Міністерство освіти і науки України
Чернівецький національний університет імені Юрія Федьковича

Кваліфікаційна наукова праця
на правах рукопису

КИРИЧЕНКО ОКСАНА ЛЕОНІДІВНА

УДК 004.415+004.942]:519.24/.25

ДИСЕРТАЦІЯ

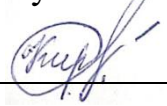
ДОСЛІДЖЕННЯ СТАТИСТИЧНИХ ХАРАКТЕРИСТИК СКЛАДНИХ МЕРЕЖ МЕТОДАМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

121 – інженерія програмного забезпечення

12 – Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.



_____ О.Л. Кириченко

Науковий керівник: **Остапов Сергій Едуардович** доктор фізико-математичних наук, професор

Чернівці – 2023

АНОТАЦІЯ

Кириченко О.Л. Дослідження статистичних характеристик складних мереж методами інтелектуального аналізу даних. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121 – «Інженерія програмного забезпечення» – Чернівецький національний університет імені Юрія Федьковича, Чернівці, 2023.

Дисертаційна робота присвячена дослідженню статистичних характеристик складних мереж та кластерної структури веб-простору з використанням методів інтелектуального аналізу даних, зокрема розробці інформаційної технології для кластеризації даних великого розміру, які були зібрані й оброблені спеціально створеним програмним забезпеченням. Також вивчено стохастичні матриці, які завдяки особливостям своїх спектральних властивостей є основним математичним об’єктом при дослідженні кластерної структури веб-простору.

Результати роботи є підґрунттям для подальших теоретичних і практичних наукових розробок із досліджень проблематики теорії складних мереж.

Дисертація складається зі вступу, чотирьох розділів, висновків, переліку використаних джерел та чотирьох додатків. У **вступі** обґрунтовано актуальність теми дослідження, сформульовано мету, завдання, предмет, об’єкт та методи дослідження, вказано наукову новизну, теоретичне та практичне значення отриманих результатів, подано та проаналізовано зв’язок роботи з науковими темами. Зазначено особистий внесок здобувача, а також наведено відомості про апробацію та публікації основних результатів дисертації. Описано структуру та обсяг дисертаційної роботи.

Перший розділ містить ключові відомості з теорії складних мереж, опис основних напрямів досліджень та завдання, якими займається теорія складних мереж. Тут проведено огляд та опис основних моделей (Ердоша–Рені, Уаттса–Строґаца, Барабаші–Альберт), які призвели до сьогоднішнього

розуміння цього напряму. Розглянуто та проаналізовано приклади реальних складних мереж (онлайнові, наукової співпраці, WWW, цитування наукових праць, Інтернет, транспортна, різні біологічні мережі тощо) та їх особливості Здійснено класифікацію та огляд методів однієї з важливих технік інтелектуального аналізу складних мереж – кластерного аналізу.

У **другому розділі** дисертаційного дослідження описано концепцію кроулінгу як одного із засобів збирання інформації, проведено огляд існуючих програмних засобів для збирання інформації у веб-просторі. Описано розроблене власне програмне забезпечення (кроулер), який сканує веб-простір, завантажує та зберігає знайдені гіперпосилання з веб-стоінок у базу даних. Перевагою даної розробки над існуючими аналогами є наявність аналітичного модуля, який надає можливість проводити статистичний та кластерний аналіз отриманого веб-графу.

Другий розділ має прикладне значення, основним його результатом є розроблене спеціалізоване програмне забезпечення – кроулер з вбудованим аналітичним модулем для інтелектуальної обробки інформації.

Третій розділ присвячений дослідженню освітніх сегментів веб-простору (українського (edu.ua), ізраїльського (ac.il) та польського (edu.pl), інформація про які була зібрана та оброблена за допомогою самостійно розробленої інформаційної технології, детальний опис якої проведено в пункті 3.3. Застосування даної розробки дозволило отримати статистичні характеристики та кластерну структуру вказаних вище сегментів веб-простору та здійснити порівняльний аналіз.

Для проведення кластеризації важливо знати оптимальну кількість кластерів, у розділі описано два класичних методи знаходження оптимальної кількості кластерів (метод «ліктя» та k-core decomposition), проведено порівняльний аналіз, який показав їх узгодженість щодо оптимальної кількості кластерів для кожного досліджуваного сегменту веб-простору.

Основні результати даного розділу можна підсумувати наступним чином:

- для збирання та проведення статистичного і кластерного аналізу даних у складних мережах розроблено інформаційну технологію;
- для підмереж edu.ua, edu.pl та ac.il проведено порівняльний аналіз статистичних характеристик та їх кластерної структури ;
- встановлено, що всі три підмережі відповідають сучасним тенденціям розвитку глобальної мережі інтернет, володіють властивостями безмасштабних графів, причому виявилось, що український сегмент edu.ua є найменш розвиненою структурою з найменшою кількістю вузлів у кластерах.

У **четвертому розділі** розглянуто питання кластеризації в графі на основі матриці суміжності. Основним об'єктом дослідження даного розділу є стохастична матриця P, що задає ймовірності переходу на графі та визначається із матриці суміжності. У даному розділі детально проаналізовано спектральні властивості стохастичної матриці P із врахуванням кластерної структури графу.

Основні теоретичні результати цього розділу можна описати наступним чином:

- доведено факт збіжності власних значень матриці P за умов, накладених на елементи матриці суміжності A (теорема 4.3.1). Причому, накладені умови послаблені порівняно із класичними результатами, де вимагається існування скінченного другого моменту для елементів матриці суміжності;
- встановлений факт про асимптотичну еквівалентність спектрів матриць P та $\tilde{P}$ дозволяє використовувати стохастичну матрицю із незалежними елементами замість відповідної стохастичної матриці P, елементи якої не є незалежними (лема 4.4.1). Даний результат дозволяє користуватися класичними результатами щодо розподілу власних значень випадкових матриць та переносити дані твердження на матриці із слабко корельованими елементами;

– у твердженнях пункту 4.5. розглянуто частинний підхід до оцінки розподілу елементів матриці P за умови показникового розподілу елементів матриці A (лемах 4.5.1 та 4.5.2). Такий підхід дозволив розробити новий алгоритм перевірки належності елементів (вершин графу) до одного кластеру.

На основі отриманих теоретичних результатів проведено порівняльний аналіз з класичними методами кластеризації, а саме: методом «ліктя», k-core decomposition та методом силуету. У результаті проведених досліджень, побудовано критерій оцінки оптимальної кількості кластерів k_{opt}, обчислення якого ґрунтується на власних значеннях стохастичної матриці P, а саме

$$\hat{k}_{opt} = \#\{\lambda_i(P): Re(\lambda_i(P)) > \max|Im(\lambda_i(P))|\}.$$

Використовуючи метод Монте – Карло, вдалося встановити, що у ряді випадків, запропонований спектральний метод визначення кількості кластерів дає більш точні оціночні значення кількості кластерів у графі в порівнянні з відомими методами («ліктя», k-core decomposition та методом силуету), що задається стохастичною матрицею P чи матрицею суміжності A. Крім того, запропонований новий метод є менш чутливим до наявності кластерів різної розмірності.

У **висновках** підсумовано основні результати дисертаційного дослідження.

У **додатках** подано наукові публікації, в яких відображено основні наукові результати роботи, відомості про апробацію результатів дисертації – акти та довідки про впровадження результатів роботи, діаграма основних класів кроулера та їх опис, лістинг частини коду програми.

Теоретичне значення. Результати теоретичних досліджень, а саме розвитку теорії графових досліджень, сформульовані та доведені леми і теореми, можуть використовуватися для подальших досліджень у цій галузі, а також у навчальних курсах кафедр математичних проблем управління та кібернетики та програмного забезпечення комп’ютерних систем

Чернівецького національного університету імені Юрія Федьковича (та інших ЗВО), пов'язаних з інтелектуальним аналізом даних, методичних розробках, навчальних посібниках для освітнього процесу та науково-дослідної роботи студентів аспірантів.

Практичне значення. Розроблені у дисертаційній роботі кроулер, інформаційна технологія та метод визначення оптимальної кількості кластерів можуть в подальшому використовуватися для практичного дослідження складних мереж. Запропоновані підходи до архітектури аналітичного модуля використовуються у компанії «Квант Азимут» для розробки власного програмного забезпечення та компанії «Qlicks B.V.» – для проведення сегментації клієнтів на різні категорії, які потім ефективно використовуються для персоналізованих маркетингових кампаній і стратегій та передбачення поведінки клієнтів на основі аналізу покупок, історії пошуку або профілей в соціальних мережах.

Ключові слова: модель (математична, економічна), моделювання, динаміка, інтелектуальний аналіз даних, кластеризація, k-means, інформаційна система, інформаційна технологія, інтелектуальна система, програмне забезпечення, тестування програмного забезпечення, рівні тестування програмного забезпечення, специфікація вимог до програмного забезпечення, функціональні та нефункціональні вимоги до програмного забезпечення, статистичні методи.

ABSTRACT

Kyrychenko O.L. The Study of Statistical Characteristics of Complex Networks by Methods of Intelligent Data Analysis. – Qualification research work published in the manuscript.

Dissertation for the degree of Doctor of Philosophy, speciality 121 – "Software Engineering" – Yuriy Fedkovych Chernivtsi National University, Chernivtsi, 2023.

The dissertation deals with the study of statistical characteristics of complex networks and the cluster structure of the web space using methods of intelligent data analysis, in particular, the development of information technology for the clustering of large data, which were collected and processed by specially created software. In addition, stochastic matrices have been studied, which, due to their specific spectral properties, are the main mathematical object in the study of the cluster structure of the web space.

The results of the research are the basis for further theoretical and practical scientific development in the research of the problems of the theory of complex networks.

The dissertation contains an introduction, four chapters, conclusions, a list of literature, and four appendices. The **introduction** substantiates the relevance of the research topic, formulates the goal, task, subject, object and research methods; indicates the scientific novelty, theoretical and practical significance of the obtained results; presents and analyzes the link between the current research and scientific topics. The personal contribution of the candidate, as well as information about the approval and publication of the main results of the research are shown. The structure and scope of the dissertation are outlined.

Chapter 1 contains key information on the theory of complex networks, a description of the main areas of research and tasks that the theory of complex networks deals with. An overview and description of the main models (the Erdős–Rényi, Watts-Strogatz, Barabási–Albert models) having led to current insight into this trend are provided here. Examples of real complex networks (online, scientific collaboration, WWW, citation of scientific works, Internet, transport, various biological networks, etc.) and their features are considered and analyzed. The methods of cluster analysis, an important technique of intellectual analysis of complex networks, are classified and reviewed.

Chapter 2 of the dissertation describes the concept of crawling as one of the means of gathering information, and provides an overview of existing software tools for collecting information in the web space. The own development of the

software (crawler) is described, which scans the web space, downloads and stores the found hyperlinks from the web sites in the database. The advantage of this development over existing analogues is the presence of an analytical module, which enables to conduct statistical and cluster analysis of the resulting web graph.

Chapter 2 has a practical value; its main result is the developed specialized software – a crawler with a built-in analytical module for intelligent information processing.

Chapter 3 deals with the study of following educational segments of the web space: Ukrainian (edu.ua), Israeli (ac.il) and Polish (edu.pl). The information on these segments was collected and processed using a personally developed information technology, a detailed description of which is provided in point 3.3. The application of this development enables to obtain the statistical characteristics and cluster structure of the above-mentioned segments of the web space and to carry out a comparative analysis.

As far as it is important to know the optimal number of clusters to carry out clustering, two classical methods of finding the optimal number of clusters (the "elbow" method and *k-core* decomposition) are described in the chapter. A comparative analysis was carried out, which showed their agreement regarding the optimal number of clusters for each studied segment of the web space .

The main results of this section can be summarized as follows:

- information technology has been developed for collecting information and conducting statistical and cluster analysis of data in complex networks;
- a comparative analysis of statistical characteristics and their cluster structure was carried out for the *edu.ua, edu.pl* and *ac.il* subnets;
- it was established that all three subnets correspond to modern trends in the development of the global Internet network, have the properties of scale-free graphs; in addition, it turned out that the Ukrainian segment of *edu.ua* is the least developed structure with the least number of nodes in clusters.

Chapter 4 deals with the issue of clustering in a graph based on the adjacency matrix. The main object of research in this section is the stochastic matrix P, which

specifies the transition probabilities on the graph and is determined from the adjacency matrix. In this chapter, the spectral properties of the stochastic matrix P are analyzed in detail, taking into account the cluster structure of the graph.

The main theoretical results of this chapter can be described as follows:

- the fact of the convergence of the eigenvalues of the matrix P under the conditions imposed on the elements of the adjacency matrix A is proved (theorem 4.3.1). Moreover, the imposed conditions are weaker compared to the classical results, which require the existence of a finite second moment for the elements of the adjacency matrix;
- the established fact about the asymptotic equivalence of the spectra of the matrices P and $\tilde{P}$ allows using a stochastic matrix with independent elements instead of the corresponding stochastic matrix P, the elements of which are not independent (lemma 4.4.1). This result enables to use classical results on the distribution of eigenvalues of random matrices and transfer these statements to matrices with weakly correlated elements;
- in the statements of point 4.5 a partial approach to the estimation of the distribution of the elements of matrix P is considered under the condition of the indicative distribution of matrix A elements (lemmas 4.5.1 and 4.5.2). This approach made it possible to develop a new algorithm for checking whether elements (graph vertices) belong to one cluster.

Based on the obtained theoretical results, a comparative analysis was carried out with classical clustering methods, namely: the "elbow" method, k-core decomposition, and the silhouette method. The research resulted in building a criterion for estimating the optimal number of clusters k_opt , the calculation of which is based on the eigenvalues of the stochastic matrix P, namely

$$\hat{k}_{opt} = \#\{\lambda_i(P): Re(\lambda_i(P)) > \max|Im(\lambda_i(P))|\}.$$

Using the Monte-Carlo method, it was possible to determine that in a number of cases, the proposed spectral method for finding the number of clusters gives more accurate estimates of the number of clusters in the graph in comparison with

known methods ("elbow", k-core decomposition and the silhouette method), given by the stochastic matrix P or adjacency matrix A. In addition, the proposed new method is less sensitive to the presence of clusters of different dimensions.

