A. Method

The Information Diffusion Model

We employ the Multi-Campaign Independent Cascade Model ($MCICM$) as our diffusion model. This model is commonly used to simulate multiple information dissemination activities simultaneously within a given network G, where $G = (V,E)$. In our experiments, we propagate two different pieces of information: the true news and the fake news.

The seed nodes for the true news and fake news (S_T, S_F) are different and responsible for spreading. The initial state of all other nodes in the network (excluding (S_T, S_F)) are "inactive". Once they are successfully infected by true news or fake news, their states will change accordingly. If a node fails to activate its neighbour, then the same news cannot propagate to this neighbour node again. We assume that there is no opinion reversal during the information propagation process, meaning that once a node's state changes, it remains unchanged and will not be further infected by different news. Additionally, we adopt the positive dominance tie-breaking rule to deal with situations where both types of news attempt to activate a particular node. As a result, when true news and fake news arrive at a node simultaneously, although both news are capable of successfully infecting the node, the node will prioritize believing in true news. In other words, when both true news and fake news have the potential to infect a node simultaneously, the state of the node will be marked as infected by the true news.

In order to simulate a more realistic network environment and acknowledging the variation in the propagation ability of each node (individual), we do not set the propagation probability for each corresponding edge to be the same. We assume that the propagation probabilities for both true news and fake news follow the normal distribution. Specifically, the propagation probability for fake news follows a normal distribution with mean equal to $\frac{N}{E}$ and standard deviation equal to $0.2 * \frac{N}{E}$. For the true news, the propagation probability follows a normal distribution with mean equal to $0.6 * \frac{N}{E}$ and standard deviation equal to $0.2 * \frac{N}{E}$.

Shortest Path Block Algorithm (S-Block)

Algorithm 1 Shortest Median

```
1: **Input:** $G$, $S_F$
2: $R_{short} \leftarrow \phi$
3: for $node_{Fi}$ in $S_F$ do
4:     for $node_i$ in $G$ do
5:         if $node$ not in $S_F$ then
6:             if There is path between $node_{Fi}$ and $node_i$ then
7:                 Append the length of the shortest path for these two nodes into $R_{short}$
8:             end if
9:         end if
10:    end for
11: end for
12: Output: the median of $R_{short}$
```

Firstly, we need to find the shortest path from the seed nodes of the false news to other nodes in the network. This step is to help determine how many steps are needed to build a wall in the subsequent algorithm. We calculated the shortest path length from $node_{Fi}$ in S_F to all other nodes(except S_F) in the network. For instance, if the shortest path length from $node_{Fi}$ to $node_i$ is 3, it means that the shortest path from $node_{Fi}$ to $node_i$ takes 3 steps. Furthermore, we have calculated the median of all these shortest paths length. In the process of information combating, if we can know the fastest infection path of false news propagation in advance, we can interfere with its spread to block its propagation. This part of the algorithm will be explained in detail in subsequent $Algorithm2$ and $Algorithm3$. The core idea of this $Algorithm1$ is the shortest path. We aim to calculate the shortest path from the S_F to other nodes in the network to determine how many steps are needed to infect the remaining nodes through the S_F. The reason to use median of all shortest paths length is that median can more accurately reflect the shortest path situation from $node_{Fi}$ to the majority of the remaining nodes in the network. In large networks, we cannot rule out the extreme situations of shortest path length, nor can we ensure that the distribution of shortest paths is relatively even.

Algorithm 2 Iterated Selection

```
1: Input: $G$, $S_F$, $P_T$, $Step$, $A_{copy}$, $R_{largest}$
2: $R_{prob} \leftarrow \phi$
3: for $node$ in $G$ do
4:     $R_{prob}[node] \leftarrow 0$
5: end for
6: $W_{reduce} \leftarrow \phi$
7: for $node$ in $G$ do
8:     if $node$ not in $S_F$ then:
9:         for $node_W$ in $A_{copy}$ do
10:            $W_{reduce}[(node, node_W)] \leftarrow 0$
11:        end for
12:    end if
13: end for
14: $R_W \leftarrow$ list($A_{copy}$.keys())
15: for $node_F$ in $S_F$ do
16:     Generate a sub-graph according to $G$, $node_F$ and $Step$
17:     for $node$ in $G$ do
18:         if $node$ not belongs to $S_F$, $R_{largest}$, and sub-graph then
19:             $P_{sum} \leftarrow 0$
20:             for $node_L$ in $R_W$ do
21:                 $P_{CTiA} \leftarrow 0$
22:                 if There exists path between $node$ and $node_L$, and the length of path
    $\leq Step$ then
23:                     Find all paths satisfy the above condition
24:                     for Each path do
25:                         Calculate the $P_{sum}$ and $P_{CTiA}$
26:                     end for
27:                 end if
28:                 $W_{reduce}[(node, node_L)] \leftarrow P_{CTiA}$
29:             end for
30:             $R_{prob}[node] \leftarrow R_{prob}[node] + P_{sum}$
31:         end if
32:     end for
33: end for
34: $V \leftarrow$ Sort $R_{prob}$ in descending order
35: Output: $V$, $A_{copy}$, $W_{reduce}$
```

