МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н.Каразіна

Факультет математики і інформатики

Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему

”Дослідження боротьби за вплив на суспільну думку за допомогою моделювання мережевих спільнот та кластерного аналізу”

Виконав: студент 2 курсу, групи МФ-61

спеціальність 122 «Комп’ютерні науки»

освітньо-наукова програма «Інформатика» Колмогоров Д. А.

Керівник Дейнега О. А.

Рецензент_____
__
(прізвище та ініціали)

Харків – 2025 року

ЗМІСТ

ВСТУП

1.1 Формулювання мети роботи, задач та обґрунтування актуальності теми

У інформаційному середовищі, що утворилося у сучасному світі, соціальні мережі є ключовим каналом розповсюдження новин та впливу на свідомість громадян. Суспільна думка є дуже цінним ресурсом, тому сьогодні зростає інформаційна конкуренція між різноманітними політичними силами, державами загалом, бізнесу. Засоби масової інформації створюють нові умови для пропаганди, поширення дезінформації, нав’язування різних наративів тощо.

За даними дослідження, проведеного Pew Research Center [1], станом на 2024-ий рік 54% дорослого населення Сполучених Штатів Америки отримують новини із соціальних мереж. Серед них найбільш розповсюдженими є такі соціальні мережі, як Facebook [2], X (кол. Twitter) [3], Instagram [4], TikTok [5], YouTube [6]. Цей показник є дещо вище, ніж за попередні роки, що вказує нам на те, що традиційні ЗМІ втрачають свій вплив на думку громадян, боротьба за яку все більше переходить у соціальні мережі. Такі процеси не є виключенням для Сполучених Штатів Америки: наприклад, в України соціальні мережі є джерелом інформації для багатьох категорій населення. Наразі соціальні мережі відіграють значну роль у формуванні думки громадян під час воєнного протистояння.

Цифровий інформаційний простір, на відміну від традиційних ЗМІ (тут мається на увазі газети, редакції та ін.), має мережеву природу, тобто люди формують різноманітні спільноти на основі таких факторів, як соціальні зв’язки, спільні погляди, інтереси та ін. Такі спільноти дуже часто є інформаційними кластерами – групами, де певні погляди лише посилюються. Такі спільноти є дуже вразливими до поширення

дезінформації, фейків, пропагандистських повідомлень. Велики спільноти є цільовими об’єктами для інформаційних кампаній, що розробляються, враховуючи специфіку певних спільнот.

Через це різко зросла необхідність у розробці інструментів для виявлення таких спільнот, аналізу поширення інформації та вплив інформаційного середовища на соціальну думку людини. Моделювання мереж, поєднане з кластерним аналізом, дозволяє будувати графи, що імітують інформаційне середовище, знаходити ключови вузли впливу, знаходити різноматнітні закономірності щодо розповсюдження дезінформації.

Метою даної роботи є дослідження боротьби за вплив на суспільну думку шляхом моделювання мережевих спільнот і застосування кластерного аналізу. У фокусі дослідження – побудова моделей та програмного інтерфейсу, що дозволяють:

- виявляти структури впливу в соціальних мережах;
- ідентифікувати кластери споживачів цільової інформації;
- аналізувати канали та ефективність поширення контенту;
- прогнозувати результати інформаційних кампаній.

Дослідження має не лише наукову цінність, а й прикладну – в рамках дослідження поставлена задача розробити програмний інтерфейс із можливостями візуалізації результатів, оцінки експериментів тощо.

Дане дослідження є логічним продовженням роботи “Моделювання боротьби за вплив на суспільну думку у мережевих спільнотах” (Колмогоров Д. А., Жолткевич Г. М.) [7] та включає в себе модернізацію запропонованого раніше методу моделювання, а також поліпшення

програмного інтерфейсу, щоб дати можливість іншим дослідникам швидко проводити необхідні експерименти за мінімальних змін у програмах.

Мета роботи: Розробити зручний програмний інтерфейс для проведення експериментів з моделювання розповсюдження інформації у мережевих спільнотах. Провести декілька експериментів. Представити результати експериментів у якості візуалізації, метрик кластеризації та висновків.

Завдання:

1. Ознайомитись з іншими середовищами для моделювання боротьби за суспільну думку.
2. Ознайомитись з різноманітними дослідженнями у сфері боротьби за суспільну думку.
3. Розробити зручний програмний інтерфейс із можливістю проведення експериментів, візуалізації результатів та збору статистики.
4. Провести декілька експериментів за допомогою розробленого програмного інтерфейсу.
5. Візуалізувати результати експериментів та представити їх у вигляді малюнків та таблиць.
6. Провести аналіз можливих покращень програмного інтерфейсу та можливих шляхів модернізації моделей.

1.2 Стислий огляд відомих результатів в області дослідження

У сфері дослідження інформаційного впливу існує низка інструментів та платформ, що дозволяють моделювати соціальні взаємодії, аналізувати мережі та оцінювати ефективність кампаній впливу. Найбільш релевантними до теми дослідження є такі інструменти, як:

- NetLogo [8]
- GAMA Platform [9]
- ORA (Organizational Risk Analyzer) [10]

NetLogo – це агентне середовище (схожа на окрему мову програмування) для створення моделей, яке дозволяє симулювати соціальні, біологічні, економічні й політичні процеси.

Це середовище може бути використане у дослідженні боротьби за суспільну думку наступним чином:

- моделювання поширення інформації серед групи агентів;
- впровадження сценаріїв впливу, дезінформації, протидії тощо;
- тестування стратегії маніпуляції громадською думкою.

Варто зазначити, що цей інструмент має декілька суттєвих недоліків, такі як:

- Затруднена інтеграція з даними користувачів;
- Низька масштабованість – дуже важко проводити експерименти з великими мережами, що мають 20000 або більше агентів;
- Обмежена візуалізація – доступні лише 2D-сцени, що у окремих випадках унеможливлює отримання релевантних висновків дослідження.

GAMA – це платформа для моделювання складних соціально-просторових систем, у якій підтримується як агентне моделювання, так і робота з даними користувачів. Побудована на основі мови GAML [11]. Ця платформа має дуже широкий функціонал, що дозволяє дуже детально виконувати моделювання електронної активності, дезінформації, інформаційних атак;

Серед недоліків цієї платформи можна виділити:

- складність мови GAML – ця мова має складний синтаксис;
- дуже складна інтеграція з різноманітними програмними середовищами, що утруднює розробку власних додатків.

ORA – це спеціалізоване програмне забезпечення, що було розроблено у лабораторії CASOS [12] (Carnegie Mellon University [13]) для аналізу динаміки соціальних, організаційних та інформаційних мереж, з особливим акцентом на інформаційну безпеку та стійкість мережі до маніпуляцій. ORA потребує ліцензії, тобто є закритим програмним забезпеченням. Через це ORA використовується здебільшого у великих організаціях та має дуже високий поріг входу, на відміну від інших подібних інструментів.

За допомогою описаних вище інструментів можливо проводити різноманітний аналіз мереж, застосовуючи сучасні підходи, такі як:

- Соціально-мережевий аналіз (Social Network Analysis, SNA);
- Моделювання динаміки інформаційного поширення;
- Когнітивне та поведінкове моделювання.

1.3 Відомості про одержані результати та їх новизна

У рамках роботи було розроблено програмний інтерфейс для моделювання боротьби за суспільну думку на основі мережевих спільнот за допомогою мови програмування Python 3.11 [14] та наступних бібліотек:

- matplotlib [15]
- networkx [16]
- numpy [17]
- scipy [18]

Програмний інтерфейс реалізовано у якості Python бібліотеки, тобто інші дослідники мають налаштовувати експерименти за допомогою мови програмування Python.

Було проведено декілька експериментів для мереж різного розміру та способу генерування самої мережі. Їх результати представлені у якості висновків, малюнків та таблиць.

Був проведений аналіз можливих поліпшень розробленого програмного інтерфейсу та запропонованого методу моделювання. Можливі модернізації також були описані у висновках.

2. ОСНОВНА ЧАСТИНА

2.1 Постановка задачі

Задачею роботи є розробка програмного інтерфейсу, що дозволяє виконувати різноманітні дослідження методів боротьби за суспільну думку. Другою задачею є проведення експериментів за допомогою розробленого програмного інтерфейсу, згідно із запропонованим методом, для дослідження впливу засобів масової інформації на суспільну думку.

Сьогодні інформація та її розповсюдження є темою багатьох досліджень, адже формування суспільної думки є дуже важливим у багатьох сферах, таких як політика, екологія, бізнес та ін.

Розробка програмного інтерфейсу включає в себе:

- визначення недоліків у програмному інтерфейсі, запропонованому у попередньому дослідженні [7];
- визначення функціоналу програмного інтерфейсу;
- визначення програмного середовища та бібліотек, що будуть використані для реалізації;
- визначення подальших кроків для розширення функціоналу та оптимізації проведення досліджень.

Проведення дослідження включає в себе:

- визначення основних тез, що були використані у попередньому дослідженні;
- опис змін, що були реалізовані у роботі;
- опис проведених експериментів;

- представлення результатів;
- опис можливих покращень.

У цій роботі пропонуються наступні терміни:

- актор – вершина графу, що є представленням конкретної людини або засобу масової інформації;
- зв’язок – ребро графу, що є представленням можливого діалогу між акторами (або ознайомлення із матеріалами ЗМІ);
- мережа G – граф, що представлений множиною акторів (вершин) N та множиною зв’язків E;
- думка актора – вектор $W \in \mathbb{R}^M$, що є чисельним представленням думки актора щодо певних питань (або ангажованість засобу масової інформації щодо певних питань);
- діалог – процес обміну інформацією між акторами (є представленням зміни думки під час діалогу з іншими людьми або під час ознайомлення із матеріалами певного засобу масової інформації). Під час діалогу думки акторів зазнають змін;
- епоха – раунд моделювання, під час якого можуть бути задіяні зв’язки між акторами один раз;
- вірогідність зв’язку – число, що представляє собою вірогідність (що має значення у діапазоні $[0, 1]$) діалогу між певними двома акторами під час однієї епохи моделювання;
- середня думка мережі W_m – середнє значення думки акторів, що не є засобами масової інформації.

