Recommendation Algorithms to Increase Network Fairness

Naisha Agarwal
Computer Science
University of California, Los Angeles
Los Angeles, California, USA
naisha1025@gmail.com

ABSTRACT

This paper introduces a novel problem of improving fairness in networks as defined by access to influencers. Most real-world networks, especially within social media, have a small fraction of highly influential nodes. Access to these influential nodes leads to better future opportunities. However, recommendation systems used in social media often maximize the total number of edges to increase engagement and advertising revenue. This creates bias and often leads to inequitable access to influencers among nodes, creating unequal access to future opportunities. This paper provides a principled formulation of this problem using concepts in graph algorithms. We first introduced a novel scale-free measure to quantify this fairness in networks. A novel recommendation algorithm is then introduced that can seamlessly work with existing recommendation algorithms to increase this fairness in networks. The algorithm introduced in this paper is inspired by the seminal work of Watts & Strogatz in the context of decentralized search. As such, the approach is to introduce random edges in a controlled fashion to create more weak ties in a network and reduce the overall distance of nodes from the influencer set. Through extensive simulations on two real-world network data sets and comparing seven different algorithms, it is shown empirically that the fairness of the graph tends to increase monotonically as the amount of controlled randomness introduced increases. This current approach assumes the graph is connected; future pursuits include generalization of the proposed method to disconnected graphs. This work provides a foundation to improve machine learning algorithms used in networks, particularly social media, using foundational concepts in graph theory.

KEYWORDS

Social networks, graph theory, network fairness, recommendation algorithms, Watts-Strogatz model

1. Introduction

Networks are ubiquitous and significantly influence daily lives. Oftentimes these networks have a highly influential small set of nodes, and having access to these nodes can be quite advantageous. For instance, on a social media site like LinkedIn, access to decision makers at large corporations can increase the likelihood of obtaining jobs, sales, and other opportunities. Similarly, on Twitter and Facebook, access to influencers can help businesses market messages more easily and extensively (1). To ensure a level playing field for all network nodes, nodes in a graph must have equal access to influencers. However, in many real-world applications, new edges (connections) are formed via recommendation systems that are based on network attributes like triangle closings (the number of mutual friends) and node characteristics like demographics, industry, school, or interests (2). While these systems are great at optimizing the number of edges that are formed in the network over time, they create unintentional biases in the system where some nodes have an unfair advantage of gaining more access to opportunities and utilities through influencers. The algorithms that power these recommendation systems further exacerbate the situation due to the attributes they rely on.

The main objective of this paper is to show that it is possible to modify machine learning algorithms used in social media to balance two competing objectives: profit for the social media companies and fair access to influencers across all nodes. Under these modified algorithms, people will not be penalized based on who they know and will instead be rewarded based on what they know. This will result in a more equitable society despite the increasing influence of social media and digitization. This paper's technical approach is inspired by the seminal work of Watts and Strogatz. In their paper, they demonstrated that introducing randomness can lead to more connections in a social network and reduce the diameter of the network (3). While Watts and Strogatz inspired the idea of adding randomness in a controlled fashion, this paper's approach applied this idea to node recommendation algorithms to provide more equitable access to influencers. A systematic study of introducing this randomness into node recommendation algorithms in the context of networks to increase fairness has not been studied before.

There have been several solutions proposed to measure fairness in networks, such as the Atkinson Index and using various other types of algorithms in networks (4,5). However, these algorithms provide variance in utility associated with a node across the entire network. In this paper, a scale-free fairness measure is calculated using the influencer set and is improved using novel node recommendation algorithms.

The efficacy of the approach is proved through extensive simulations on two real-world datasets comparing seven different algorithms. It was shown that adding controlled randomness results in fairness increasing monotonically with increasing randomness. This paper will motivate further research that can leverage well established mathematical models in networks and graphs to improve machine learning algorithms used for graphs in social media.