The main results of the dissertation research are summarized in the **conclusions**.

The **appendices** present scientific publications reflecting the main scientific results of the work, information on the approval of the results of the dissertation: acts and certificates on the implementation of the results of the research, a diagram of the main classes of the crawler and their description, and listing of part of the software code.

Theoretical significance. The results of theoretical research, namely the development of the theory of graph research, formulated and proven lemmas and theorems, can be used for further research in this field. They can also be applied in the educational courses of the Departments of Mathematical Problems of Management and Cybernetics and Software of Computer Systems of Yuriy Fedkovych Chernivtsi National University (and other higher educational institutions), related to the intellectual analysis of data, methodological developments, teaching aids for the educational process and research work of graduate and postgraduate students.

Practical significance. The crawler, information technology, and method for determining the optimal number of clusters developed in the dissertation can be used for further practical research of complex networks. The proposed approaches to the architecture of the analytical module are used by the company "Kvant Azimuth" for the development of its own software and by the company "Qlicks B.V." for segmenting customers into different categories, which are then effectively used for personalized marketing campaigns and strategies and for predicting customers' behavior based on purchase analysis, search history or social media profiles.

Keywords*:* model (mathematical, economic), simulation, dynamics, intelligent data analysis, clustering, k-means, information system, information

technology, intelligent system, software, software testing, software testing levels, specification of software requirements, functional and non-functional software requirements, statistical methods.

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці у періодичних наукових виданнях, проіндексованих у наукометричних базах даних Scopus:

1. Kyrychenko O., Ostapov S., Kanovsky I. Investigation of the certain internet domain statistical characteristics / Статистичні характеристики деяких зон інтернету та їх дослідження. *Eastern-European Journal of Enterprise Technologies*. 2013. Vol. 6, no. 12(66). P. 91–96. (Scopus)

Наукові праці у виданнях, включених до переліку наукових фахових видань України:

2. Кириченко О.Л., Малик І.В., Остапов С.Е. Стохастичні моделі в задачах штучного інтелекту. *Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки*. 2021. № 2. С. 53–57.
3. Kyrychenko O. Information technology for statistical cluster analysis of information in complex networks. *Computer Systems and Information Technologies*. 2022. No 4. P. 47–51.
4. Кириченко О. Особливості архітектури програмного забезпечення для збору та аналізу статистичної інформації в глобальній мережі. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2023. № 2. С. 107–112.

Наукові праці, які додатково відображають наукові результати дисертації:

5. Kyrychenko O.**,** Ostapov S., Kanovsky I. Comparison of Statistical Characteristics of Certain Internet Subdomains. Chapter 1. Monograph. *Internet in the information society. Insights on the information systems, structures and applications* / ed. by M. Rostanski, P. Pikiewicz. Scientific

Publishing of the Academy of Business in Dabrowa Gornicza : Wydawnictwo Naukowe, 2014. 138 p. ISBN: 978-83-62897-91-9.

6. Кириченко О.Л., Малик І.В., Остапов С.Е. Аналіз кластерної структури Інтернет-мереж на основі випадкових матриць. *Проблеми керування та інформатики.* 2022. №1. С. 37–46.
7. Кириченко О. Л., Kanovsky I., Остапов С. Е. Програмне забезпечення для дослідження статистичних характеристик глобальної мережі WWW. *Системи обробки інформації*. 2013. Том 2. Вип. 3. С. 99–104.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Кириченко О. Л., Остапов С. Е. Статистичні характеристики деякої частини українського домену глобальної мережі WWW. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2012)* : тези доповідей Всеукр. наук.-практ. конф., м. Чернівці, 3–5 травня 2012 р. Чернівці : Золоті литаври, 2012. С. 93–94.
9. Кириченко О. Л., Остапов С. Е., Kanovsky I. Статистичні характеристики українських доменів edu.ua, net.ua глобальної www-мережі та їх дослідження. *Актуальні проблеми інформаційних технологій, економіки та права (ІТЕП–2013)* : тези доповідей Міжнар. наук.-практ. конф., м. Чернівці, 3–5 квітня 2013 р. Чернівці : Книги-ХХІ, 2013. С. 84–85.
10. Кириченко О. Л., Kanovsky I., Остапов С. Е. Складні мережі та їх статистичні характеристики: аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2013)* : тези доповідей Всеукр. наук.-практ. конф., м. Чернівці, 27–31 травня 2013 р. Чернівці : Видавничий дім «Родовід», 2013. С. 16–22.
11. Кириченко О. Л., Kanovsky I. Y., Остапов С. Е. Дослідження статистики складних мереж на прикладі українських та ізраїльського доменів www-простору. *Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано- та мікроелектроніки* : матеріали ІІІ-ої

Міжнар. наук.-практ. конф., м. Чернівці, 24–26 жовтня 2013 р. Чернівці, 2013. С. 125–126.

12. Кириченко О. Л., Остапов С. Е. Статистичні характеристики польського домену Інтернет edu.pl. *Інтелектуальні технології в системному програмуванні (ІТСП-2014)* : збірн. наук. праць ІІІ Всеукр. наук.-практ. конф. молодих учених та студентів, м. Хмельницький, 23–25 квітня 2014 р. Хмельницький : Гонта А.С., 2014. С. 269–270.
13. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Моделювання, дослідження та аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2014)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 27–30 травня 2014 р. Чернівці : Видавничий дім «Родовід», 2014. С. 174–176.
14. Кириченко О. Л., Остапов С. Е. Дослідження структури Веб-простору за допомогою кластерного аналізу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2016)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 21–24 травня 2016 р. Чернівці : Видавничий дім «Родовід», 2016. С. 112–114.
15. Кириченко О.Л., ОстаповС.Е., Кановський І.Я. Проведення оптимальної кластеризації структури веб-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2017)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 05–08 жовтня 2017 р. Чернівці : Видавничий дім «Родовід», 2017. С. 68–70.
16. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Проведення оптимальної кластеризації структури деяких зон веб-простору за допомогою методу K-Core Decomposition. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2018)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 11–14 жовтня 2018 р. Чернівці : Видавничий дім «Родовід», 2018. С. 48–50.
17. Кириченко О. Л., Остапов С. Е. Застосування методу k-Core Decomposition для проведення оптимальної кластеризації. *Інформаційні*

технології: наука, техніка, технологія, освіта, здоров'я : тези доп. XXVII Міжнар. наук.-практ. конф. MicroCAD-2019 (м. Харків, 15–17 травня 2019 р.) : у 4 ч. Ч. IV. / за ред. проф. Сокола Є. І. Харків : НТУ «ХПІ». С. 155.

18. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Розробка інформаційної технології проведення статистично-кластерного аналізу інформації у складних мережах. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2019)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 03–06 жовтня 2019 р. Чернівці : ЧНУ, 2019. С. 92–94.
19. Кириченко О. Л., Остапов С. Е. Інформаційна технологія для проведення досліджень у складних мережах. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXVIII Міжнар. наук.-практ. конф. MicroCAD-2020 (м. Харків, 28–30 жовтня 2020 р.) : у 5 ч. Ч. І. / за ред. проф. Сокола Є.І. Харків : НТУ «ХПІ». С. 324.
20. Кириченко О. Л., Малик І. В., Остапов С. Е. Кластеризація великих даних на основі спектрального аналізу матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2020)* : праці IX Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2020 р. Чернівці : ЧНУ, 2020. С. 83–84.
21. Кириченко О. Л., Малик І. В., Остапов С. Е. Асимптотичний розподіл власних значень матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2021)* : праці X Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2021 р. Чернівці : ЧНУ, 2021. С. 22–24.
22. Kyrychenko O. L., Knignitska T. V., Ostapov S. E. Stochastic models in artificial intelligence development. *Modern stochastics: theory and applications V* : Materials of the Intern. Conf., Kyiv, Ukraine, June 1–4, 2021. Kyiv, 2021. P. 35.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 18
ВСТУП 19
РОЗДІЛ 1. СКЛАДНІ МЕРЕЖІ ТА ЇХ РОЛЬ В ІНФОРМАЦІЙНОМУ ПРОСТОРІ 24
1.1. Теорія складних мереж 24
1.2. Моделі складних мереж 25
1.2.1. Класичний випадковий граф (модель Ердоша-Рені) . . . 26
1.2.2. Мережа малого світу (модель Уаттса – Строґаца) 27
1.2.3. Безмасштабні мережі (модель Барабаші–Альберт) 28
1.3. Приклади складних мереж 29
1.3.1. Соціальні мережі 30
1.3.1.1. Онлайнові соціальні мережі 30
1.3.1.2. Мережа наукової співпраці 32
1.3.2. Інформаційні мережі 33
1.3.2.1. WWW мережа 33
1.3.2.2. Мережі цитування наукових праць 36
1.3.3. Технологічні мережі 36
1.3.3.1. Інтернет мережа 36
1.3.3.2. Транспортна мережа 39
1.3.4. Біологічні мережі 39
1.4. Методи та засоби інтелектуального аналізу даних 41
1.5. Методи кластеризації у дослідженні складних мереж 44
ВИСНОВКИ ДО РОЗДІЛУ 1 51
РОЗДІЛ 2. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБИРАННЯ ДАНИХ 52
2.1. Концепція кроулінгу як засобу збирання інформації 54
2.2. Огляд існуючих програмних засобів для збирання інформації у веб-просторі 58

2.3. Визначення та аналіз вимог до кроулера 69
2.4. Архітектура та структура побудованого програмного забезпечення 72
2.5. Засоби розробки програмного продукту 85
2.6. Результати роботи розробленого програмного забезпечення 91
ВИСНОВКИ ДО РОЗДІЛУ 2 93
РОЗДІЛ 3. СТАТИСТИЧНИЙ ТА КЛАСТЕРНИЙ АНАЛІЗ ВЕБ-ПРОСТОРУ МЕТОДАМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ 95
3.1. Статистичні характеристики інформації у веб-просторі 97
3.2. Особливості застосування методів кластерного аналізу до сегментів веб-простору 103
3.3. Розробка інформаційної технології для збирання статистичної інформації та проведення кластерного аналізу у веб-просторі 111
3.4. Інтелектуальний аналіз статистичних даних сегментів веб-простору 116
3.4.1. Статистичний та кластерний аналіз даних сегменту edu.ua. 117
3.4.2. Статистичний та кластерний аналіз даних сегменту edu.pl 121
3.4.3. Статистичний та кластерний аналіз даних сегменту ac.il 125
3.5. Обговорення отриманих результатів 129
ВИСНОВКИ ДО РОЗДІЛУ 3 139
РОЗДІЛ 4. ОЦІНКА ОПТИМАЛЬНОЇ КІЛЬКОСТІ КЛАСТЕРІВ СКЛАДНИХ МЕРЕЖ НА ОСНОВІ ВИПАДКОВИХ МАТРИЦЬ 141

4.1. Основні відомості та твердження 141
4.2. Застосування теорії випадкових матриць 150
4.3. Застосування до оцінки оптимальної кількості кластерів . . . 152
4.4. Аналіз кластерної структури інтернет-мереж на основі випадкових матриць 158
4.5. Про один клас стохастичних випадкових матриць 163
4.6. Моделювання та порівняння з класичними результатами. . . 172
ВИСНОВКИ ДО РОЗДІЛУ 4 179
ЗАГАЛЬНІ ВИСНОВКИ 180
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 183
ДОДАТКИ 208
ДОДАТОК А. ДІАГРАМА ОСНОВНИХ КЛАСІВ КРОУЛЕРА ТА ЇХ ОПИС 208
ДОДАТОК Б. ЛІСТИНГ ЧАСТИНИ КОДУ ПРОГРАМИ. . 210
ДОДАТОК В. АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ 222
ДОДАТОК Г. СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ 226

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

- $\mathbb{N}$ – множина натуральних чисел;
- $\mathbb{R}^d$ – d-вимірний Евклідовий простір;
- $\mathbb{C}$ – множина комплексних чисел;
- β_X – сукупність борелівських множин в X;
- $\#\{B\}$ – кількість елементів в множині B;
- $\mathrm{A}^{(\mathrm{N})}$ – матриця розмірності $N \times N$;
- $A_{N\times M} -$ матриця розмірності $N \times M$;
- $\lambda_i(A)$ – i-те власне значення матриці A, причому власні значення будемо впорядковувати за спаданням абсолютних величин, тобто

$$|\lambda_1(A)| \geq |\lambda_2(A)| \geq \cdots \geq |\lambda_N(A)|;$$

- $e_i(A)$ – власний вектор матриці A, що відповідає власному значенню $\lambda_i(A)$;
- $\Pr(A)$ – ймовірність множини A;
- $\mu_N \rightrightarrows \mu$ – слабка збіжність розподілів μ_N до розподілу μ при $N \to \infty$;
- $\xi \sim Dist(\theta)$ – випадкова величина має розподіл $Dist$ із параметром θ;
- $\boldsymbol{N}(\mu, \Sigma)$ – нормальний розподіл із середнім $\mu \in R^d$ та коваріаційною матрицею $\Sigma \in R^{d\times d}$;
- $\boldsymbol{Pois}(\lambda)$ – розподіл Пуассона з параметром λ;
- $\boldsymbol{Exp}(\lambda)$ – показниковий розподіл із параметром λ;
- $i.i.d.$ – сукупність незалежних однаково розподілених випадкових величин;
- $Pr(\cdot)$ – ймовірнісна міра.

ВСТУП

Актуальність та обґрунтування теми дисертаційного дослідження.

Характерною рисою сучасного етапу розвитку суспільства є накопичення та обробка величезних обсягів інформації, для чого найчастіше використовують методи інтелектуального аналізу даних. Сьогодні ми живемо і працюємо в інформаційному просторі, який, без сумніву, являє собою складну мережу з притаманними таким структурам особливостями та зв'язками. Дослідження цих статистичних особливостей сприяє кращому розумінню організації таких мереж, їх використанню в різних компаніях та інших галузях аналізу суспільства. Статистичною основою досліджень служить адекватна сегментація складних мереж на організовані за різними критеріями структури меншої розмірності, які дозволяють спростити аналіз великих обсягів даних, дослідити структуру мережі в цілому та її кластерів зокрема.