2.2 Огляд існуючого рішення та його недоліків

У минулому дослідженні було запропоновано програмний інтерфейс для проведення експериментів, що будувався на спеціально розробленій

для вирішення поставленої проблеми реалізації графу, що мала свої недоліки, такі як:

- майже повна відсутність функціоналу генерації графу (для генерації більшості моделей використовувалася бібліотека networkx);
- відсутність легкої конфігурації експерименту (користувач не міг використовувати інші алгоритми обміну думками та керувати процесом моделювання);
- відсутність конфігурації для візуалізації результатів.

Ця робота включає в себе виправлення цих недоліків та пропонує користувачу (досліднику) більш ширший функціонал для проведення експериментів.

Метод дослідження, що був запропонований у минулому дослідженні [7], мав наступні ключові тези:

- вірогідність зв’язку між акторами рахувалася як косинусна відстань або розходження Кульбака-Лейблера (відносна ентропія);
- усі актори під час однієї епохи моделювання змінюють свою думку;
- кожен актор мав “силу думки” – коефіцієнт, що визначав, наскільки сильно певний актор може змінювати думку інших акторів;
- у якості представлення результатів експерименту використовувався малюнок та середня думка мережі;

Основним недоліком методу, запропонованого у минулій роботі, є той факт, що всі актори змінюють свою думку під час епохи моделювання, що не є реалістичним, адже у сучасному світі існують дуже багато джерел інформації, що пропагандують певну точку зору без її зміни (заангажовані ЗМІ, боти, тощо).

2.3. Огляд використаного програмного середовища

Для реалізації програмного інтерфейсу для моделювання процесів зміни суспільної думки була обрана мова програмування Python через те, що вона має наступні переваги:

- Python є стандартом в сфері наукових досліджень, соціального аналізу, штучного інтелекту, обробки мови тощо;
- Велика кількість наукових бібліотек та публікацій базується саме на Python, що полегшує порівняння та відтворення результатів;
- Завдяки інтуїтивному синтаксису Python дозволяє зосередитися на логіці моделі, а не на деталях реалізації, що особливо актуально у нашому випадку, коли досліднику потрібно швидко провести певний експеримент та внести зміни відповідно до результатів експериментів;
- Python має дуже багато наукових бібліотек, зокрема ті, що були використані при реалізації програмного інтерфейсу, що пропонує ця робота – numpy, networkx, matplotlib та ін.

Варто зазначити, що у контексті наукових досліджень Python має вагомий недолік, а саме швидкодію. Через те, що мова програмування Python має динамічну типізацію (тобто певна зміна під час свого життєвого циклу може змінювати свій тип) та є інтерпретованою мовою програмування (тобто код спочатку конвертується у байт-код, а саме він далі компілюється у машинний код), виконання складних розрахунків є дещо ускладненим. Багато наукових бібліотек нівелюють цей недолік за допомогою використання більш швидкодіючих та низькорівневих мов програмування, такі як Fortran [19], C++ [20] або Rust [21]. Так, наприклад NumPy використовує Fortran як основу для виконання розрахунків та

пропонує програмний інтерфейс на Python для використання користувачами.

При розробці програмного інтерфейсу для моделювання були використані такі наукові бібліотеки, як:

- numpy;
- networkx;
- matplotlib;
- scikit-learn.

Numpy це бібліотека для виконання розрахунків, що має значно більшу швидкодію, ніж використання звичайних структур, реалізованих на мові програмування Python. Бібліотека пропонує дуже зручний програмний інтерфейс для виконання розрахунків для багатовимірних масивів та векторів. У цьому дослідженні ця бібліотека була використана задля прискорення розрахунків під час моделювання та можливості швидкого внесення змін у процес моделювання (наприклад, зміна розмірності вектора, що представляє собою думку актора).

Networkx – бібліотека для роботи з графами. У цій роботі networkx була використана як інструмент візуалізації (тобто усі розрахунки виконуються за допомогою своєї реалізації класу графа, адже networkx має суттєвий недолік у вигляді масштабованості, а задля представлення результатів експериментів у вигляді малюнків використовується програмний інтерфейс, що пропонує networkx).

Matplotlib – бібліотека, що була створена для різноманітних видів візуалізації (графіки, гістограми, кругові діаграми тощо). Саме ця бібліотека використовується у бібліотеці networkx для візуалізації графів. У цій роботі бібліотека matplotlib була використана задля розширення можливостей візуалізації графів.

Scikit-learn – бібліотека, що містить в собі дуже багато алгоритмів машинного навчання, та додаткових інструментів. У рамках цієї роботи ця

бібліотека використовується для кластеризації (використаний алгоритм кластеризації DBSCAN [33]) та оцінки кластеризації.

У якості середовища для реалізації програмного інтерфейсу була обрана IDE PyCharm [22], що має дуже багато переваг для розробки на мові програмування Python, особливо у контексті виконання наукових досліджень, а саме:

- підтримка різноманітних наукових бібліотек, такі як numpy, pandas [23], scikit-learn [24], scipy [18], plotly [25], seaborn [26] та ін.;
- забезпечення зручною навігацією проектом, автозаповненням, рефакторингом, підказками типів і перевіркою помилок у реальному часі;
- дуже зручний у використанні інструмент налагодження (Debugger), що дозволяє відстежувати процес виконання програми покроково;
- підтримка Jupyter Notebook [27], що є дуже популярним інструментом у науковій спільноті;
- PyCharm Pro надає інтеграції з різноманітними мовами програмування, такі як JavaScript [28], R [29] та ін., а також із інструментами для розгортання моделей, такі як Docker [30].

2.4. Генерація мережі та ініціалізація акторів

У якості методу генерації мереж була обрана обрана модель Барабаші-Альберта [31] (адже вона чудово відображає природу інтернет-спільнот та засобів масової інформації), що будується на покроковому додаванні нових вершин до існуючого графу із певною вірогідністю, що розраховується як:

$$P_i = \frac{k_i}{\sum_N k_j}$$

У цій формулі вірогідність появлення зв’язку (ребра) між новою вершиною та існуючою виражена як відношення степені існуючої вершини (кількість ребер, що виходять з цієї вершини) до суми степенів усіх вершин у мережі.

Модель Барабаші-Альберта є безмасштабною мережею, тобто відношення вершин (вузлів) графу мережі, що мають k зв'язків (граней), до числа усіх вершин для великих значень k визначається як:

$$P(k) \sim k^{-\lambda}$$

λ у безмасштабних мережах є сталою та зазвичай має значення у діапазоні $[2,\ 3]$, хоча інколи значення λ може не належати цьому інтервалу.

Модель Барабаші-Альберта підпорядковується цьому закону та має наступний степеневий розподіл:

$$P(k) = k^{-3}$$

Також цікава властивість моделі Барабаші-Альберта щодо середнього шляху між двома випадковими вершинами, залежність середнього шляху до розміру мережі є логарифмічною та виглядає наступним чином:

$$l = \frac{\ln N}{\ln \ln N}$$

Середню довжину шляху можна інтерпретувати як середню кількість кроків, якими передається певна інформація між довільними акторами у мережі.

Модель Барабаші-Альберта додає нові вершини у вже існуючий граф, що може бути довільним (варто зазначити, що вершина, що має ступінь 0, після додавання нових вершин, гарантовано не буде мати жодних ребер).

Генерація мережі за допомогою моделі Барабаші-Альберта реалізована у бібліотеці networkx [32], проте не є оптимальною. У розробленому програмному інтерфейсі така генерація була оптимізована за рахунок зберігання степеней вершин у хеш-таблиці. Таким чином, операція розрахунку ступеня вершини (що є основною операцією під час генерації мережі за моделлю Барабаші-Альберта) була замінена на операцію отримання елементу за ключем у хеш-таблиці, що є значно дешевшою операцією.

У цій роботі генерація за допомогою моделі Барабаші-Альберта починається з повного графу з N_0 вершинами. Такий граф перед початком генерації має наступну кількість ребер:

$$N_E = \frac{N_0 \cdot (N_0 - 1)}{2}$$

Таким чином, вірогідність появи зв’язку (ребра) між першою новою вершиною та кожною існуючою є:

$$P_0 = \frac{N_0 - 1}{N_0 \cdot (N_0 - 1)} = \frac{1}{N_0}$$

Модель Барабаші-Альберта дуже часто має наступну властивість: у мережі існують вершини-хаби, що мають значно вищі ступені, у порівнянні з іншими.