2. Technical Approach

Graphs used in this paper represent social networks. A given graph is assumed to have nodes with the same attributes (ex. skill level, age, gender, location), and edges that represent the connections among those nodes. In other words, the likelihood of any pair of nodes connecting is the same in terms of the attribute properties of the different nodes. The difference in the likelihood of any pair of nodes connecting comes from graph structure and the differences in the number of mutual edges. The graph is also assumed to be connected; hence a path always exists. Disconnected graphs are discussed in future work.

2.1 Finding an appropriate measure of fairness

There are three main components to finding an appropriate measure for fairness:

1) finding the influencers in the graph using graph theoretic measures;
2) finding the fairness measure of the overall graph; and
3) normalizing the graph fairness measure.

These methods are summarized and explained in further detail below.

2.1.1 Determining influencers

When measuring the influence of a node on the network, betweenness centrality was used as the measure of centrality. Betweenness centrality measures the proportion of times a node acts as a bridge along the shortest path between two other nodes (6). The equation for betweenness centrality for a node v is

$$c(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \quad (1)$$

σ_{st} is the total number of shortest paths from a source node, s, to a target node, t, and σ_{st} (v) is the number of σ_{st} that pass through v (7).

Influencers were found by calculating the betweenness centrality of every node in the graph and taking the top k percent of nodes as the influencers.

2.1.2 Finding the fairness measure

Fairness is defined to be the minimum shortest path to the influencer set of an existing path. To compute the fairness of the entire network, the fairness of a single node is first found, and then an average is taken of each node (described in further detail below) to compute the fairness of the entire network. Using Dijkstra's algorithm (8), the fairness of a node can be measured using

$$fairness(v) = \min_{\{s \text{ in } S\}} dist\ (v, s) \quad (2)$$

where S is the influencer set, s is the source node in S, and v is a node in the graph. If no path exists from a node to the influencer set, individual node fairness is not defined. In this paper, it was assumed the graph is connected, hence there is always a path.

2.1.2.2 Getting fairness of total graph

The fairness of the graph is defined to be the average of the top t percent of node fairness values. We do this so that we are modeling the fairness of the graph only using the best node fairness values (defined by t). This can be modeled by the equation:

$$fairness(G) = \frac{1}{\text{top } t\%_{\{v \text{ in V -S}\}} \text{fairness}(v)} \quad (3)$$

where G is the graph in question, v is a node in the graph, and V-S is the set of vertices in the graph not including influencers. As the denominator becomes smaller, more nodes have better access to influencers and the fairness of the graph increases.

2.1.3 – Calculating fairness index

Since graphs have different numbers of nodes and edges, it is important to normalize the fairness values to ensure it is scale-free and easily comparable across different graphs. To normalize the fairness measure, the fairness of a random graph with the same number of nodes and edges as the observed graph was computed. A random graph was used to obtain a scale free fairness index that makes it possible to compare fairness after adjusting for the varying number of edges and nodes in the graph. The fairness index is defined as:

$$fairnessindex = \frac{\text{fairness(G)}}{\text{fairness(random graph)}} \quad (4)$$

The Erdos-Renyi graph was used to produce random graphs. The Erdos-Renyi graph randomly connects nodes with every edge included in the graph with selected probability p. The probability for generating a graph with n nodes and m edges is modeled by

$$p^{m}(1-p)^{(C_2^n - m)} \qquad (5)$$

where the likelihood of adding more edges to the given graph increases as probability p increases (9).

The Barabasi-Albert was another method used to produce random graphs. This graph begins with an initial connected network of n nodes, where new nodes are added to existing nodes with probability p_i. p_i is the probability that a new node is connected to node i. This is modeled through the equation

$$p_i = \frac{k_i}{\sum_j k_j} \qquad (6)$$

where k_i is the degree of node i and the baseline sum is made over all pre-existing nodes j (10).

When generating Erdos-Renyi and Barabasi-Albert graphs, different numbers of random graphs (e.g. 5, 10, and 100) were tested and fairness values were averaged for each graph. The fairness values typically converged and stabilized around 10 random graphs and above. There was an average of 10 random graphs when calculating the fairness index.