Не дивлячись на існуючий математичний, алгоритмічний та програмний апарати у галузі інтелектуального аналізу складних мереж, залишається нез'ясованим цілий ряд питань, як от: оптимальний вибір кластерів складної мережі, вивчення статистичних особливостей підмереж (наприклад, окремих зон веб-мережі), методики збирання та аналізу інформації про такі мережі тощо.

Зважаючи на наведене вище, розробка та вдосконалення методів та інформаційних технологій збирання та інтелектуального аналізу даних вважається актуальною задачею.

Об'єктом дослідження є складні мережі та процес збору й обробки інформації.

Предметом дослідження є моделі та методи інформаційної технології збору та обробки статистичної інформації, отриманої з веб-простору.

Метою дисертаційного дослідження є проведення статистичного та кластерного аналізу окремих зон веб-простору на основі програмного

забезпечення власної розробки та нових підходів для визначення оптимальної кількості кластерів складних мереж.

Відповідно до поставленої мети в дисертаційній роботі розв'язуються такі **основні задачі**:

- провести аналітичний огляд існуючих програмних засобів для статистичного та кластерного аналізу інформації у веб-просторі;
- розробити програмне забезпечення з урахування сучасних засобів та методів розробки, яке включає аналітичний модуль статистичної обробки інформації;
- розробити інформаційну технологію збирання та обробки інформації у складних мережах;
- за допомогою розробленої технології виконати порівняльне дослідження статистичних характеристик та кластерної структури окремих зон веб-простору;
- використовуючи новий підхід до теорії дослідження графових структур, розробити метод визначення оптимальної кількості кластерів у складних мережах;
- дослідити адекватність запропонованого методу визначення оптимальної кількості кластерів за допомогою методів моделювання;
- впровадити результати дисертаційної роботи у діяльність кафедр та підприємств.

Методи дослідження. В основу дисертаційної роботи покладені методи інтелектуального аналізу даних, а саме: кластерний аналіз; класичні математичні методи (побудова графової структури мережі), статистичні методи (обчислення статистичних характеристик мережі); теорія матриць (побудова стохастичних матриць та дослідження їх спектральних характеристик).

Наукова новизна. В дисертаційній роботі

1) вперше:

– на основі сучасних методів розробки програмного забезпечення створено кроулер, який, на відміну від існуючих, містить аналітичний модуль інтелектуальної обробки інформації, підтримує контейнеризацію, роботу в багатопотоковому режимі, легко масштабується, і може бути використаний для аналізу споріднених мереж;
– розроблено інформаційну технологію збирання та інтелектуального аналізу даних у складних мережах;
– проведено порівняльні дослідження статистичних характеристик та кластерної структури українського освітянського сегменту edu.ua, польської підмережі edu.pl та ізраїльської академічної зони ac.il.

2) набуло подальшого розвитку:

– теорія дослідження графових структур: сформульовано і доведено твердження, які дозволяють оцінювати розподіл власних значень випадкових матриць та переносити класичні результати на матриці із слабко корельованими елементами; на основі розширення границь теорії дослідження графових структур розроблено алгоритм перевірки належності елементів до одного кластеру, що надає можливість визначити її оптимальну кластерну структуру.

Зв’язок роботи з науковими програмами, планами, темами.

Дисертаційне дослідження виконано на кафедрі програмного забезпечення комп’ютерних систем Чернівецького національного університету імені Юрія Федьковича. Її зміст відповідає тематиці науково-дослідних робіт: «Математичне та програмне забезпечення обчислювальних систем» (Державний реєстраційний номер 011U007046) та «Дослідження, моделювання та розробка програмного забезпечення складних динамічних систем» (Державний реєстраційний номер 0121U109232).

Публікації. За темою дисертації опубліковано 21,5 роботи, в яких подано результати досліджень. З них 6 статей у рецензованих виданнях (1 з яких – в журналі, що індексується у наукометричній базі SCOPUS, 5 – в

українських фахових виданнях, 1 – в розділі монографії міжнародного видання), у збірниках матеріалів міжнародних та всеукраїнських наукових конференцій – 15 робіт.

Особистий внесок здобувача. Дисертантка брала активну участь в постановці задачі, визначенні мети роботи та виборі методів досліджень, обговоренні та інтерпретації результатів, підготовці матеріалів до опублікування в усіх працях [1-22]. Так, в роботах [1; 5] проводила основні розрахунки, здійснювала аналіз отриманих результатів. У роботі [2] займалась питанням застосування теоретичного результату до визначення оптимальної кількості кластерів складних мереж, при дослідженнях структури веб-простору та Grid system. Моделюванням мережі зв'язків, що розподілені за законом Пуассона, визначенням оптимальної кількості кластерів дисертантка займалась у [6]. У працях [3; 4; 7] займалась розробкою та вдосконаленням програмного забезпечення. Результати дисертаційної роботи [8–22] доповідались і обговорювались на Всеукраїнських та Міжнародних наукових і науково-практичних конференціях.

Апробація результатів. Основні результати роботи доповідались та обговорювались на наукових семінарах кафедр програмного забезпечення комп'ютерних систем та математичних проблем управління і кібернетики, а також на Всеукраїнських та Міжнародних наукових і науково-практичних конференціях: «Актуальні проблеми інформаційних технологій, економіки та права» (Чернівці, 2013), «Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано– та мікроелектроніки» (Чернівці, 2013), «Інтелектуальні технології в системному програмуванні» (Хмельницький, 2014), «Інформаційні технології: наука, техніка, технологія, освіта, здоров'я (MicroCAD)» (Харків, 2019, 2020), «Проблеми інформатики та комп'ютерної техніки (ПІКТ)» (Чернівці, 2012 – 2014, 2016-2021)», «Сучасна стохастика: теорія та застосування V» (Київ, 2021).

Дисертантка брала участь у роботі Міжнародної літньої школи ECODAM (м. Ясси, Румунія, 2019 р.), де доповідались результати дисертаційного дослідження. У вересні 2022 року було зроблено доповідь на Міжнародному науковому семінарі «Advances & Challenges in Computing (A2C)», де було представлено основні наукові результати, які увійшли у дисертаційне дослідження.

Структура та обсяг роботи. Дисертаційна робота є завершеним дисертаційним дослідженням, загальним обсягом 229 сторінок (з них: 182 сторінки основного тексту, 25 сторінок – література, 22 сторінки – додатки). Дисертація складається зі вступу, чотирьох розділів з підрозділами, висновків, списку використаних джерел (252 позиції) та додатків.

РОЗДІЛ І. СКЛАДНІ МЕРЕЖІ ТА ЇХ РОЛЬ В ІНФОРМАЦІЙНОМУ ПРОСТОРІ

1.1. Теорія складних мереж

Складною називають мережу (граф) з нетривіальними топологічними особливостями, які не зустрічаються в простих мережах (таких як решітки або випадкові графи), але часто зустрічаються в мережах, які репрезентують системи реального світу (наприклад, комп'ютерні мережі, біологічні мережі, технологічні мережі тощо).

Теорія складних мереж (Complex Networks) вивчає характеристики мереж, враховуючи не тільки їх топологію, а й статистичні феномени, розподіл ваг окремих вузлів і ребер, ефекти протікання і провідності в таких мережах, як мережі струму, рідини, інформації тощо [1; 2].

Прикладами важливих концепцій складних мереж є мережі малого світу та безмасштабні мережі, моделі випадкових графів, моделі зростаючих мереж тощо.

Можна виокремити два важливих питання, які розглядаються у цій галузі:

– Чи існують певні об'єднуючі принципи, що лежать в основі топології складних мереж?

– Якою є поведінка складних мереж при комунікації з іншими складними мережами.

Одним із інтуїтивно зрозумілих підходів до охоплення глобальних властивостей таких складних систем є їх моделювання у вигляді графів, де вузли представляють динамічні одиниці, а зв'язки (ребра) відображають взаємодію між вузлами. Виявилося, що властивості багатьох реальних мереж істотно відрізняються від властивостей класичних випадкових графів [3]. Топологія та еволюція складних мереж реального світу контролюються різними принципами організації топології та динаміки. Дослідники вивчають структурні та топологічні проблеми складних мереж, наприклад:

характеристики складних архітектур взаємозв'язку для розуміння принципів об'єднання, які є основою складних мереж у реальному світі; моделювання зростання та відтворення структурних властивостей складних мереж [4-7].

У теорії складних мереж розглядають три основних напрями досліджень [3]:

– дослідження статистичних характеристик, які описують поведінку мереж;

– створення моделі мережі;

– прогнозування поведінки мережі при зміні її структури.

Активний розвиток такої області досліджень призвів до вивчення характеристик мережі, враховуючи не тільки її топологію, а й статистичні характеристики, які описують поведінку мережі при зміні структурних властивостей.

Сьогодні досліджено статистичні характеристики різноманітних мереж: енергетичних, транспортних, комп'ютерних, співавторства, авіаперевезень, соціальної мережі, всесвітньої мережі Інтернет, бізнес-мережі, мережі хімічних речовин, мережі цитувань документів або веб-сторінок тощо [1; 8-11].

1.2. Моделі складних мереж

З появою та розвитком теорії графів науковці відкрили багато фундаментальних величин і понять. Застосування та використання теорії графів поширилось на різні галузі науки: математики, фізики, соціології, біології, інформатики та кібернетики та ін. Виявилось [12], що найважливіші мережі, які виникають унаслідок людської життєдіяльности, та природні мережі мають специфічну структуру, яка характеризується розподілом ступенів вузлів із товстим хвостом (fat-tail distribution) і сильно відрізняється від структури вже дослідженого в математиці класичного випадкового графа. Зазвичай, ці мережі не статичні, а такі, що розвиваються, і для розуміння їхньої структури необхідно знати принципи їх розвитку. Існує багато

моделей, покликаних пояснити ті явища, що відбуваються в складних мережах.

Основними моделями, які спричинили сьогоднішнє розуміння складних мереж, є: класичний випадковий граф Ердоша–Рені (як уже згадувалося вище, мережі реального світу, як правило, не описуються цим графом), мережа тісного світу Уаттса–Строґаца та безмасштабна мережа Барабаші–Альберта. На відміну від двох перших, остання модель є прикладом зростаючої мережі. Як виявилося, саме моделі, що враховують зростання мереж, приводять до степеневих розподілів ступенів вузлів.

1.2.1. Класичний випадковий граф (модель Ердоша–Рені)

Вперше випадкові графи були вивчені угорськими математиками Полом Ердошем і Альфредом [13; 14]. Ця модель була введена для опису властивостей випадкового графа. При моделюванні досліджується мережа з загальною кількістю N вершин. Розглядають дві моделі класичного випадкового графа:

1) вважається, що M ребер розподілені довільно та незалежно між парами з N вершин графа;

2) фіксується ймовірність p, з якою може об’єднуватися кожна пара вершин.

У теорії графів модель Ердоша–Рені – це модель для генерації випадкового графа з постійною ймовірністю появи ребра між двома вершинами незалежно від інших ребер. Граф будується за допомогою з'єднання пар вершин випадковим чином. Кожне ребро включається в граф з ймовірністю p незалежно для кожної пари вершин (при $p = 0$ матимемо порожній граф, а при $p = 1$ – повнозв'язний граф). Розподіл ступенів для такої моделі є біноміальним. При великих значеннях N ймовірність того, що вершина матиме ступінь k , дорівнюватиме

$$P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!},$$

де $\langle k \rangle$ – середнє значення ступеня вузла (для першої моделі $\langle k \rangle = \frac{2M}{N}$, для другої моделі $\langle k \rangle = pN$). Для моделі Ердоша–Рені можна визначити й інші характеристики, наприклад:

– середня довжина найкоротшого шляху $\langle l \rangle = \frac{\ln(N)}{ln\langle k \rangle}$;

– коєфіцієнт кластерності $C \sim \frac{\langle k \rangle}{N}$.

Опис цієї моделі наведено в роботах: [1; 10; 12; 15-17].

1.2.2. Мережа малого світу (модель Уаттса–Строґаца)

Більшість моделей складних мереж являють собою граф. Д. Уаттс (Watts, D. J.) і С. Строгац (Strogatz, S. H) виявили феномен, характерний для багатьох реальних мереж, названий ефектом «тісного світу» [18]: відстань між будь-якими парами вузлів відносно мала, а в той же час рівень транзитивності або кластеризації відносно високий.

Модель тісного світу (малого світу) можна побудувати наступним чином (рис. 1.2.2.1) [12; 16]:

– розглядаємо одновимірний періодичний ланцюг із N вершин, замкнутий у кільце. З'єднаємо кожну вершину з k сусідами (k – додатне, парне число);

– уведемо процедуру перез'єднання – з ймовірністю p кожне ребро перекидаємо в довільну позицію. Така модель зводиться до канонічного ансамблю графів, оскільки кількість ребер є сталою, а значення ймовірності реалізації графів – різні (варіювання кількості випадково перекинутих ребер).

