data to anyone else with an internet connection. This design grants anyone the ability to start a website for any purpose, without anyone's explicit permission. The Moore-like [55] exponential increase in the number of websites on the internet [20] may be attributable to this very freedom of participation.
This open participation model of the internet unfortunately also invites malicious actors to make their own websites. To keep the internet as safe and positive as possible these ill-disposed websites need to be tracked down so that users may be warned or so that law enforcement can catch the malicious actors behind them. This thesis approaches this task of finding dishonest websites through Web-Graph clusters [33].

1.1 Web-Graph

The Web-Graph is a graph structure formed by websites and the hyperlinks between them. These hyperlinks allow users to hop from website to website, in what is classically called *surfing the web*. All these websites linking to other websites form a directional graph structure which is referred to as the Web-Graph.
This Web-Graph can be investigated in methods similar to social network analysis [43]. SNA, originating from sociology, investigates people and the connections that they have with other people. These people are all different, and so are the purposes of the connections that they may have between them. Either way, the social network around a person can be used to say something about that person.
The same concept holds on the Web-Graph. Many different sites can link to other sites for a variety of reasons with a variety of contexts. The community that a website exists in can be used to make certain judgments about that website. As such, a website may be judged "by the company they keep".

1.2 Research Questions

Since there is a wide range of ways that a graph section can present itself there is a need to find a consistent method to reduce the whole Web-Graph structure to usable components. The

current research investigates the use of Web-Graph clustering for this specifically. By applying methods to make clusters using the graph-structure, it may be possible to draw conclusions about a website based on its clusters. The main question that this research intends to answer is how graph-clustering can be applied to the Web-Graph and how these clusters can be used to identify malicious websites.
This research question contains two core parts. Firstly, it addresses a method of clustering. Second, it attempts to demonstrate value of these clusters by identifying malicious websites with them.

1.3 Clustering

There are several methods available for the former. Chapter 2 explores this by attempting to improve an existing Genetic Algorithm for Graph Clustering. Specifically, it takes a critical look at the design of CC-GA [53]. CC-GA is a Genetic Algorithm that attempts to optimize modularity [41], a metric for clustering quality. That chapter therefore aims to answer the question of how CC-GA may be improved to achieve a modularity as high as possible, as fast as possible. Chapter 2 describes some theoretical issues that CC-GA has, leading to a hypothesis that there is indeed room for improvement.

1.4 Malicious Website Identification

Chapters 3 and 4 investigate how constructed clusters may be applied to identify malicious websites. Specifically, they respectively explore a Trust Score regression task where an existing trust-score is improved using graph clusters, and a fake-news classification task where dedicated fake-news sites are identified by the known fake news sites in their cluster.

1.4.1 Trust Score Re-estimation

As a demonstration of possible use for Web-Graph clusters an investigation into their use for Trust Score re-estimation is performed. For this an existing Trust Score estimation method already exists [39] and needs to be improved. The existing Trust Score considers intrinsic properties about the website, for example whether they have an SSL certificate. The new research attempts to improve upon this by collecting nearby websites, either with BFS or from the cluster, and try to re-estimate a website's Trust Score based on the Trust Scores of nearby websites. Specifically, it aims to answer whether collecting these nearby websites from a cluster gives a better accuracy than collecting nearby websites using BFS. It is hypothesized that using clusters can filter out *neighbours* that are less relevant as they belong to a different community.

1.4.2 Fake News Classification

Another demonstration of what Web-Graph clusters may be used for is given as a Fake News site classification task. This is considered a particularly potent candidate as the Fake News topics may form specific communities, and link to each other as *sources*. Research on Fake News site classification is currently rather slim, as most research is focused on Fake News through Social Media. To get a relative value of the clusters a classifier based on Stochastic Block Model (SBM) clusters [31] as features is compared to a classifier using extracted keyphrases as features. Chapter 4 subsequently intends to answer the question whether SBM clusters are a good feature for Fake News site classification compared to extracted keyphrases. It is expected that extracted keyphrases are a better reflection of the content on the site than the clusters, thus resulting in a better classifier. However, considering the idea that Fake News persists in *echo-chambers* it is expected that the clusters could also result in a rather good classifier. Ultimately, since both these features reflect different information sources, it is expected that a classifier with both feature sets performs best.

Chapter 2

Improvements to CC-GA for Web-Graph Clustering

Abstract The links between websites can be agglomerated into a graph structure. Graph-clustering Genetic Algorithms are applied to maximize Modularity, a measure of community structure. An efficiency improvement to CC-GA is found by applying a stochastic variation of the initial population generation. This vastly speeds up convergence. Nonetheless, the research concludes that genetic algorithms are not the best fit for maximizing modularity. The parallel research area of statistical inference of Stochastic Block Models outperforms any genetic algorithm for graph clustering. These two fields work with different measures of quality and are therefore rarely compared. The conclusion from this comparison is that Genetic Algorithms are inferior to statistical inference algorithms for graph clustering.

2.1 Introduction

The World Wide Web is named after the connective structure that underpins the internet architecture. This network structure not only exists as the system routing packages and connecting devices, it also exists on a macro scale as websites linking to each other. "Surfing the web" as it was called in the first decade consisted of hopping from website to website through links.

For most of the websites that people visited the user doesn't need to know the *url*, instead we rely on the websites that we do know to *link* us to the unknown websites with more information that we want. A great example of this can be found in the sources of Wikipedia. The user doesn't need to know the urls of the sources from a Wikipedia article, but they can click the relevant links to read further into the topic.

Aside from these *hubs* an appreciable amount of linking is used for referring users to affiliated websites. A charity might link to sponsoring companies, or an online retailer might link to their suppliers. These links provide a more interesting effect, in that they show some contentual relationship between a sender and receiver. Several aquarium retailers may link to their fish-supplier, which might also share some links with aquarium maintenance advice. By collecting the cluster of websites that are connected through these links it may be possible to isolate a set of websites that are related to each other.

2.2 Clustering

Clustering a graph is far from trivial. Several surprisingly challenging problems come up. The first problem is to exactly define what a good clustering is. The definition of a good clustering is often problem specific. For one: some problems might prefer larger clusters, while others prefer smaller clusters. The second part in this is that what constitutes a valid clustering is also problem dependent. There are definitions and algorithms for overlapping clustering, where any vertex may be part of multiple clusters, but there are also cluster definitions where any vertex may or may not be part of a cluster [54]. There are even hierarchical clustering tasks where each cluster is clustered into sub-clusters [32], which in turn gives an entirely different clustering result and challenge.

2.2.1 Solution Definition

The current research focuses on non-overlapping, non-hierarchical clustering. This is a comparatively simple problem definition, which most importantly gives a very simple to interpret result. It finds that every vertex is in one cluster, which can simply be indexed. While the results are easy to work with, they are not easy to achieve. With a *valid* clustering defined, a *good* clustering still needs to be defined. On small graphs people can easily make an intuition of where to cut clusters, provided that the graph is intuitively visualized, but a formal definition is not trivial. Depending on the formal definition certain aspects and situations might not be clustered as the human intuition would have wanted it [54]. A commonly accepted option is the Modularity measure for graph clustering [41]. Here, Modularity and thus the goodness of a clustering is defined as follows:

$$Q = \sum_{c=1}^{n} \left[\frac{L_c}{L} - \left(\frac{K_c}{2L} \right)^2 \right] \tag{2.1}$$

Where n is the number of clusters, L_c is the number of edges within cluster c, L is the total number of edges within or between clusters, and K_c is the average number of edges (within and between clusters) of the vertices in c. Taking a look at this definition shows that indeed having more edges within the cluster relative to the severed edges outside the cluster results in a higher modularity score.

While modularity is commonly used in Social Network Analysis [41] to make clusters similar to the task at hand here, it is still flawed. Modularity suffers from a resolution limit. Modularity fails to identify clusters smaller than $\sqrt{2L}$ [34] so that they get merged together. Optimistically, this bug may be dubbed a feature, as it ensures that the size of the clusters is consistently proportional to the size of the graph. Regardless, modularity is generally accepted as an acceptable solution [41], while its flaws are acknowledged.

This allows a formal solution to be defined, which is the clustering for a given graph which maximizes Q. Now a solution may be proposed and evaluated, and it is even theoretically (though not practically) possible to find the best solution through an exhaustive search.

2.2.2 Proposing Solutions

Accepting modularity as the quality of a solution, the most difficult task is to propose the right solution. Evaluating a proposed solution is now trivial, though slightly costly depending on the size of the graph. Any proposed solution should exist within what is referred to as the *search* *space*. The search space is the space of all possible solutions that may be proposed and evaluated. Within this search space lies at least one globally optimal solution, which is ultimately the desired outcome. For the task of clustering a graph with v vertices, therefore allowing up to v clusters, there are v^v possible solutions. For a graph with $v = 10$ vertices, assuming 1 second to calculate the modularity it would take over 300 years to try all possible solution with the exhaustive search previously suggested. Graph clustering has been proven to be an NP-hard problem [15], but no polynomial time algorithms are available. Between polynomial algorithms, and exhaustive searches exists a middle ground for clever solution sampling strategies.

2.2.3 Genetic Algorithm: General Concept and Clustering Implementation

Genetic Algorithms [38] are a biologically inspired class of algorithms that can be used to sample solutions from large search spaces provided that a performance can be quantified. They rely on Darwin's theory of evolution to propose various solutions that may be evaluated by their quantified performance. In the biological version this sample space is a genetic encoding, and the quantifiable performance is whether the genetic encoding produces a creature that is able to produce offspring so that their genes may be propagated.
When Genetic Algorithms are re-applied to solve other tasks some key components are required:

- A genetic encoding and a way to generate one
- A fitness function to evaluate an encoding
- An offspring function to create a new encoding
  - with crossover function based on parents with fitness values
  - and a mutation function

The following sections will describe how these can be implemented in general cases, as well as how they can be specifically implemented for the task of graph clustering.

Genetic Encoding and Generation

In order to propose a solution some formalized and consistent encoding is required. This may be designed in any way provided that the subsequent fitness and offspring functions can be applied to them. In biology we see that the genetic encoding is (mainly) the actual sequence of nucleobases adenine, cytosine, guanine and thymine that make up the DNA.
An example of an encoding can be found for the iconic Traveling Salesman Problem, where a salesman has to visit a number of locations and return home in the shortest distance possible. A simple encoding for this is to enumerate all the locations and encode a proposed solution as the sequence of locations visited [13]. This can then easily be translated back into the actual route. A TSP encoding is subject to the constraint that every city occurs exactly once in the encoding. Therefore a simple way to make an initial sample is to shuffle the cities and encode the path as such.
A wholly different example is Cooperative Synapse NeuroEvolution, where the weights of a Neural Network are learned to optimize any kind of task, including fighting forest fires [22]. For this an encoding can simply consist of the weight for each synapse in the Neural Network. In order to generate an encoding in this sense a weight for each synapse needs to be sampled

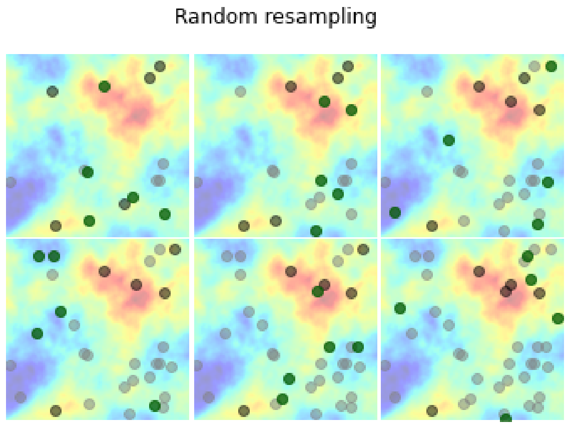
from some distribution. Any distribution will technically work, but some will work better than others. For a practical application a sensible range is essential, but a sensible distribution could also be very important.
For the graph-clustering problem at hand *locus-based adjacency representation* is typically the representation of choice [53, 56]. In this representation each node in the graph is connected to exactly one other node in the graph. When a set of nodes are linked to each other they are considered as one cluster. By connecting two nodes to each other, or nodes connecting to nodes that are already in their *cluster*, the graph gets disconnected into disjoint clusters. This encoding allows for an easy generation of an initial sample by selecting a single neighbor for each vertex as their connection. Selecting a connection only from the neighbors already makes a lot of bad clusterings impossible. By only allowing vertices to cluster with their neighbors it becomes impossible for a cluster to be split into two disconnected sections of the same cluster. This was technically possible in the problem definition, but would definitely give a bad result.

The Fitness Function

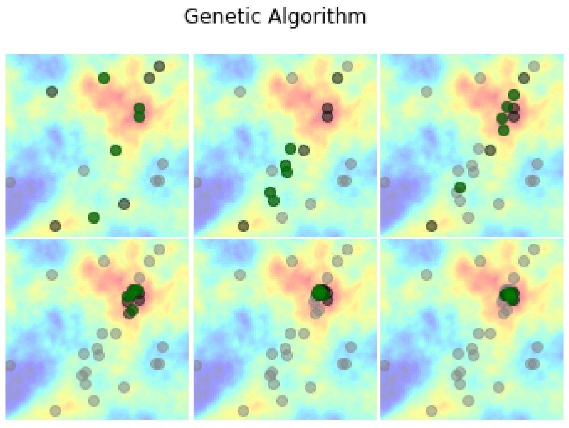
With a genetic encoding of a solution available, the next task is to evaluate that solution. This is done according to a fitness function. Since the Genetic Algorithm is exclusively tuned to score maximally on the fitness function it is important that the fitness function exactly reflects the desired goal.
For the task of the TSP this is trivial: take the series of location and sum the distances traveled from one location to the next. For a practical application of the same problem it should be noted that human common sense is generally neglected by algorithmic optimization so practicality like travel time or cost are ignored if they are not defined in the fitness.
A slightly challenging fitness function is found in hyperparameter optimization for stochastic learning [63]. In this case the fitness can simply be the accuracy that the learning system achieved for a given set of hyperparameters. The challenge here is that learning processes are often stochastic, so that even with the same encoding it is possible to get different fitnesses. Nonetheless, the genetic algorithm will still converge around a local optimum as the random sampling is more likely to persist in high fitness regions.
The fitness function for the clustering is of course the Modularity with its features and its flaws. Fortunately this is fully deterministic, so synonymous encodings that produce the same clusters in locus-based adjacency representation can be known to have identical fitnesses without requiring the modularity to be re-computed.

Offspring

The improvement that the Genetic Algorithm makes against random sampling comes from the way offspring is generated. Rather than simply trying solutions and seeing how well they perform, the Genetic Algorithm needs a way to generate candidate solutions based on the success of solutions it already knows. This is how a Genetic Algorithm is able to reasonably effectively explore the search space to find decent solutions.
Figure 2.1 illustrates how this producing of offspring compares against random sampling. The Genetic Algorithm ensures that the search space is mostly explored around areas which have been shown to give good performance. This works on the assumption that solutions that are similar (close to each other in search space) will have a performance that is also similar. Exploring solutions that are similar to once with good performance helps ensure that the new solutions will have a similarly good performance.



(a) Search with Random Sampling



(b) Search with Genetic Algorithm using euclidean crossover

Figure 2.1: The figures above show 6 generations of the search process of Random Sampling and a Genetic Algorithm across an artificial 2D performance landscape (Perlin Noise [48]). Light gray points are samples that have been explored and rejected. Dark gray points are points that are tried and among the top 50%. Green points are samples that are newly introduced and have not been evaluated.