2.2 Recommendation algorithms to increase the fairness index in a network

The main idea of the proposed recommendation algorithm was motivated by a similar idea in the Watts-Strogatz model in the context of the small-world experiment and decentralized search (3). The high-level mathematical model behind this approach states for a given node n, with some probability P, the proposed algorithm recommends a node based on the number of mutual friends between n and the candidate node. The larger the number of mutual friends, the more likely it is to recommend the node. With probability Q = (1-P), the algorithm adds some randomness and creates a weak tie by forming a new edge with a randomly selected target node. This experiment introduced a modification where the weak tie is selected via an importance sampling algorithm that is proportional to some function based on degree and distance to the influencer set of the candidate node. This algorithm is illustrated in pseudocode in the appendix.

2.2.1 – Details

For a given node n, consider the array M[n] = {(m, i(m), w(i(m))}. Here, m denotes the target candidate nodes that n is not yet connected to. For each node m, i(m) denotes the number of mutual friends (edges) between n and m, and w(i(m)) denotes the weight of m. The influencer set is not included in M[n] to avoid trivial solutions.

2.2.2 – Calculate w(i(m)) values

In order to give people with more mutual friends a higher probability of connecting, a weight function for each node was calculated. The weight w(i(m)) should be an increasing function of i(m) because the more mutual friends there are between n and m, the more likely they are to connect. Therefore, they should be recommended with a higher probability. One popular w function used is the sigmoid function q(i)=1/(1+ e^{a-i}) where a is a constant. Since q(0)=1/(1+e^{a}) is a constant, the weight function is

$$w(i) = \frac{q(i)}{q(0)} = \frac{(e^{-a}+1)}{(e^{-a}+e^{-i})} \approx e^{i} \qquad (7)$$

The approximation was made assuming a was very large due to the probability of two nodes connecting with zero mutual friends being very small. Hence, w(i)= e^{i}. This equation is derived in the subsequent section.

Consider another array for a given node n as R[n] = {m, D(m), d(m), wr(a, b))}. Here, D(m) is the degree of m, d(m) is the minimum distance of m to the influencer set, wr(a,b) is the weight function written as wr(a,b)=$\frac{(D(m))^a}{(d(m))^b}$ and a and b are nonnegative constants. Recall m refers to the target candidate nodes that n is not yet connected to. The reasoning behind choosing this weight function is described in the appendix.

2.2.3 – Algorithm details

For probability P, the algorithm recommends a node for a given visit node n by sampling a node from M with weights proportional to w(i)’s. Probability Q = 1-P recommends a node by sampling from R with weights proportional to wr(a,b). Note that wr(0,0) = 1. Just like in the Watts-Strogatz model, for probability P, the new algorithm recommends a node that is close to n’s neighborhood, while for probability Q, the algorithm selects a node at random (assuming a=b=0). These random nodes shorten the path to the influencer set and create more fairness.

For simplicity and efficiency in the simulations, the algorithm always connects to the recommended node. In practice, this does not happen and recommendations have different connection probabilities that are not constant. For instance, the connection probability to a node with zero mutual friends is typically lower than connecting to a node with ten mutual friends. Since the goal of this paper is only to study the impact of adding diversity to fairness, this assumption can be made without loss of generality. In the end, what matters is the budget in terms of adding nodes at random and the fairness index it produces.

Similarly to the Watts-Strogatz model, diversity can be introduced in a controlled manner. Two intuitions were used. First, sampling higher degree nodes with higher weights is helpful since they are likely to create more paths to the influencer set. Additionally, if two nodes are given that have the same high degree, the one that is closer to the influencer set is more likely to increase fairness.

However, it is not clear what is better: a very high degree node far away from the influencer set (e.g., a node with a degree of 100 that is a distance of 3 from an influencer) or a less high degree node closer to the influencer set (e.g., a node with a degree of 50 that is a distance 1 from an influencer). Simulations tested multiple a and b values, which will be described in further detail in the experiments section (Table 1).