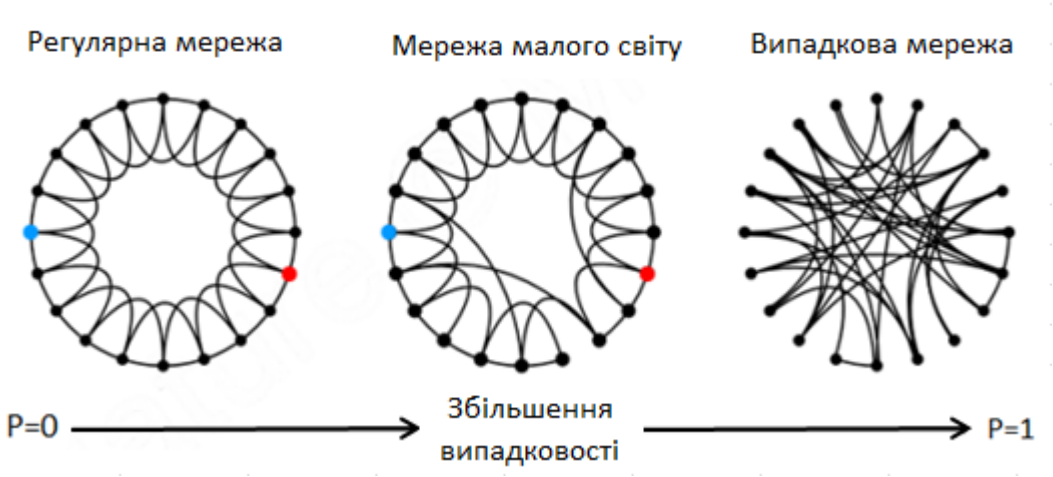


Рис.1.2.2.1. Модель Уаттса–Строґаца

Граф моделі тісного світу реалізується при малих значеннях p ймовірності перез'єднування, за яких розподіл ступенів вузлів не відрізняється від закону Пуассона. Модель Уаттса–Строґаца реалізує інтуїтивне уявлення про реально існуючі мережі, топологія яких не цілком регулярна, але й не випадкова.

Дослідження та опис моделі Уаттса–Строґаца проведені в роботах [3; 12; 15; 19-21].

1.2.3. Безмасштабні мережі (модель Барабаші – Альберт)

Багато великих мереж реального світу не мають масштабу. Більшість мереж реального світу описують відкриті системи, які ростуть шляхом постійного додавання нових вузлів. Сценарій побудови моделі, яка генерує безмасштабну мережу запропонували науковці А.-Л. Барабаші (Albert-László Barabási) і Р. Альберт (Réka Albert) [22]. Побудова цієї моделі ґрунтується на двох механізмах, що притаманні багатьом мережам. Це зростання та переважне приєднання:

1) Зростання: починаючи з невеликої кількості (n_0) вузлів, на кожному часовому кроці, додаємо новий вузол із $n < n_0$ зв'язками, які приєднуються до вже існуючих вузлів.

2) Переважне приєднання: при виборі вузлів, до яких приєднується новий вузол, ми припускаємо, що ймовірність $P(k_i)$ того, що новий вузол буде підєднаний до існуючого вузла i залежить від ступеня k_i вузла i, так що

$$P(k_i) = \frac{k_i}{\sum_j k_j}.$$

Підсумовування у знаменнику відбувається за всіма вузлами. Ключова особливість безмасштабної мережі (масштабно-інваріантної мережі) полягає в алгоритмі з'єднання вузлів. Новий вузол тим ймовірніше приєднається до деякого існуючого вузла мережі, чим більше інших вузлів вже приєдналося раніше до нього. Вузол тим швидше накопичуватиме нові зв'язки, чим у нього більше вже існуючих зв’язків, оскільки найімовірніше, нові вузли будуть приєднуватися саме до нього. Отже, особливість цієї структури – це наявність великої кількості вузлів, у яких мало зв'язків, і мала кількість вузлів із великою кількістю зв'язків. Як комп’ютерні симуляції, так і аналітичні розв’язки моделі Барабаші–Альберт дають степеневу асимптотику розподілу ступенів вузлів $P(k) \approx \frac{1}{k^\gamma}$ $(k \neq 0,\ \gamma > 0)$ із показником $\gamma = 3$. Дослідження підтвердили, що саме безмасштабні розподіли часто спостерігаються в складних реально існуючих мережах.

При дослідженні мереж реального світу виявилося, що інколи принцип переважного приєднання працює лише локально, тобто в множині певних вузлів [23]. Тоді в мережі утворюються так звані тісні світи (small worlds), або мережі Уаттса-Строґаца.

Безмасштабні мережі та їх характеристики досліджувались у роботах: [1; 5; 10; 12; 15-17; 24].

1.3. Приклади складних мереж

Дослідники складних мереж встановили істотні відмінності властивостей багатьох реальних мереж від властивостей класичних випадкових графів. Для розуміння структури мережі потрібно враховувати топологію та принцип еволюції мережі, тому до розгляду беруться не тільки

статичні, а й динамічні мережі. Сьогодні широкого використання в повсякденному житті набули соціальні та інформаційні мережі, цікавими для досліджень є технологічні мережі, а використання теорії складних мереж призвело до бурхливого розвитку біологічних мереж.

1.3.1. Соціальні мережі

Складні мережі знайшли також широке застосування при дослідженнях соціальних мереж. Досліджуються мережі наукової співпраці, мережі знайомств, мережа кіноакторів, онлайнові мережі та ін.

1.3.1.1. Онлайнові соціальні мережі

Набули популярності онлайнові соціальні мережі (Facebook, MySpace та ін.). У таких мережах вузли представляють людей, а ребрами є онлайн зв'язки з іншими людьми (соціальні відносини). У таких мережах реального світу динамічно з’являються вузли і ребра. Паралельно з популярністю онлайнових соціальних мереж зростає інтерес науковців до властивостей таких мереж, до збору статистичних даних про них, також цікавим є питання побудови моделей, що моделюють їх еволюцію [11].

Властивості моделі малих світів для соціальних мереж представлені у роботі [18] і є центральним поняттям у вивченні складних мереж. Властивість малого світу передбачає низьку середню відстань між вузлами (або діаметр) і високу кластеризацію, що спостерігалося в широкому спектрі складних мереж.

Все більше уваги науковців зосереджено на властивостях моделі малого світу та дослідженнях властивостей онлайнових соціальних мережах. Так, у роботі [25] було проведено раннє дослідження онлайнової соціальної мережі в Стенфордському університеті, де було з'ясовано, що вказана мережа має властивості моделі малого світу. У роботі [26] автори досліджували еволюцію онлайнових мереж Flickr та Yahoo. Виявилось, що середня відстань між користувачами фактично зменшується з часом, і що ці мережі

демонструють степеневий розподіл ступенів. Для аналізу мережі Facebook (вибірка 4,2 мільйона користувачів) у роботі [27] вивчалась модель обміну повідомленнями між друзями. Тут також виявлено підпорядкування степеневому розподілу ступенів і властивостям моделі малого світу. Подібні результати отримали науковці в [28; 29], де проводились дослідження різних онлайнових соціальних мереж (Cyworld, MySpace, Orkut, Flickr, YouTube, LiveJournal).

У роботах [29-31] досліджувались структури спільнот в соціальних онлайнових мережах (Facebook, Twitter тощо) завдяки доступності даних для аналізу. Аналіз соціальних мереж, представлених графами, відображено в роботах [26; 32]. Результати виокремлення структури спільноти з онлайнової мережі сильно залежать від статистичних і топологічних характеристик набору даних графа, таких як середній ступінь, коефіцієнт кластеризації та рівень фрагментації. Ці характеристики також залежать від алгоритму, який застосували для виокремлення спільноти. У статті [33] описано два різних алгоритми для отримання структури спільноти з соціальних мереж, представлених у вигляді графа зі збереженням структури (характеристик глобальної мережі).

Концепції інтелектуального аналізу даних у соціальних мережах, які подаються графовими структурами, побудова графів і окреслення тем для продовження досліджень у сфері соціальних мереж відзначені в роботі [34].

Застосування структурного підходу при моделюванні соціальних мереж розглядалося у роботі [35]. У ній було обґрунтовано необхідність застосування такого підходу – визначення найбільш важливих вершин, зв'язків, спільнот і країн, регіонів мережі, що розвиваються. Зазначалось, що такий аналіз надає можливість здійснювати огляд глобальної еволюційної поведінки мережі. Під час структурного аналізу та аналізу поведінки зв'язків автором використано методи статистичного аналізу, визначення спільнот, алгоритми класифікації. З’ясовано, що саме слабкі зв’язки є тим феноменом, який зв’язує мережу в єдине ціле. Досліджено ефект «малих світів». Для

виокремлення спільнот використано як спеціалізовані алгоритми, наприклад алгоритм кластеризації Маркова, так і просто поділ об'єктів за класом модульності.

1.3.1.2. Мережа наукової співпраці

Опис дослідження мережі наукової співпраці наведено в [36]. У цій мережі вершинами виступають науковці, а зв'язки – спільні опубліковані наукові праці. В роботі зроблено висновок про динамічні та структурні механізми, які керують еволюцією та топологією даної складної системи. Автори розкрили топологічні показники, які характеризували мережу і показали, що мережа не масштабується, а також – що еволюція мережі регулюється пріоритетним приєднанням, що впливає як на внутрішні, так і на зовнішні зв’язки. Однак, на відміну від більшості прогнозів моделі, середній ступінь збільшується з часом, а відокремлення вузлів зменшується.

У роботі [37] показано, що для мережі наукової співпраці розподіл ступенів вершин, близький до степеневого закону і такі мережі володіють властивостями мереж тісного світу (високий коефіцієнт кластерності та коротка дистанція між двома довільно обраними вершинами).

При дослідженні мережі наукової співпраці потрібно звернути увагу на деякі особливості при дослідженні мережі наукової співпраці, які можуть вплинути на результати:

- з якої галузі науки науковець (вузькоспеціалізована галузь чи ні);
- якого характеру публікація (теоретична чи експериментальна);
- кількість співавторів.

Виявилось, що розподіли параметрів мережі співавторства не відповідають чистому степеневому закону, а швидше – степеневому закону з експоненційним обрізанням [37]. Автор пояснює це кількома причинами: дані, які розглядались, взяті лише за певний часовий проміжок; природною обмеженістю активного робочого віку науковців тощо.

З практичної точки зору дослідження мереж наукової співпраці можуть бути корисними при визначенні наукових шкіл,аналізі наукових тематик, співпраці науковців всередині країни та за її межами, вплив на глобальну наукову спільноту, прогнозування розвитку науки в країні.

Так, у статті [38] досліджувався вплив структури мережі наукової співпраці на ефективність національних досліджень і розробок. Автори довели, що структура мережі співпраці впливає на продуктивність процесу науково-дослідних робіт та наукові публікації на рівні країни, виміряли показники ефективності досліджень і розробок за допомогою індексу продуктивності Малмквіста, пов’язаного з аналізом охоплення даних.

Науковцями вивчається вплив наукової співпраці на розвиток наукових напрямів. Такі дослідження проводились [39] та свідчать про те, що близькість між дослідниками та їхнє структурне розташування в мережі співпраці може впливати на продуктивність і результативність спільних досліджень. У статті проаналізовано мережі співавторства для трьох країн (США, Китаї та Індія) в галузі штучного інтелекту та машинного навчання, яка зараз набуває швидкого росту. Аналіз спирається на спостереження, отримані в результаті порівняння характеристик статистичних властивостей мереж, що розвиваються, щоб визначити властивості мережі, які сприяють високій дослідницькій продуктивності.

1.3.2. Інформаційні мережі

1.3.2.1. WWW мережа

Експоненційний ріст об’єму інформації в мережі ставить перед користувачами питання про раціональність її використання. Інформаційний шум, спам, дублювання інформації знижує ефективність роботи в Інтернеті без залучення інтелектуальних програмних засобів, які повинні вирішити проблеми навігації та пошуку, виходу на дійсно потрібну, корисну і достовірну інформацію. Очевидно, для цього необхідні знання про структуру WWW [40].

Бродер та ін.в роботі [8] описали методику дослідження, яке вони проводили, з визначення структури Web-простору в цілому. Була зібрана інформація про більше ніж 200 млн. веб-сторінок та декілька мільярдів посилань. Автори запропонували графову структуру Web-простору (рис. 1.3.2.1), яка базується на зв'язках кожної сторінки з більш сильно зв'язною компонентою. Вони розглядали кожну сторінку як вузол, а кожне гіпертекстове посилання як ребро на графі. Умовно простір можна розділити на такі основні компоненти: ядро (MAIN) – сильно зв’язна компонента, вхідні (IN) та вихідні (OUT) компоненти, тунелі (TUNNEL) та «вуса» (TENDRILS). З’ясувалось, що при значному збільшенні загального обсягу веб-ресурсу, пропорції цих компонент протягом кількох місяців залишалися незмінними. Топологія і характеристики моделі виявилися приблизно однаковими для різних підмножин веб простору, підтверджуючи тим самим, що структурні властивості веб-простору будуть розповсюджуватись і на його підпростори. [41-45].

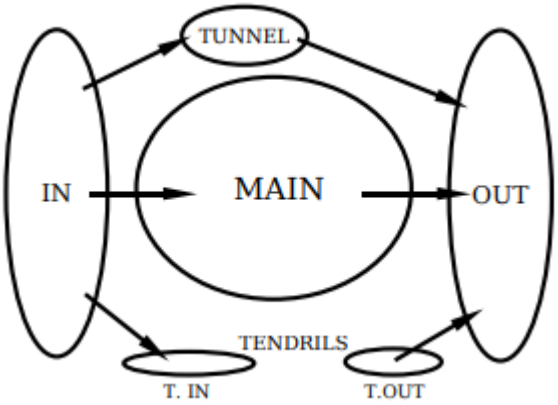


Рис. 1.3.2.1. Графічна структура веб-простору

Як бачимо, веб-простір складається з деякого ядра (MAIN) або, як його називають в роботі [46-47] опорної мережі (сукупності веб-сторінок зі степенем не менше двох), та периферійних сторінок, які посилаються на сторінки ядра (IN) або, на які є посилання з нього (OUT). Периферійні сторінки пов’язані між собою так званими тунелями (TUNNEL) та мають «вуса» (TENDRILS) зі сторінок, що не входять до структури ядра. [42-43].

Подібні дослідження структури WWW проводились також авторами [48]. У статті наведені розрахунки відносних розмірів усіх гігантських зв’язних компонент орієнтованого графа та показано, що великий сильно зв'язний компонент може бути менш стійким до випадкових пошкоджень, ніж великий слабко зв'язний компонент. Крім цього наведена графова структура WWW, продемонстровано, що кореляції між вхідними та вихідними ступенями згодом впливають на глобальну топологію мережі.

Проведений аналіз топології мережі показав, що великі вузли мають більше зв'язків між собою, ніж із малими вузлами, тоді як малі вузли мають більше зв'язків з великими вузлами, ніж між собою. Ці дослідження дають підстави думати, що залежність веб-простору від великих вузлів значно більше, ніж передбачалося раніше, тобто він ще більш вразливий відносно зловмисних атак [16].