Нижче наведені діаграмі, що показують розподіл ступеней вершин у мережі, що має 100 вершин, для різних значень N_0 :

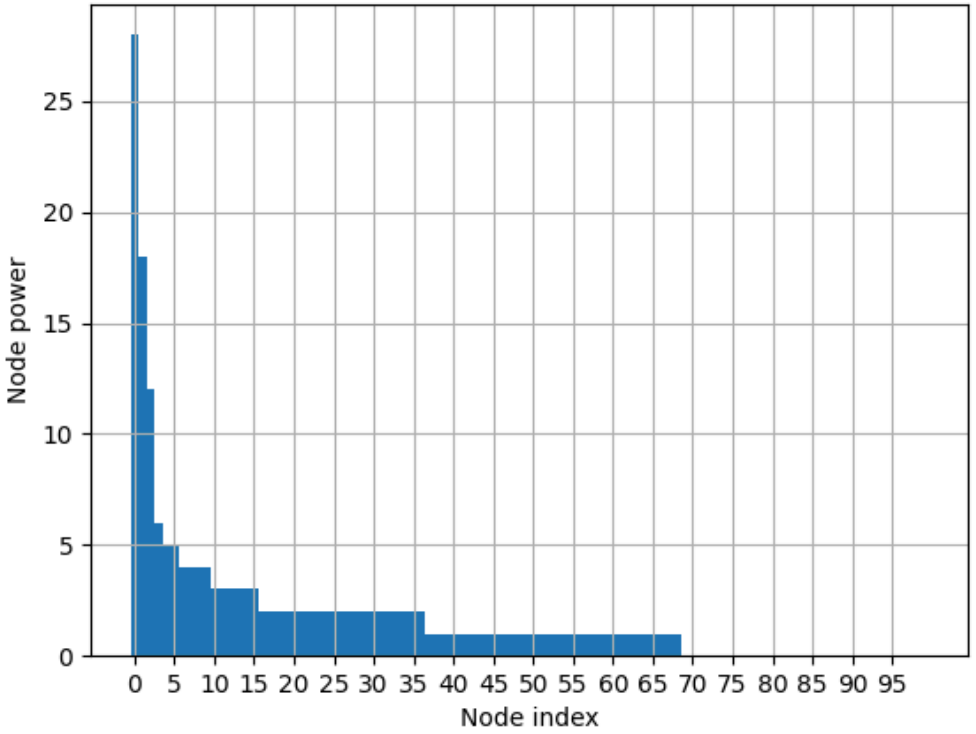

Рис 1. Ступені вершин у мережі при значенні $N_0 = 2$

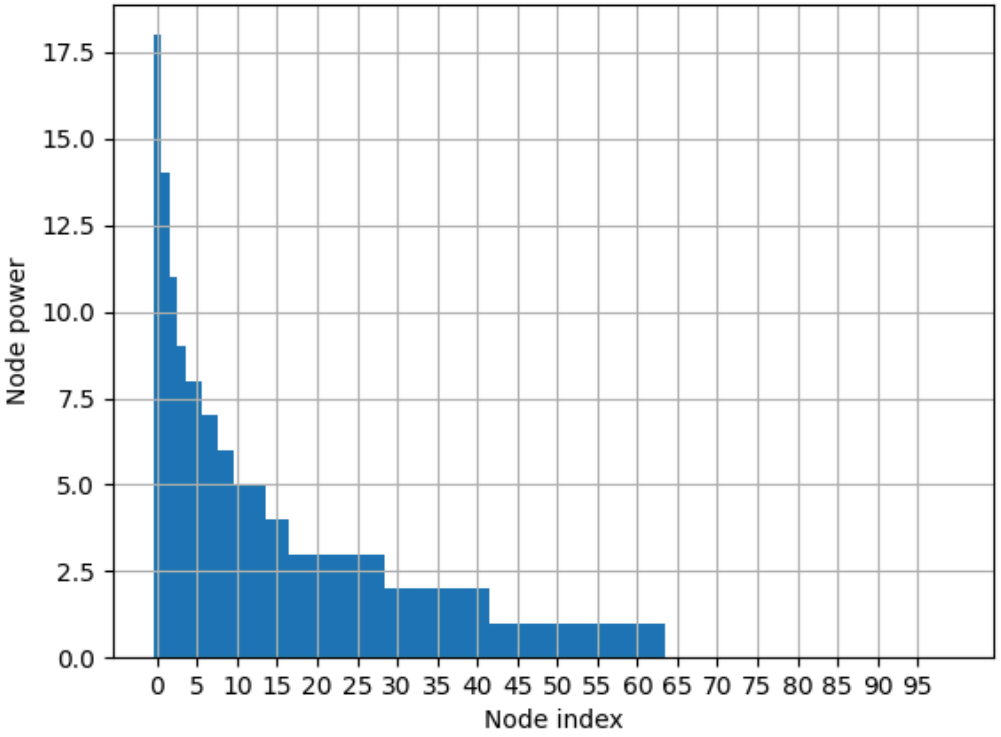

Рис 2. Ступені вершин у мережі при значенні $N_0 = 3$

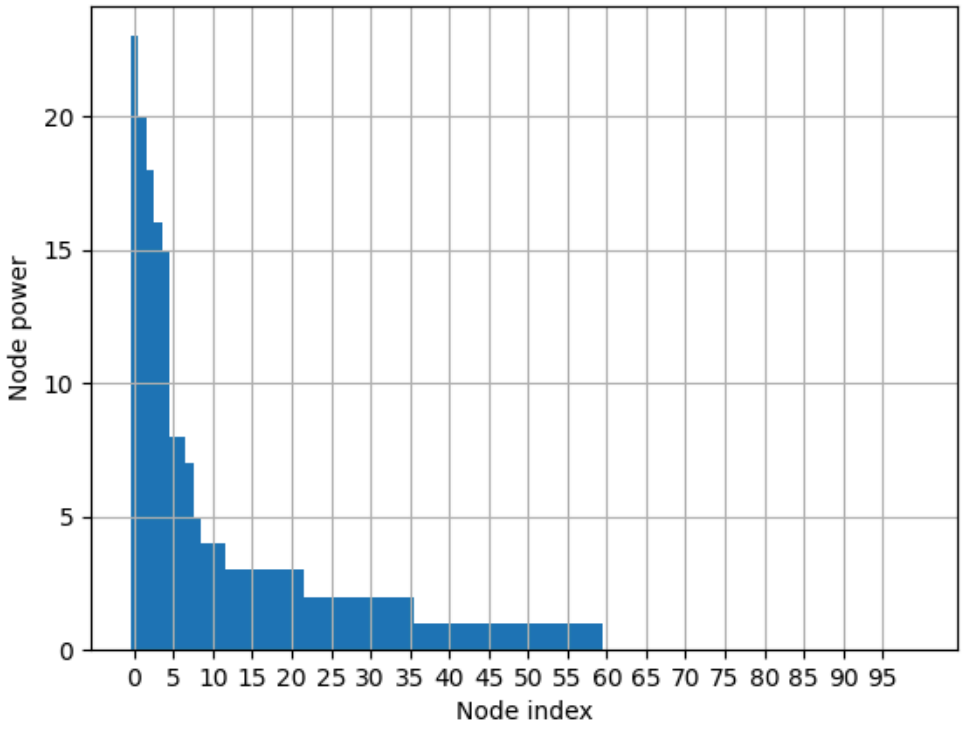

Рис 3. Ступені вершин у мережі при значенні $N_0 = 5$

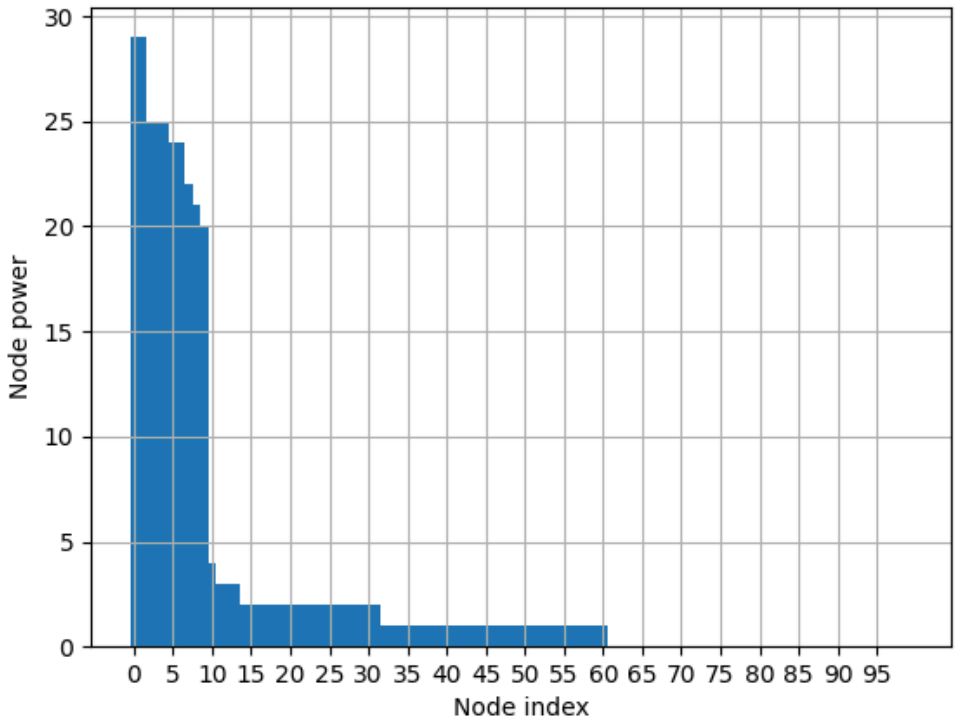

Рис 4. Ступені вершин у мережі при значенні $N_0 = 10$

У випадку генерації мережі, починаючи з повного графу з N_0 вершинами, вірогідно саме вони будуть вершинами-хабами у результуючій мережі. Це припущення використовується при ініціалізації акторів – процесі перед моделюванням, що визначає думки акторів, як вектори, визначає, чи є певний актор засобом масової інформації у мережі та встановлює вірогідність зв'язку між акторами протягом однієї епохи моделювання. Тобто при генерації мережі з $N_0 = 2$, алгоритм ініціалізації обере двох акторів із найвищим ступенем як засоби масової інформації.

Думка актора виражена вектором $W \in \mathbb{R}^M$. Під час ініціалізації думка актора стає виражена вектором із певною розмірністю та значеннями з інтервалу $[0, 1]$. У реалізованому програмному інтерфейсі є опція ініціалізації думки акторів як випадкових одиничних (unit) векторів, тобто :

$$\sqrt{\sum W_i^2} = 1$$

Такі одиничні вектори можуть бути інтерпретовані як “напрямок думки”.

2.5 Моделювання зміни суспільної думки

У минулому дослідженні були введені наступні функції:

- сила думки актора;
- псевдо-сигмоідальна функція;
- функція діалогу.