To get more intuition into the algorithm, the distance between w(*i*) (the mutual friends algorithm) and wr(a,b) was computed using the Kullback-Leibler distance function. The distance is given by

$$\mathrm{KL}(w, wr(a = 0, b = 0)) = A \sum_{k=1}^{K} f(k) e^{k} + C \qquad (8)$$

where k denotes the number of mutual friends, K is the maximum value of mutual friends there are in the data, f(k) denotes the frequency of target nodes that have k mutual friends with the source node, and A and C are constants. We assume a=b=0 for comparing the distance with a completely random distribution as the baseline. As can be seen above, the distance from the random distribution increases when the source node has a higher concentration of target nodes with a large number of mutual friends. These are the scenarios where the randomization is breaking the "rich gets richer" characteristics of the usual recommendation algorithm and creating more fairness.

2.2.4 – Deriving the Kullback-Leibler equation

For a given source node *n*, let *m* subscript be the target node the source node can connect to and let P and Q be the probability distributions for weights w and wr (a=b=0), respectively. Therefore, P = Aw and Q=B(wr), where A and B are normalizing constants:

$$\mathrm{KL}(P, Q) = \sum_{m} P(m) \log\left(\frac{P(m)}{Q(m)}\right) \qquad (9)$$

Since w(m)= $e^{i(m)}$ and wr(a=b=0) = 1, where i(m) is the number of mutual friends between n and target node m, the expression simplifies to:

$$\mathrm{KL}(P, Q) = A \sum_{m} i(m) e^{i(m)} + C, \text{where C is a constant} \quad (10)$$

Let *k* be the number of mutual friends and f(*k*) denote the frequency of target nodes with *k* mutual friends with the source node n. Doing a frequency tabulation of i(m)$e^{i(m)}$over all target nodes, the expression becomes:

$$\mathrm{KL}(P, Q) = A \sum_{k=1} f(k) k e^{k} + C \qquad (11)$$

2.3 Experiments

Simulations were conducted on multiple datasets to show the efficacy of the proposed methods: a Facebook social circles graph and a LastFM graph (Figure 1). Both data sets had different network structures and provided complementary insights to the methodology in this paper. For each data set, the D^a / d^b algorithm was run for seven pairs of a and b (Table 1).

The Barabasi-Albert random graph was used to calculate the fairness index of nodes (Figure 2).

Additionally, the influencers of the LastFM graph were manually changed to show the robustness of this algorithm (Figure 3). The new influencers had very low betweenness centralities, representative of situations where influencers who don't have a big social media presence are still influential in the real world.

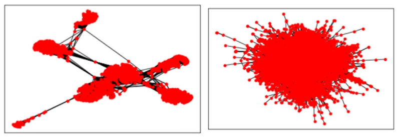


Figure 1: Visualization of Facebook and LastFM Networks

Nodes and edges of the graph were visualized using the NetworkX draw feature, which draws the graph using Matplotlib. Nodes (dots) on the graph represent a person in a social media network, and edges (lines) represent a connection in the network. The graph on the left representing the Facebook social network shows several dense communities loosely connected by weak ties. The graph on the right representing the LastFM network shows one central community with some nodes on the periphery.

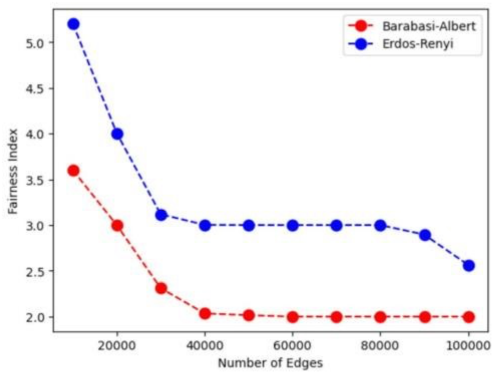


Figure 2: Fairness Convergence for Random Graphs for Different Amounts of Edges

This graph compares the Erdos-Renyi and Barabasi-Albert random graphs. The x-axis indicates the different number of edges and the y-axis represents the fairness index of these graphs. The random graphs’ fairness indices begin to converge to a singular value as the number of edges increases. The Barabasi-Albert random graph converges faster than the Erdos-Renyi graph; while the Erdos-Renyi graph starts to converge at 50,000 edges and then changes values at 100,000 edges, the Barabasi-Albert graph begins to converge at 40,000 edges and stays at the same fairness value through the 100,000 edges. Therefore, the Barabasi-Albert random graph serves as a better normalizing value for calculating the fairness index.