This algorithm serves as an intermediate procedure to calculate three required elements in $Algorithm3$, which are the sorted $R_{prob}(V)$, A_{copy} and W_{reduce}. A_{copy} is an iteratively updated element in each run of the algorithm. W_{reduce} is required because in each step, after selecting a node add to S_T from C_T, we need to subtract the probability of that node can propagate to the node on the wall (R_{Wi}) from the accumulated probability (P_{WiA}) of the R_{Wi}. This can help our algorithm dynamically update the true seed selection criteria to find the optimal solution in every iteration. In this algorithm, we calculated the partial probability ($P_{C_{Ti}A}$) for each pair of nodes (from node in candidate set(C_{Ti}) to the node on the wall(R_{Wi})), which is subsequently added to W_{reduce}. Additionally, we also calculated the sum probability (P_{sum}), which indicates the total expected probability of C_{Ti} can affect R_W and this is treated as the final criteria to select the true seeds (V). In other words, P_{sum} shows the comprehensive potential influence of C_{Ti} on the R_W. In order to determine which paths meet the criteria for calculation of the aforementioned probabilities, we utilised the $Step$ as the condition to ensure that the propagation length is less or equal to the $Step$, thereby identifying all eligible paths. To decide which path satisfies the condition to calculate the above two probabilities, we used the input step as the condition to control the propagation length less or equal to the step as the eligible paths.

Algorithm 3 Shortest Path Block

```
1: **Input:** $G$, $P_F$, $S_F$, $P_T$, $T1$, $T2$, $T3$, $T4$
2: $Step \leftarrow$ round-down value of the median of $R_{short}$ / 2
3: $A \leftarrow \phi$
4: **for** $node$ in $G$ **do**
5:     $A \leftarrow 0$
6: **end for**
7: **for** $node_F$ in $S_F$ **do**
8:     Generate a sub-graph according to $G$, $node_F$ and $Step$
9:     Construct wall $R_W$ according to the sub-graph and $Step$
10:    Calculate the accumulated probability for each node $P_{WiA}$ in $R_W$ according to $S_F$
11: **end for**
12: Remove unused node in $R_W$
13: Select top $T_1$ proportion of nodes in $A$ according to the accumulated probability
14: Select top $T_2$ proportion of nodes in $A$ according to the in-degree of each node
15: $A_{copy} \leftarrow A$.copy()
16: $R_{largest} \leftarrow \phi$
17: Call **Algorithm 2** to acquire $V$, $A_{copy}$, $R_W$
18: Append the first element in $V$ to $R_{largest}$
19: **for** each node in $A_{copy}$ **do**
20:     **if** ($V$, node) in $R_W$ **then** $A_{copy} \leftarrow A_{copy}$ - $R_W$[($V$, node)]
21:     **end if**
22: **end for**
23: **for** each node in $A_{copy}$ **do**
24:     **if** 1 - $A_{copy}$[node] / $A$[node] $> T_3$ **then**
25:         Remove this node from $A_{copy}$
26:     **end if**
27: **end for**
28: **while** 1 - length($A_{copy}$) / length($A$[node]) $< T_4$ **do**
29:     Repeat **line 17 - 27**
30: **end while**
31: **Output:** $R_{largest}$
```

This algorithm is designed to identify the seed nodes of true news from the candidate set that can combat fake news by blocking their spread. Firstly, the algorithm builds a wall R_W, which is composed of nodes that can be reached from $node_{Fi}$ with a length of step. The $step$ is determined by the median of R_{short} which is output by Algorithm1. The number of steps to construct the wall is obtained by dividing the median of R_{short} by 2 and rounding down. Traditionally, true seeds are always designed to directly put in the path of false news propagation (like $Largest\ Infectees$ strategy), more true seeds are needed to build this block wall. However, our methods used only a small amount of true seed nodes to spread to build the wall which is more efficient. In the network, if we set the spread like this, we can utilise true news to protect approximately more than half of the nodes in the network. Taking $node_{Fi}$ as the source node, a sub-graph with the calculated $step$ length is established, and the sum propagation probability (P_{WiA}) of propagating fake news from $node_{Fi}$ to the nodes R_{Wi} on R_W is calculated.