Отже, WWW-простір являє собою безмасштабну мережу, що підпорядкована степеневому закону розподілу, для якого підтверджений феномен малих світів.

Web можна охарактеризувати з багатьох точок зору, використовуючи численні метрики. Це складне завдання, головним чином через його великі розміри та безперервну еволюцію [49]. Складно охарактеризувати реальну загальну характеристику, і отримати деякі статистичні дані в результаті аналізу глобального Web, тому краще зменшити масштаби дослідження, розглянувши менші області. Web містить частини зі специфічними характеристиками, які, враховуючі їх невелику присутність, не будуть впливати на загальну характеристику веб-простору. Однак, такі частини можуть зацікавити відносно великі спільноти, наприклад ті, що представляють національні або культурні групи. При проведенні таких досліджень автори [50] використовували іншу методику ніж А. Бродер та ін. в [8], а саме: розглядали лише зв’язки між окремими сайтами, що розміщені в національному домені. Розглядався неорієнтований граф і було виявлено, що 73% сайтів пов’язані з іншими сайтами. Цей результат свідчить, що при

розгляді меншої частини Web (національний домен Португалії), то зв'язність графа зменшилась (на противагу 91% з досліджень А. Бродера та ін). Також розглядались орієнтовані графи (по вхідних та вихідних зв'язках) та встановлено, що для графа по вхідних зв'язках, 45% сайтів мали зв'язки і були доступні, а 55% — ніяк не пов'язані (сайти-сироти). Для графа побудованого по вихідних зв'язках, з'ясувалось, що 95% сайтів не пов'язані з сайтами з національного домену (тобто містять внутрішні посилання), а 66% посилань не вказували на документи національного веб-простору [50].

1.3.2.2. Мережі цитування наукових праць

У роботі [15] наведено опис мережі цитувань наукових праць. Вершинами цих мереж є наукові праці, а ребра (орієнтовані) – цитати. Процес зростання мережі цитування дуже простий – майже кожна нова стаття містить ненульову кількість посилань на стару статтю. Це єдиний спосіб розширити мережу. В той самий час неможлива поява нових зв'язків між старими вершинами. Кількість посилань на певну статтю дорівнюватиме вхідному ступеню відповідної вершини мережі.

Дослідження мереж цитування наукових праць та їх характеристики також описані в роботах [51-53]. У вказаних працях емпірично досліджувався процес отримання цитувань статтями у зростаючій мережі цитування. Було продемонстровано, що нові цитати (вхідні ребра) розподіляються між документами (вершинами) з ймовірністю, пропорційною ступеню вершини. Це вказує на те, що в цьому графі цитувань діє лінійний механізм переважного приєднання.

1.3.3. Технологічні мережі

1.3.3.1. Інтернет мережа

Мережу Інтернет, як сукупність телекомунікаційних каналів та комп‘ютерів, які вони з‘єднують, вважають представником технологічних

мереж. Постійно змінюючись, Інтернет став складною мережею, яка характеризується неструктурованістю та безмасштабністю [1].

Вивчення топології Інтернету цікаве з багатьох причин. Найважливішими з них вважаються [23]:

− розробка технологій, що дозволять підвищити «продуктивність» Інтернету;

− покращення розуміння тенденцій зростання трафіку мережі, які впливають і на користувачів, і на глобальну інфраструктуру мережі;

− вдосконалення можливостей для Інтернет-провайдерів щодо керування мережами за допомогою поглибленого аналізу трафіку та засобів візуалізації;

− достовірне моделювання та забезпечення подальшого розвитку Інтернету.

Вивчаються різноманітні характеристики, за допомогою яких можна характеризувати досліджуваний сегмент, наприклад: яка структура сегменту та зв'язків, розподіл документів за IP-адресами, розподіл сайтів по серверах, розміри документів, визначення відсотків сайтів, в яких контент повторюється, відсотки мов, якими поданий контент сайтів, кількість користувачів.

Проводяться різноманітні дослідження топології мережі Інтернет. У роботі [54] Зубок В.Ю та Головін А.Ю запропонували для дослідження топології мережі Інтернет використання наступних методик:

1. Аналіз таблиць маршрутизації на рівні взаємодії автономних систем.

2. Метод traceroute – трасування маршрутів.

3. Аналіз баз даних реєстрів маршрутизації.

При використанні наведених методик, можна отримати наочну інформацію про топологію ділянки мережі і зробити висновки щодо її оптимальності, а також визначити «вузькі місця».

Але на сучасному етапі відкритість Інтернету в плані підключення нових мереж, а також використання основних принципів глобальної

маршрутизації спричинили ряд топологічних феноменів, що характеризують Інтернет як складну мережу. Вивченню феномену безмасштабності, самоподібності, малого світу приділяється значна увага.

На основі визначеної топології Інтернету автори [55] сформулювали два механізми, які необхідні для правильного моделювання топології Інтернету на рівні автономних систем: інтерактивне зростання нових вузлів і нових внутрішніх посилань, а також нелінійне переважне приєднання, де ймовірність переваги описується механізмом позитивного зворотного зв'язку. В результаті одержується модель, яка дає уявлення про еволюційну динаміку реальних складних мереж.

Також проводяться дослідження топології та основних характеристик локальних комп'ютерних мереж [12; 56; 57]. Локальні мережі в процесі розвитку та еволюції в часі проходять етапи становлення від класичного випадкового графа до безмасштабних мереж. Під час досліджень виділяють основні фактори, які впливають на процес росту локальної комп'ютерної мережі, формулюють послідовність дій, необхідних для побудови моделі системи. Здійснюється побудова імітаційної моделі, що, в свою чергу, надає можливість отримати зображення мережі для різних початкових умов, динамічно візуалізувати процес її структуризації та відслідковувати його в довільний момент часу. З допомогою імітаційного моделювання досліджують вплив початкових умов – напрямків розгалуження мережі, кількості вузлів з різними ступенями приєднання споживачів на ріст локальної мережі.

У праці [58] описується методика моделювання топологічних змін певного сегменту мережі Інтернет та порівняння параметрів мережі до та після змін. Авторами було з'ясовано, що попри високу вразливість до видалення «центральних» вузлів, Інтернет є стійким до навіть спрямованого видалення окремих зв'язків між будь-якими вузлами, незалежно від навантаження цих зв'язків.

1.3.3.2. Транспортна мережа

Теорія мереж знайшла застосування в повсякденному житті, зокрема, у вивченні транспортних мереж. У багатьох наукових працях досліджуються топологічні властивості різноманітних транспортних мереж [19; 59-64]: мережі аеропортів, залізничні мережі, електромережі, метро. Окремого розгляду потребують мережі громадського транспорту. Детальний огляд мереж громадського транспорту великих міст світу (Берлін, Париж, Рим, Лондон та ін.) і 22 польських міст проведено в [65-67]. Показано, що для названих міст притаманний степеневий розподіл ступенів вузлів. Водночас широко проаналізовано характеристики мереж (кластерність, посередництво, асортивність).

Щодо досліджень мереж громадського транспорту в Україні, то відзначимо роботу [68], в якій наведено систематичний огляд статистичних властивостей мереж громадського транспорту чотирьох міст західного регіону України: Львова, Тернополя, Івано-Франківська та Чернівців, розглянута топологія, локальні та глобальні характеристики цих мереж. Продемонстровано, що розподіл ступенів вузлів підпорядковується степеневому закону в L-просторі та експоненційному в P-просторі.

1.3.4. Біологічні мережі

У галузях біології та медицини застосування складних мереж до аналізу включають, наприклад, ідентифікацію ліків, визначення функції білка чи гена, розробку ефективних стратегій лікування різних захворювань або забезпечення ранньої діагностики розладів. Мережі білок-білкової взаємодії (PPI), біохімічні мережі, або метаболічні мережі є виділеними категоріями мереж у системній біології, які часто мають спільні характеристики та властивості [9]. Головним рушієм у галузі системної біології є розвиток розуміння того, як взаємодія між компонентами впливає на функціонування системи в цілому. Аналіз мережі – це підхід, який унікально підходить для виявлення закономірностей і принципів організації в різноманітних складних

системах. Застосування методів мережевого аналізу до досліджень біологічних мереж розглідаються в [69].

Визначення та дослідження різних властивостей мереж дає цінне уявлення про внутрішню організацію біологічної мережі, перерозподіл молекул між клітинними процесами, а також еволюційні обмеження, які сформували білкову чи метаболічну структуру [9].

Наприклад, з'ясовано, що метаболічна мережа демонструє властивість маленького світу. В метаболічних мережах часто трапляється, що шляхи кількох (трьох-чотирьох) реакцій пов'язують більшість метаболітів. Як наслідок, локальні зміни концентрації метаболіту, локальні збурення в цих мережах поширюватимуться по всій мережі.

У роботі [70] представлено систематичний порівняльний аналіз метаболічних мереж 43 організмів, які представляють усі три сфери життя. Показано, що, незважаючи на значні варіації в їх індивідуальних складових, ці метаболічні мережі мають однакові властивості топологічного масштабування та демонструють подібність до внутрішньої організації складних небіологічних систем. Отже, метаболічна мережа подібна для всіх живих організмів і відповідає принципам безмасштабних мереж (надійність та стійкість).

Автори роботи [71] досліджували мережу взаємодії білків, де вузли – це білки, а зв'язки – фізична взаємодія між двома білками. Представлено детальний статистичний аналіз взаємодій білків на основі кількох масивів даних великої пропускної здатності. Зроблено оцінки швидкості для двох ключових еволюційних процесів, що формують мережу: дублювання генів і посилення та втрата взаємодій через мутації в існуючих білках, які називаються динамікою зв'язку. Побудовано та емпірично обґрунтовано кількісну модель еволюції мереж взаємодії білків. Згідно з цією моделлю, динаміка зв'язку є домінуючою еволюційною силою, яка формує статистичну структуру мережі, а динаміка дублювання генів головним чином впливає на її розмір.

Мережі широко визнані як популярний метод представлення складних біологічних процесів шляхом фіксації взаємозв'язків між різними біологічними компонентами за допомогою бінарних взаємодій або зв'язків. Вивчення різноманітних баз даних, які містять дані про біологічні мережі, призводять до проведення досліджень розуміння динаміки захворювань з використанням мережевої модульності, такі дослідження описані [72].

1.4. Методи та засоби інтелектуального аналізу даних

Інтелектуальний аналіз даних (Data Mining) – це сучасна концепція, яка дозволяє шукати та аналізувати дані різної природи, у тому числі величезних обсягів, що можуть бути неточними, неповними, суперечливими, різнорідними [3]. Інтелектуальний аналіз даних – напрямок, пов'язаний з обробкою інформації та виявленням в ній закономірностей і тенденцій, які можуть застосовуватися при підтримці прийняття рішень.

Використання математичних та програмних методів і засобів в інтелектуальному аналізі даних дозволяє виявляти невідомі раніше закономірності і тенденції в даних, які неможливо або важко виявити при традиційному аналізі через великі обсяги даних або їх складність.

Великі обсяги даних (Big Data), їх складність, мережева природа, динаміка і різноманітність інформації привели до вибухового зростання кількості та потужності методів і засобів інтелектуального аналізу даних [3].

Інтелектуальний аналіз даних визначається як набір завдань, які використовуються для систематичного та автоматизованого виявлення раніше невідомої, неявної, прихованої інформації з даних, присутніх у базах даних. Інтелектуальний аналіз даних часто розглядається як синонім процесу виявлення (відкриття) знань. Ці два терміни взаємозамінні [73]. Процес виявлення знань описані у роботах [74; 75] і поданий на рис. 1.4.1.

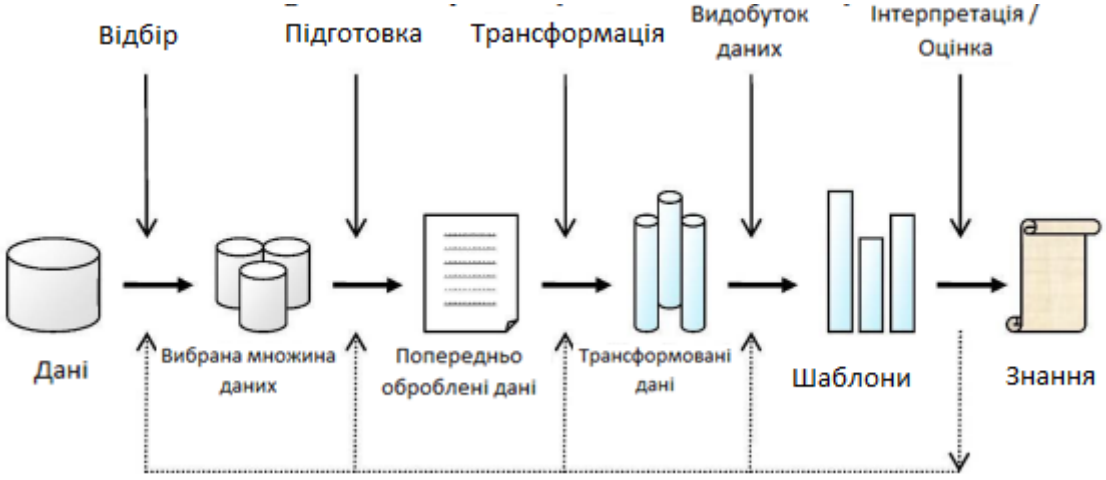


Рис. 1.4.1. Процес виявлення знань

Спочатку збираються відповідні дані для вирішення поставленого завдання. Дані можуть бути зібрані з різних джерел, таких як бази даних, текстові файли, сенсори, веб-сторінки тощо. Тоді, враховуючи мету видобутку знань, з цих даних робиться відбір. Після попередньої обробки вибраної підмножини даних, виконується процес трансформації, що перетворює отриманий набір у форму, до якої можна застосувати інтелектуальний аналіз даних, котрий включає використання різноманітних статистичних методів та алгоритмів машинного навчання для виявлення прихованих шаблонів і залежностей в даних. Одержані під час аналізу шаблони є ключовими елементами для отримання знань з даних. Цей процес допомагає виявляти тенденції та важливі взаємозв'язки, що дозволяють краще розуміти предметну область і приймати обґрунтовані рішення на основі отриманих результатів.