The way offspring is generated is subsequently a significant factor in performance of a Genetic Algorithm. Generating offspring always involves some choices to be made. Firstly the choice stands of how to select parents. A universally acceptable strategy is to pick any 2 random parents from a set of top-performers [53, 56, 63, 22], but other strategies exist and may perform better [52].
The crossover and mutation operations jointly decide what offspring comes from two parents. The way these can be implemented fully depends on the task and genetic encoding. The simple demonstration in figure 2.1 can take the mean of both parents with some normally distributed noise in both dimensions to create a child that is around the average of both parents.
Other problems, like the TSP and the clustering task at hand do not have a valid solution in their average vector. Instead for the clustering task at hand ”Uniform Crossover” is used [53]. With uniform crossover each parameter is randomly selected from either parent. As from the genetic encoding described in 2.2.3 each vertex' preserved connection is randomly selected from the vertex' preserved connection of either parent. The mutation is then done by randomly changing some preserved connections in the locus-based adjancency representation.

2.2.4 A Genetic Algorithm for Clustering

The genetic encoding, population generation, fitness function, and crossover come together to form a ”basic” genetic algorithm to perform the clustering task. Algorithm 1 demonstrates this reasonably simple procedure. The algorithm shows the steps of generating an initial population, selecting parents based on their performances, and generating offspring based on the encoding of the parents as described above. A slight addition is the *patience* of the genetic algorithm.
The optimal score that can be achieved is not known for this problem, so the GA could continue searching forever as there is no end state. Instead, a stopping criteria is implemented. This is fairly trivial: if there has not been any improvement for the last t iterations, it seems unlikely

that there will be an improvement over more iterations.

Algorithm 1: Basic Genetic Algorithm for graph clustering

```
Input : Graph: $G$
Population size: $N = 200$
Parent rate: $p = 0.5$
Mutation rate: $m = 0.15$
Patience: $t = 20$
Result: Chromosome in locus-based adjacency

for $N$ do
    $Population \leftarrow Population \cup$ createRandomAdjacency($G$)
end
while $i < t$ do
    for $0 \rightarrow N \cdot p$ do
        $Candidates \leftarrow Population \cap Parents'$
        $Parents \leftarrow Parents \cup \text{argmax}_{c \in Candidates} Q_G(c)$
    end
    for $N \cdot p \rightarrow N$ do
        $p_a \leftarrow$ selectRandom($p \in Parents$)
        $p_b \leftarrow$ selectRandom($p \in Parents$) such that $p_b \neq p_a$
        $child \leftarrow$ uniformCrossover($p_a, p_b$)
        $child \leftarrow$ adjacencyBasedMutate($child$, $m$)
        $Children \leftarrow child \cup Children$
    end
    if $\exists c \in$ Children $\wedge \exists p \in$ Parents $\wedge Q_G(c) > Q_G(p)$ then
        $i \leftarrow 0$
    end
    $Population \leftarrow Children \cup Parents$
    $Parents \leftarrow \emptyset$
    $Children \leftarrow \emptyset$
end
```

The simple GA for clustering described in Algorithm 1 provides some solution for the task at hand. Unfortunately, this GA may take quite a few iterations to converge to an optimum. As an improvement to this version Said et al. (2018) proposes a Cluster-Coefficient based Genetic Algorithm in order to achieve faster convergence and better found optima.

2.3 CC-GA

The basic-GA provides a very naïve way of exploring the search space. Samples are constructed without any meaningful graph understanding. CC-GA[53] intends to improve upon that by incorporating the *clustering coefficient* of vertices in the graph into the sample generation process.

2.3.1 Clustering Coefficient

The clustering coefficient is a measurement from Social Network Analysis that describes the connectedness of a vertex [61]. It is formally defined as:

$$C_i = 2\frac{L_i}{K_i(K_i - 1)} \tag{2.2}$$

Where K_i is the number of neighbours that vertex i has, and L_i is the number of links between those neigbours. Note here that the clustering coefficient is not a measure that describes made clusters, it only describes the strength of local community structure on the original graph. This is a reasonably intuitive concept when thinking about social networks: suppose *Isabelle* has three friends *John*, *Kyle* and *Lei.* If *John*, *Kyle* and *Lei* are all friends between each other then $C_I = 1$. If *j, k & l* do not know each other then $C_I = 0$.
With this definition the clustering coefficient gives an indication of how clusters may be formed. If *John* finds that they're connect to *Isabelle* with $C_I = 1$, then *John* should likely be in a cluster with *Isabelle.*

2.3.2 Incorporating the Clustering Coefficient

Said et al. [53] applies this clustering coefficient to the initial population generation step in the Genetic Algorithm. Since the initial population can have a big impact on the behaviour of the population in later epochs they make some additional adaptations to fit their new initial population.
The initial population is determined by connecting each vertex to the neighbour with the higher clustering coefficient. If a vertex has multiple neighbours with the same clustering coefficients one of them is randomly chosen. This generates an initial population of near-identical samples that are all very good candidates. By connecting each vertex to its neighbour with the highest clustering coefficient the locus-based adjacency encoding will start with smaller good clusters. Said et al. [53] demonstrates that the CC-based initialization also identifies local bridges and disconnects clusters there. A local bridge is a vertex of which none of the neighbours are connected between themselves, i.e. they have a clustering coefficient of 0.

Adjusting for small Clusters

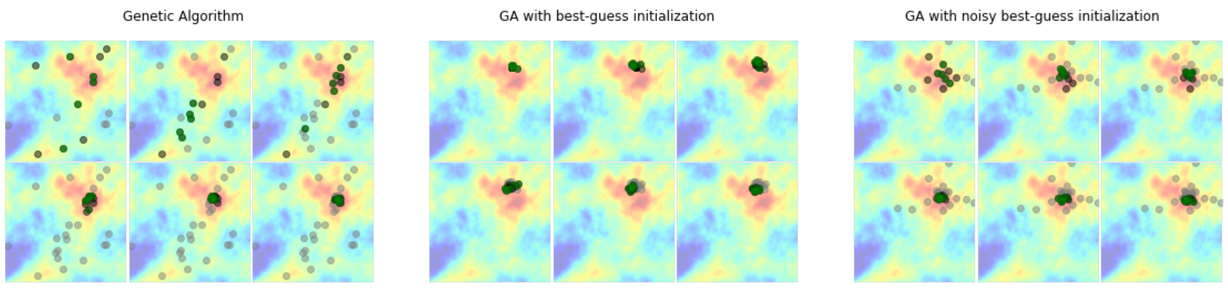
The initial population generated by CC-GA consists of many small clusters, often smaller than the resolution of modularity Q. In order to ensure that small clusters are merged into appropriately size clusters CC-GA implements an additional mutation step.
Next to the traditional mutation discussed in section 2.2.3, CC-GA also mutates the children by randomly connecting two clusters. This is done by randomly selecting a node $v_i \in C_x$ where C_x is a randomly selected cluster. From v_i's neighbours one node is randomly selected under the requirements that this not is not in the same cluster. In the case that all neighbours are in the same cluster, then another random vertex $v_j \in C_x$ is chosen for the same operation.

2.3.3 Additional Modularity Checks

An additional feature that Said et al. [53] implement in their CC-GA are intermittent evaluations of their offspring. After crossover, after traditional mutation and after the extended mutation the modularity of the chromosome is evaluated. Only when the change provides an improvement the improved chromosome is added to the population. This also means that members of the population are only removed by substitution with a better chromosome. Additionally, rather than looping over the parents until the population is refilled each parent is selected once per epoch to mate with a random other parent.

2.3.4 Weak Points

The current research intends to improve upon the CC-GA by identifying theoretical improvements to CC-GA, and testing these improvements on the Web-Graph.
For that purpose, this section describes potential issues that CC-GA may have, so that improvements can be suggested in section 2.4.
The first vulnerability is the very short field of view that the clustering coefficient considers. The clustering coefficients as used in CC-GA only considers first order neighbours for the scoring, but especially larger graphs may end up having their clusters significantly larger than that. A clustering coefficient that considers higher order neighbours is proposed in 2.4.1.
The second issue that the current research considers is the minimal spread of the initial population. Typically, Genetic Algorithms build their initial population from random variations across the search space. This is done to build a very rough impression of the whole performance landscape, to stochastically converge into one or more optima. The initialization procedure for CC-GA makes all samples in the population nearly identical to ensure that all samples in the initial population start at a high-performance strategy. The disadvantage of this may be that this puts the system at risk of reaching a limited-performance local optimum as it does not sufficiently explore the search space. It also means that the population first needs to diverge to start exploring the local search space, before it can converge to a local optimum. This comparison is visualized in figure 2.2, which shows 6 epochs with random initialization, a *best-guess* initialization which CC-GA has, and a best-guess initialization with injected noise to increase the spread. Figure 2.2 suggests that an improvement may be made by adding noise to the CC-GA initialization procedure. Section 2.4.2 proposes such a method.



(a) Typical GA initialization

(b) Best-guess initialization as CC-GA

(c) Best-guess initialization with injected noise

Figure 2.2: The figures show 3 different initialization methods and how they affect convergence behaviour over 6 epochs. Light gray points are samples that have been tried and rejected. Dark gray points are points are tried and among the top 50%. Green points are samples that are newly introduced and have not been evaluated. The typical GA starts over the whole search space and converges to an optimum. The best-guess initialization diverges from its initial position to converge to a nearby optimum. Best-guess initialization with injected noise starts with a spread around the high-performance part and converges to an optimum.

Third, the crossover procedure is evaluated. The uniform-crossover method used in CC-GA might not be the best fit for locus-based adjacency representation. Successful Genetic Algorithms for the TSP use crossover methods that preserve some meaningful parts of the encoded solution [35]. For the TSP such meaningful parts are sections of the path that an encoding makes. If a certain segment is already perfect, then the Genetic Algorithm is able to pass that perfect segment on and pair it with another perfect segment from a different chromosome to reach an even better offspring. The uniform-crossover does not ensure such idealized crossover. Instead, the uniform-crossover performs crossover at the smallest atomic level, rather than in some meaningful large sections of the solution. Section 2.4.3 proposes a crossover alternative that preserves larger meaningful sections.
Lastly, the mutation step also leaves some room for improvement. A uniform mutation probability ignores the fact that certain parts of the entire encoding are quantifiably better than others. This ties onto the previous issue, as they are both focused on acknowledging the fact that certain parts of the clustering may be "solved", while other parts may be hardly decent. Section 2.4.4 provides a mutation rate that is variant to the quality of a certain gene.

2.4 Improved CC-GA

Having identified several potential limitations of CC-GA the current section describes solutions to these limitations. These improvements can then be implemented as adaptations of CC-GA and evaluated on a set of Web-Graph samples in section 2.6.

2.4.1 Extended-CC-GA

The first proposed improvement is to use an extended clustering coefficient instead of the first-order clustering coefficient used in CC-GA. This addresses the issue of small-scoped clustering coefficients, particularly for larger clusters. The extended-CC determines the clustering coefficient as the link density between the neighbours of a node at any given depth. This introduces an additional hyper-parameter d to indicate the depth of the extended-CC. The following equation defines the extended-CC [1] for a vertex i at depth d:

$$C_i^d = \frac{|\{\,\{u,v\}; u,v \in N_i | d_{G(V_i)}(u,v) = d\,\}|}{\binom{|N_i|}{2}} \tag{2.3}$$

Note that the extended cluster coefficient, like the original clustering coefficient, describes the strength of community structure around a vertex on the original graph. It does not build on a generated cluster, but can be used as a heuristic to help build clusters.
It is trivial that the CC-GA can be considered as a special case of Extended-CC-GA with $d = 1$. The extended-CC-GA may pose an improvement with $d \geq 2$, though the performance of extended-CC-GA will not be expected to change after d exceeds the diameter [9] of the graph.

2.4.2 Stochastic Initialization CC-GA

The SI-CC-GA addresses the issue of initializing the population at a single point as visualized in figure 2.2. A good solution here would still be centered around the point that CC-GA initializes its population, but adds some amount of variance to the initial population.

CC-GA initializes the population by connecting each vertex to its neighbour with the highest Cluster Coefficient. Stochastic-Initialization CC-GA instead connects to a neighbour by a weighted probability determined by the Cluster Coefficient. By applying SoftMax [64] to the Clustering Coefficients vector a probability vector can be made and used for selecting a good but randomized neighbour.

2.4.3 Cluster Crossover CC-GA

Clucro CC-GA implements a crossover alternative to preserve sections of meaningful size. Whereas CC-GA applies crossover at the atomic level – taking each vertex randomly from either parent – clucro CC-GA applies crossover at the level of clusters.
While TSP crossover methods can use *partial matching crossover*[35] to select sections as a series of the sequence, the order of items in locus-based adjacency encoding is not meaningful. Instead, sections should be selected based on the meaningful sets found in the solution. This is done by selecting a cluster from a parent and copying all the vertices in the cluster to the child. The Cluster Crossover then alternates between parents to find a cluster of which none of the vertices are encoded in the child yet, so that cluster may be encoded in the child. When no more clusters meet this constraint the remaining vertices are encoded according to Uniform Crossover. This ensures that resulting encodings are still valid, and preserve larger parts of the solutions from parents.

2.4.4 Quality Based Mutation CC-GA

A uniform probability of mutation ignores a quantifiable knowledge that certain parts of a proposed clustering may be better than others. By determining which parts of the clustering are better and which parts are worse, it is possible to modify the mutation probability as a function of quality.
This quality can be assessed by a cluster's modularity contribution. This is given in [12] as:

$$q_c = \frac{m_i}{m} - \left(\frac{m_i}{m} + \frac{m_e}{2m}\right)^2 \tag{2.4}$$

Where m_i is the number of edges within the cluster, m_e is the number edges exiting the cluster and m is the total number of edges in the graph. This is different from the part of the sum of the modularity for the given cluster, it is in fact $\frac{m_e m_i}{2m^2}$ less. This difference follows from the varying cluster sizes that may bias them towards smaller or larger parts of the modularity.
The mutation probability for each vertex can be determined by its normalized modularity contribution. In order to still allow some mutation chance at the best clusters, while also maintaining a limited mutation chance at the worst clusters, the probability of mutation for each vertex is determined as:

$$(1 - q_c) * s + b \tag{2.5}$$

Where s is a spread factor that indicates how far the highest and lowest risks range (set at 0.2), and b is a base probability (set at 0.05). q_c here is the cluster contribution normalized to range $[0, 1]$. This gives the vertices in the worst performing cluster a mutation probability of 0.25, while the vertices in the best cluster have a mutation probability of 0.05. This results in an average mutation rate of 0.15, equal to that of CC-GA [53].

2.5 Stochastic Block Model Based Clustering

While the current research proposes suggestions to improve the available Genetic Algorithms for clustering, it is important to keep an eye on alternative solutions. Comparisons between various GA-based clustering algorithms are intuitive to make as they live in the same domain. However, the entire clustering task has also been addressed from a statistical inference perspective, rather than as an optimization task.
This perspective creates a model of cluster-like blocks with certain probabilities to link between and within them [31]. Different models can be proposed by putting different nodes together in different blocks, they are then evaluated on their entropy.
In this scope efficient Markov chain Monte Carlo sampling [45] has been applied to block model optimization. Here, changes to a previous block model are randomly proposed and accepted according to the Metropolis-Hastings algorithm [30], where the probability of accepting a change is given as a function of the entropy difference.
While MCMC based block model inference and GA based clustering are both valid approaches in their own right, by being defined in different domains they are not normally compared. In order to give some perspective to the GA solutions Peixoto's Python library Graph Tool [46] also provides a modularity oriented variation of the MCMC based block model inference. This allows the MCMC based approach to be compared against the GA approaches.