Сила думки актора – певний коефіцієнт, що виражає його впевненість щодо якогось певного питання, або іншими слова його радикальність у тому чи іншому питанні. Це поняття було введене задля більш точного моделювання боротьби за суспільну думку, адже більш радикальні люди рідше змінюють свою думку, аніж “центристи”. Функція, що відображає силу думки актора, визначається наступним способом:

$$O(x)\colon [0,\ 1] \rightarrow [0,\ 1] = 2 \cdot |0.5 - x|$$

Ця функція повністю відображає наше припущення щодо зміни думки у радикально-налаштованих людей та людей, що мають неоднозначну точку зору щодо певного питання, що виражається у наступних її властивостях:

$$O(0) = 1$$

$$O(1) = 1$$

$$O(0.5) = 0$$

Нижче наведений графік функції, що є відображає силу думки певного актора у мережі:

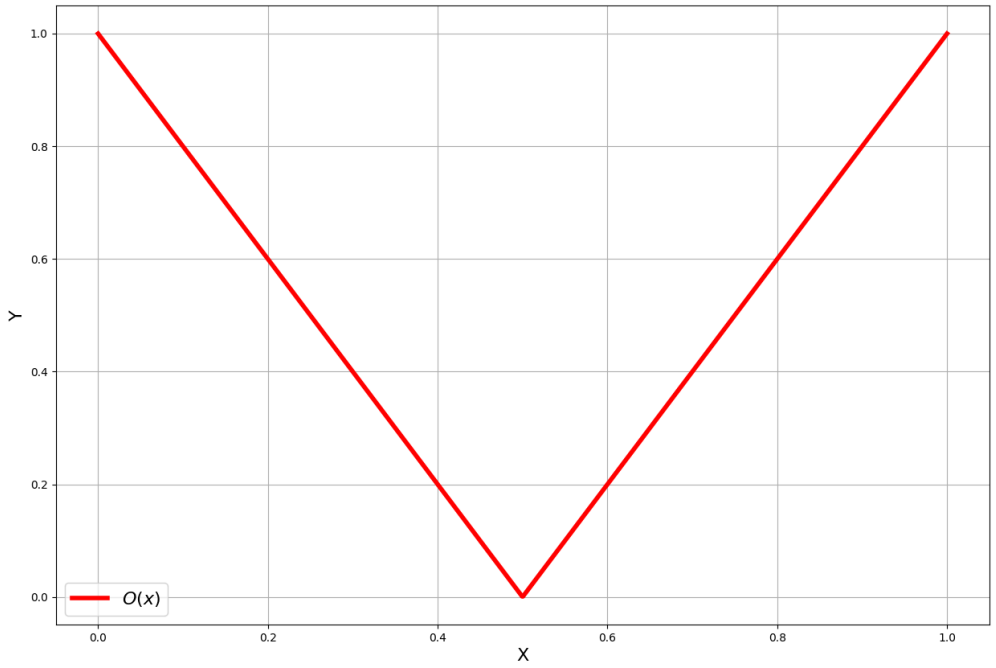

Рис. 5. Графік функції сили думки актора

Реалізація функції сили думки актора за допомогою мови програмування Python та бібліотеки numpy виглядає наступним чином:

```
def O(x: np.ndarray) -> np.ndarray:
    return 2 * np.abs(0.5 - x)
```

Псевдо-сигмоїдальна функція – функція, що відображає коефіцієнт впливу зовнішніх джерел інформації на думку певного актора в залежності від сили його думки. Назва була обрана через схожість цієї функції з сигмоїдальною функціює, що широко використовується у машиному навчані. Ця функція зберігає всі властивості класу сигмоїдальних функцій, є неперервною та визначена наступним чином:

$$P\ (o)\colon [0,\ 1] \rightarrow [0,\ 1] \ = \ 2\ \cdot o^{2},\ o\ <\ 0.5 \mid 1\ -\ 2(1\ -\ o)^{2},\ o \geq 0.5$$

Аргументом цієї функції у процесі моделювання є зворотнє значення сили думки актора (тобто, $1\ -\ O(x)$).

Нижче наведений графік псевдо-сигмоїдальної функції у залежності від зворотної сили думки актора o:

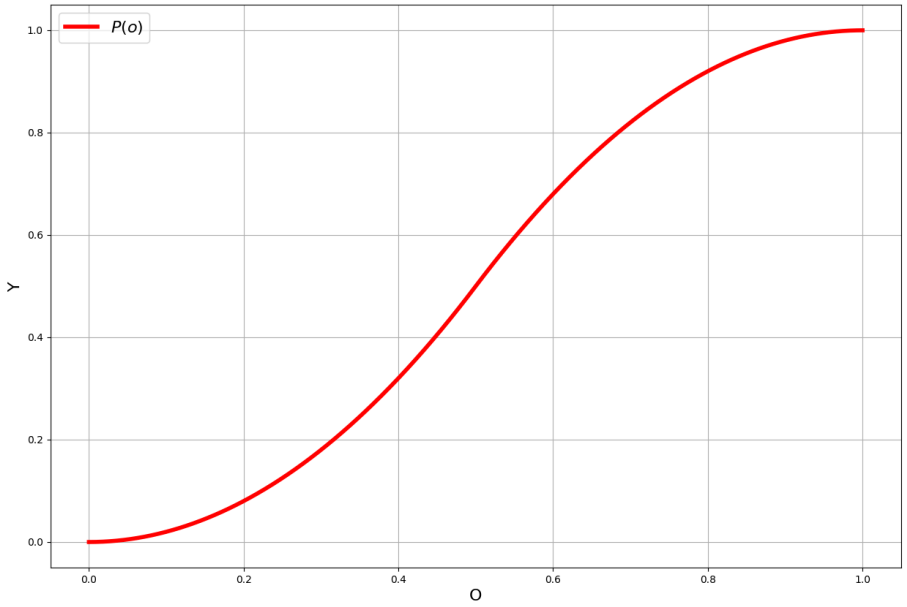

Рис. 6. Графік псевдосигмоїдальної функції у залежності від зворотної сили думки актора

Також наведено графік псевдо-сигмоїдальної функції в залежності від думки актора:

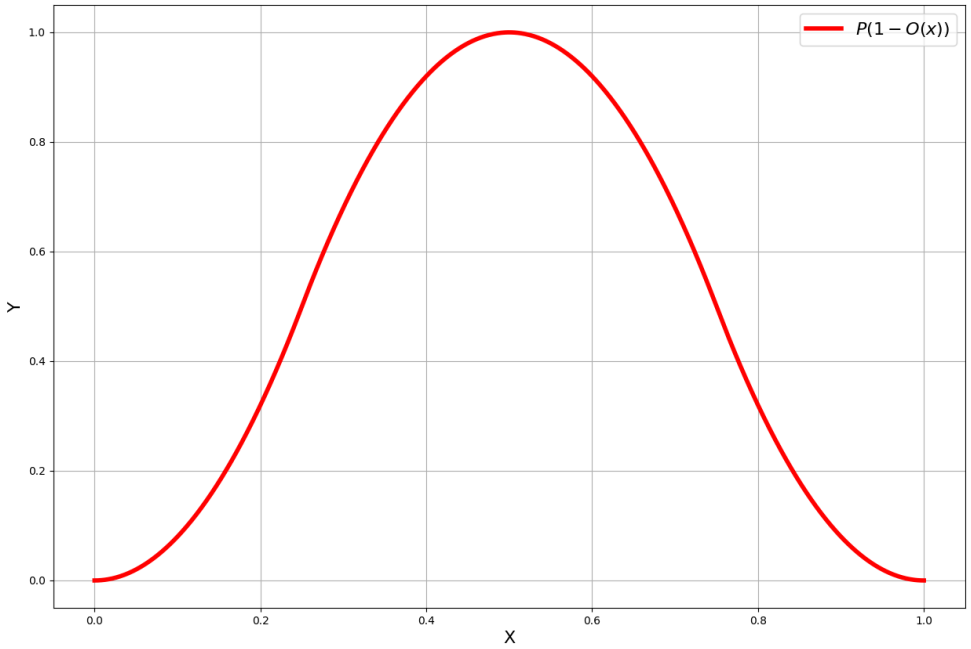

Рис. 7. Графік псевдо-сигмоїдальної фукнції у залежності від думки актора

Імплементація псевдо-сигмоїдальної функції за допомогою мови програмування Python та бібліотеки numpy виглядає наступним чином:

```
def P(x: np.ndarray) -> np.ndarray:
    opp_o: np.ndarray = 1 - O(x)

    return np.where(
        opp_o < 0.5,
        2 * opp_o**2,
        1 - 2 * (1 - opp_o)**2
    )
```

У минулому дослідженні вірогідність зв’язку між акторами протягом однієї епохи моделювання була виражена через косинусну

відстань або розходження Кульбака-Лейблера, використовуючи вектори думок акторів. У цій роботі пропонується замінити вірогідність зв’язку між акторами на константу, що визначається на етапі ініціалізації. Ця вірогідність виражена наступним чином: якщо один з акторів є засобом масової інформації (він має топ-K ступінь у мережі), то такая вірогідність є випадковим числом з інтервалу [0, 1], помножене на певний коефіцієнт (запропонований програмний інтерфейс має опцію налаштування цього коефіцієнта, за замовчуванням у цій роботі він дорівнює 4), та приведене до інтервалу [0, 1]. Реалізація розрахунку вірогідності зв’язку між акторами у рамках однієї епохи моделювання за допомогою бібліотеки numpy виглядає наступним чином:

```
def get_connection_probability(
     actor1: NodeInfo,
     actor_2: NodeInfo,
     media_coefficient: float = 4.
) -> np.float32:
     if actor1.is_media or actor_2.is_media:
         return np.clip(
              media_coefficient * np.random.uniform(0.0, 1.0),
              a_min=0.0,
              a_max=1.0
         )
     else:
         return np.random.uniform(low=0.0, high=1.0)
```

Основою моделювання є функція діалогу – процес обміну інформацією між акторами та зміни думки актора, якщо це можливо. У цій роботі запропонована наступна функція діалогу:

```
def dialog(
      actor1: NodeInfo,
      actor2: NodeInfo,
      factor: float = .5
) -> tuple[NodeInfo, NodeInfo]:
      x1: np.ndarray = actor1.weights
      x2: np.ndarray = actor2.weights

      actor1_mean_repr = .5 - actor1.weights
      actor2_mean_repr = .5 - actor2.weights

      if not actor1.is_media:
            new_actor1 = x1 - factor * actor2_mean_repr * P(1 -
O(x1))
          new_actor1 = np.clip(new_actor1, 0.0, 1.0)
          actor1.weights = new_actor1

      if not actor2.is_media:
            new_actor2 = x2 - factor * actor1_mean_repr * P(1 -
O(x2))
          new_actor2 = np.clip(new_actor2, 0.0, 1.0)
          actor2.weights = new_actor2

      return actor1, actor2
```

Ця функція зазнала змін з минулого дослідження лише у умові, що актори, що є засобами масової інформації, не можуть змінювати свою думку. Також на поведінку цієї функції впливають усі зміни, що були внесені у метод дослідження та описані вище.