a value	**b value**	**representation**
0	0	completely random
1	0	proportional to degree
1	1	proportional to D/d
1/2	1/2	proportional to $\sqrt{D/d}$
0	2	proportional to $1/d^2$
2	2	proportional to $(D/d)^2$
1	2	proportional to D/d^2

Table 1: Different values of a and b being tested

This table details the different values of a and b that are being tested in our overall $\boldsymbol{D^a/d^b}$ recommendation algorithm, and that are graphed in Figure 3.

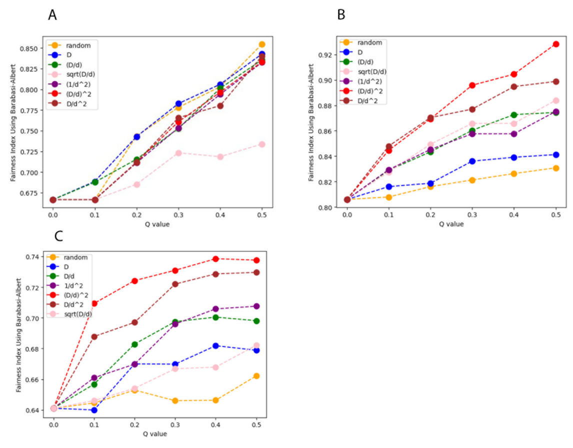


Figure 3: Changes in Fairness with Different Recommendation Algorithms for Social Networks

The x-axis in graphs A,B, and C is the probability Q that the recommendation algorithm recommends a node based on randomness. The y-axis is the fairness index, where a higher value indicates more fairness in the graph. Each graph compares numerous recommendation algorithms of the form $\boldsymbol{D^a/d^b}$, with various values of a and b plotted. Graph A compares recommendation algorithms in the Facebook social network. The blue line with the degree algorithm (a = 1, b = 0) consistently appears to be the highest line of the graph, indicating that it achieves the greatest degree of fairness. Graph B compares recommendation algorithms in the LastFM social network. The red line, or the $(D/d)^2$ algorithm (a =2, b=2), consistently earns the highest fairness index, indicating it produces the greatest fairness. Graph C also compares different recommendation algorithms for the LastFM social network, except with a different choice in influencers (low betweenness centralities). It appears that $(D/d)^2$ still produces the highest fairness indexes, proving the robustness of the algorithms with a change in influencers.

3. Results

3.1 – Random graph

It was found that the Barabasi-Albert random graph converged faster than the Erdos-Renyi graph and provided a better normalizing value for calculating the fairness index (Figure 3). The fairness index of nodes were calculated using the Barabasi Albert graph.

3.2- Facebook data

It was observed that degree (a =1, b = 0) was the best algorithm for the Facebook graph as it consistently produced the highest data points. D/d^2 and $(D/d)^2$ were close seconds (Figure 3).

Each of the algorithms had the same value for Q=0 due to the locations’ algorithms being based on mutual friends and not randomness. The fairness index increased when there was a higher chance of randomness in the network, indicated by the decreased P and increased Q.

3.3- LastFM data

It was observed that $(D/d)^2$ resulted in the highest fairness values for the LastFM data (Figure 3).

3.4- Studying the impact of changing influencers on the fairness index

The algorithm was robust to a change in influencers by showing that $(D/d)^2$ resulted in the highest fairness values (Figure 3).