2.6 Experiments

With 6 Genetic Algorithms for clustering (classic, CC-GA, and 4 improvements), and an MCMC blockmodel inference method, the subsequent goal is to find the best! The problem at hand is the clustering of websites in a graph. Since this graph would actually consist of over 300 million vertices no Genetic Algorithm can reasonably be ran on the entire network. Instead, 6 sub-graph samples are collected from the internet with more reasonable sizes. Each sub-graph is collected by selecting a single node in the entire network and traversing all incoming links in Breadth First Search until 8192 vertices are collected. Since some vertices quickly have far more neighbours than that a limit is set that only 1024 neighbours are allowed for each vertex. Table 2.1 provides some metrics for an understanding of the graphs.

Root Hostname	Edges	Mutuality	Transitivity	Density	Global CC
genderlinks.org.za	8571	0.00281	0.00000	0.00013	0.00037
www.dataprovider.com	10752	0.03704	0.00041	0.00016	0.02090
www.gnatus.com.br	10899	0.06512	0.00088	0.00016	0.02570
www.thedogbakery.com	13713	0.17588	0.01201	0.00020	0.18140
zeelearn.com	12388	0.22952	0.01006	0.00018	0.16637
zest.net.au	10418	0.04514	0.00243	0.00016	0.00664

Table 2.1: Metrics of the 6 internet sub-graphs collected.

Some of these metrics should come with a slight clarification. The number of edges is an intuitive concept, but it should be noted that since 8192 vertices are collected, each graph will have at least 8192 edges. Mutuality is the ratio of directed edges that also have an oppositely directed edge, that is: $p[(y,x) \in E | (x,y) \in E]$. It is interesting to find the mutuality for certain sub-graphs is much larger (up to 100x) than that of others. This is partly due to the number of edges, but may also indicate that different parts of the internet behave differently. The transitivity indicates the chance that when a website is a second order neighbour of a website, that it is also a first order neighbour. That is: $p[(x,z) \in E|(x,y) \in E, (y,z) \in E]$. This seems to follow a trend that is similar to the mutuality. The density $(p((x,y) \in E|x \in V, y \in V))$ is fairly low for all nodes. This follows expectations, as websites tend to only link to a fairly limited number of websites, even though there are millions. The Global Clustering Coefficient [40] is defined as:

$$C = 3 \times \frac{\text{number of triangles}}{\text{number of connected triples}} \tag{2.6}$$

This is a measure of how clustered the graph is in a range $[0,1]$. While the graphs don't all have a high clustering coefficient, this does not mean that the graph cannot be clustered well. It only indicates the extend to which the graph is inherently clustered.
With the datasets and the algorithms defined an experiment can easily be performed. In order to ensure a fair measurement each algorithm is applied to each graph sample 10 times. The parameters are kept consistent with the parameters proposed for CC-GA[53]. The performances will be evaluated on the distribution of final modularities, but the computational time will also be considered.

Algorithm	Start site	Score	Std.dev(score)	CPU time	Std.dev(time)
CC-GA	www.dataprovider.com	0.8736	0.0009	9672	1904
GA	www.dataprovider.com	0.8737	0.0015	2896	1755
CC-GA	zeelearn.com	0.8684	0.0009	9695	1711
GA	zeelearn.com	0.8682	0.0019	3132	1872
CC-GA	genderlinks.org.za	**0.9060**	0.0003	11199	2493
GA	genderlinks.org.za	0.9049	0.0006	4037	1483
CC-GA	www.thedogbakery.com	0.7887	0.0004	12906	2107
GA	www.thedogbakery.com	0.7881	0.0007	4152	1945
CC-GA	zest.net.au	**0.7795**	0.0019	4205	2361
GA	zest.net.au	0.7756	0.0035	3226	1321
CC-GA	www.gnatus.com.br	**0.8343**	0.0012	7967	1750
GA	www.gnatus.com.br	0.8322	0.0017	3226	1321

Table 2.2: Modularities and CPU seconds for CC-GA compared to GA for 6 graph samples.

2.7 Results

Firstly, the performance from the CC-GA[53] is reproduced in table 2.2. This does indeed show a marginally better performance compared to standard Genetic Algorithms, however, it also shows that computational cost is about 3-4x as large. The effect generally persists throughout the different graph samples. In only 3 cases (genderlinks.org.za, p ¡ 0.0001, zest.net.au, p = 0.0062, gnatus.com.br, p = 0.0051) it can really be said that the CC-GA got a signicantly higher final modularity ($\alpha = 0.008$ after Bonferroni correction [62]). .

In order to assess the value of the proposed extended-CC variation to CC-GA various extended clustering coefficient depths are displayed in table 2.3. This shows a minimal difference in performance between the different depths. The variations do not seem to keep any consistent trend, and are minimal compared to the standard deviations, so they may be fully attributed to noise. However, the computational costs do follow a clear trend. The computational cost seems to steadily increase with the depth of the extended-CC. This effect holds generally across the different sub-graphs. It is worth noting here that ext1 is actually identical to the original CC-GA, as this also uses a CC-depth of 1.

Table 2.4 demonstrates the performances of the various proposed adaptations to CC-GA. While stochastic initialization performs on-par with CC-GA modularity wise, none of the proposed alternatives get consistently higher modularities. However, they do all converge consistently faster. Here, clustered crossover converges the fastest, but also has the lowest final modularities. From these variations, stochastic initialization seems to perform the best overall. It gives a respectable reduction in time cost compared to CC-GA, while keeping the modularity on-par.

To investigate the effect that combining the changes have, figure 2.3 compares the combination of SI, QD and clucro to their individual effects. The combination of SI and clucro is much faster than the individual speed improvements. However, they do reach a lower final modularity than CC-GA. As also shown in table 2.4, clucro gives a much worse final modularity while SI remains on-par with CC-GA. Their combination actually relieves part of the decreased performance that clucro has. Adding QD appears to possibly give a marginal improvement in performance, but

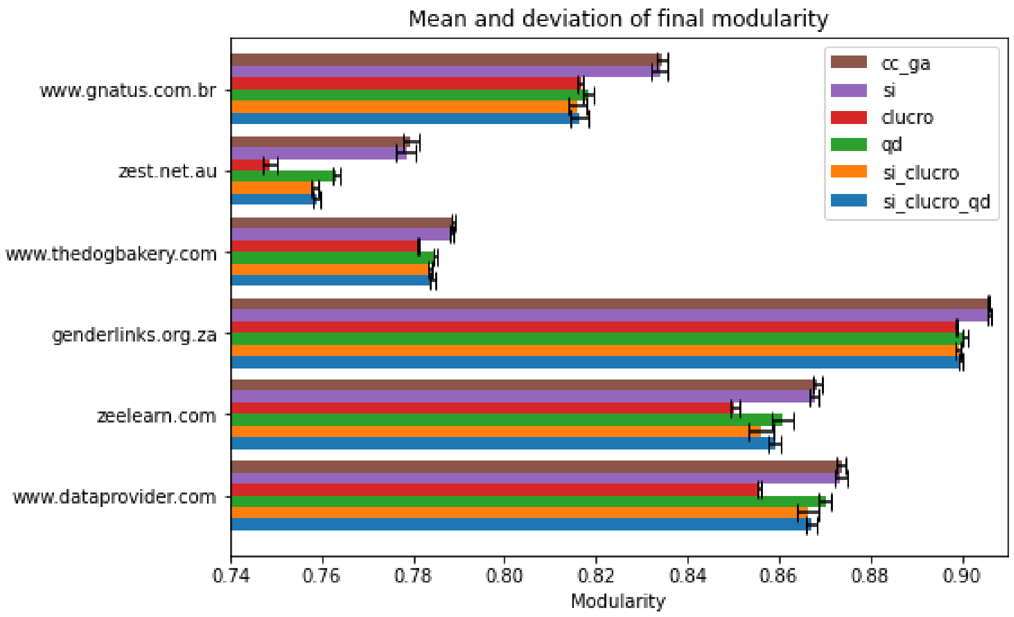
Algorithm	Start site	Score	Std.dev (score)	CPU time	Std.dev (time)
ext1	www.dataprovider.com	0.8736	0.0009	9672	1904
ext3	www.dataprovider.com	0.8742	0.0011	13135	1979
ext5	www.dataprovider.com	0.8744	0.0012	15334	3217
ext1	zeelearn.com	0.8684	0.0009	9695	1711
ext3	zeelearn.com	0.8682	0.0013	11626	2967
ext5	zeelearn.com	0.8685	0.0012	12433	3028
ext1	genderlinks.org.za	0.9060	0.0003	11199	2493
ext3	genderlinks.org.za	0.9060	0.0003	12977	2005
ext5	genderlinks.org.za	0.9060	0.0002	12440	1737
ext1	www.thedogbakery.com	0.7887	0.0004	12906	2107
ext3	www.thedogbakery.com	0.7888	0.0003	10752	2134
ext5	www.thedogbakery.com	0.7886	0.0004	14067	1946
ext1	zest.net.au	0.7795	0.0019	4205	2361
ext3	zest.net.au	0.7801	0.0011	13280	36367
ext5	zest.net.au	0.7809	0.0020	14084	2546
ext1	www.gnatus.com.br	0.8343	0.0012	7967	1750
ext3	www.gnatus.com.br	0.8334	0.0012	10405	2253
ext5	www.gnatus.com.br	0.8317	0.0032	12274	3842

Table 2.3: The performance of the extended-CC variations for different depth are given. ext i indicates an extended-CC depth of i. The final modularities in (a) have minimal differences between the extended-CC depths. On the other hand, the computational costs seems to increase steadily with the depth.

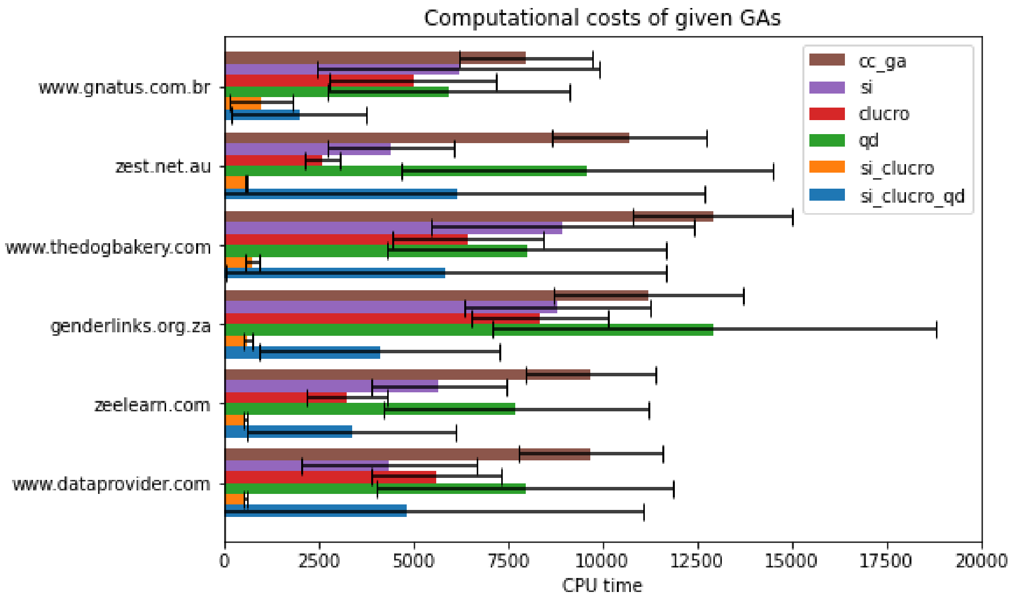
does so at a vastly increased computational cost.
While table 2.4 and figure 2.3 show that SI and clustered-crossover-SI give respectable improvements to previous GA-based clustering, figure 2.4 shows that the MCMC-based clustering outperforms them still. The MCMC-based clustering in fact gives an often much higher final modularity, while still taking only a fragment of the CPU time of the best GA-based solution.

Algorithm	Start site	Score	Std.dev (score)	CPU time	Std.dev (time)
CC-GA	www.dataprovider.com	0.8736	0.0009	9672	1904
SI	www.dataprovider.com	0.8735	0.0013	4367	2318
QD	www.dataprovider.com	0.8701	0.0014	7950	3918
Clucro	www.dataprovider.com	0.8556	0.0003	5598	1701
CC-GA	zeelearn.com	0.8684	0.0009	9695	1711
SI	zeelearn.com	0.8678	0.0011	5654	1781
QD	zeelearn.com	0.8608	0.0022	7699	3499
Clucro	zeelearn.com	0.8504	0.0010	3248	1050
CC-GA	genderlinks.org.za	0.9060	0.0003	11199	2493
SI	genderlinks.org.za	0.9062	0.0004	8796	2451
QD	genderlinks.org.za	0.9007	0.0005	12937	5841
Clucro	genderlinks.org.za	0.8989	0.0003	8328	1813
CC-GA	www.thedogbakery.com	0.7887	0.0004	12906	2107
SI	www.thedogbakery.com	0.7885	0.0004	8918	3471
QD	www.thedogbakery.com	0.7848	0.0006	8000	3672
Clucro	www.thedogbakery.com	0.7811	0.0016	6426	1984
CC-GA	zest.net.au	0.7795	0.0019	4205	2361
SI	zest.net.au	0.7785	0.0021	4400	1661
QD	zest.net.au	0.7631	0.0008	9587	4928
Clucro	zest.net.au	0.7487	0.0016	2611	459
CC-GA	www.gnatus.com.br	0.8343	0.0012	7967	1750
SI	www.gnatus.com.br	0.8340	0.0018	6189	3716
QD	www.gnatus.com.br	0.8182	0.0013	5928	3210
Clucro	www.gnatus.com.br	0.8166	0.0005	4988	2205

Table 2.4: he performance of different proposed CC-GA modifications. QD indicates the Quality Driven mutation variation, SI indicates the Stochastic Initialization variation, and clucro indicates the Clustered Crossover variation. Overall, SI performs on-par with the original CC-GA in terms of modularity, while QD performs slightly worse and clustered crossover performs worse still. However, the variations do offer time benefits. Clustered crossover is generally the fastest to converge, followed by SI, QD and CC-GA.

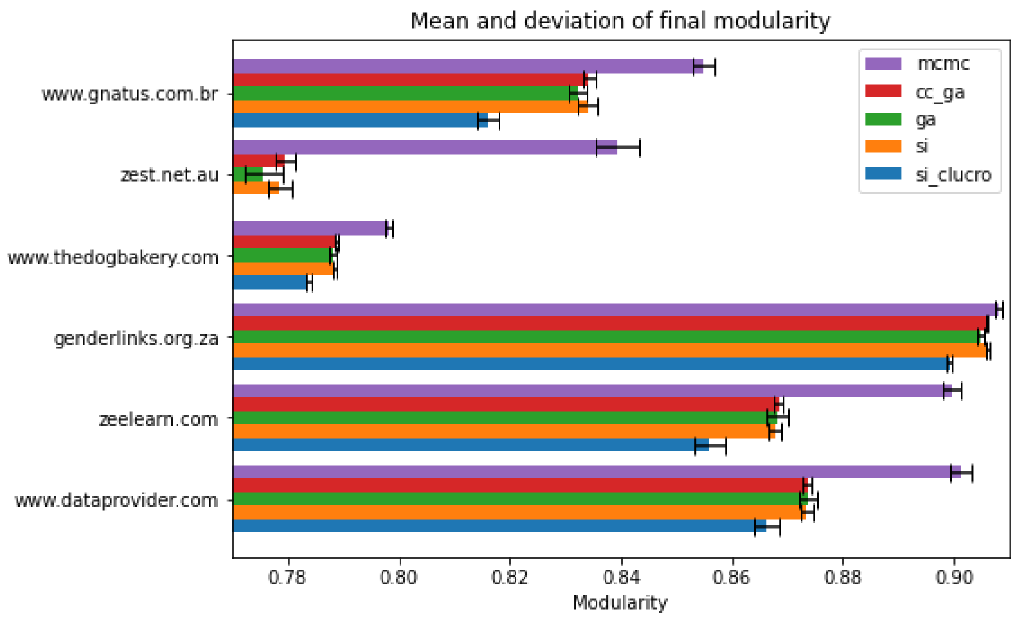


(a) Final modularities for different combinations of SI, QD and clucro compared to CC-GA.