Можна виділити основні завдання, які найчастіше вирішує інтелектуальний аналіз даних [3], [73]:

– *Асоціація* (виявлення залежностей між пов'язаними подіями) – це процедура побудови асоціативних правил, які допомагають нам передбачити появу конкретного елемента на основі появи інших елементів у даних.

– *Класифікація* (віднесення об'єктів до одного із заздалегідь відомих класів). Ці класи визначаються до того, як дані перевіряються, це вважається технікою навчання під наглядом. Класи часто визначаються на основі атрибутів або характеристик даних, які необхідно класифікувати. Класифікація може бути якісною, кількісною, географічною та хронологічною.

– *Кластеризація* (групування об'єктів на основі властивостей, що описують сутність об'єктів). Кластеризація відрізняється від класифікації. У класифікації класи попередньо визначені, тоді як у кластеризації кластери не визначені заздалегідь, і ми не маємо про них попередньої інформації.

– *Регресія* (встановлення функціональної залежності між вхідними і неперервними вихідними змінними). Регресія допомагає зрозуміти, як на звичайне значення залежної змінної може вплинути зміна будь-якої з незалежних змінних, тоді як інші незалежні змінні залишаються фіксованими. Існує кілька типів регресії, як-от лінійна регресія, поліноміальна регресія, степенева регресія.

У сучасному інтелектуальному аналізі даних виділяють два основних класи моделей Data Mining:

1) *описові* (дескриптивні), які необхідні для кращого розуміння досліджуваної системи, відомих фактів і спостережень.

Описові моделі пов'язані із залежностями в наборі даних, взаємного впливу різних чинників, тобто базуються на побудові емпіричних моделей різних систем. У першу чергу до описових моделей належать асоціативні правила і кластери.

2) *Передбачувальні* (предиктивні), необхідні для розуміння нових фактів щодо системи.

Прогнозуюче моделювання дозволяє передбачати нові стани об'єктів, для чого використовуються алгоритми класифікації і регресії.

Для побудови розглянутих моделей використовуються різні математичні алгоритми і методи: класифікації, моделювання і прогнозування, кластерного аналізу, штучні нейронні мережі, статистичні методи аналізу, різноманітні методи візуалізації даних тощо [3].

Одним з найбільш ефективних методів інтелектуального аналізу даних, виявлення спільних рис та відмінностей, сьогодні є кластерний аналіз.

1.5. Методи кластеризації у дослідженні складних мереж

Для видобутку цінної інформації з великих і різних категорій набору даних використовують алгоритми та методи інтелектуального аналізу даних. Отримані дані потрібно перетворити у зрозумілу структуру для подальшого використання і проведення досліджень одним із методів інтелектуального аналізу. Мета техніки інтелектуального аналізу даних полягає в тому, щоб отримати інформацію з великого набору даних і перетворити її в прийнятну форму для додаткових цілей. Інтелектуальний аналіз даних можна виконувати, проходячи через різні фази [76].

При проведенні аналізу даних та їх дослідженні важливою задачею є процес кластеризації даних. Процес кластеризації [77-81] – це техніка інтелектуального аналізу даних, що має широку сферу застосування, як, наприклад, комп’ютерна графіка, сегментація зображень, комп'ютерний зір, нейронні мережі, розпізнавання образів, обробка зображень, аналіз генів, кросмаркетинг, кібербезпека та багато ін. Кластеризація полягає у виявленні природних груп схожих елементів у наборах даних. Можна виділити три фундаментальні питання, які необхідно вирішити в будь-якому типовому сценарії кластеризації, а саме [81]:

– яка модель набору даних або, який алгоритм кластеризації підходить краще для даного набору даних;

– який порядок моделі даних, тобто яка оптимальна кількість кластерів для такого набору даних;

– наскільки реальною чи якісною є сама кластеризація.

Вибір моделі для проведення кластеризації складається з двох основних кроків:

1) визначення методу кластеризації для конкретного набору даних;

2) визначення кількості кластерів та їх валідності.

Так в роботі [82] проведено оцінку якості кластерізації за допомогою трьох класичних алгоритмів у поєднанні з чотирма індексами валідності, досліджується їхня ефективність для визначення кількості кластерів. Порівняльну характеристику застосування семи індексів валідності кластерів при кластеризації даних відомими класичними алгоритмами та визначенні правильної кількості кластерів проведено в роботі [81]. Інший підхід описаний в роботі [83], де у дослідженнях використовуються не традиційні методи кластеризації на основі відстаней, враховуються міри подібності для зменшення складності обчислень, розглядається проблема якості кластеризації та вимірювання валідності кластерів за допомогою статистичних тестів.

Наведемо необхідний функціонал, яким має володіти кластерний аналіз [73]:

1. Можливість масштабовання: алгоритми кластеризації добре працюють з невеликими наборами даних, але вони повинні мати можливість працювати і з великими наборами даних, що містять мільйони елементів.

2. Одне сканування набору даних.

3. Здатність справлятися з шумом: набори даних складаються з помилкових, відсутніх і зашумлених даних, які надходять із кількох джерел. Алгоритми кластеризації повинні мати здатність надійно працювати з такими даними.

4. Можливість інтерпретації: алгоритми кластеризації повинні давати результати таким чином, щоб результати були зрозумілими та придатними для використання.

5. Здатність мати справу з довільними формами: алгоритми кластеризації можуть виявляти кластери довільної форми.

6. Здатність працювати з різними типами даних.

Загальноприйнятої класифікації методів кластеризації не існує, але можна виділити ряд підходів, при цьому деякі методи та алгоритми можна віднести відразу до декількох груп.

У кластерному аналізі можна виділити такі методи [73; 84-86]:

1. Ітераційні методи (Partitioning algorithms)

Такі алгоритми засновані на оптимізації деякої цільової функції, що визначає оптимальне в певному значенні розбиття множини об'єктів на кластери. Для алгоритмів цього методу потрібна попередня інформація про кількість кластерів і для отримання оптимального рішення використовуються техніка ітераційного переміщення.

Алгоритми, що підтримують ітераційні методи: DMCT, k-means, k-mediods, k-modes, Fuzzy k-means, PAM, CLARA, CLARANS.

2. Ієрархічні методи (Hierarchy algorithms)

Такі методи використовується для організації даних у набір груп, розміщених на різних рівнях, щоб створити ієрархію. Вони засновані на використанні бінарних дерев або дендрограм для ілюстрації різних кластерів. Існує два типи ієрархічних методів (рис. 1.5.1): агломеративний підхід («знизу вгору») і дівізімний підхід («згори вниз»). Ієрархічні методи кластерного аналізу використовуються при невеликих обсягах наборів даних. Перевагою ієрархічних методів кластеризації є їхня наочність. Недоліком ієрархічної кластеризації є те, що вони чутливі до викидів і шуму. Існує проблема визначення числа кластерів, яке іноді можна визначити апріорно. Однак у більшості випадків число кластерів визначається в процесі агломерації/поділу об’єктів.

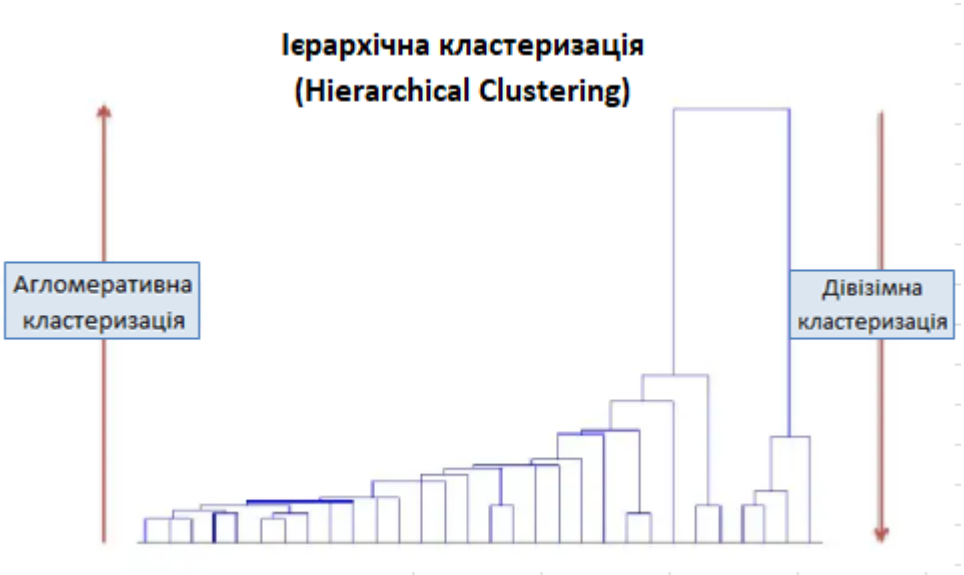


Рис. 1.5.1. Представлення ієрархічної кластеризації

Алгоритми, що підтримують ієрархічні методи:

1) агломеративні: AGNES, ROCK, CURE, CHAMELEON; BIRCH;

2) дівізімні: DIANA, MST.

3. *Грід-методи (Grid-based methods)*

Методи полягають у квантуванні об’єктів у грід-структуру, на якій виконуються всі операції для кластеризації. Вони використовують єдину сітку для розподілу всіх даних у комірки, а об’єкти даних, розташовані в цих комірках, символізуються статистичними характеристиками об’єктів. Ці методи мають високу швидкодію, оскільки переглядають дані лише один раз для обчислення значень.

Алгоритми, що підтримують грід-методи: STING, CLIQUE, ENCLUS, STIRR, WaveCluster.

4. Модельні методи (Model-based)

Використання моделі для знаходження кластерів, що найбільше відповідають даним.

Алгоритми, що підтримують модельні методи: EM Algorithm, Auto class, COBWEB, GMM.

5. Методи на основі щільності (Density-based)

У цих методах моделюють кластери як щільні області в просторі даних, розділені розрідженими областями. Вони засновані на тому, що для кожної точки даних у кластері має бути мінімальний критерій точок даних, присутніх на заданій відстані. Отже, точки даних, які знаходяться поза згаданою відстанню, розглядаються як шум або викиди. Кластери можна розглядати як щільні області щільності ймовірності в наборах даних.

Алгоритми, що підтримують методи на основі щільності: DBSCAN, DBCLASD, OPTICS, DENCLUE, SUBCLU.

6. Графові методи (Graph based methods)

Є багато різних підходів до проведення кластеризації графів [87], [84], [88], [44], [89]: засновані на оптимізації модулярності; розмітці графів; випадкових блуканнях; спектральні алгоритми.

Алгоритми, що підтримують графові методи: OPPOSUM, SNN Similarity Clustering, Infomap, Louvain, DEMON, EVSA, PIC.

За типами кластерні алгоритми можна розділити наступним чином:

1. Ієрархічні та пласкі.

Пласкі алгоритми:

– починають роботу розділенням елементів по групах випадковим чином;

– ітеративно покращують результат.

Ієрархічні алгоритми:

– знизу-вгору (агломеративні);

– згори-вниз (дівізімні).

2. Чіткі та нечіткі

Нечіткі алгоритми:

– елемент може належати до кількох кластерів.

Чіткі алгоритми:

– кожен елемент належить строго до одного кластеру.

Зазвичай, усі алгоритми ґрунтуються на різних вхідних параметрах, таких як кількість кластерів, критерій оптимізації/побудови, умова

завершення, міра близькості тощо. Різні алгоритми кластеризації, застосовані до одного набору даних, дають різні результати:

1) не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд досить прийнятних критеріїв, а також ряд алгоритмів, що не мають чіткого критерію, але здійснюють досить розумну кластеризацію безпосередньо при побудові. Всі вони можуть давати різні результати;

2) число кластерів, як правило, заздалегідь невідоме і встановлюється відповідно до деякого суб'єктивного критерію;

3) результат кластеризації істотно залежить від метрики, вибір якої, як правило, також суб'єктивний і визначається дослідником.

Кластерний аналіз є одним із основних методів аналізу даних, який широко використовується для багатьох практичних застосувань. Хороший метод кластеризації створить високоякісні кластери з високою внутрішньокластерною та низькою міжкластерною подібністю [90]. Якість результату кластеризації залежить як від міри подібності, яка використовується методом, так і від його реалізації.

Деякі популярні та широко поширені у використанні алгоритми кластеризації інтелектуального аналізу даних, такі як ієрархічні алгоритми та алгоритм k-means, часто застосовуються до масивів даних великої розмірності. Зменшення розмірності – це перетворення багатовимірних даних у значущі представлення зменшеної розмірності. Наприклад, для покращення ефективності роботи алгоритму k-means автори [91] застосували метод головних компонент до вхідного набору даних і отримали зменшений набір даних для подальшого проведення процесу кластеризації алгоритмом k-means.

Незважаючи на те, що було розроблено багато ефективних алгоритмів кластеризації, для більшості з них потрібно знати оптимальну кількість кластерів. Так у роботі [92] проводиться кластеризація ієрархічним алгоритмом, а оптимальну кількість кластерів визначають як міру, яка базується на квадраті суми помилок у ході виконання алгоритму

кластеризації. Таке вимірювання кластеризації показало хорошу продуктивність і розбиття на кластери в евклідовому просторі згідно з експериментальними результатами. Крім того, автори зазначають, що цей показник можна використовувати для оцінки бажаної кількості кластерів для методів не ієрархічної кластеризації.

До методів, що знаходять оптимальну кількість кластерів для проведення кластеризації можна віднести метод силуету [93], метод «ліктя» [94], k-core decomposition [95-97] та ін.

В останні роки алгоритми спектральної кластеризація стали одними із найпопулярніших та ефективних сучасних методів кластеризації [98]. Алгоритми спектральної кластеризації працюють шляхом перетворення вихідного простору даних у новий простір меншої розмірності, де проблему кластеризації можна легше вирішити вже відомим методом кластеризації, наприклад k-means. Метод базується на аналізі множини власних значень матриці подібності. До переваг методів спектральної кластеризації можна віднести його ефективність при утворенні кластерів, що не мають фіксованої форми чи шаблону.