4. Discussion

As the amount of randomness, Q, in the network increased, the overall fairness in the network in terms of access to influencers increased. This is the most critical and basic observation, and it shows the efficacy of the approach. Based on this experiment, it appeared that introducing randomness as some polynomial function of D and d makes a difference relative to complete randomness. $(\mathrm{D}/d)^2$ seemed to be the best recommendation algorithm to increase fairness in the network, although it is specific to the dataset at hand. This result is significant and important to find algorithms that can achieve high fairness with lower randomization. More randomization would lead to more loss in number of edges and hence engagement and revenue. Additionally, the algorithm found is robust to small changes in the quantity and set of influencers. The overall increase in fairness with increasing randomness deteriorated but the monotone increasing pattern remained intact. This is again significant because the definition of influencers in the real-world could be defined in various ways. For instance, it could include a famous personality who is known to be an influencer but does not necessarily engage in a social network, and hence, they could have low betweenness centrality. This result shows networks can add such nodes in the influencer set based on relevant domain knowledge and still obtain fairness using the algorithm proposed.

5. Conclusion

A novel problem of measuring network fairness in terms of access to influencers was introduced in this paper. Novel methods to measure this fairness and increase this fairness with recommendation algorithms were proposed. The efficacy of the approach was tested through extensive simulations on two real world datasets comparing several variations of the algorithm. First, experiments conducted proved the efficacy of the proposed approach: fairness tends to increase monotonically with increasing randomness. This is important as increased randomness will also monotonically increase the number of edges lost compared to the canonical recommendation algorithm. Hence, the outlined approach provides a clear mechanism to incorporate fairness in social media through a simple randomization mechanism that can be easily incorporated into existing systems. This study also showed that introducing randomness via importance sampling through a polynomial function of D and d can yield better fairness with lower randomness. Other functions based on node attributes can improve this even further. Machine learning applications can be explored in future work. Finally, this study showed that the monotone relationship is intact with changes in the influencer set. This is a critical finding as in the real-world, influencers may not be necessarily determined solely based on the graph theoretic measure we introduced. Some domain knowledge may also be important. For instance, a famous personality who is not very active on social media could still be considered as an influencer.

Previous work studied bias and fairness in supervised machine learning algorithms that address fairness by modifying the training data and the supervised algorithms (5). While this work is similar in motivation, it is highly specific to networks. Additionally, there is very recent work being done on dyadic preferences in a graph, where the authors of the paper address potential bias in user recommendation (11). However, their work is specific to ensuring demographic parity with respect to node attributes in the network. This work addresses a more specific problem of ensuring fairness with respect to access to influencers. While it is more specific, it does require new methods to measure fairness and modify the recommendation algorithms. The area of studying fairness in networks is relatively new and hence the literature in this area is sparse.

The methods discussed only apply to connected graphs as the fairness measure would not be defined if the graph is disconnected. Future investigations include studying generalizations to disconnected graphs. Additionally, since the simulations in the experiment were based on small datasets, it encourages future studies of behavior with very large graphs. This will require large scale distributed computing with graphs that we plan to explore. The activeness of a user can also be considered when running simulations. Currently, it is assumed all nodes are equally likely to visit the network in the simulations. It is also assumed that the number of nodes in a graph remains constant. The number of nodes visiting a network depends on how active a user is, and nodes in a graph are not constant due to users constantly joining and leaving.

ACKNOWLEDGMENTS

We would like to thank MIT Professor Julian Shun for his guidance throughout this research.

REFERENCES

[1] Thorpe, Jessica, 2022. How Influencers Can Help Your Business in Times of High Inflation. *Fast Company.*

[2] How We Analyzed Twitter Social Media Networks, 2014. PewResearchCenter. Social Media Research Foundation.

[3] D. Watts, S. Strogatz, 1998. Collective dynamics of ‘small-world’ networks. *Nature.* **393,** 440-442. DOI: 10.1038.30918.

[4] G. Saint-Jacques, A. Sepehri, , I. Perisic, N. Li, 2020. Fairness through Experimentation: Inequality in A/B Testing as an approach to responsible design. *arXiv*. DOI: 2002.05819.