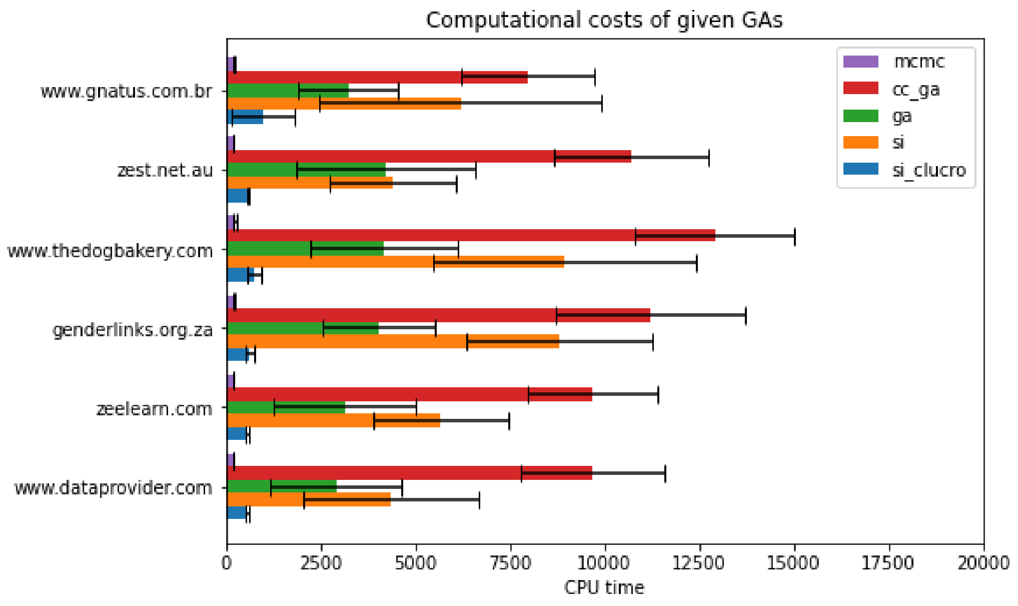


(b) CPU time different combinations of SI, QD and clucro compared to CC-GA.

Figure 2.3: The performance of different of SI, QD and clucro, compared to CC-GA. SI keeps a modularity on par with CC-GA, but the rest all perform worse. Clustered crossover gives the worst modularity, followed by its combination with SI and QD. In the computational costs (b) the combination of stochastic initialization combined with clustered crossover performs best. QD does not provide a reliable decrease in computational time.



(a) Final modularities for different best GA-based contenders compared to MCMC-based clustering.



(b) CPU time for different best GA-based contenders compared to MCMC-based clustering.

Figure 2.4: The performance of CC-GA and the suggested variations are compared to MCMC-based clustering. It is clear the stochastic initialization, or stochastic initialization combined with clustered crossover gives an appreciable improvement in overall performance compared to previous GA-based clustering solutions. However, the MCMC-based clustering algorithm finds a much better final modularity, while still taking less CPU time than the best GA-based solution.

2.8 Discussion

The results show that the Stochastic Initialization variation to CC-GA is better than CC-GA was. It therefore provides a new best Genetic Algorithm for modularity-driven graph clustering. This improvement is in the vastly decreased computational cost, without trading anything on the quality of the solution. This speed improvement can only be attributed to faster convergence, as the operations are otherwise nearly identical.
The clustered-crossover combined with Stochastic Initialization gives an even larger speed improvement, but at the cost of some quality of the solution. This makes the comparison to CC-GA a bit more difficult. The improvement in speed follows from the number of iterations that are required until convergence, but CC-GA might just be *slower* here as it continues to find better solutions where the new algorithm fails to provide these solutions.
It is as expected that the Stochastic Initialization gives this time improvement. By spreading around the initial population as demonstrated in figure 2.2 it takes fewer iterations to find the same solution that would also otherwise be found by CC-GA. Since the convergence behaviour remains the same, and since the starting population starts in the same *area* it should be expected that the same solutions will be found in fewer iterations. Therefore, it is as expected that the SI adaptation improves on CC-GA. Figure 2.5 shows how stochastic initialization is able to perform so much faster than CC-GA. By having more variation in the initial population, at least one of the members of the population will be really good. From this great start there is no-longer a lot of space for improvement.

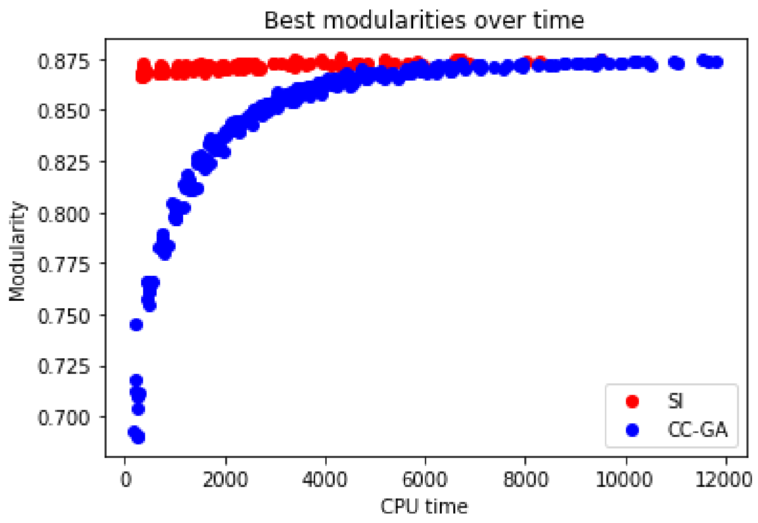


Figure 2.5: Modularities over time for the Dataprovider.com graph sample plotted for 10 different runs. SI starts with a much higher modularity, but both SI and CC-GA converge to roughly the same similarity. Over time, SI hardly improves from its starting position.

Unfortunately, the extended-CC adaptations did not provide any respectable improvement over CC-GA. The theory was that the extended-CC would give a "better" suggestion for an initial population, but this does not seem to hold up. Since the CC-GA does not really consider the actual value of the local CCs in the initial population step, but only considers their relative ranking, it is possible that the highest ranked first-order CCs are also the highest ranked higher-order CCs. This would be an explanation of why it does not provide any improvement on the scores. An alternative is that the initial population for CC-GA does not benefit from extra accuracy. The CC (extended or not) is only a proxy for what would make a good initial population, but the later convergence steps still need to explore away from that initial population. This believe matches well with the result for Stochastic Initialization, where a less accurate initial population does not persist in the final result. Naturally, the higher-order extended CC-GAs also come with more computational cost simply to compute the higher-order CCs. Figure 2.6 shows the effect that the higher-order CC has on the convergence behaviour. The lower order CC initialization actually appears to give a better starting position, though the higher order solutions converge to the same final quality. It could be the case that the lower order CC is better because the clusters form as an agglomeration of these local connections, so the higher-order CC is a worse advisor for local connections.

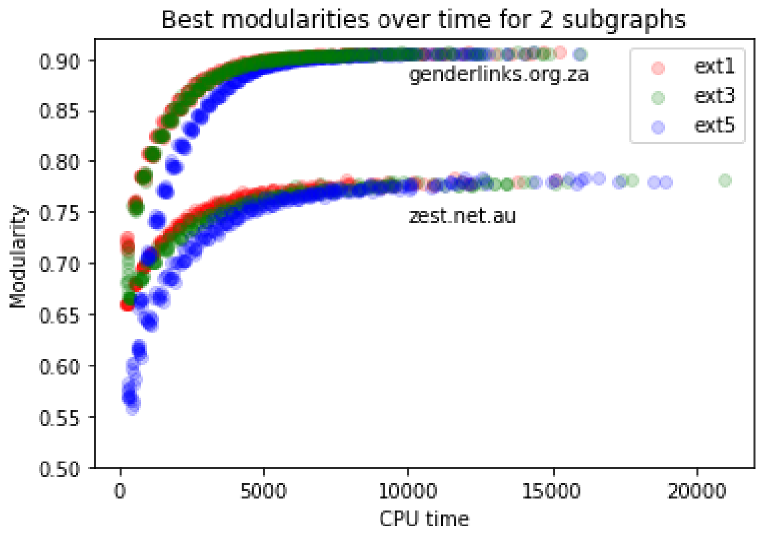


Figure 2.6: Modularities over time for 2 of the graph samples plotted for 10 different runs. The higher-order CCs have worse starting modularities, but they all converge to the same quality final solution.

To investigate the lack of improvement from quality-driven mutation and clustered crossover figure 2.7 presents their convergence behaviours compared to CC-GA. It appears that they do indeed converge faster, but that they converge to some lower-quality local optimum. To further investigate this in the future these alternatives may be run with some alternative parameters that ensure more spread. This convergence behaviour does however show that these adaptations may still be better than CC-GA, but that they need some parameter tuning to really show that effect. Specifically, both of these adaptation are well suited for a temperature-based version where over some number of epochs they *turn into* CC-GA. This would allow a future adaptation to reap the benefit of the fast convergence upfront, while still preserving the detailed convergence later.
For an explanation of the clucro-SI combination a quick look at figures 2.5 and 2.7 will suffice. The modularity at the start of SI is higher than what clucro is able to converge to. The result is that after initialization, no more improvements are found in the following epochs. This lets the genetic algorithm end very quickly, explaining the impressive speed, while retaining the good modularity from the stochastic initialization. In practice though, this means that clucro-SI is no better than just doing the initial population step for stochastic initialization and neglecting any other steps of the genetic algorithm.

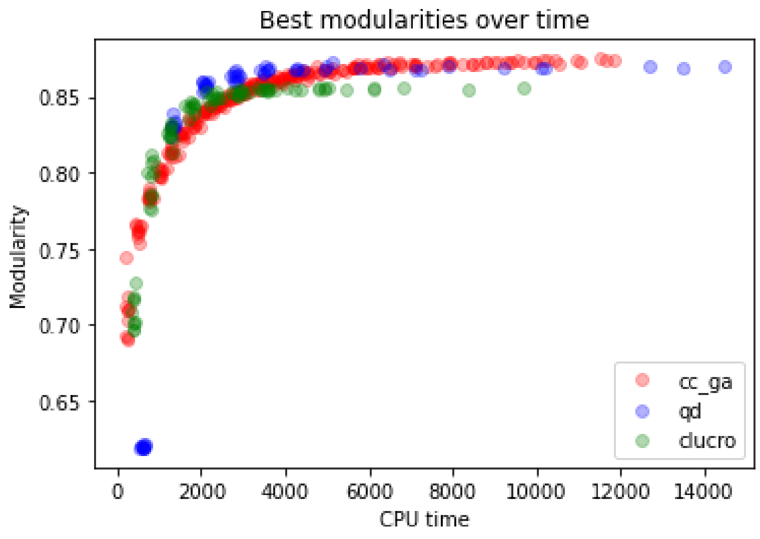


Figure 2.7: Modularities over time for the Dataprovider.com graph sample plotted for 10 runs of QD, clucro and CC-GA. QD and clucro converge faster, as indicated by the steep slope at the start, but they settle at lower final modularities than CC-GA.

The consistency at which the various GAs get similar final modularities might suggest that they may have reached a near-optimal solution. However, the far better MCMC-based clustering shows that this far from true. It appears that the searching behaviour of any of these GA solutions is not able to really find optimal solutions. The distinguishing factor that may be making the MCMC-based solutions better is the more directed *searching* for a better solution that entropy difference based Metropolis-Hastings sampling provides. This may allow it to effectively search in all dimensions where an improvement could exist, while the GA solutions depend on some low probability of sampling the correct changes without simultaneously sampling negative changes.

2.8.1 Future Research and Conclusion

A simple and clear lesson that can be drawn from this research is that Stochastic Initialization makes CC-GA better. Simultaneously, it could be concluded that the other solutions may provide some benefit over CC-GA if their parameters are refined.
The more potent lesson to be learned from this however, is the cost that comes with the parallel distinct paradigms that are used to address the same problem. In this case, a substantial body of research exists for graph-clustering by using genetic algorithms to maximize modularity. However, in complete disconnect from the previous, a similar body of research for graph clustering focuses itself on SBM optimization techniques to minimize MDL. Ultimately both of these research areas serve the same purpose: make good clusters from graphs. However, because they approach the task from different perspectives (problem-solving vs. statistical) they invent different solutions, but also different methods of evaluation. The current research puts these two niche sub-fields of parallel research side-by-side and finds that SBM optimization shows much more potential than the current clustering.
For future research, the current paper recommends to avoid genetic algorithms for modularity optimization. This is a heavy demand, as it suggests dropping an entire section of research. At the very least, it should be considered a good recommendation to evaluate solutions from the

Modularity optimization domain against solutions from the SBM domain. This brings some perspective to claims of new state-of-the-art genetic algorithms for modularity optimization. While new bests in a subfield have some value, they should be kept in perspective to the ultimate goal of finding the best method for clustering, not the best Genetic Algorithm for clustering.

This lesson applies generally for scientific research, expanding beyond the niche of graph-clustering. Many scientific fields depend on various paradigms, but the somewhat organic way that new research is produced may risk that certain topics are extensively researched within a certain paradigm while a better solution already exists in another paradigm. This shows that there is a lot of value to be gained by ensuring some collaboration between various scientific disciplines.

If there is a specific need on maximizing GA-based graph clustering, research should be directed at maximizing exploration in a valueable sense. Perhaps mutation rate can be increased for variations where a heuristic indicates a potential for improvement, borrowing the idea from the entropy improvement based metropolis-hastings sampling. As stochastic initialization showed some very respectable improvement to performance, further building on CC-GA should always use some variation of Stochastic Initialization.

It would also be interesting to further explore how much the variance on the starting population may affect to performance after convergence. While the publication for CC-GA [53] claims that the initial population is very important for determining the final result after convergence, figures 2.5 and 2.6 seems to indicate that the GAs converge to the same quality, regardless of the initial population. By applying various bases for the Softmax formula or by adding some noise to the CCs the initial population may be generated with a larger or smaller spread, which may or may not have a meaningful effect on the quality of the converged population.

Chapter 3

Trust Score Re-estimation with Web-Graph Clusters

Abstract To improve upon previously developed website Trust Score estimation the current research uses the Web-Graph to predict Trust Scores so that the error distribution of both systems may be combined to provide a theoretically better final estimate. Two methods of collecting websites as features from the Web-Graph are explored: one cluster-based and the other BFS-based. While both perform better than random guessing, there does not seem to be a significant difference between them. Additional, it is shown that the size of the cluster does not show any correlation to the accuracy of the estimator. Future research may verify the theoretically better final Trust Score by collecting a new annotated dataset.

3.1 Introduction

Previous work by Mostard et al. [39] developed a classifier for identifying fraudulent e-commerce websites. Based on these findings, a more general classifier was developed to offer a trust indication for any website. This trust is determined as a Random Forest's class probability of a website being fraudulent. Such a Trust Score can allow individual shoppers to be cautious when visiting a website, but it may also be used by governmental organizations or DNS providers to actively combat untrustworthy websites.