Зважаючи на наведене вище, актуальним вважається дослідження методами інтелектуального аналізу складних мереж та їхніх характеристик. Зокрема, розглянути та охарактеризувати реальну підмережу веб-простору: розробити спеціалізоване програмне забезпечення для збирання статистичних даних, провести статистичний та кластерний аналіз цих сегментів, визначити оптимальну кількість кластерів.

ВИСНОВКИ ДО РОЗДІЛУ 1

У даному розділі наведено ключові відомості з теорії складних мереж, описано основні моделі (Ердоша-Рені, Уаттса– Строґаца, Барабаші–Альберт), процес їх побудови, моделювання та динаміку розвитку.

Розглянуто та проаналізовано приклади реальних складних мереж (онлайнові, наукової співпраці, WWW, цитування наукових праць, Інтернет мережа, транспортна і біологічні мережі) та їх особливості.

Проаналізовано методи та засоби інтелектуального аналізу даних в контексті складних мереж. Проведено класифікацію та огляд методів кластерного аналізу.

Основні наукові результати розділу опубліковані в працях [42-45; 47; 89; 96; 97].

РОЗДІЛ 2

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ЗБИРАННЯ ДАНИХ

Пошукова система – це комплекс програм, основним функціональним призначенням якого є здійснення пошуку інформації в мережі Інтернет, збереження відомостей про неї в базах даних і надання користувачу переліку посилань відповідно до його пошукового запиту, тобто такого, що відповідає заданим критеріям пошуку.

Головним завданням пошукових систем є здатність надавати користувачам швидко і у простому вигляді саме ту інформацію, яку вони шукають.

На даний час найпопулярнішими пошуковими системами, що задовольняють основні вимоги (релевантність результату, зручний та простий інтерфейс, функціональні можливості для розширення або звуження пошуку), є: Naver, AskJeeves, DuckDuckGo, Bing, Baidu, Yahoo!, Google та ін.

Пошук інформації з допомогою пошукової системи відбувається в 3 етапи [99]:

1. *Сканування.* На цьому етапі пошукові роботи виявляють сторінки в мережі. Пошукова система постійно відшукує нові та оновлені сторінки, щоб додати їх до списку відомих сторінок (виявлення URL-адреси). Після виявлення сторінки, сканер перевіряє її вміст. Пошукова система використовує деякий алгоритм, для визначення переліку та частоти сканування сторінок.
2. *Індексація.* Після процесу сканування сторінки текстовий вміст обробляється, аналізується та позначається атрибутами, метаданими, які допомагають пошуковій системі зрозуміти про що йдеться. Це надає можливість видаляти дублікати сторінок і збирати інформацію про вміст.

3. *Пошук і ранжування*. Коли користувач уводить запит, то система в індексі знаходить збіг із відповідними сторінками та повертає найбільш релевантні результати на сторінці результатів пошукової системи. Система проводить ранжування вмісту за низкою факторів, таких як авторитетність сторінки, зворотні посилання на сторінку, ключові слова, які містить сторінка.

Для автоматичного й ефективного перегляду інформації в мережі Інтернет найчастіше використовують так званий кроулер – це веб-сканер, який є частиною пошукової системи. Сьогодні, коли відбувається швидкий ріст інформації, а саме кількості інтернет-сторінок в інформаційній мережі, людям стало надзвичайно складно і затратно за часом відшуковувати потрібну інформацію. Для спрощення процедури пошуку і потрібна допомога кроулерів. Їх основною задачею є перегляд гіперпосилань та їх індексування. Оскільки веб-сканування є основним етапом у роботі пошукових систем, то питання збільшення точності та швидкодії пошуку стало важливим напрямком досліджень у галузі сканерів і привертає значну увагу багатьох дослідників [100], [101].

Основні принципи роботи та характеристики веб-сканерів, їх класифікацію та ключові властивості описано у роботах [101; 102].

У роботі сучасних пошукових машин можна відмітити деякі недоліки, які мають негативний вплив на процес збору інформації [103; 104]:

1. Досить часто у вебі розглядають публічний контент, в той час як існує прихований або глибокий контент, який доступний звичайному користувачу, але недоступний для пошукових машин. Насамперед, це сторінки з динамічним контентом, доступ до якого захищений паролем і який генерується безпосередньо за конкретним запитом користувача, або сторінки.

2. Роботи часто негативно впливають на пропускну здатність мережі, хоча деякі з них враховують цей вплив, певною мірою зберігаючи швидкісні параметри мережі.

Актуальними є питання, що стосуються пропускної здатності кроулерів, оскільки роботи можуть охоплювати відносно велику частину веб-простору за короткі періоди. Вузькі місця можуть виникати локально, враховуючи високий рівень використання пропускної здатності, особливо, якщо пошуковий робот використовується часто чи взагалі постійно, або в години пікового навантаження мережі. Проблема ускладнюється, якщо частота запитів на ресурси не регулюється.

3. Відомо, що так звані «інтенсивні» запити (послідовні HTTP-запити до одного сервера без затримки) вимагають дуже багато ресурсів від сервера для поточної реалізації HTTP. Тут знову, нерегульований робот може викликати проблеми. Вдало підібрані затримки й алгоритми обходу мережі можуть допомогти вирішити цю проблему.

4. Ще одне питання, яке викликає занепокоєння, – засмічення (спотворення) лог-журналів сервера. Дії робота, що індексує весь сайт, будуть мати своє відображення у журналах подій, а це, у свою чергу, ускладнить можливість відрізнити активність робота від активності звичайних користувачів.

5. Існує невелика кількість роботів-шкідників (їх ще називають неетичними роботами), основним завданням яких є ускладнити роботу сервера. До таких завдань відносять розсилку спаму, що додатково навантажує систему електронної пошти, викрадання великих списків адрес електронної пошти, які можуть бути продані рекламодавцям, порушення авторських прав шляхом копіювання цілих сайтів.

2.1. Концепція кроулінгу як засобу збирання інформації

Основною задачею кроулерів є навігація по всесвітній павутині та завантаження контенту веб-сторінок. Пошукова машина розрізняє й оцінює лише контент – скільки разів зустрічаються ключові слова, в яких розділах сайту вони знаходяться, наскільки близько до початку сторінки вони розташовані. Кроулери можуть використовуватись як самостійно, так і в

складі пошукових машин. Більшість кроулерів обробляють контент веб-сторінки для подальшого індексування сайту, тобто працюють не тільки зі структурою знайдених посилань, а й з іншими частинами сторінки, які можуть містити цікаву для користувача інформацію. Кроулери можуть використовувати спеціальні комп'ютерні програми, які налаштовані на вивантаження контенту потрібних веб-сайтів і наступне вивчення гіперпосилань, що знаходяться на цих сайтах. Можливість вибору простору дослідження, контроль за кроулерами та керування ними залишаються за дослідником – це і є їхньою перевагою. Але є і недоліки кроулерів в порівнянні з пошуковими машинами – це велике споживання ресурсів для збирання даних, а також значно менше охоплення WWW-простору для проведення аналізу [104].

Згідно з серією публікацій Najork, M. та Heydon, A. найбільш поширеним є таке означення кроулера [105;106]:

Веб-кроулер – це програма, яка за допомогою однієї або кількох початкових URL-адрес завантажує веб-сторінки, пов'язані з цими URL-адресами, видобуває будь-які гіперпосилання, що містяться в них, і рекурсивно продовжує завантажувати веб-сторінки за цими гіперпосиланнями. Веб-кроулери є важливими компонентами будь-яких аналітичних програм, котрі обробляють велику кількість веб-сторінок [105-106].

Незважаючи на простоту ідеї, розробка високопродуктивних веб-кроулерів створює серйозні проблеми через розмір мережі. Для того, щоб просканувати значну частину сегменту мережі за прийнятний проміжок часу, веб-кроулери повинні завантажувати велику кількість сторінок за незначний проміжок часу. Набір URL-адрес, які ще не просканово, і набір виявлених URL-адрес – зазвичай, не вміщуються в оперативну пам'ять, тому необхідно використовувати ефективні засоби збереження даних у постійній пам'яті. Також необхідно забезпечити незначне навантаження на будь-який окремий веб-сервер [105; 106].

Побудова веб-кроулера, здатного масштабуватися в залежності від кількості оброблюваних даних, є надзвичайно складною інженерною задачею. Для її розв’язання, кроулер повинен бути розподіленою системою з декількох компонент, здатних завантажувати сторінки паралельно. З іншого боку, кроулеру необхідно підтримувати спеціальну політику, яка обмежує обсяг трафіку, спрямованого на конкретний веб-сервер. Так, наприклад, можна заборонити одночасні запити до того ж самого веб-сервера, або реалізувати період очікування, пропорційний до часу останнього завантаження, перед повторним з’єднанням із даним веб-сервером [106].

У деяких архітектурах веб-кроулерів процеси завантаження сторінок розподілені, а основні структури даних – набір відкритих URL-адрес і набір URL-адрес, які потрібно завантажити, – обслуговуються однією компонентою. Такий підхід є простим але слабко масштабованим, зрештою центральні структури даних стають вузьким місцем [106].

Альтернативою є розділення основних структур даних на декілька компонент, максимально ізольованих одна від одної. Поділ URL-адрес між компонентами можна здійснити за іменем хоста. Це дозволить здійснювати планування сканування, враховуючи політику щодо обсягу трафіку, спрямованого на будь-який конкретний веб-сервер, в межах однієї компоненти [106].

Після того, як гіперпосилання було знайдено на веб-сторінці, веб-кроулер повинен перевірити, чи зустрічалася ця URL-адреса раніше, щоб уникнути додавання кількох екземплярів однієї URL-адреси до набору URL-адрес, які очікують на сканування. Такий набір зазвичай занадто великий, щоб підтримувати його в оперативній пам’яті (враховуючи, що середня URL-адреса має довжину приблизно 100 символів), тому для його реалізації необхідно використати чергу. Це дозволить регулювати навантаження компонент для завантаження веб-сторінок, а також визначати пріоритет сканування [106].

Враховуючи дослідження, проведені науковцями при розробці кроулерів [105-113], концепцію процесу кроулінгу можна описати наступним чином:

- вибирається довільним чином вхідна URL-адреса,
- завантажується відповідна веб-сторінка, з якої видобуваються наявні гіперпосилання для подальшого завантаження,
- рекурсивно повторюється вищеописаний процес.

Отже, якісний кроулер повинен володіти наступними властивостями [113]:

- *Надійність*: оскільки веб-сканування передбачає завантаження багатьох різних веб-сторінок, то веб-сканери повинні бути відмовостійкими та мати змогу відновити і продовжити свою роботу в разі виникнення різноманітних помилок або інших проблем.
- *«Ввічливість»*: кроулеру необхідно підтримувати спеціальну політику, яка обмежує обсяг трафіку, спрямованого на будь-який конкретний веб-сервер.
- *Розподіленість*: веб-сканери мають вміти виконувати сканування на кількох віртуальних машинах у розподіленому режимі.
- *Масштабованість:* структура веб-сканерів повинна дозволяти прискорення процесу сканування, шляхом використання додаткових машин або збільшення пропускної здатності.
- *Продуктивність та ефективність*: веб-сканер повинен мати можливість використовувати різні системні ресурси (наприклад: процесори, пристрої зберігання даних, пропускну здатність мережі) ефективно.
- *Актуальність*: реалізація можливості веб-сканера працювати в безперервному режимі.
- *Розширюваність*: веб-сканери повинні мати архітектуру, яка дозволяє легко вносити необхідні зміни.

Далі проведемо огляд розвитку програмних засобів для збирання інформації у веб-просторі.

2.2. Огляд існуючих програмних засобів для збирання інформації у веб-просторі

Наведемо перелік кроулерів загального призначення з коротким описом їх важливих характеристик та особливостей:

The Wanderer (Matthew Gray, 1993) – перша пошукова система Інтернету, яка здійснювала пошук по вебсайтам, була написана мовою Perl з метою вимірювання розміру World Wide Web (WWW). Незважаючи на те, що The Wanderer був, ймовірно, першим веб-роботом і, завдяки власній системі індексування, явно мав потенціал стати універсальною пошуковою системою WWW, розробники мали іншу мету, а саме – пошук нових сторінок. The Wanderer активно використовувався до кінця 1995 року [110; 114].

Excite (1993) – перша пошукова система, яка дійсно проводила аналіз контенту сторінки (ідея базувалась на використанні статистичного аналізу відношень слів). На початку такий аналіз був призначений для пошуку по сайту і передбачена можливість налаштування та створення власного каналу, де виводиметься тільки та інформація, яка цікава користувачу [115].

На відміну від інших пошукових систем, Excite пропонує кілька унікальних і особливих функцій, які відрізняють його від конкурентів, таких як Google. Наприклад, алгоритм пошуку Excite спеціально розроблений для отримання найкращих і найрелевантніших результатів кожного запиту користувача. Крім того, Excite визначає пріоритетність популярних тем на основі того, що шукають інші користувачі, а не покладається виключно на традиційні веб-сканери, як багато його конкурентів. Нарешті, ця пошукова система також дозволяє легко фільтрувати та звужувати результати пошуку за різними факторами, такими як мова чи країна походження. Загалом, Excite пропонує швидкий і комплексний пошук, який продовжує приваблювати

користувачів навіть на сучасному переповненому онлайн-ринку [111; 115; 116].

RBSE (David Eichmann, 1994) – перший опублікований веб-кроулер. Розроблений для проведення дослідження структури WWW та індексування знайденого корисного. Він базується на двох програмах: перша програма, Spider – підтримує черги в реляційній базі даних, і друга програма – Mite, що є WWW ASCII браузером, який завантажує сторінки з Інтернету [117-119].

RBSE був першим веб-кроулером, який індексував документи за вмістом. За допомогою програми Mite він отримує документи і використовує чотири механізми локального пошуку:

- пошук у ширину за заданою URL-адресою,
- пошук спочатку з обмеженою глибиною за заданою URL-адресою,
- пошук у ширину за невідвіданими URL-адресами в базі даних,
- пошук з обмеженою глибиною за невідвіданими URL-адресами в базі даних.