Загалом процес моделювання реалізований наступним чином:

```
def modeling(
      graph: Graph,
      epochs: int = 100,
      factor: float = 1e-3,
```

```
    dialog_function = None
) -> Graph:
    if dialog_function is None:
        dialog_function = dialog

    for _ in range(epochs):
        for left_index, right_index in graph.edges:
            actor1 = graph.get_node(left_index)
             actor2 = graph.get_node(right_index)

             connection = graph.get_edge(
                 left_index,
                 right_index
             )

          if random() < connection.connection_probability:
               dialog_function(actor1, actor2, factor=factor)

    return graph
```

Як можна помітити із імплементації функції моделювання, реалізований у рамках роботи програмний інтерфейс пропонує використання різноманітних функцій діалогу, що можуть бути реалізовані користувачем (дослідником). За замовчуванням буде використовуватися запропонований у рамках цієї роботи метод.

2.6 Кластеризація акторів за думкою

Кластеризація акторів за векторами думок є дуже важливим у контексті роботи, адже це дає можливість виокремити окремі спільноти, відстежити шляхи поширення думок, вплив засобів масової інформації на певні кластери. За допомогою різноманітних алгоритмів кластеризації актори мережі можуть бути відокремленими за різними властивостями.

Один з найбільш розповсюджених алгоритмів кластеризації є K-Means [37]. Цей алгоритм реалізує ідею розбиття певних семплів (у контексті цієї роботи – векторів думок) таким чином, щоб мінімізувати відстань семплів до центру кластера, до якого вони були віднесені. Таким чином, алгоритм K-Means має за мету знайти для семплів (x_1, x_2, ..., x_n) такий набір кластерів (S_1, S_2, ..., S_k), щоб знайти наступне значення:

$$arg\,min \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 = arg\,min \sum_{i=1}^{k} |S_i|\, Var\, S_i$$

У цій формулі μ_i є середнім (також використовується термін центроїд) точок для певного кластеру S_i:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

У рамкаї цієї роботи алгоритм кластеризації K-Means має два дуже суттєвий недоліка:

необхідність вказувати необхідну кількість кластерів (k) перед початком кластеризації;

алгоритм не знаходить outliers у вхідних семплах, що є критичним, адже під час процесу моделювання деякі актори можуть мати аномальну поведінку, що призведе до погіршення якості

У якості представлення результатів експериментів пропонується використати алгоритм кластеризації DBSCAN [33], що базується на оцінці щільності розташування точок у просторі. Його головна ідея полягає в тому, що кластери – це області з високою щільністю точок, відокремлені областями з низькою щільністю (шумом або між-кластерним простором).

Оригінальний алгоритм можна записати наступним чином за допомогою псевдокоду:

```
DBSCAN(DB, distFunc, eps, minPts) {
    C=0
    for each point P in database DB {
        if label(P) ≠ undefined then continue
        Neighbors N=RangeQuery(DB, distFunc, P, eps)
        if |N| < minPtsthen{
             label(P)=Noise
             continue
        }
        C=C + 1
        label(P)=C
        Seed set S=N \ {P}
        for each point Q in S {
            if label(Q)=Noise then label(Q)=C
            if label(Q) ≠ undefined then continue
            label(Q)=C
                                                Neighbors
N=RangeQuery(DB,distFunc,Q,eps)
            if |N| ≥ minPts then{
                S=S ∪ N
            }
        }
    }
}
```

У рамках цієї роботи використовується імплементація DBSCAN з бібліотеки scikit-learn [34], через те що цей алгоритм не потребує встановлення певної кількості кластерів, а також дозволяє фільтрацію

аномальних точок. Кластеризація проводиться лише використовуючи акторів, що не є засобами масової інформації.

Для оцінки кластеризації була використана така метрика, як силует (The Silhouette Coefficient) [35]. У рамках цієї роботи була використана імплементація цієї метрики з бібліотеки sklearn [36].

Ця метрика може бути виражена наступним чином для точок у гіпер-просторі $i \in C_I$ (у контексті цієї роботи – векторів думок), що були розбиті на кластери (C_1, C_2, …, C_k):

$$a(i) = \frac{1}{|C_I| - 1} \sum_{j \in C_I,\, i \neq j} d(i, j)$$

$$b(i) = min_{J \neq I} \frac{1}{|C_J|} \sum_{j \in C_J} d(i, j)$$

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}},\ |C_I| > 1$$

$$s(i) = 0,\ |C_I| = 1$$

Сам Silhouette Coefficient може бути виражений через $s(i)$, як максимальне середнє значення $s(i)$ серед усіх кластерів:

$$SC = max_k\, s`(k)$$

2.7 Проведення експериментів

Експерименти були проведені на процесорі AMD Ryzen 5 7600x (6x12). Для різних експериментів визначались такі гіпер параметри, як:

- кількість вершин у початковому повному графі;
- кількість епох моделювання;
- загальна кількість акторів у мережі.

Був проведений аналіз швидкодії моделювання у залежності від загальної кількості акторів у мережі та кількості епох моделювання. Нижче приведений графік залежності часу виконання моделювання від кількості епох для мереж, що мають 100, 200, 500, 1000, 2000, 5000 та 10000 вершин. Результати є усередненими задля позбавлення даних від шуму. Також результати аналізу часу виконання моделювання були представлені у вигляді таблиці.

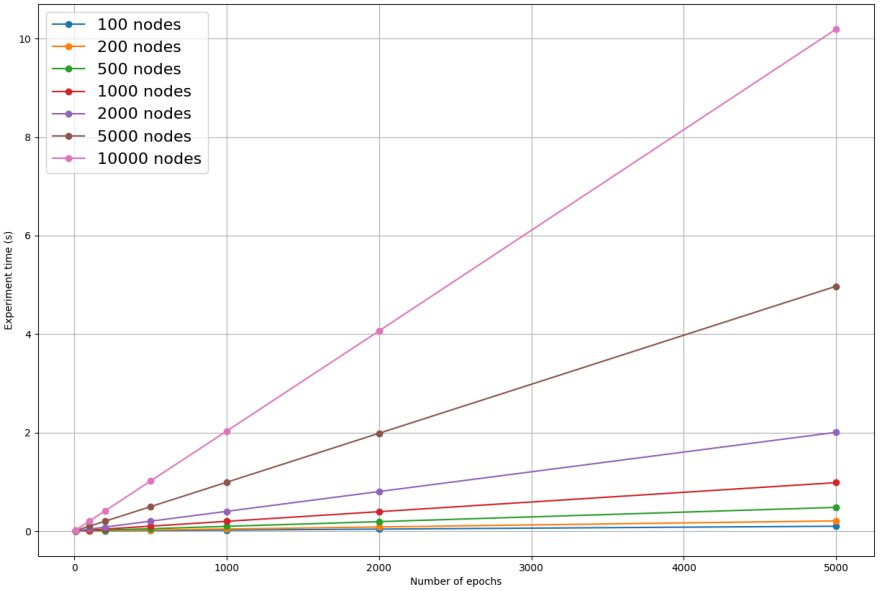

Рис. 8. Залежність часу виконання моделювання від кількості епох для мереж, що мають різну кількість вершин

Кількість вершин у мережі	Кількість епох моделювання	Час виконання моделювання (секунди)
100	100	0.0016
	200	0.0032
	500	0.0082

	1000	0.016
	2000	0.032
	5000	0.08
200	100	0.0031
	200	0.0062
	500	0.0158
	1000	0.031
	2000	0.063
	5000	0.157
500	100	0.008
	200	0.016
	500	0.04
	1000	0.08
	2000	0.16
	5000	0.41
1000	100	0.0184
	200	0.0367
	500	0.0916
	1000	0.183
	2000	0.365
	5000	0.92
2000	100	0.035
	200	0.07
	500	0.176
	1000	0.352

	2000	0.706
	5000	1.77
5000	100	0.089
	200	0.178
	500	0.444
	1000	0.883
	2000	1.773
	5000	4.404
10000	100	0.1757
	200	0.353
	500	0.877
	1000	1.777
	2000	3.566
	5000	8.883

Табл. 1. Залежність часу виконання моделювання від кількості вершин у мережі та кількості епох моделювання

2.8 Огляд результатів експериментів

У цьому розділі наведені результати одного з експериментів, що були проведені у рамках роботи. Для експерименту були обрані наступні гіпер-параметри системи:

- початкова кількість вершин у повному графі – 3;
- загальна кількість акторів (вершин) у мережі – 500;
- загальна кількість епох моделювання – 1000.

Початкова мережа має наступні властивості:

- середня думка мережі (без засобів масової інформації) – 0.4855;
- середня думка засобів масової інформації у мережі – 0.5678;
- за допомогою алгоритму DBSCAN було виявлено 14 кластерів;
- Silhouette score результатів кластеризації дорівнює -0.2206.