[5] Mehrabi, N,. Morsatter, F., Saxena, ., Lerman, K,. Galsytan, A, 2019. A Survey on Bias and Fairness in Machine Learning, *arXiv: 1908.09635*

[6] Brandes, U, A Faster Algorithm for Betweenness Centrality, 2001. *J Math Sociol*, *25*(2), 163-177

[7] Freeman, L, 1977. A Set of Measures of Centrality Based on Betweenness. *Sociometry*, 40(1), 35-41. doi: 10.2307/3033543. [8] Dijkstra, E. W, 1959. “A Note on Two Problems in Connexion with Graphs”. *Numer Math.* **1(1)**, 269–271. doi: 10.1007/BF01386390

[9] Erdős, P.; Rényi, 1959. A, "On Random Graphs. I". *Publ Math-Debrecen*. 6: 290–297

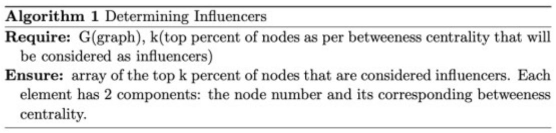
[10] Barabási, A.-L., Albert, R 1999. Emergence of scaling in random networks. *Science*, *286*(5439), 509-512. doi: 10.1007/BF01386390

[11] Li, Pezhou., Wang, Yifei., Zhao, Han., et.al, 2021. On Dyadic Fairness: Exploring and Mitigating Bias in Graph Connections. *International Conference on Learning Representations.*

APPENDIX

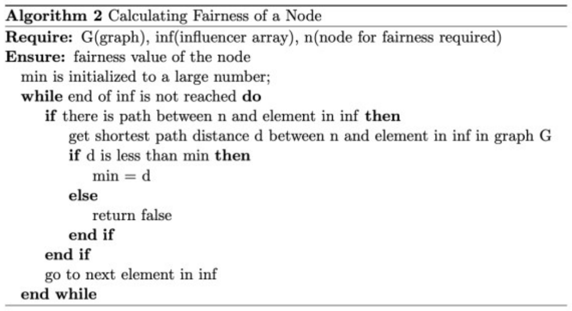
A.1 Pseudocode

1. Determining Influencers



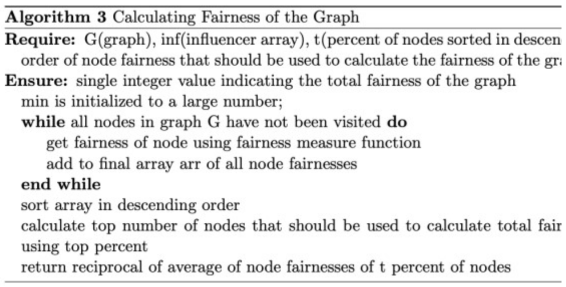
```
Algorithm 1 Determining Influencers
Require: G(graph), k(top percent of nodes as per betweeness centrality that will
  be considered as influencers)
Ensure: array of the top k percent of nodes that are considered influencers. Each
  element has 2 components: the node number and its corresponding betweeness
  centrality.
```

2. Calculating Fairness of a Node



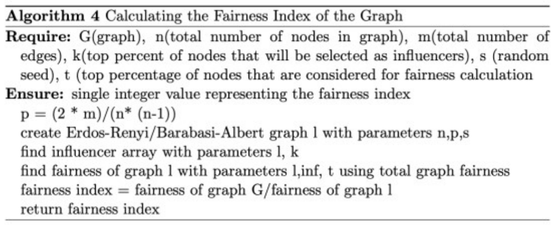
```
Algorithm 2 Calculating Fairness of a Node
Require: G(graph), inf(influencer array), n(node for fairness required)
Ensure: fairness value of the node
  min is initialized to a large number;
  while end of inf is not reached do
      if there is path between n and element in inf then
          get shortest path distance d between n and element in inf in graph G
          if d is less than min then
              min = d
          else
              return false
          end if
      end if
      go to next element in inf
  end while
```