The work by Mostard et al. [39] is exclusively but extensively directed at the content of the individual website. Since the malicious web-developers behind fraudulent websites have control over the content and properties of their site, they have the ability to adapt their websites in such a way that these classifiers fail to detect them. For example, Mostard et al. [39] found that the lack of an SSL certificate can indicate an untrustworthy website. Since desktop browsers have generally adopted security warnings when SSL certificates are missing [5] malicious actors have been incentivized to get such certificates. In this style, anything that is within a website's control to be adapted, may be adapted to benefit the website owner.

To overcome this, the estimation of Trust Scores may be augmented with features outside the control of the website. The current research uses the Web-Graph [33] to further improve the Trust Score as a reflection of the trustworthiness of websites. This is done by collecting other websites that are *near* a website in question, and training a model to predict the Trust Score of a website based on nearby websites.

3.1.1 Joining Conditional Estimators

Both a Trust Score based on the website itself, as well as the Trust Score estimated based on other websites are valid conditional estimators that can give an indication to the underlying actual trustworthiness of a given website.
While estimators are typically used for their maximum likelihood prediction of the true value, they can also be used to estimate a probability distribution. By considering the Trust Score that is estimated and the distribution of errors that the estimator has shown, a probability distribution can be constructed for the true trustworthiness with the mean as the estimated Trust Score.
With two different estimators like this, one should find two distributions of where the true trustworthiness of a website should actually lie. Assuming that both of these distributions have normal errors [37], and making the critical assumption that both errors are i.i.d., the variance of product of the two distributions [42] is:

$$\sigma_C = \sqrt{\frac{\sigma_M^2(\sigma_M^2 + \sigma_N^2)}{2\sigma_M^2 + \sigma_N^2}} \tag{3.1}$$

Where σ_M is the variance of the Trust Score estimator from Mostard et al. [39] and σ_N is the newly developed estimator based on the Web-Graph. The effect of variances σ_M and σ_N on the resulting variance σ_C is visualized in figure 3.1. This shows that the final error will decrease with σ_N. At the best possible new classifier with $\sigma_N = 0$, we will find that $\sigma_C = \frac{\sigma_M}{\sqrt{2}}$. This shows a particular imbalance in the final result between the two estimator. Such an imbalance is caused by the fact that the Web-Graph estimator doesn't estimate the true trustworthiness, but instead estimates the Trust Score from Mostard et al. [39], therefore, there is a limit to σ_C determined by σ_M.
Nonetheless, it can be concluded that the best joined estimator can be made by trying to minimize the errors that the new estimator makes.

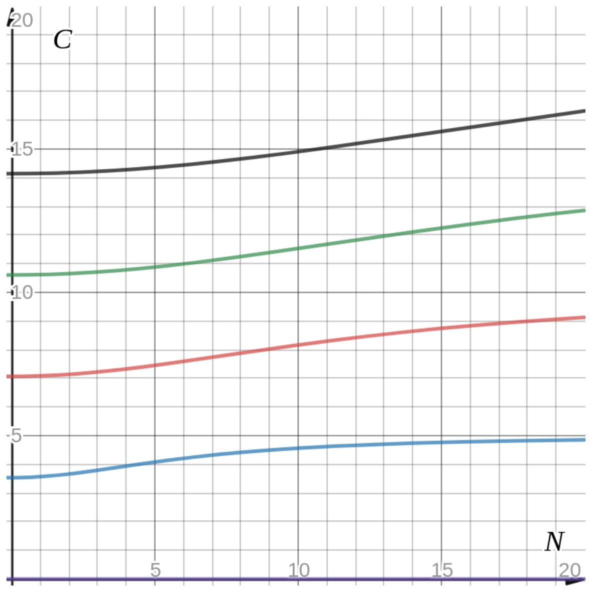


Figure 3.1: Combined variance C as determined by variance N of the newly developed estimator N for 5 values $[0, 5, 10, 15, 20]$ for the variance M of the original estimator.

This joined estimator concept leads this research to attempt to develop an estimator that attempts to predict the Trust Scores for websites to be as close to that of Mostard et al. [39] as possible. It would then be possible to combine this newly developed model with the original model. The resulting prediction can then be determined with:

$$\mu_C = \frac{\sigma_M^2(\mu_N + \mu_M) + \mu_M \sigma_N^2}{2\sigma_M^2 + \sigma_N^2} \tag{3.2}$$

Here μ_C is the predicted Trust Score from the combined models. μ_M and σ_M are the predicted Trust Scores and the variance of the error distribution from Mostard et al. [39]. μ_N and σ_N are then the predictions from the Web-Graph based model and the variance of its error distribution. From here on, the current research will explore how it can construct an optimal model N that creates a μ_N with minimal error σ_N. With μ_M and μ_N it is then possible to calculate μ_C, which would theoretically be better than the components it is made from. Unfortunately, the current research will not be able to evaluate this since a ground truth dataset is missing.

3.1.2 Selecting Sites from the Web-Graph

An interesting part of the Web-Graph based prediction of Trust Scores is the selection of which websites are considered to determine the Trust Score of a given specific website. Some sites should be more relevant to determine the score of another website than others. By selecting websites that are more relevant to a given website as input for a new regression model, its accuracy may be improved.
The simple method for this is collecting some number of websites with Breadth-First Search [6]. This can provide any requested number of websites as input for the regression model. Unfortunately, this takes very little consideration into the actual structure of the Web-Graph. For example, websites that have a lot of connections will be used as input for many sites, despite not being specifically relevant for all of them. Groups of websites that form a community may give a stronger reflection of a websites trustworthiness, but this community is not actively considered by Breadth-First Search.
An alternative and more elaborate method is to first apply community detection to the Web-Graph. This assigns every website to a community with relatively densely connected websites. The websites that are in the same community (or cluster) can then be used as the the input for predicting the Trust Score. This alternative overcomes the issues for Breadth-First Search, w.r.t. highly connected websites and insight from the graph structure. Additionally, it eliminates the parameter for deciding the number of relevant websites, as it attempts to infer this from the graph structure.

3.1.3 Research Goal

Having defined that predicting the Trust Score as accurately as possible has a purpose to improve a joined conditional estimator in section 3.1.1, the goal of the current research is to construct an accurate regression model using the Web-Graph. As section 3.1.2 proposes two alternative methods for finding inputs to such a model, this research attempts to answer whether *nested-SBM cluster* based or BFS based website selection results in a better Trust Score re-estimation model.
It is expected that cluster-based website selection outperforms BFS based website selection, as it considers more information from the graph structure. To verify that both inputs are indeed

relevant, the current research will compare both regression models to one without any websites as input. It is expected that the difference between the models with cluster based and BFS based website selection is fairly small compared to their difference with the no-input model.

3.2 Method

The methodology for this research concerns three major components. Firstly, the dataset is discussed. This consists of selecting the websites for training and testing data, but also explains the Trust Scores as generated by Mostard et al. [39]. Second, and most important, is the collecting of websites from the Web-Graph for the BFS and cluster-based regression models. Lastly, the design of the regression model is discussed.

3.2.1 Dataset

The dataset for the current research is collected from the Dataprovider.com database. 2000 random websites are selected on the condition that they have a response code 200 to indicate the website is functioning normally. Additionally, each website should have at least 1 incoming link, to ensure that the Web-Graph can actually be applied.
Since the website-relevant features are already used to generate the Trust Scores, no features are collected from the dataset, other than the dependent variable Trust Score.

Trust Score

The Trust Score is the dependent variable to be predicted by the models in the current research. For an understanding of these generated Trust Scores the simplified model from Mostard et al. [39] is discussed.
Mostard et al. [39] accumulated a dataset of 5115 websites, of which 2022 were identified as untrustworthy by consumer association *Consumentenbond* in 2018. Another 1791 websites were identified as trustworthy since they were registered with e-commerce association *Thuiswaarborg*. The remaining websites were collected from manual annotation by professional annotators from Dataprovider. It should be noted that this dataset is unbalanced, with 82.2% of websites being trustworthy.
On this dataset a Random Forest with 500 trees is trained using 18 features collected by Dataprovider. These features include for example whether a website has an SSL certificate, the number of outgoing links a website has, or whether they have a phone number listed on their website. The resulting classifier has a precision of 71% and a recall of 80%.
The Trust Scores are then determined by the class probability as determined by the Random Forest. A Trust Score of 100 is 100% certainty of being trustworthy, while a Trust Score of 0% is no chance of being trustworthy.
While identifying whether a website is trustworthy or not is at its core a classification task, the current research instead chooses to look at it as a regression task of class probability. This stays close to what a Trust Score may be used for, which is quantifying the risk that a website brings. Since it is impossible to create a classifier with 100% accuracy, there is always some information loss when reducing the output from a model to a binary classification. If a classification is necessary, the Trust Score can be set against a threshold that is set to satisfy the necessary precision or recall for a specific usage.

3.2.2 Website Collecting

As described in section 3.1.2, there is a potential to be gained in selecting which websites from the Web-Graph are used to predict a Trust Score. The current section describes the two methods that this research compares against each other for the purpose of optimizing Trust Score predictions.

Community Detection

The community detection based method firstly concerns a concept of a community. Using the Stochastic Blockmodel [31] framework, a community or cluster is a block of websites with some probability to have links within another block. With this idea, one can determine the likelihood of the Web-Graph based on the statistical model of these blocks. The model for which the current Web-Graph has the best likelihood is then the best model. However, using this definition the best model is one where each individual website is an individual block. Instead Minimum Description Length [27] is optimized, which gives a certain cost to having too many blocks. This is still flawed as it has a resolution limit where the average cluster size is largely determined by the number of vertices in the whole graph [44]. This is resolved by applying a Nested Stochastic Blockmodel, where each block is re-considered as a whole graph on its own, which then gets clustered again.

An effective method for optimizing such a Nested Stochastic Blockmodel is described by Peixoto [45], and implemented and provided by Peixoto [47]. This method uses MCMC sampling to sample changes to a SBM, which convergences to a good final state.

Since the whole Web-Graph has vastly expanded beyond the 50 million sites it contained in 2006 [6], applying the clustering to the whole graph is not feasible. Instead a graph sample may be collected using BFS, where as many websites as possible are used as starting nodes [6]. For this, the 2000 websites from the dataset are used, from which the graph sample is expanded to 109,210 vertices. This graph is then clustered into different communities using MCMC. Because the MCMC clustering does not specifically enforce the condition that disconnected components of the graph should also be disconnected clusters, this condition was manually applied. This is because disconnected components have no graph-structure relation to other disconnected components, which means there is no grounds for them to be in the same cluster.

The result is a staggering 102781 clusters, which means that only a very limited number of no more than 6 thousands websites are placed in a cluster with at least 1 other site.

Breadth First Search

A simpler method of collecting websites for Trust Score estimation is through Breadth First Search. Rather than collecting websites by forming clusters one could simply use Breadth First Search to collect any number of websites that are nearby in the graph structure. This gives a hyperparameter to determine how many websites should be collected to estimate a Trust Score.

In order to ensure a fair comparison between BFS and Community based website collection the current experiment determines the number of websites to be collected with BFS as the number of websites that were found through the community based method. This makes sure that neither method can outperform the other due to having more or fewer websites to make an estimate with.

3.2.3 Random Forest Regression

Since Random Forests [14] have been shown to be rather robust for a wide range of classification and regression tasks [10] they will also be used to perform the regression for Trust Scores. Using the mean and standard deviation of the Trust Scores of the collected websites as well as the number of websites collected as features, the Random Forest will estimate a Trust Score. This estimated Trust Score would then be μ_N as described in equation 3.2
For the Random Forest 500 decision trees are used. This hyperparameter does not need to be thoroughly tuned, as Random Forests have been shown to give good results regardless of the number of trees provided that there are plenty [50].

3.2.4 Evaluation

The models will be trained and evaluated only on sites with 1 or more other sites in their cluster, to ensure that only relevant websites are considered.
To ensure reliable results 10-fold cross validation is used to train and test both models. Additionally a baseline Random Forest without any features is also trained and tested across these 10 sections. This baseline without any features will fit to the mean of the distribution, which will have a smaller error than a random guessing baseline. The models will be assessed on their average MAE across the 10 folds, to align with the intuition of how many points the models are off. The variance across these folds will be considered the consistency of the different models.
Since the same folds are used for each model, the significance of the differences is determined with a one-way repeated measures ANOVA. Due to the limited number of folds it should be considered that small effect sizes may not show as significant.

3.3 Results

From the 2000 websites to start collecting the graph structure, only 378 were placed in clusters with at least 1 other site. Figure 3.2 shows that the size of the clusters are usually very small, with far fewer clusters being up to a couple hundred websites in size.
The results from the 3 models in the 10 fold cross validation are shown in table 3.1. This shows the lowest error rate for the BFS model, followed by the Cluster model. Lastly, the model without any input has the highest error. A one-way repeated measures ANOVA shows that the difference between the model's performances is significant ($F(2, 18) = 11.931, p < 0.001$). Ad-hoc paired t-tests between the models with Bonferroni correction [11] for multiple tests showed no significant difference between the BFS and cluster models ($T(9) = 2.16, p > 0.1$), but it does show that the baseline model performs significantly worse than the other models ($T(9) = 3.48, p = 0.02$ and $T(9) = 3.76, p = 0.01$).
As a process measure (and as a brief side-track) figure 3.3 investigates the error of a BFS-based model for different cluster sizes. For this, a model was trained on 80% of the data, and all samples of the data are shown. Since there are many more small clusters than large clusters, the scatter plot suggests that there may be larger errors for smaller clusters, but this is only because there is a higher chance to draw some exceptionally high values from the error distribution when there simply are more values. To give a more accurate representation of the average error a LOWESS [18] curve was fitted (red), which shows that the local average error is actually consistent throughout the cluster size. One outlier (indicated in red) was removed here.

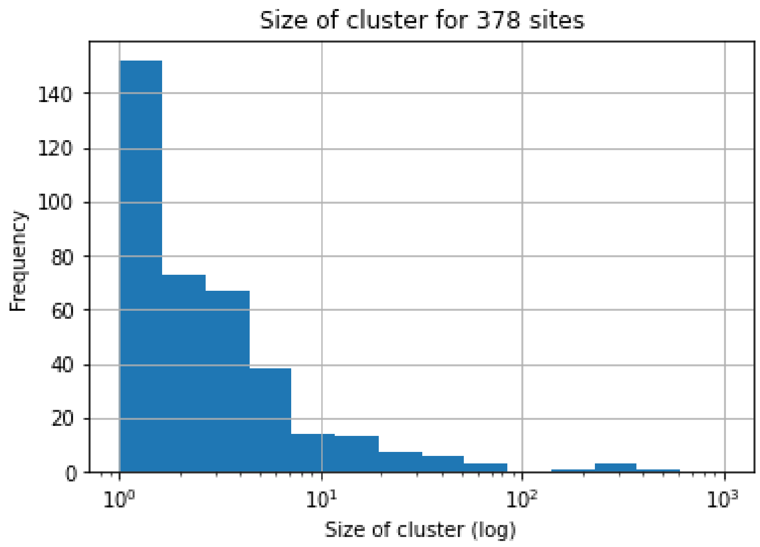


Figure 3.2: Size of clusters for the websites in the dataset after filtering out all the clusters with only 1 site. The log-x histogram shows that most clusters still have less than 10 sites, but that some even have several hundreds.

Website collection	MAE	Standard deviation
BFS	10.70	0.38
Cluster	11.25	0.41
None	**12.30**	0.41

Table 3.1: MAEs for the different models across the 10 folds. BFS and Cluster both get a significantly lower error than the model without input. No statistically significant difference is found between the BFS and Cluster features.