Based on the sorted value of PWiAP$_{WiA} foreachRWiR_{Wi}, selectthetopT1AccordingtotheVvalueofcandi$
$rankednodeischosentobeaddedtotheseednodesoftruenewsST.Thisprocessisrepeateduntilthefollowingt$

First, when $T_3\%$ of P_{WiA} of the node on R_W" is eliminated by S_T's V, and second, when the $T_4\%$ of P_{WiA} of the node on R_W" is eliminated by S_T's V. In summary, this algorithm provides a strategic way to block the spread of fake news by utilising the influence of true news.

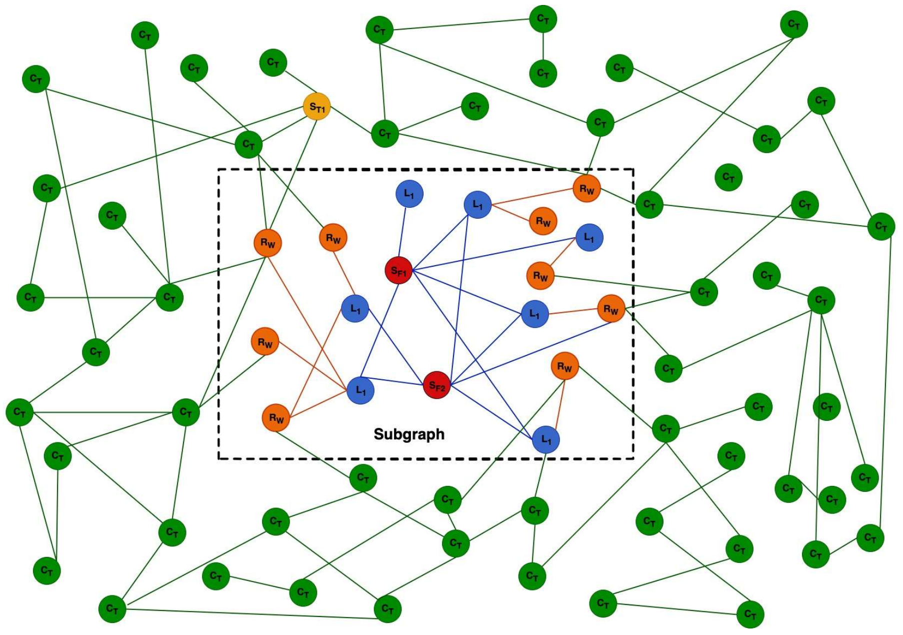

Fig. 1: Shortest-path block (S-block) visualization

Demonstration

This is a sample demonstration of how our method worked on the influence blocking maximisation problem. S_{F1} and S_{F2} represent the seed nodes for propagating the fake news and L_1 represents the first layer of nodes influenced by S_{F1} and S_{F2} , R_W refers to the block wall we designed to limit the fake news spreading, in this case, the second layer of fake news spreading, and ST_1 represents the selected true seed nodes in the candidate set (C_T) to infect the nodes on the wall to block the spread of fake news. It is evident that S_{T1} can infect two R_W after the second step, which is the most in this network structure. Our strategy further considers the propagation probability on these paths to avoid extreme cases of propagation (e.g., very high or low probability on an edge).

When designing the block wall, it is a collection of nodes that could potentially be infected after a certain number of steps. We calculated the shortest path from all S_F to the rest of the nodes, as it represents the minimum number of steps between two nodes, and used the rounded-down average value as the step length to build the wall. This can ensure the effectiveness of preempting the nodes which are useful to interpret the spreading of fake news. Furthermore, because our multi-diffusion model employs the positive dominant strategy which rules that true news always has advantage during the propagation process. Therefore, we want to find the nodes in the candidate set that can leverage this advantage as S_T . By propagating in the opposite direction with a step length less than or equal to the step it takes for fake news propagating to the nodes on the wall, we aim to propagate the true news to the wall to hinder the spread of fake news. The final step is to maximise the influence of the chosen true seed node from the candidate set (C_T) to ensure it affects as many nodes on the wall as possible. We consider two factors during this process. First is the accumulated probability of all fake news reaching the node on the wall. Second is the expected probability that a node in the C_T can affect the nodes on the wall. This ensures that the selected true seed node can exert a significant influence on the nodes on the wall.