3. Calculating Fairness of Graph



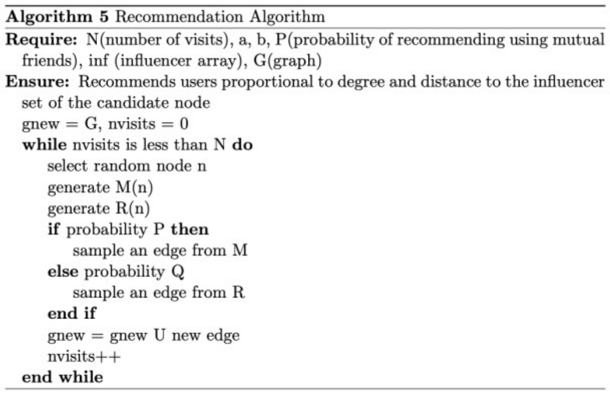
```
Algorithm 3 Calculating Fairness of the Graph
Require: G(graph), inf(influencer array), t(percent of nodes sorted in descen
  order of node fairness that should be used to calculate the fairness of the gr
Ensure: single integer value indicating the total fairness of the graph
  min is initialized to a large number;
  while all nodes in graph G have not been visited do
      get fairness of node using fairness measure function
      add to final array arr of all node fairnesses
  end while
  sort array in descending order
  calculate top number of nodes that should be used to calculate total fair
  using top percent
  return reciprocal of average of node fairnesses of t percent of nodes
```

4. Calculating Fairness Index of Graph

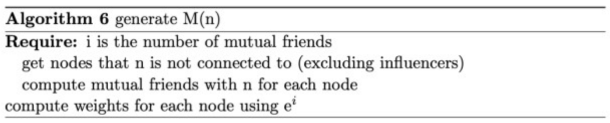


```
Algorithm 4 Calculating the Fairness Index of the Graph
Require: G(graph), n(total number of nodes in graph), m(total number of
  edges), k(top percent of nodes that will be selected as influencers), s (random
  seed), t (top percentage of nodes that are considered for fairness calculation
Ensure: single integer value representing the fairness index
  p = (2 * m)/(n* (n-1))
  create Erdos-Renyi/Barabasi-Albert graph l with parameters n,p,s
  find influencer array with parameters l, k
  find fairness of graph l with parameters l,inf, t using total graph fairness
  fairness index = fairness of graph G/fairness of graph l
  return fairness index
```

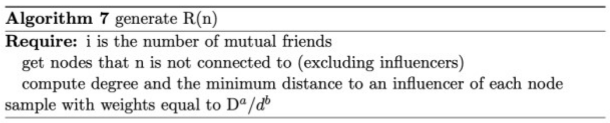
5. Recommendation Algorithm



```
Algorithm 5 Recommendation Algorithm
Require: N(number of visits), a, b, P(probability of recommending using mutual
  friends), inf (influencer array), G(graph)
Ensure: Recommends users proportional to degree and distance to the influencer
  set of the candidate node
  gnew = G, nvisits = 0
  while nvisits is less than N do
      select random node n
      generate M(n)
      generate R(n)
      if probability P then
          sample an edge from M
      else probability Q
          sample an edge from R
      end if
      gnew = gnew U new edge
      nvisits++
  end while
```



```
Algorithm 6 generate M(n)
Require: i is the number of mutual friends
  get nodes that n is not connected to (excluding influencers)
  compute mutual friends with n for each node
compute weights for each node using $e^i$
```



```
Algorithm 7 generate R(n)
Require: i is the number of mutual friends
  get nodes that n is not connected to (excluding influencers)
  compute degree and the minimum distance to an influencer of each node
sample with weights equal to $D^a/d^b$
```

A.2 Reproducibility

2.1 – Data In this paper, 2 data sets were used: a Facebook social circles graph and a Last FM Network. The Facebook social circles graph can be found here: https://snap.stanford.edu/data/egoFacebook.html, and the LastFM network can be found here: https://snap.stanford.edu/data/feather-lastfm-social.html.

2.2 – Code

The link to the code used in this research can be found here: https://colab.research.google.com/drive/1mjletqli0W1oJd0qDc1u8SGLmUYNJ5n_?usp=sharing.