3.4 Discussion

The results from the 10 fold cross validation show that the method of predicting Trust Scores based on other websites is indeed functional, as it is significantly better than random guessing. In contrast to the hypothesis, the cluster-based website collection does not appear to be any better than the BFS based website collection.

This unexpected result may be attributable to the size of clusters. The Nested SBM MCMC may be making clusters that are too small, but this is unlikely considering that figure 3.3 suggest that this does not affect accuracy. To formally answer the research question, it does not appear that either BFS based or cluster based website selection is any better than the other, though both are better than a no-input model.

Another problem that may have negatively impacted the performance of the cluster-based method is that the taken graph sample may not be large enough. Larger graph samples come with additional computational cost, particularly in the clustering part, but they may result in clusters that better reflect the actual graph structure.

The original purpose of making a model that predicts Trust Scores based on collected websites was to allow for further improvements to the Trust Score. While figure 3.1 suggests that this theoretically works, future research may verify this with a new annotated dataset. Nonetheless,

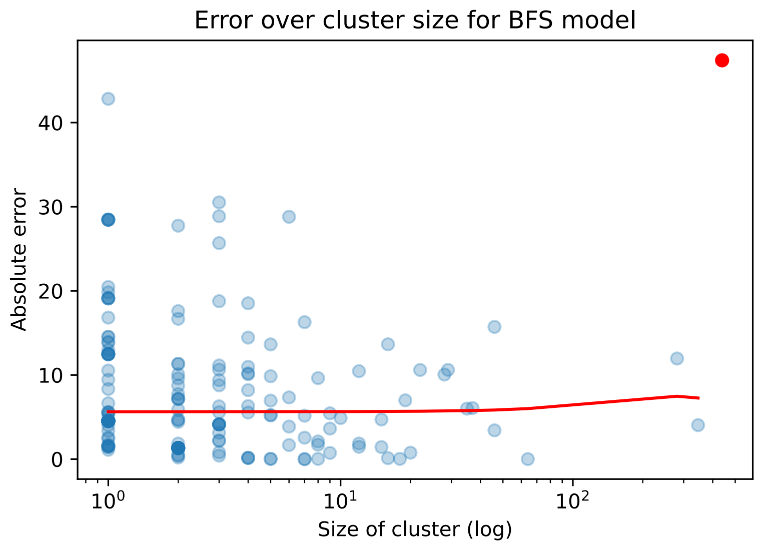


Figure 3.3: Cluster size against regression error for BFS model. A LOWESS [18] curve is given to demonstrate that the mean error remains consistent over cluster size. An outlier (indicated in red) is not taking into consideration for the LOWESS curve.

we do now have a σ_N and σ_M, which means that we can indeed calculate how much the combined model would theoretically be better. Considering equation 3.1 and taking $\sigma_N \approx 11$ and $\sigma_M = 7.07$ (measured on Mostard et al. [39] original dataset), we can calculate that $\sigma_C \approx 6.16$. Giving the combined model an error reduction of 12.8%. This does however rely on the errors being i.i.d.. If they are partially dependent, the error reduction will be less.
With an eye on the value of Web-Graph clusters it appears that the Web-Graph clusters were not shown to give additional value over the Web-Graph an-sich for this application. Other classification or regression tasks may find more benefit in the Web-Graph clusters. It may also be the case that Web-Graph clusters may be used for Trust Score estimation using some different feature extraction method. It may be interesting to explore cluster-index based approaches for this, or methods that preserve additional insight about the graph structure for the model.

Chapter 4

Fake News Discovery through the Web-Graph

Abstract As social media platforms have been cracking down on Fake News, misinformation providers and consumers are forced to migrate to dedicated Fake News sites. While research for Fake News classification on Social Media is interesting and extensive, research on Fake News Site classification has been lacking. The current research uses the Web-Graph to find candidate Fake News sites, which are then classified based on extracted keywords or cluster indices. These cluster indices are generated by applying statistical inference methods for graph-community detection. The results show that the cluster-based Random Forest slightly outperforms the keywords-based one (AUC .93, AUC .91). Unfortunately, neither classifier generalized beyond the constructed training and testing dataset. However, this failure to generalize is not believed to be a problem with the proposed method, but rather a problem with the constructed dataset.

4.1 Introduction

Previous research on Fake News mostly considered it as a symptom of Social Media [57]. This has been a very good strategy throughout the last decade, as more and more people get news content served on social media. Between 2013 and 2016 the percentage of people who reported getting news from Facebook increased from 47% to 66% [26]. It is reasonable to assume that social-media news has had more fake-news than classical media, as the barrier of entry is so much lower. In that time, anyone could reasonably produce, post and distribute news-content independently on their social media accounts.
2021 started with a shift in tides on social media freedom with the ban of the US president Donald J. Trump [17]. Mr. Trump's account on Twitter, Facebook, Instagram, YouTube, Snapchat, Twitch and even Shopify were suspended or removed. The same week, Twitter banned 70,000 accounts it believed to propagate misinformation related to the QAnon movement.
QAnon refers to a wide conspiracy theory (or perhaps more of a theoretical framework) with a large alt-right following [29]. The harm in this not only lives in the onset of an "Information Dark Age", the QAnon-following has actually been mobilized to storm the US Capitol and attempt to disrupt the 2020 US Presidential Election [4].
The QAnon ban presents a cat-and-mouse game of alt-right conspiracy theorists and large tech companies. Parler, a free-speech focused social media platform attempted to provide a safe haven to QAnon followers, but after having rapidly acquired 4 million active users it got taken down by joint action from AWS, Google's Play Store and Apple's App Store for violent content

and lack of moderation. With Parler removed, alt-right users found their refuge in alternatives such as Gab, which got subsequently removed from the Google Play Store.
This rapid demonstration of power from large tech platforms, while questionable for the future of democracy, shows that fake-news may not have a long future on social media platforms. Instead, fake news sources may need to have their own (smaller) platforms, possibly in the form of independent websites.
Independent fake-news websites are often the backbone behind the fake-news that is propagated on social media. Fake-news presented on social media usually consists of a headline with a link to the news website where the content is actually hosted. Since social media may be taken out of this fake-news propagation equation, new fake-news classification and discovery solutions should not rely on social media context.
The current research demonstrates and evaluates a fake-news site discovery and classification system that foregoes the dependency on social media. Instead, it uses techniques known from fake-news content detection in social media and re-applies them to fake-news website detection in the Web-Graph.

4.1.1 The Web-Graph

The Web-Graph is the graph-structure perspective on the websites and hyperlinks between them [33]. This works similar to the social networks that social media has, which makes it an excellent candidate for fake-news detection.
The current size of the Web-Graph is impossible to know exactly. Determining the size of the entire Web-Graph is expensive, and the structure as a whole is rather volatile. For some indication of the size: all the way back in 2006 over 50 million vertices already existed [6]. The naïve strategy of taking every single site through a classifier is excessively expensive. Instead, the current research uses the graph structure as a method for discovering fake-news candidates.

4.1.2 Feature Potency

Research on fake-news identification in social media has defined, among others, content and network features as valuable for identifying fake-news. It is clear that content features will translate to website-based fake-news identification just as well.
The translation of network based features however is more questionable. Friend networks on social media are a lot more dense, and there is a lot of information that can be used as network features, such as co-occurrences, stances and diffusion networks [57]. Comparatively, the Web-Graph as a network feature may be much more sparse in information.
To extract valuable features from the Web-Graph for fake-news identification a community detection algorithm is applied. Community detection here is a clustering method for graph structures. These communities that websites may be in may be an effective feature for fake-news classification.

4.1.3 Hypothesis

The current research intends to investigate the possible value of network features for fake-news classification. With the knowledge that this is effective for social media based fake-news detection, and with the intuition that fake-news sources are less likely to receive links from

legitimate sources, it is hypothesized that Web-Graph clusters are indeed a good feature for fake-news classification.
The clusters are particularly good candidates as they support an intuition that fake-news persists in *echo-chambers*, and is mostly connected to more fake-news. This would allow the clustering to make clear *fake-news* clusters.
To provide some perspective to the value of the cluster features, they are compared to Haarman-keyphrases[28].
The Haarman-keyphrases should provide a simple but reliable content-based feature. It is hypothesized that Haarman-keyphrases will be more effective features than the clusters, as they are a more direct reflection of the possibly-fake content. However, since the Haarman-keyphrases and the cluster features reflect different indications for fake-news it is also believed that they may be combined to make an even better classifier.

4.2 Method

The methodology will be explained according to the pipeline developed. Naturally, this starts with acquiring a dataset. Subsequently, a subset from the Web-Graph is collected. This collection is then clustered and used to train a classifier. The classifier then labels some newly collected websites, so that the classifier may be evaluated according to new human-annotation. The entire pipeline is visualized in figure 4.1, and discussed step-by-step in the following sections.

4.2.1 Dataset

Some impressive datasets exist for fake news detection. Unfortunately, none of them are ready-to-use for classifying fake-news sites. The BuzzFeedNews annotated dataset [49] is oriented to individual articles in a social media context. The LIAR dataset [60] is oriented to individual statements.
A more potent candidate for this specific task is the FakeNewsCorpus that powers BS Detector [59]. The advantage here is that it gives specific fake-news websites. However, the problem is that the dataset is labeled by a Deep Learning model, rather than a human. This may have a larger risk of faulty samples. Moreover, this lacks any negative samples. The 430 websites from FakeNewsCorpus can be compensated by supplementing them with other sources of fake and real news.
356 additional fake-news sites are made available by NewsGuard [58]. These websites specifically concern covid-19 related fake news, but because they are expert-labeled they are more likely to be consistently accurate than the FakeNewsCorpus.
Getting non-fake news sites for comparison is a tricky task. The negatives (non-fake news sites) should only differ from the positives (fake news sites) in their truthfulness. Any other differences may cause a classifier to learn these biases, rather than the actual fake-news identifiers. The difficulty here largely depends on the features that a classifier is given. For example, if the positives are collected later than the negatives, a feature such as the WhoIs registration date may be a very simple and reliable way for the classifier to distinguish the positives from the negatives. However, the classifier will fail miserably when applied to new data. Being conscious of the biases that are present in the features, can offer an opportunity that a classifier generalizes well.

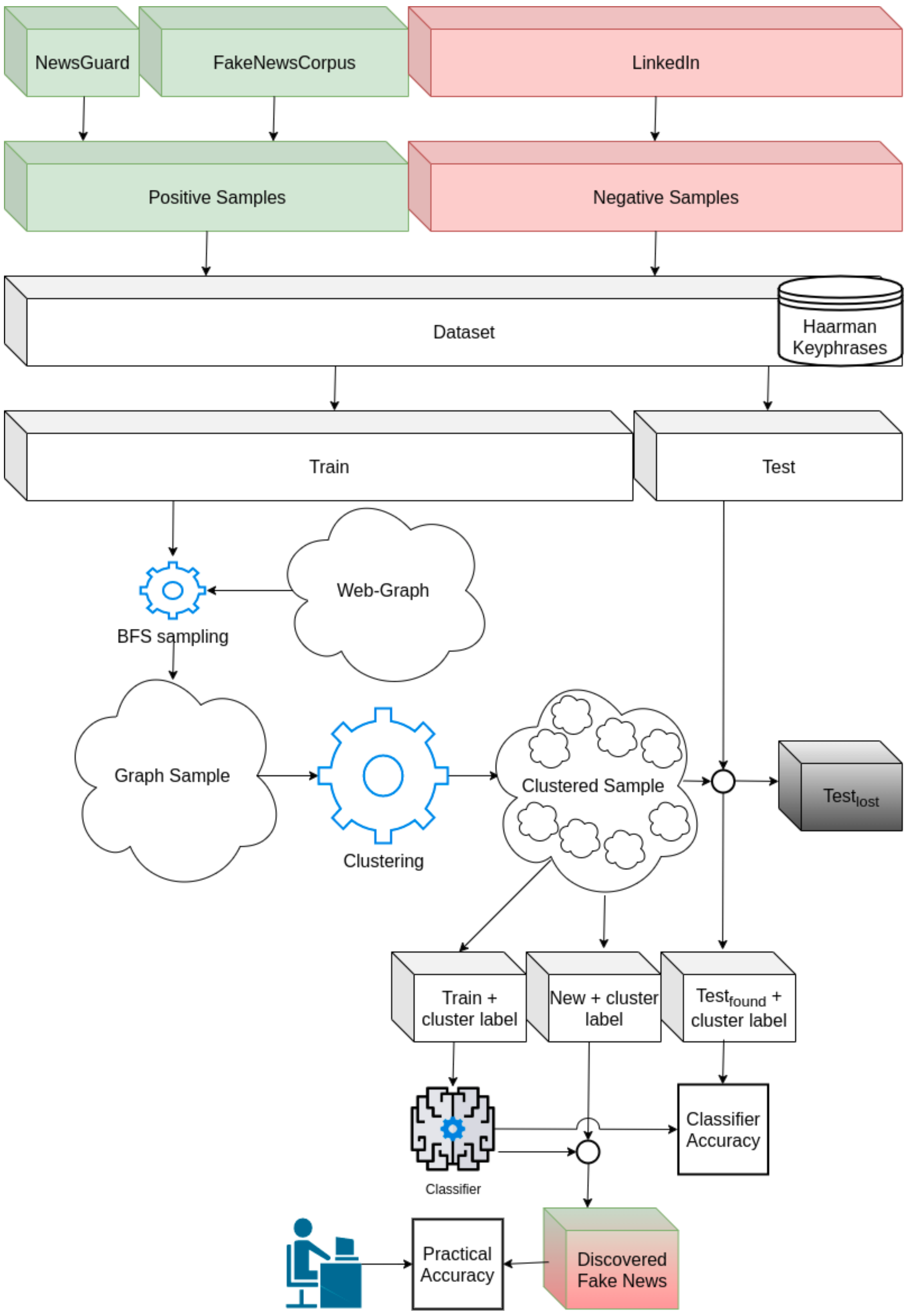


Figure 4.1: Full pipeline of the methodology. Starting at the construction of the dataset (section 4.2.1), following into the BFS graph sample (section 4.2.2). This is then used for the cluster (section 4.2.3), proceeding to classifier construction (section 4.2.5) and ending in evaluation (section 4.2.6). The details about the Haarman-Keyphrase features (section 4.2.4) are omitted.

The current research constructs a set of negatives as websites with a coupled LinkedIn page that is labeled as "Newspapers". This leaves a strong bias in the dataset. The LinkedIn negatives can be from all over the world, while the positives are mostly English written sites. To alleviate this, the LinkedIn negatives were filtered to only allow US-based sites. Of course this will still remain some bias, but this is believed to be a lot less detectable in the keywords or Web-Graph. To ensure some balance 997 sites from this set were randomly sampled.

Any bias that remains here will negatively affect the model generalizability, but this will be tested for with human annotation as described in section 4.2.6.

Naturally, the dataset is split into training and testing. Specifically, a randomly selected 70% of the samples are labeled as training data. Another 30% of remaining samples are set aside as testing data. The testing data will only be used to evaluate the system's performance.

4.2.2 Graph Sample

The Web-Graph as a whole is too large for many practical applications. To address this problem Becchetti et al. [6] have researched methods for taking a sample from the Web-Graph. While it introduces some biases to the perceived graph structure, Breadth-First-Search seems to be a simple and effective method for taking a graph sample. Becchetti et al. [6] recommends starting the BFS with as-many-as-possible websites. For this, all the websites from the training set are set as the starting points for the BFS. In this sampling approach the BFS spreads around evenly from all starting nodes, creating a (possibly disconnected) graph sample.