B. Data Collection

According to the project description, the client provided a Twitter dataset at the beginning of the project. It was obtained by researchers via the Twitter API. However, due to the sparsity problem in the Twitter dataset. The client asked our team to find other real datasets on the Internet for experiments. After conducting a large number of experiments to test the usability of the network, we used the open source network dataset provided by Stanford University (Jure Leskovec) - Stanford Large Network Dataset Collection (SNAP), and selected some representative networks to test the capabilities of the algorithm as shown in the Table II:

TABLE II: Statistical data for datasets

Networks	**Degree**	**Edges**	**Type**	**Average clustering coefficient**
ego-Facebook	4039	88234	Undirected	0.6055
ca-GrQc	5242	14496	Undirected	0.5296
lastfm-asia	7624	27806	Undirected	\
wiki-vote	7115	103689	Directed	0.1409
email-Eu-Core	1005	25571	Directed	0.3994

Ego-Facebook [40] was collected from survey participants using this Facebook app and consists of 'circles' (or 'friends lists') from Facebook. Facebook data includes node features (profiles), circles, and ego networks. It includes 4039 users and 193 social circles, these social circles include college students, sports teams, etc.

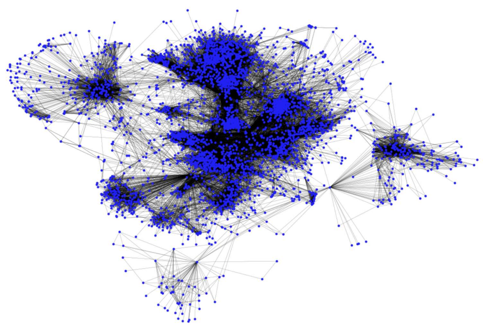
Fig. 2: ego-facebook dataset visualization

For ca-GrQc (Collaboration network of Arxiv General Relativity) [41], author i is undirectedly connected to author j if they co-authored the same paper on general relativity and quantum cosmology. If the paper is co-authored by k authors, this generates a completely connected (sub) graph on k nodes. It is a dataset of scientific papers on general relativity and quantum cosmology with 5242 nodes and 14496 edges. The data covers papers in the period from January 1993 to April 2003 (124 months). It begins within a few months of the inception of the arXiv and thus represents essentially the complete history of its GR-QC section.

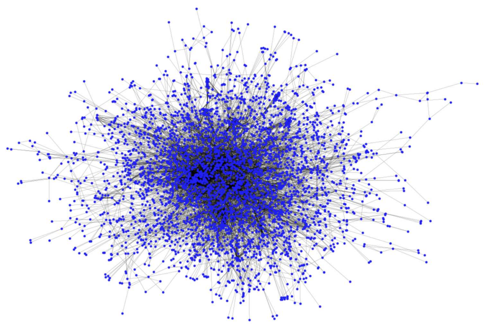

Fig. 3: ca-GrQc dataset visualization

LastFM Asia Social Network [42] was collected from the public API in March 2020, which contains LastFM users from Asian countries. Like the Facebook dataset, the node represents each user, and the edge represents their following relationship. It is an undirected graph that has 7624 nodes and 27806 edges.

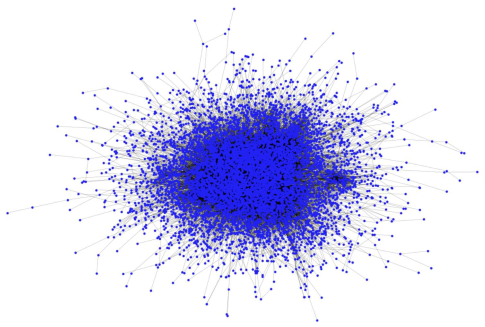

Fig. 4: lastfm asia dataset visualization

The email-Eu-Core [41] dataset is a network formed of Universitat Rovira i Virgili email data. It includes relationships with professors, students, administrators, researchers, etc. In this dataset, each node is an email address. If person v has sent an email to person t, then an edge will be formed. In this network, there are a total of 1005 nodes and 25571 edges. Notably, There are a large number of cycles, as shown in the figure below. We remove nodes if the node has a cycle.

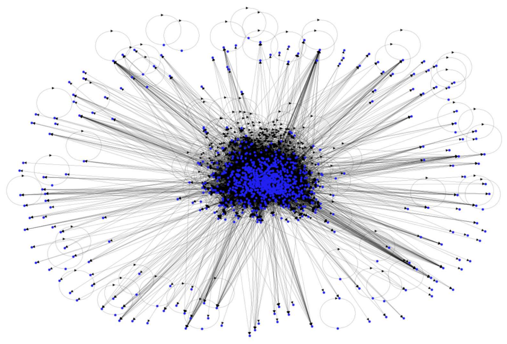

Fig. 5: Email-Eu-Core dataset visualization

Wiki-vote [43] is a database based on Wikipedia users and voters. Nodes in the network represent Wikipedia users and edges represent voting relationships between users. There are 7115 nodes and 103689 edges in total.