Перед початком експерименту мережа мала наступний вигляд (на рисунку засоби масової інформації виділені колами більшого радіусу, а відтінок вершини позначає думку певного актора (чим світліше вершина, тим середнє значення вектору думки актора ближче до 1):

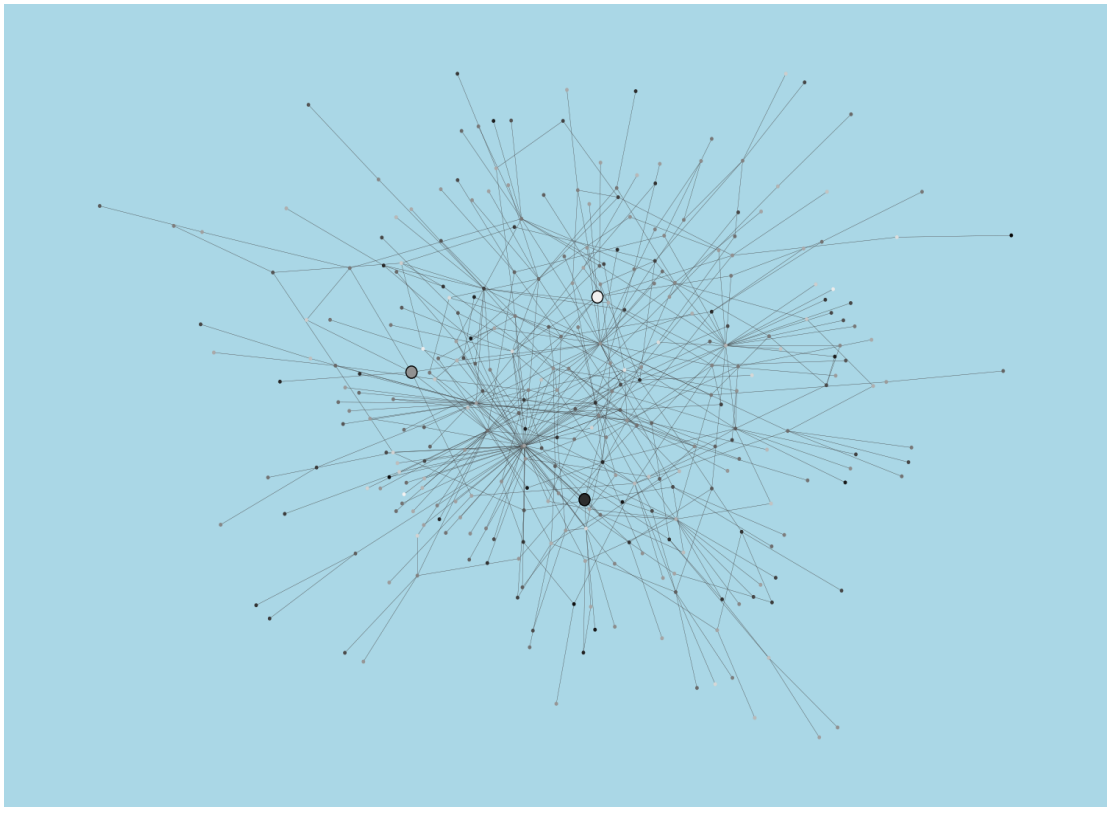

Рис. 9. Початковий вигляд мережі у описаному експерименті

Далі наведені рисунки, що відображають стан мережі після завершення певної кількості епох моделювання:

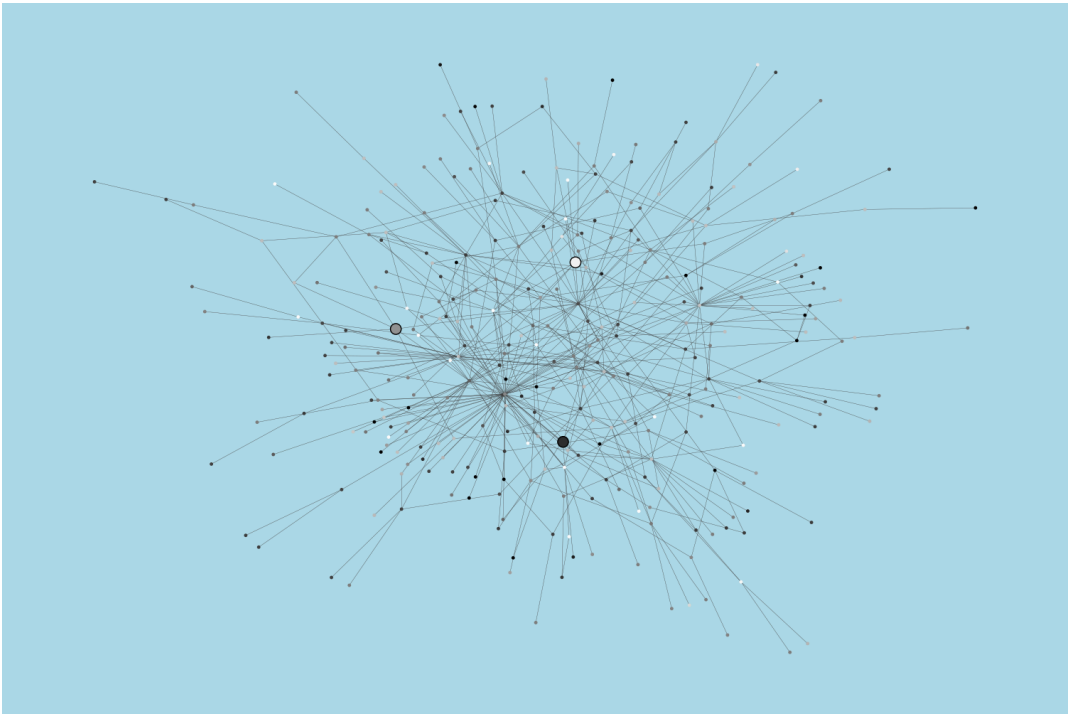

Рис. 10. Стан мережі після завершення 100 епох моделювання

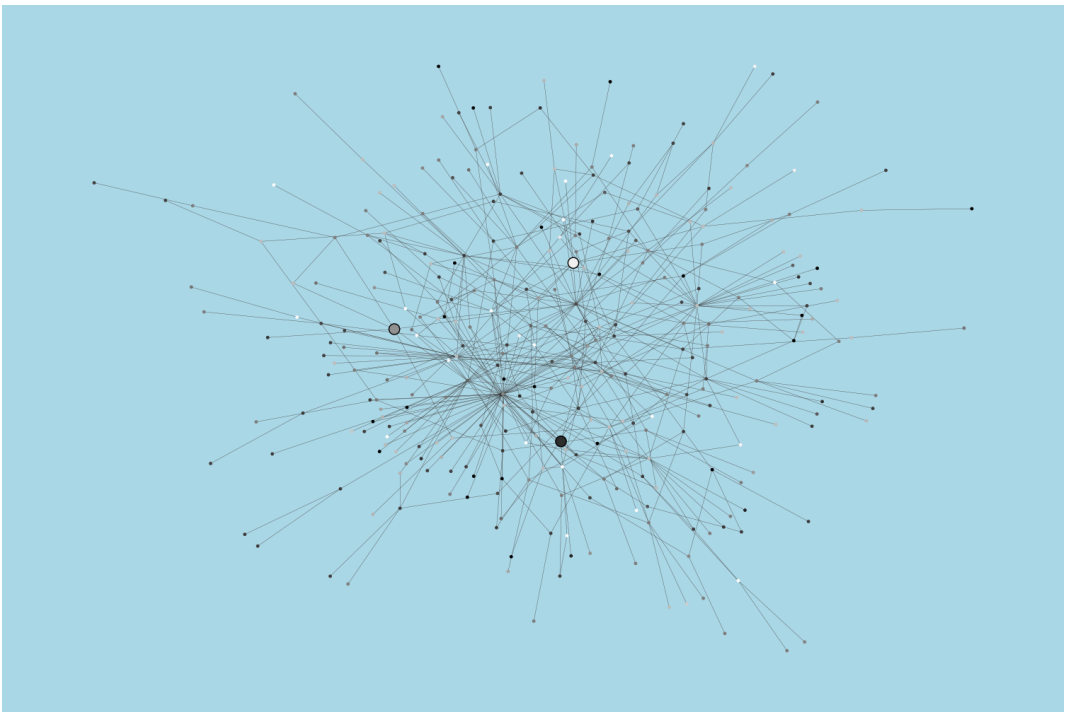

Рис 11. Стан мережі після виконання 250 епох моделювання

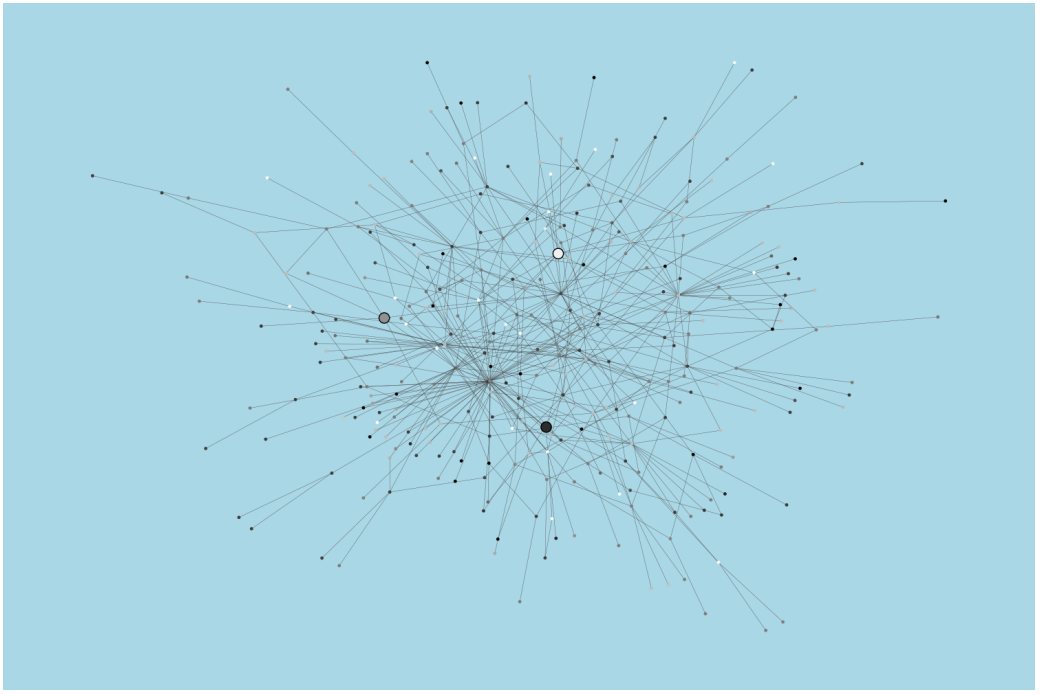

Рис 12. Стан мережі після завершення 500 епох моделювання

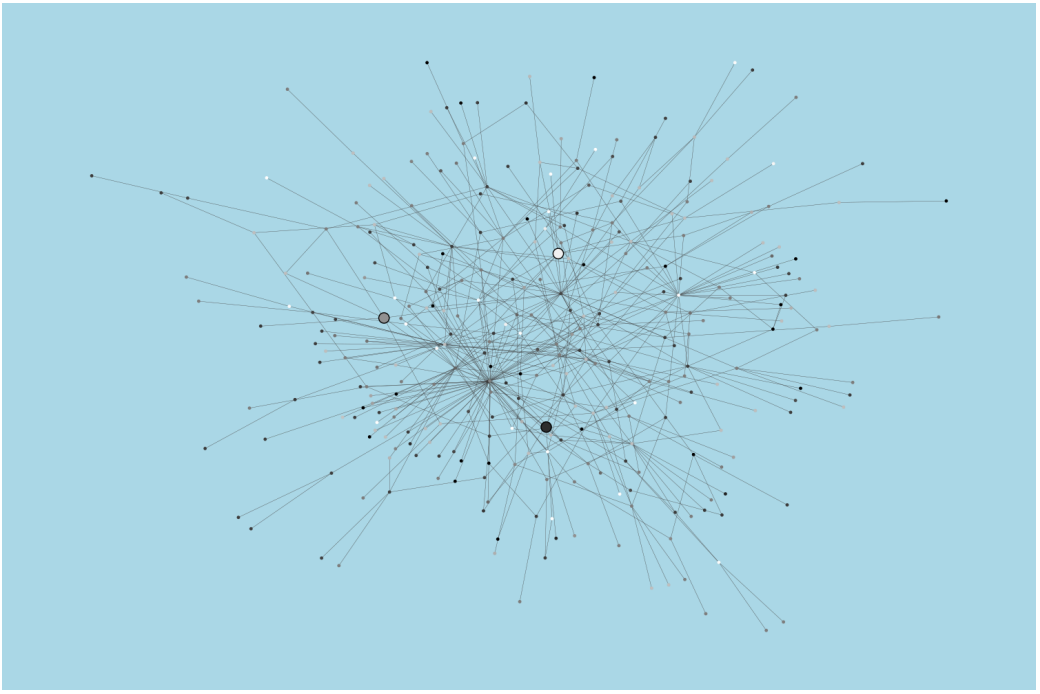

Рис 13. Стан мережі після завершення 1000 епох моделювання

Також результати експерименту представлені у якості таблиці, у якій розглянуті кількість знайдених кластерів за допомогою алгоритма

DBSCAN, Silhouette score та середня думка мережі (без урахування засобів масової інформації).

Кількість епох моделювання	Кількість знайдених кластерів	Silhouette score	Середня думка мережі
0	14	-0.2206	0.4855
100	8	0.6022	0.4758
250	8	0.6606	0.4742
500	10	0.7091	0.4728
1000	9	0.7267	0.4728

Табл. 2. Результати експерименту для різної кількості епох моделювання

Таким чином можна помітити, що під впливом зовнішніх джерел інформації та під впливом комунікації з іншими акторами, мережа з кожною епохою моделювання стає все більш кластеризованою. Тобто під впливом інформації люди об’єднуються у інформаційні кластери, про що йшла мова у початку роботи.

Аналіз таких кластерів є дуже важливим, адже вони є найбільш вразливими до дезінформації, пропаганди та проштовхування певних наративів. Запропонований метод моделювання зміни соціальної думки, а також і програмний інтерфейс, що було розроблено для проведення експериментів, дозволяють проводити аналіз на предмет закономірностей появи таких кластерів, зв’язків між певними кластерами, появою outliers тощо.

Також наведені точкові діаграми векторів думок акторів для різної кількості епох моделювання:

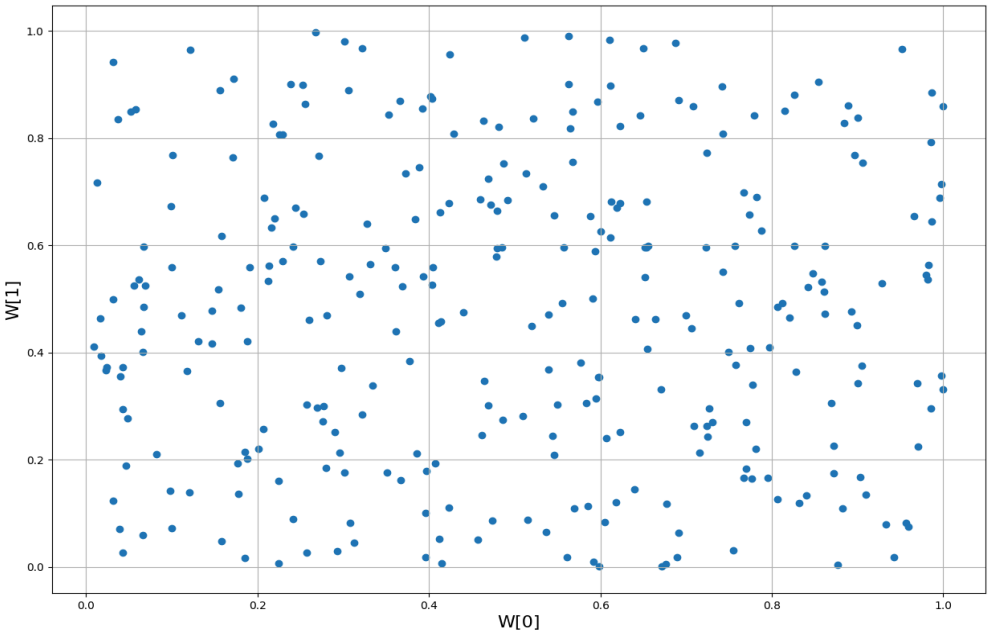

Рис. 14. Точкова діаграма векторів думок акторів перед початком моделювання

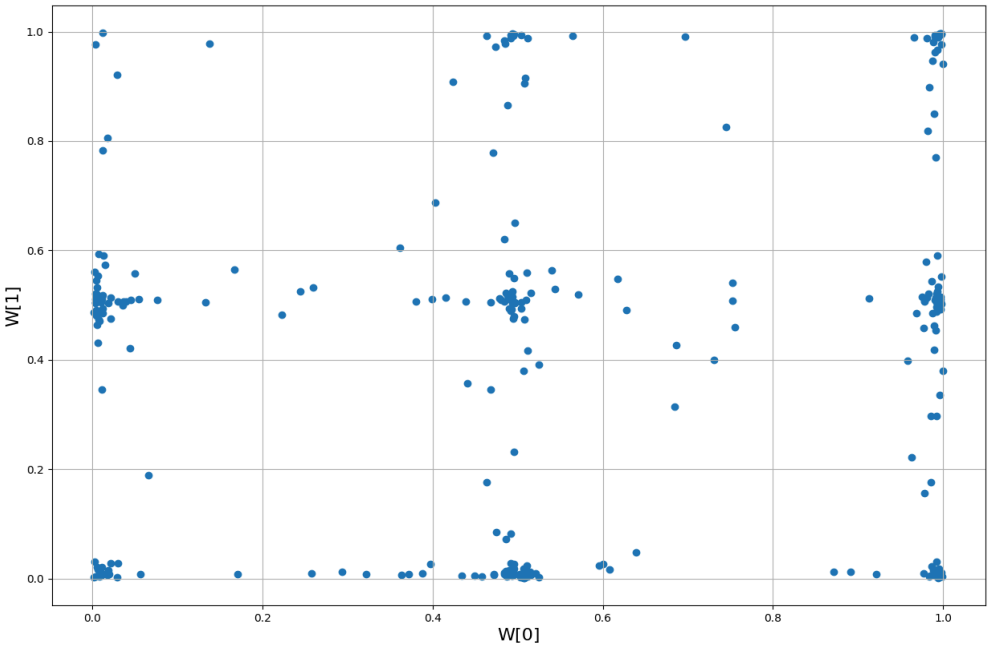

Рис. 15. Точкова діаграма векторів думок акторів після 100 епох моделювання

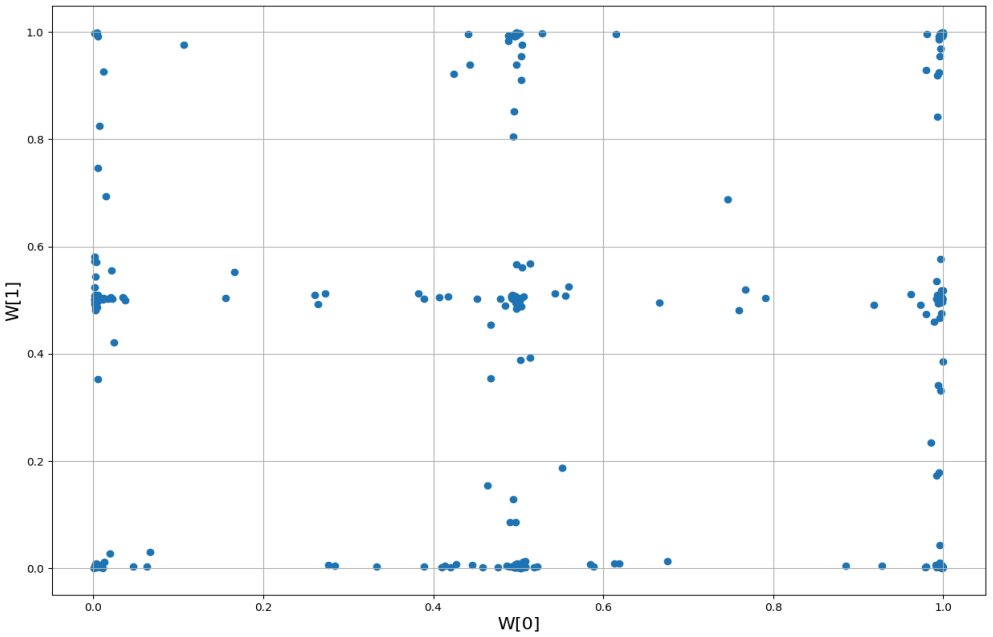

Рис. 16. Точкова діаграма векторів думок акторів після 250 епох моделювання

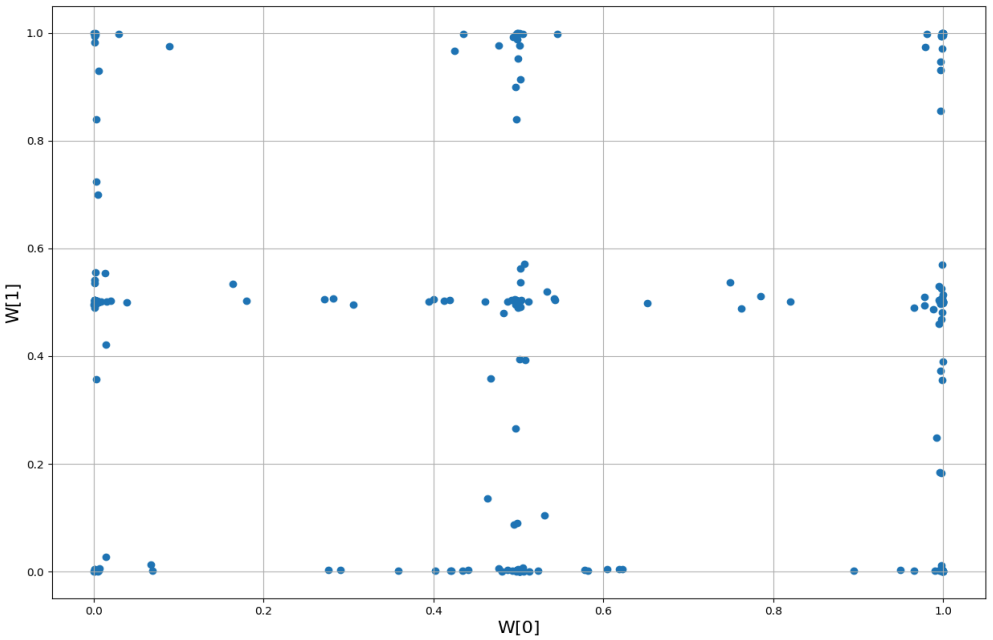

Рис. 17. Точкова діаграма векторів думок акторів після 500 епох моделювання

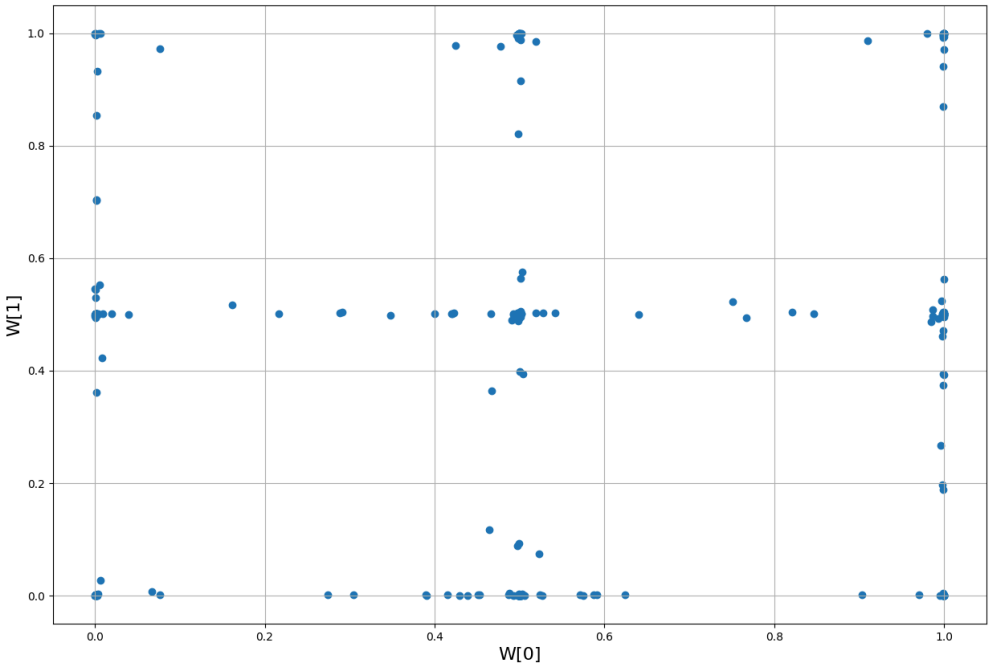

Рис. 18. Точкова діаграма векторів думок акторів після 1000 епох моделювання

3. ВИСНОВКИ

У межах виконання поставлених задач було повністю реалізовано програмний інтерфейс для дослідження методів боротьби за суспільну думку, а також проведено низку експериментів згідно із запропонованим методом. У процесі розробки було:

- проаналізовано недоліки інтерфейсу, запропонованого в попередньому дослідженні, що дозволило уникнути їх повторення;
- визначено необхідний функціонал для забезпечення гнучкого проведення експериментів;
- обрано програмне середовище та бібліотеки, що забезпечують ефективну реалізацію поставлених завдань;
- сформульовано подальші кроки щодо розширення функціональності інтерфейсу та підвищення ефективності досліджень.

Проведені експерименти підтвердили працездатність інтерфейсу, дозволили на практиці перевірити вплив засобів масової інформації на суспільну думку, а також забезпечили можливість візуалізації та аналізу отриманих результатів. Внесені зміни у метод дослідження сприяли підвищенню точності моделювання та полегшенню інтерпретації результатів.

Однією із задач роботи є визначення можливих поліпшень програмного інтерфейсу та запропонованого методу моделювання боротьби за суспільну думку. Серед можливих модернізацій можна виділити:

- можливість використання програмного інтерфейсу як Python модуля, що може бути встановлений через пакетний менеджер pip [39] або локально за допомогою setuptools [40];
- подальше розширення опцій для візуалізації, генерації та кластеризації мереж;
- реалізація більш гнучкого інтерфейсу для керування процесом моделювання;
- розширення можливих ролей акторів у мережі, наприклад, ботами. Боти дуже сильно впливають на моделювання боротьби за суспільну думку, адже за даними Imperva Bad Bot Report [38], майже половина усього трафіку у мережі генерується ботами, отже вони можуть бути використані для впливу на суспільну думку (наразі вже можна спостерігати дуже багато ботів у таких соціальних мережах як Tik-Tok, що налаштовані на поширення дезінформації).

Таким чином, поставлені завдання були повністю виконані, а результати роботи можуть слугувати основою для подальших досліджень у сфері моделювання інформаційного впливу в соціальних мережах.

Реалізований програмний інтерфейс у вигляді GitHub репозиторію [41] можна знайти за посиланням у списку використаних джерел.

4. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PewResearchCenter, Social Media and News Fact Sheet (2024) – www.pewresearch.org/journalism/fact-sheet/social-media-and-news-fact-sheet/
2. Facebook – www.facebook.com
3. X – x.com
4. Instagram – www.instagram.com
5. TikTok - www.tiktok.com
6. YouTube – www.youtube.com
7. Колмогоров Д. А., Жолткевич Г. М – “Моделювання боротьби за вплив на суспільну думку у мережевих спільнотах”
8. NetLogo – ccl.northwestern.edu/netlogo
9. GAMA Platform – gama-platform.org
10. ORA (Organizational Risk Analyzer) – www.casos.cs.cmu.edu/publications/papers/carley_2004_oraorganizationrisk.pdf
11. GAML – gama-platform.org/wiki/GamlLanguage
12. CASOS – www.casos.cs.cmu.edu
13. Carnegie Mellon University – www.cmu.edu
14. Python – www.python.org
15. matplotlib – matplotlib.org
16. networkx – networkx.org
17. numpy – numpy.org
18. scipy – scipy.org
19. Fortran – fortran-lang.org
20. C++ – en.cppreference.com
21. Rust – www.rust-lang.org

22. PyCharm – www.jetbrains.com/pycharm
23. Pandas – pandas.pydata.org
24. Scikit-learn – scikit-learn.org
25. Plotly – plotly.com
26. Seaborn – seaborn.pydata.org
27. Jupyter Notebook – jupyter.org
28. JavaScript – javascript.info
29. R – www.r-project.org
30. Docker – www.docker.com
31. Albert R. and Barabási A.-L., «Statistical mechanics of complex networks» , Rev. Mod. Phys. 74, 47–97 (2002).
32. networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.barabasi_albert_graph.html
33. Ester, Kriegel, Sander, Xu, 1996, с. 226–231.
34. Scikit-learn DBSCAN implementation – scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html
35. Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65
36. scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
37. Kriegel, Hans-Peter; Schubert, Erich; Zimek, Arthur (2016). "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?". Knowledge and Information Systems. 52 (2): 341–378.
38. imperva.com/resources/resource-library/reports/2024-bad-bot-report
39. PIP – pypi.org/project/pip
40. setuptools – pypi.org/project/setuptools
41. github.com/myDiamondsDancing/graph-modeling-for-society