Starting with 1248 websites from the training set, the graph is collected to around 1 million websites (1,035,758 to be exact). A spectral projection of this graph can be seen in figure 4.2.

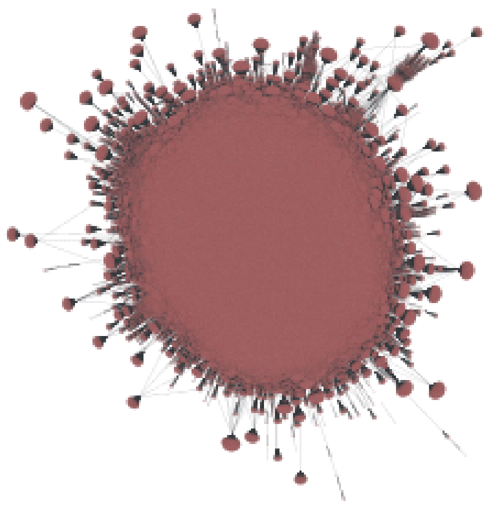


Figure 4.2: Spectral visualization of the 1 million websites collected with BFS.

This graph will necessarily contain all the websites from the training set. A sufficient number of collected websites should also contain at least some of the websites for the testing set. In fact, the 1 million websites contained all of the 538 websites from the testing set. Typically however, it may be expected that some of the websites from the testing set are lost here.

4.2.3 Clustering

In order to extract meaningful information from the graph structure a community detection (i.e. clustering) algorithm is applied.
The goal of clustering is defined in the *Minimum Description Length* (MDL)[27]. The idea here is to fit a probabilistic model to predict the graph structure where the amount of information to express that model is minimized. The Stochastic Blockmodel (SBM) [31] is the base for this, where *blocks* (or clusters/communities) have random edges between them. Optimizing the MDL for the SBM has a problem with large networks such as the current one. It can only find a limited number of blocks dependent on the number of vertices V as $B_{max} = \mathcal{O}(\sqrt{V})$[44]. Fortunately, this problem can be solved with nested clustering.
With a nested SBM each *cluster* found can be clustered again using the same MDL optimization that was used for the whole graph. The actual optimization is performed using an agglomerative Mixed chain Monte Carlo algorithm [45]. Table 4.1 shows the various layers of nested clustering. Moreover, the clustered graph is visualized in figure 4.3.

Layer	Members	Clusters
0	3	1
1	10	3
2	23	10
3	68	23
4	356	68
5	1035758	356

Table 4.1: Different depths of Nested Clustering. The number of members in each layer follow from the clustering done in the layer below it. The 0th layer condenses the clusters into a single node. The 5th layer condenses all 1 million sites into 356 clusters.

Since the finest layer still contains very large clusters (almost 3000 websites per cluster on average), the finest layer should be considered as the discovered communities.

4.2.4 Features

Each website in the graph is given the index (1-356) of the cluster they are in. This is directly used as a categorical feature for a classifier. The idea behind this is that Fake News and real news will be separable by cluster. The classifier may learn that some cluster c consists mostly of Fake News sites based on the training data, and they can then infer that all websites from the test data that are in cluster c must also be Fake News. This feature and the inferences based on this may be simple enough to be manually constructed, which would be more explainable, but may open up more space for human error.
The Haarman-keyphrases [28] used as the other features are a bit more intricate. Haarman-keyphrases are extracted using an unsupervised embedding similarity to the document as a whole. It differs from previous keyphrase extractions such as EmbedRank [8] in that it specifically considers web-formatting to affect the quality of a keyphrase. This prefers keyphrases that occur in the hostname, title, headings or description.
These keyphrases are considered as singular tokens to which a TF-IDF weighting is applied. TF-IDF has previously been shown to be effective in other news classification tasks [21]. Since

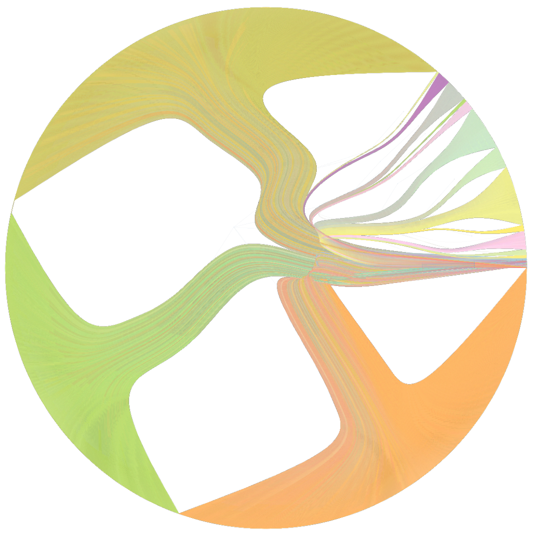


Figure 4.3: Visualization of the nested clustering of 1 million websites. 3 large clusters seem to be responsible for the majority of the websites.

1 million documents will have many different keyphrases, ANOVA feature selection was applied to preserve only the 1000 most distinguishing keyphrases. Such feature selection has previously been found successful in other text classification tasks [24]. The 1000 values for each document form the keyphrase features.

4.2.5 Classifier

The classification is done using a Random Forest [14] of 500 trees. Random Forests are chosen as they are consistent and are not very sensitive to noisy variables [10]. The accuracy of a Random Forest has been shown to not be very sensitive to the number of trees beyond a sufficient size [50], therefore this does not need to be explored into much detail.
An added benefit to the Random Forest as a classifier is that it can offer insight into the importance of certain features, which can help in evaluating the value of cluster indices compared to the keyphrase features.

4.2.6 Evaluation

Naturally, the hypothesis needs to be tested according to some quantitative performance. The current research entails making a comparison between keyphrase feature based classification and cluster index based classification.
In order to establish the value of each feature set, a Random Forest as described in section 4.2.5 is trained on the training data. Additionally, a Random Forest is trained on the joined feature sets: both keyphrase and cluster index features. All 3 classifiers can be evaluated on the test data. The evaluation will be done based on an ROC curve, with the AUC as the most important metric. This approach is chosen because Fake-News classification is not likely to be an even-cost task. If a classifier is used to determine the final result, a very high specificity should be desirable so that oppositional speech will not be hindered. However, if the classifier is used to detect Fake-News candidates for further processing a high sensitivity might be preferable.

Section 4.2.1 already indicates some issues with this approach. Specifically, the testing set (nor the training set) is a perfectly fair fake-news classification. There may be biases in the positive and negative samples that make classification easier, without making the classifier good for classifying new sites. In order to address this issue, newly classified sites are evaluated by humans.
The annotation is simple in design, but challenging in execution. From the 1 million newly discovered sites a sample of 500 sites, distributed uniformly across class probability from the most promising classifier (by test AUC), is taken. These 500 sites are then given to professional website annotators, who classify each website as whether they are fake news or not. Of-course, the annotators are not informed of which classes the classifier assigned to which websites. All classifiers can be evaluated with this dataset, but since it is tailored to one of them, this may or may not give an unfair advantage.
This gives 2 indications of the value of the cluster indices as features: the test-set accuracy and the generalization-set accuracy. By comparing these for the 3 classifiers one may conclude whether cluster features are valuable for this classification task. A third indicator may be constructed from the feature importances of the forest that is given the combined sets of features. The feature importances from this combined forest can indicate whether cluster features of keyphrase features are more effective.

4.3 Results

The ROC curves in figure 4.4 give an overview of the test performance of each classifier. All 3 classifiers show a good AUC and are able to achieve good specificity and sensitivity. The cluster based classifier has a slightly higher AUC. It appears this classifier is able to maintain a better specificity than the others. The classifier with both types of input seems to perform worse than the cluster-only classifier. Model performances are known to be worse when additional bad features are available. This is one of the reasons that feature selection is often considered essential. A more close-up quantitative assessment of the test scores is given in table 4.2. This also shows that the cluster based classifier has a higher accuracy, and quite a bit higher F1 score.
Figure 4.5 shows the feature importances of the features given to the combined classifier. The `Cluster feature` is clearly considered much more important than the best keyword features. These findings combined show that the clusters as made are excellent for good classification on the testing data.

Classifier	Accuracy	AUC	F1
Keyphrase	0.827	0.909	0.782
Cluster	0.867	0.928	0.848
Combined	0.836	0.916	0.797
Random guess	0.561	0.574	0.561

Table 4.2: Performance measures for the 3 classifiers, as well as random guessing. The cluster based classifier has the highest accuracy.

After the testing accuracies, figures 4.6 and 4.7 indicate the performance of the classifiers on the human-annotated generalization data. This is shown as a classifier-predicted probability of fake news for the Fake and Real news annotation labels. Since the predicted class probabilities are

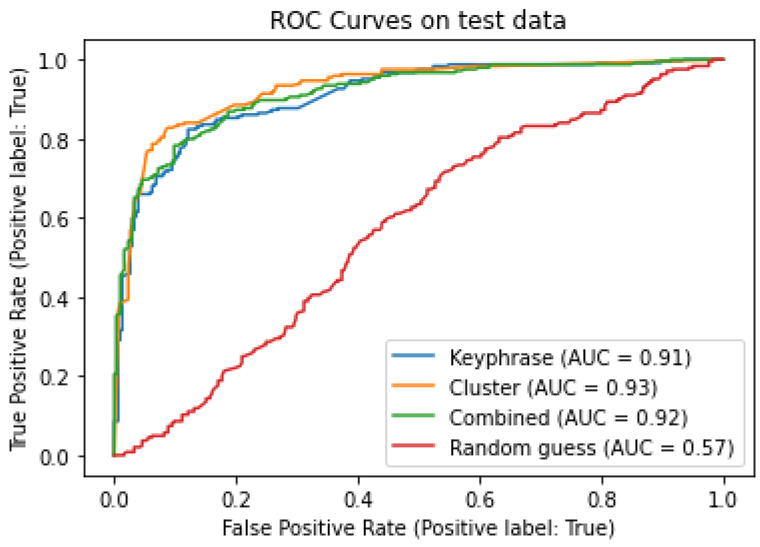


Figure 4.4: Receiver Operator Curve for the 3 classifiers on the test set.

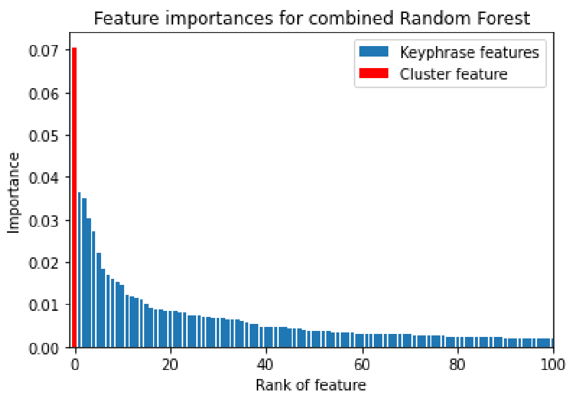


Figure 4.5: Feature importances for the combined Random Forests. The cluster feature is respectably more important than any of the keyphrases, though the total importance of the keyphrases combined is more important than the cluster feature. Only the most important 100 keyphrases are shown, as the subsequent ones follow the trend.

roughly identical for the Real and Fake news, neither classifier was able to generalize to these newly annotated websites. It should be noted that of the 500 websites annotated, only 127 were actual News sites, from those 127, 25 were found too difficult to classify. The remaining 102 News sites were found to be 21% Fake-News. The cluster and keyword classifiers got accuracies of 55% and 53% respectively, though this is not quite an appropriate metric for such imbalanced classes. As an aside it may be noted that the annotators reported the Fake-News labeling as a rather tricky and subjective task, suggesting that their annotations may not be fully objective and consistent.

4.4 Conclusion & Discussion

The results overall show 3 classifiers that perform excellently on the testing set. Unfortunately, the classifiers do not generalize to websites outside the original dataset. This adds some com-

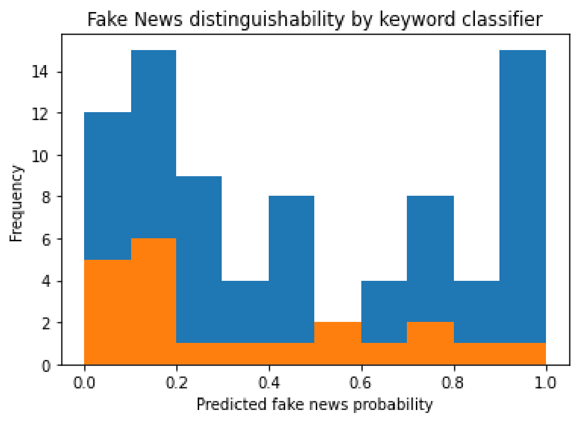


Figure 4.6: Distribution of class probabilities according to the keywords classifier for Real and Fake News.

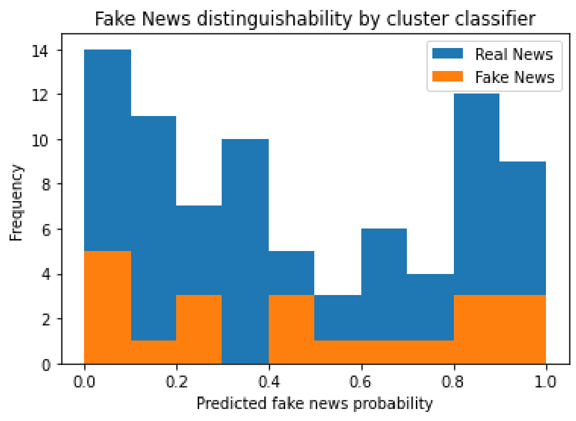


Figure 4.7: Distribution of class probabilities according to the cluster classifier for Real and Fake News.

plication to answering the Research Question about whether the clusters can be used for Fake News classification.

From the fact that the cluster-based classifier outperformed the key-phrase classifier on the test data we can conclude that the clusters are a good feature to distinguish between the classes in the dataset. The fact that these models do not generalize to new sites indicates that the classes in the constructed dataset are not quite identical to the classes as collected and annotated by humans. This difference is not necessarily on the definition of what is or isn't fake news, but may instead come from a bias in the samples.

This does not specifically say anything about the cluster classifier, as the effect is consistent with the keywords classifier. Instead, the answer should be approached by assessing what biases the constructed dataset has and how they might affect the classifiers. For instance, some of the Fake News sites in the constructed dataset may have been collected by traversing the graph structure. This means that they will be more likely to be in a cluster together, and are therefore easier to classify. If this would be the case, then the classifier might learn to classify on that bias, rather than on the actual distinction that we want it to make. All in all, the research question cannot be properly answered with the current dataset, as it contains too much bias.

It should be noted that these generalizability tests are not always standard practice for Machine

Learning research. Models are often trained on a section of a sample (dataset) of the real problem it is actually intend to address. When such a sample has certain biases, the testing section of that dataset will also present the same biases so a modeller will not be aware of the actual performance on the real world. Of course deploying a model on the real world and annotating an evaluation sample on that is a simple step to measure the generalizability of the model. Unfortunately this simple step is also often expensive and often results in a less positive outcome of a project.

Future research for Fake News classification should focus on attempting to draw a real representative sample from the News Sites and annotating those manually. It will likely be rather expensive to actually collect such a dataset, as the classes are highly imbalanced. In order to acquire a sufficient number of Fake News sites one would need to label many more Real News sites. Additionally, a good Fake News site identification pipeline also relies on a step that filters out non-news sites. Since Fake News classifiers are trained exclusively for news sites, their performance can also only be guaranteed for news sites. A classifier that can identify News sites exclusively by their home page can help in identifying Fake News sites. Here it may be noted that many legitimate news sites have rather similar visual layouts. However, some Fake News sites have a vastly different visual layout. Such a news-site classifier should therefore be trained with a mix of fake and real news sites.