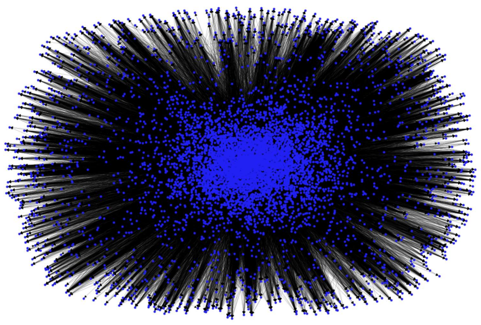

Fig. 6: Wiki-vote dataset visualization

C. Data Analysis

In data analysis, centrality and shortest path analysis were applied. Centrality analysis is used to quickly determine the importance and influence of each node in the network.

Through statistic analysis on ego-facebook, ca-GrQc and lastfm-asia, the maximum degree is 1045, 81 and 216 respectively. The minimum degree is 1 for all datasets.

The shortest path indicates the degree of connectivity between nodes in the network. As the Block algorithm is based on the shortest path, investigating the shortest path frequency of different networks helps in analysing the algorithm's advantages and disadvantages.

Observing the results of the shortest path length frequency network in Figure 7, it can be found that in the three undirected graph data sets, the frequency of the shortest path is approximately normally distributed. The average shortest path is 4 and 5, which has a frequency of 35% and 34% respectively. In the ca-GrQc and lastfm-asia datasets, there are nodes that have more than 10 shortest paths length.

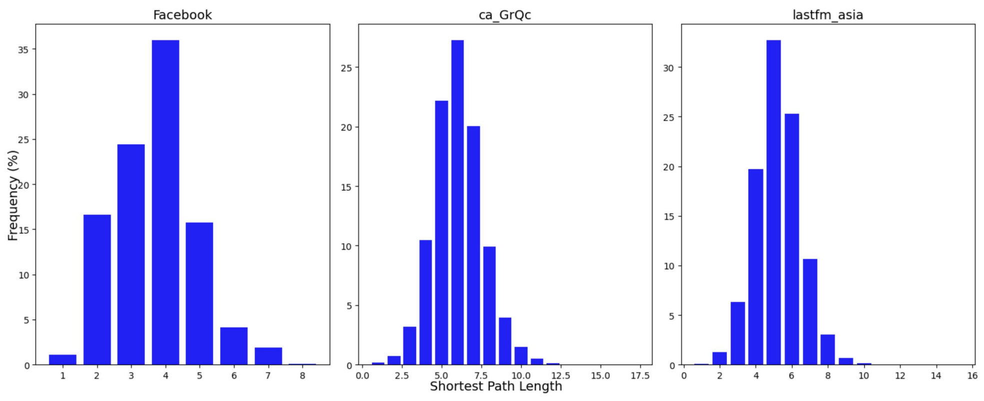

Fig. 7: The frequency of shortest path length in each dataset.

To analyse the difference between the performance of the algorithm in the undirected graph and the directed graph, we plot a scatter diagram of the out-degree minus the in-degree of each node in Figure 8, for the directed graph mail-Eu-Core and Wiki-Vote data set. When the out-degree and in-degree of nodes are equal, their difference is 0. However, there is a large difference between the out-degree and in-degree of nodes in these two datasets, with the largest difference being as high as 800.

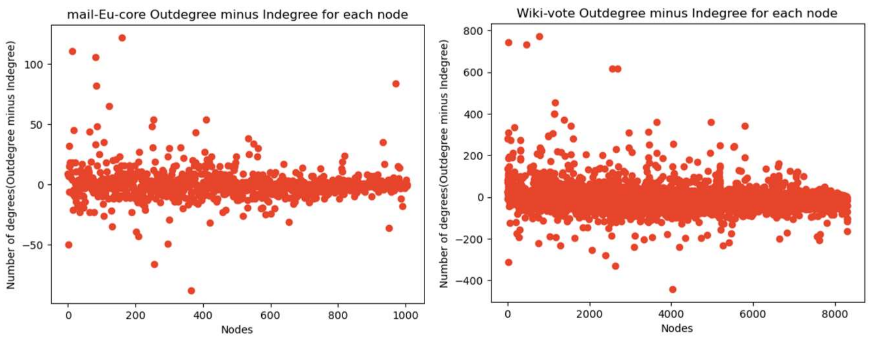

Fig. 8: Outdegree minus Indegree graph for the directed network. the x-axis represents all nodes in the dataset, while the y-axis represents the Outdegree minus Indegree for each node.