Alternatively, a slightly less labour-intensive approach may be applied by attempting to classify the articles on a News Site, and then drawing a conclusion about the fake-ness of the news site on that. For this it is also important to focus on generalizability. There are currently Fake News articles datasets available [2, 19]. In order to ensure generalizability it is best to attempt to train a model on one dataset, with its biases, and then test on another dataset which should have different biases. News articles can then be extracted from News Sites, so that Fake News sites may be identified by the rate of Fake News articles they post.

Chapter 5

Conclusion

Chapter 2 has shown multiple algorithms for Web-Graph clustering. It was able to answer quite definitively that CC-GA can indeed be improved for its application to the Web-Graph. It concludes that the initialization procedure should be adapted to a variation with spread in the initial population. This change results in a respectably faster convergence to the same final modularity. Ultimately it concludes that statistical inference algorithms such as MCMC optimization of Stochastic Block Models is able to achieve an even higher modularity and converges even faster than any of the tested Genetic Algorithms.
It is likely that these two conclusions hold for graphs other than the Web-Graph as well, but that is not proven by the current research. To further solidify this idea, the algorithms need to be applied to different graph structures. The Web-Graph is particularly anomalous due to its large size and incredibly low density. The algorithms may behave very differently on graph with much higher density.
Additionally, it is not clear whether the conclusions about convergence speed and solution quality would persist across different clustering quality metrics. Entropy is often used for defining the quality of Stochastic Block Model solutions, so it would be interesting to see whether SBM oriented algorithms fare comparatively better when optimizing entropy compared to modularity.

5.1 Trust Score

With respect to Trust Score re-estimation using the Web-Graph, chapter 3 was not able to show a significant difference between SBM cluster based and BFS based website selection. Despite not technically being able to answer the research question, the results do suggest that with the current method there is not a substantial difference between BFS and cluster collected websites. Since the clustering is rather computationally expensive, BFS would be preferred. However, it should be noted that the number of neighbours collected with BFS was determined by the size of the cluster in this experiment. This ensures that any observed differences are not caused by different cluster sizes, but also means that the BFS website collection as shown here cannot directly be used outside the experimental environment without introducing a hyperparameter to determine the number of neighbours used.
Since the graph sample for the Trust Score experiment was collected using BFS from multiple starting nodes the sample consisted of several disconnected components. With more computational availability a larger graph sample could be collected until the entire graph sample would

be connected. This would certainly affect the clusters that are produced, and might even result in better accuracy for the Trust Score re-estimation.
Ultimately, this Trust Score experiment needs future research dedicated to measuring the accuracy of the combined estimator. Currently, it is only theoretically established that the combined estimator is better, but it is not shown on actual data. Additionally, that would open avenues to measure the generalizability of both models.

5.2 Fake News

With regards to the question whether nested SBM clusters are a good feature for Fake News classification compared to extracted keyphrases it can be concluded that they are an excellent feature. While it was hypothesized that the cluster indices might be nearly as good as the keywords, they actually received a higher accuracy.
Unfortunately, it should be noted that due to biases in the dataset neither model generalized to unseen data. This does not necessarily say anything about the methodology, but only specifically shows that the dataset was not a representative sample of the classification task.
Nonetheless, it may be noted that since the dataset did not represent a Fake News classification task well enough, the conclusions drawn may also not generalize to an actual Fake News classification task. For example, if the Fake News websites were collected by traversing the Web-Graph the cluster indices may be given an advantage on this dataset that does not hold for the actual problem.
To really draw consistent conclusions a dataset should be made by drawing a random sample of news sites and having those annotated by humans. Any constructed dataset such as the current one has an inherit risk of bias as a result from the way the dataset is constructed. Unfortunately, this is rather expensive due to the unbalanced classes. This means one would need to annotate lot of news sites in order to get a sufficient number of Fake News sites.

5.3 In Closing

The core research question of this thesis was dedicated to finding a method to apply graph-clustering to the Web-Graph, and creating ways to use these clusters to identify malicious websites. The first part in this addresses the clustering method. For this any Genetic Algorithm was found to be inferior to SBM optimization using Markov-Chain Monte Carlo. This shows that MCMC clustering is a valid way to do Web-Graph clustering. However, the current research was not able to apply it to the whole Web-Graph, only to several (sizable) samples. Since the MCMC clustering has a complexity of $\mathcal{O}(V \ln(\ln V))$ where V is the number of nodes [45], the whole Web-Graph may be clustered within reasonable time provided sufficient computational resources.
Following a successful method of clustering, a task remains of applying these clusters to malicious website identification. The Trust Score experiments were not able to show specific value to the clusters beyond alternative methods. However, this does not mean that the clusters are not usable for this, only that they may be rather expensive to compute when there are simpler alternatives available. If the clusters are already available and computed for another purpose, the clusters may be used so that no additional hyperparameters for alternative methods need to be explored.

The clusters also appear to be effective for Fake News site identification, by using the cluster index as a feature for a classifier. This then learns which clusters contain fake news sites and which ones contain real news sites. It should be noted here that it was found that the dataset was not a proper representation of the task, and that the conclusions might not generalize to news sites beyond the dataset.

These findings show a sufficient, but not complete answer to the Research Question. We've shown that graph-clustering can be done on the Web-Graph using MCMC, and that these clusters can be used by index on cluster peers to make classifiers or estimators for malicious site identification. While these are interesting findings, they are not a complete picture. Having found 1 effective method of clustering and 2 effective methods of malicious site identification does not mean there are not (many) more possible methods.

5.4 Future Research

The current research has looked at clustering as an independent task, attempt to optimize some cluster quality exclusively based on graph structure. However, the clusters are subsequently used for methods that might not be as affected by the cluster quality. It would be interesting to see whether the quality of a cluster solution, for example defined by modularity, would correlate to the quality of a classifier or estimator based on these clusters. If it were to be found that the quality of the models is not strongly influenced by the quality of the clusters then the clustering method may be adapted to safe computational cost.

Alternative clustering algorithms may also be considered in such a way. The Label Propagation Algorithm [51] is an interesting path for this. Because it is nearly linear in time it may be much easier to compute. This will definitely alter the clusters that are formed, but it should be investigated whether this would also affect the quality of subsequent models.

5.4.1 Other Cluster Definitions

This entire thesis has worked with a consistent definition of what a clustering may be. In this definition every website belongs to exactly one cluster, with some number of other sites in that same cluster. This is a very intuitive definition to partition the graph with, but it may not necessarily be the best foundation to build subsequent models. An interesting alternative is the overlapping stochastic block model [25]. In this cluster definition each site can be a member of multiple communities. This aligns better to an intuition of social communities, where an individual can be both a member of a fish-keeping community as well as a local-businesses community. This aligns quite naturally with the proposed uses of the cluster here, but may also open avenues for new features to construct.

Another interesting avenue would be clustering methods where not every website needs to be part of a cluster. Particularly due to the low density of the Web-Graph forcing every website to be in some cluster may muddy the meaning of the clusters for the subsequent model. Clique detection could be an example of this, but triangle counting has already shown useful for detecting spam sites [7]. In triangle counting the transitivity around a website is measured, which is then used as an indication of genuine community structure rather than a spam site linking to random sites.

5.4.2 Other Classifications

The current research only focused on two hard-to-define types of malicious websites. Focusing on Fake News and untrustworthy websites is a good start, but it does not cover the entire scope of malicious sites that may be out there. Of course there are also counter-fitters, malware sites, hate-speech sites, and of course entire criminal organizations. These all need dedicated research to identify them. Some of these types of malicious sites may benefit more or less from insights from the Web-Graph.

For large-scale criminal activity, like illegal arms and drug trade, research may instead find itself to the dark web [16]. This would have a distinct Dark-Web-Graph [3] From this research direction it would also be really interesting to compare the Dark-Web-Graph to the regular Web-Graph. It already appears that they hold similar graph structures, such as the bow-tie decomposition [23]. It would be interesting to see whether malicious website identification methodologies on the surface Web-Graph would also work well on the Dark-Web-Graph.

Bibliography

[1] Alexandre H Abdo and APS de Moura. Clustering as a measure of the local topology of networks. *arXiv preprint physics/0605235*, 2006.

[2] Hadeer Ahmed, Issa Traore, and Sherif Saad. Detecting opinion spams and fake news using text classification. *Security and Privacy*, 1(1):e9, 2018.

[3] Abdullah Alharbi, Mohd Faizan, Wael Alosaimi, Hashem Alyami, Alka Agrawal, Rajeev Kumar, and Raees Ahmad Khan. Exploring the topological properties of the tor dark web. *IEEE Access*, 9:21746–21758, 2021.

[4] Max Aliapoulios, Emmi Bevensee, Jeremy Blackburn, Emiliano De Cristofaro, Gianluca Stringhini, and Savvas Zannettou. An early look at the parler online social network. *arXiv preprint arXiv:2101.03820*, 2021.

[5] Chaitrali Amrutkar, Patrick Traynor, and Paul C Van Oorschot. Measuring ssl indicators on mobile browsers: Extended life, or end of the road? In *International Conference on Information Security*, pages 86–103. Springer, 2012.

[6] Luca Becchetti, Carlos Castillo, Debora Donato, Adriano Fazzone, and I Rome. A comparison of sampling techniques for web graph characterization. In *Proceedings of the Workshop on Link Analysis (LinkKDD'06), Philadelphia, PA*, 2006.

[7] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24, 2008.

[8] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. Simple unsupervised keyphrase extraction using sentence embeddings. *arXiv preprint arXiv:1801.04470*, 2018.

[9] Claude Berge. *The theory of graphs.* Courier Corporation, 2001.

[10] Gérard Biau. Analysis of a random forests model. *The Journal of Machine Learning Research*, 13(1):1063–1095, 2012.

[11] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.

[12] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.

[13] Heinrich Braun. On solving travelling salesman problems by genetic algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 129–133. Springer, 1990.

[14] Leo Breiman. Random forests. *UC Berkeley TR567*, 1999.

[15] Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[16] Hsinchun Chen. *Dark web: Exploring and data mining the dark side of the web*, volume 30. Springer Science & Business Media, 2011.

[17] Farhan Asif Chowdhury, Dheeman Saha, Md Rashidul Hasan, Koustuv Saha, and Abdullah Mueen. Examining factors associated with twitter account suspension following the 2020 us presidential election. *arXiv preprint arXiv:2101.09575*, 2021.

[18] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American statistical association*, 74(368):829–836, 1979.

[19] UTK Machine Learning Club. Fake news, Feb 2018.

[20] Kerry G Coffman and Andrew M Odlyzko. Growth of the internet. In *Optical fiber telecommunications IV-B*, pages 17–56. Elsevier, 2002.

[21] Seyyed Mohammad Hossein Dadgar, Mohammad Shirzad Araghi, and Morteza Mastery Farahani. A novel text mining approach based on tf-idf and support vector machine for news classification. In *2016 IEEE International Conference on Engineering and Technology (ICETECH)*, pages 112–116. IEEE, 2016.

[22] Ivo de Jong. Adapting CoSyNE to fight forest fires, Juli 2019.

[23] Debora Donato, Stefano Leonardi, Stefano Millozzi, and Panayiotis Tsaparas. Mining the inner structure of the web graph. In *WebDB*, pages 145–150. Citeseer, 2005.

[24] Nadir Omer Fadl Elssied, Othman Ibrahim, and Ahmed Hamza Osman. A novel feature selection based on one-way anova f-test for e-mail spam classification. *Research Journal of Applied Sciences, Engineering and Technology*, 7(3):625–638, 2014.

[25] Prem K Gopalan, Sean Gerrish, Michael Freedman, David Blei, and David Mimno. Scalable inference of overlapping communities. *Advances in Neural Information Processing Systems*, 25:2249–2257, 2012.

[26] Jeffrey Gottfried and Elisa Shearer. News use across social media platforms 2016. 2016.

[27] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.

[28] Tim Haarman, Bastiaan Zijlema, and Marco Wiering. Unsupervised keyphrase extraction for web pages. *Multimodal Technologies and Interaction*, 3(3):58, 2019.

[29] Matthew Hannah. Qanon and the information dark age. *First Monday*, 2021.

[30] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[31] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[32] Mirko Křivánek and Jaroslav Morávek. NP-hardf problems in hierarchical-tree clustering. *Acta informatica*, 23(3):311–323, 1986.

[33] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, Dandapani Sivakumar, Andrew Tompkins, and Eli Upfal. The web as a graph. In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–10, 2000.

[34] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6), Dec 2011. ISSN 1550-2376. doi: 10.1103/physreve.84.066122. URL `http://dx.doi.org/10.1103/PhysRevE.84.066122`.

[35] Pedro Larranaga, Cindy M. H. Kuijpers, Roberto H. Murga, Inaki Inza, and Sejla Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

[36] Barry M Leiner, Vinton G Cerf, David D Clark, Robert E Kahn, Leonard Kleinrock, Daniel C Lynch, Jon Postel, Larry G Roberts, and Stephen Wolff. A brief history of the internet. *ACM SIGCOMM Computer Communication Review*, 39(5):22–31, 2009.

[37] Guillermo Mendez and Sharon Lohr. Estimating residual variance in random forest regression. *Computational statistics & data analysis*, 55(11):2937–2950, 2011.

[38] Melanie Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[39] Wouter Mostard, Bastiaan Zijlema, and Marco Wiering. Combining visual and contextual information for fraudulent online store classification. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 84–90, 2019.

[40] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2): 167–256, 2003.

[41] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[42] Ben O'Neill. Exchangeability, correlation, and bayes' effect. *International statistical review*, 77(2):241–250, 2009.

[43] Evelien Otte and Ronald Rousseau. Social network analysis: a powerful strategy, also for the information sciences. *Journal of information Science*, 28(6):441–453, 2002.

[44] Tiago P Peixoto. Parsimonious module inference in large networks. *Physical review letters*, 110(14):148701, 2013.

[45] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[46] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194. URL `http://figshare.com/articles/graphtool/1164194`.

[47] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194.

[48] Ken Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682, 2002.

[49] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. *arXiv preprint arXiv:1702.05638*, 2017.

[50] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *J. Mach. Learn. Res.*, 18(1):6673–6690, 2017.

[51] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[52] Noraini Mohd Razali, John Geraghty, et al. Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong, 2011.

[53] Anwar Said, Rabeeh Ayaz Abbasi, Onaiza Maqbool, Ali Daud, and Naif Radi Aljohani. CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks. *Applied Soft Computing*, 63:59–70, 2018.

[54] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[55] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[56] Chuan Shi, Zhenyu Yan, Xin Pan, Yanan Cai, and Bin Wu. A posteriori approach for community detection. *J. Comput. Sci. Technol.*, 26:792–805, 09 2011. doi: 10.1007/s11390-011-0178-z.

[57] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1): 22–36, 2017.

[58] Matt Skibinsi. Coronavirus misinformation tracking center. URL `https://www.newsguardtech.com/coronavirus-misinformation-tracking-center/`.

[59] Maciej Szpakowski. Fakenewscorpus, 2018.

[60] William Yang Wang. "liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[61] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[62] Eric W Weisstein. Bonferroni correction. *https://mathworld. wolfram. com/*, 2004.

[63] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pages 1–5, 2015.

[64] Dejian Yu. Softmax function based intuitionistic fuzzy multi-criteria decision making and applications. *Operational Research*, 16(2):327–348, 2016.