Однак, RBSE так і не став публічним індексом, його головною метою була побудова веб-індексу для певного домену, використання профілів домену для ідентифікації документів та індексування їх локальних сусідів [117-119].

Yahoo! Directory (David Filo and Jerry Yang, 1994-теперішній час) – створений каталог з веб-сторінок, який поруч з URL-адресою містив їх опис, складений веб-майстрами або редакторами, які проглядали сторінки. У відповідь на запит каталог виконує пошук по цих описах. У каталогах не використовується безпосередньо програма-кроулер, а тому вони не можуть в автоматичному режимі виявляти зміни веб-сторінок.

Yahoo намагається не відставати від Google. Станом на початок 2011 року частка Yahoo на світовому ринку становила 12%, це було максимальне значення якого вона досягала за історію свого існування. З цього року

прослідковується тенденція до її постійного зниження. Зокрема, у 2022 цей показник становив близько 3%.

У 2016 році хакери змогли отримати доступ до даних і паролів усіх трьох мільярдів облікових записів Yahoo, що пояснює причину посилення заходів безпеки. Компанія нарешті визнала проблему після трьох попередніх зломів між 2013 і 2014 і 2016 роками.

Ще одна особливість Yahoo, яка виділяє його серед інших пошукових систем, – це Yahoo News, що надає користувачам найновіші новини на такі теми, як розваги, політика, міжнародні справи та технології. Завдяки зручним у використанні категоріям і вдалому макету Yahoo News швидко став одним із найпопулярніших джерел актуальної інформації [111; 115; 120].

WebCrawler (Pinkerton, 1994) був використаний для складання першого загальнодоступного повнотекстового індексу підмножини the Web. Він базується на lib-WWW для завантаження сторінки, та іншій програмі для розбору та впорядкування URL-адрес при дослідженні веб-графа. Він також працює в режимі реального часу, слідуючи за посиланнями на основі подібності посилань до запиту.

WebCrawler реалізує найпростішу стратегію сканування і використовує чергу FIFO, скануючи посилання в тому порядку, в якому вони виявлялися. При такому підході**,** коли черга стає заповненою, сканер може додати лише одне посилання із просканованої сторінки. Оскільки він не використовує жодних знань про інформацію, яка сканується, його ефективність буде нижче у порівнянні з будь-яким із більш складних алгоритмів [115; 118; 121-123].

WorldWideWebWorm (O. McBryan, 1994) – кроулер, який використовується для створення простого індексу назв документів і посилань. Він використовував алгоритм індексування інформації заснований на зважених посиланнях (weighted links) та статичних описах, що надавало кращу оцінку важливості сторінок [115; 118].

Lycos (Michale Mauldin, 1994) – одна з перших пошукових комерційних систем, володіла функціоналом подібним до WebCrawler, але також проводила ранжування результатів, базуючись на релевантності та дозволяла користувачам налаштовувати параметри пошуку для отримання кращих результатів. Однією з популярних функцій було надання веб-розробникам можливості додавати сторінки в пошуковий індекс у режимі реального часу.

Lycos має чудову репутацію в галузі пошукових систем. Його критично важливими сферами є електронна пошта, веб-хостинг і сайти соціальних мереж, як-от YouTube або Facebook, для творців розважального контенту, які хочуть, щоб їх сайт отримував більше трафіку від Lycophiles (фанатів).

Lycos пропонує широкий спектр функцій і послуг для користувачів у всьому світі. Деякі з його ключових функцій включають швидкий, надійний пошук, персоналізовані результати на основі вподобань користувача та розширені параметри фільтрації для більш цілеспрямованих результатів. Крім того, Lycos також надає інші послуги, такі як оновлення новин, облікові записи електронної пошти, форуми тощо [111; 115; 124; 125].

У системі Infoseek (Steve Kirsch, 1994) впроваджено технології ранжування сторінок за релевантністю. Це дало користувачам кращі результати пошуку, оцінюючи інформацію на основі ваги ключових слів та структури зв'язків між сторінками. Недоліком цієї реалізації було те, що Infoseek мав функцію «миттєвого додавання», яка дозволяла будь-кому додавати свої сторінки до його пошукового індексу за лічені секунди. Це відкривало (двері для пошуку) можливість для поширення спаму: люди просто переписували свої сторінки, доки не досягали вершин рейтингу [111; 115; 126].

Internet ArchiveCrawler (M. Burner, 1997) – кроулер розроблений з метою архівування періодичних знімків великої частини Інтернету. Він використовує кілька процесів у розподіленому режимі, і фіксоване число веб-сайтів обробляється кожним процесом. Internet ArchiveCrawler також відслідковує зміну DNS-записів, зберігаючи архіви з трансляцією імен в IP-

адреса [127]. Головною технічною проблемою при архівуванні вебу є той факт, що велика кількість веб-сторінок є важкодоступною; вони називаються «глибоким Інтернетом» і можуть бути в 400-500 разів більшими за поверхневий Інтернет. Інша технічна проблема пов'язана з неузгодженістю зібраних веб-сторінок. Це трапляється, якщо під час процесу збирання веб-сканером (що може тривати кілька тижнів) частини веб-сайту оновлюються, а веб-вміст у верхній частині початкової URL-адреси більше не збігається з вмістом на рівнях, які знаходяться нижче, тому отримана колекція не є узгодженим представленням веб-сайту [112; 128].

Web SPHINX (R. Miller and K. Bharat, 1998) складається з бібліотеки класів Java, які реалізують багатопотоковий веб-пошук сторінок і HTML-парсинг, та графічного інтерфейсу користувача для встановлення початкового URL, отримання завантажених даних і для здійснення пошуку [129].

WebSPHINX розроблено для досвідчених користувачів Інтернету та Java-програмістів, які хочуть сканувати невелику частину Інтернету (наприклад, окремий веб-сайт) автоматично.

WebSPHINX не призначений для сканування всієї мережі. Пошукові системи зазвичай використовують розподілені сканери, що працюють на серверах із повним мережевим каналом і розподіленою файловою системою або базою даних для керування межами сканування та зберігання даних сторінок. WebSPHINX призначений більше для особистого використання, щоб сканувати, можливо, сотню чи тисячу веб-сторінок [130].

GoogleCrawler (S. Brin and L. Page, 1997) – мова йде лиш про ранню версію архітектури, що базувалась на C++ і Python. Кроулер інтегрований з процесом індексування (алгоритм ранжування PageRank), оскільки аналіз тексту було зроблено для повнотекстового індексування, а також для отримання URL. В реальності існує URL-сервер зі списками URL-адрес, які необхідно обробити декількома процесами кроулінгу. Під час аналізу знайдені URL-адреси передаються URL-серверу, який перевіряє, чи дана

адреса раніше оброблялася. Якщо ні, то URL-адреса додається в чергу URL-сервера. [118; 131; 132].

CobWeb (daSilva et al., 1999) – кроулер з розподіленою та високомасштабованою архітектурою, використовує центральний "планувальник" та серії розподілених "колекторів". Колектори аналізують завантажені веб-сторінки і відправляють виявлені URL-адреси планувальнику, який, у свою чергу, закріплює їх за певним колектором. Планувальник забезпечує пошук відповідно до спеціальної політики, щоб уникнути перевантаження веб-серверів. Кроулер написаний на Perl.

Головними характеристиками CoBWeb є ефективність, надійність та “ввічливість” у використанні спільних ресурсів. Незважаючи на ефективність поточної реалізації CoBWeb, є деякі моменти, де її можна ще вдосконалити. Зокрема, це стратегія планування, яка дозволить зменшити вартість операцій з черги, але збереже дотримання стандарту для виключення роботів [133].

Mercator (Heydon and Najork, 1999-2001) являє собою модульний веб-кроулер написаний на Java. Його модульність виникає від використання змінних «модулів протоколу» і «модулів обробки». Модулі протоколу визначають спосіб завантаження веб-сторінок (наприклад, за HTTP), модулі обробки визначають спосіб обробки веб-сторінок. Стандартний модуль обробки просто аналізує сторінки для отримання нових URL-адрес, а інші модулі обробки можуть бути використані для індексування тексту сторінок, або для збирання статистики з Інтернету.

Mercator – це додаток із високими можливостями конфігурації зі змінними компонентами, які дозволяють налаштувати його для різних сценаріїв. Ефективно обробляє великі набори даних і забезпечує гнучкість за допомогою компонент, що підключаються за потреби. Mercator зберігає централізовану архітектуру, яка обмежує його можливість до легкого масштабування. Навіть якщо кілька екземплярів Mercator запускаються на кількох віртуальних машинах, вони повинні спільно використовувати структури даних на диску. Через це розподіл кожного кроку алгоритму на

кількох машинах є нетривіальним, оскільки вимагає певної форми крос-машинної синхронізації [118; 134; 135].

WebFoundation (Edwards et al., 2001) представляє собою розподілений, масштабований модульний кроулер, схожий на Mercator, але написаний на C++. Метою розробки було не лише індексування Інтернету, але й створення його локальної копії. Особливістю цього кроулера є "контролер", який координує групи "ant"-машин. У разі неодноразового завантаження сторінок, для кожної сторінки виводиться змінений рейтинг і застосовуються методи нелінійного програмування для визначення ступеню оновленості веб-сторінки. Такий підхід рекомендовано використовувати на ранній стадії сканування, а потім перейти до порядку з єдиним обходом, в якому всі сторінки відвідуються з однаковою частотою [118; 132].

PolyBot (Shkapenyuk and Suel, 2002) представляє собою розподілений кроулер написаний на C++ і Python, який складається з «кроулера-менеджера», одного або більшої кількості "завантажувачів" і одного або більше “DNS resolvers”. Зібрані URL-адреси додаються в чергу на диску, і в подальшому обробляються в пакетному режимі.

Кожен із компонентів відповідає за чітко визначене завдання. Така архітектура легко масштабується з точки зору обчислень шляхом додавання додаткових компонент.

Тим не менш, хоча така архітектура й демонструє певний ступінь конфігурації, це не завжди достатньо, оскільки конфігурація можлива шляхом заміни відносно великих блоків. Зокрема, програма сканування не розділена на менші компоненти, які підключаються, хоча певне функціональне узагальнення або декомпозиція були б можливими у випадку застосування, наприклад, системи сканування.

Передача даних PolyBot може бути досить значною за розміром, що обмежує розподіл системи у глобальній мережі. Таким чином, PolyBot не може ефективно використовувати географічне розташування серверів [132; 135].

Web RACE (D. Zeinalipour-Yazti and M. D. Dikaiakos, 2002) складається з модулів сканування і кешування, реалізованих на Java, і використовується в якості частини більш загальних систем. Коли такі системи отримують запити від користувачів для завантаження веб-сторінок, то кроулер діє як смарт-проксі-сервер. Найбільш відмінна риса Web RACE полягає в тому, що в той час як більшість кроулерів починають “обхід” з певного набору URL-адрес, Web RACE може самий отримувати нову точку входу для сканування.

Web RACE продемонстрував відмінні результати щодо швидкодії у локальній мережі, але використання цього веб-кроулера у глобальній мережі потребує додаткових оптимізацій [118; 136].

FAST Crawler (Risvik and Michelsen, 2002) використовується як швидка пошукова машина, що дотримується децентралізованого підходу, коли кожній віртуальній машині в кластері призначається окрема частина мережі для сканування. Він являє собою розподілену архітектуру, де кожна машина має «документ-планувальник», який підтримує чергу документів, котрі будуть завантажені за допомогою «документ-процесора», і зберігає їх у локальній підсистемі. Кожен кроулер спілкується з іншими кроулерами через «дистриб'ютор» – модуль, що відповідає за обмін інформацією при гіперпосиланні. При відповідному масштабуванні FAST Crawler може створити індекс значної частини Інтернету менш ніж за тиждень [137].

WIRE (Baeza-Yates and Castillo, 2002) – веб-кроулер, написаний на C++. Включає декілька політик планування завантаження сторінок і модуль для генерації звітів і статистичних даних про них, внаслідок чого може бути використаний для отримання характеристик вебу.

Вказаний веб-кроулер має розподілену архітектуру. Він щільно інтегрований з індексом знань кожної сторінки. Політика сканування передбачає перевагу якості-свіжості інформації проти кількості сторінок [118; 138].

Ubicrawler (Boldi et al., 2004) представляє собою масштабований, розподілений кроулер, написаний мовою Java, основними особливостями

якого є незалежність від платформи, відмовостійкість, децентралізація кожного завдання (відсутність центрального процесу). Він складається з декількох ідентичних "агентів", кожному з яких призначаються посилання для обробки за допомогою спеціальної функції. Ця функція забезпечує відсутність перекриття для посилань, що гарантує відсутність подвійного обходу сторінок. Кроулер призначений для досягнення високої масштабованості і низької чутливості до збоїв.

Тим не менш, таке використання агентів може стати недоліком у деяких випадках. Так агенти не використовують спеціальні механізми зберігання чи кешування даних, вони жертвують цим задля низького використання ресурсів. Зокрема, завдання, що вимагають інтенсивних обчислень, які можна масштабувати шляхом подальшого розподілу, важко включити в модель. Наприклад, робота з обробки зображень не може бути зроблена агентом, відмінним від того, який завантажив зображення простим способом. Насправді UbiCrawler не має високий ступінь конфігурованості [109; 118].

На сьогодні існує багато розроблених пошукових систем, кроулерів, фреймворків та бібліотек, функціональним призначенням яких є індексування, збір даних з веб-сторінок, видобування певної інформації з сайтів та ін. Наприклад: Heritrix, Apache Nutch, BeautifulSoup, Selenium, Scrapy, Crawler4j та ін.

Heritrix (Her 2003; Mohr et al. 2004) – це веб-сканер з відкритим вихідним кодом. Він написаний на Java і також має модульну архітектуру. Це дає наступні переваги: по-перше, для роботи Heritrix немає накладних витрат на мережу. По-друге, він не потребує встановлення, користувачеві потрібно лише включити файл Heritrix.jar до свого проєкту. Нарешті, Heritrix працює на одній віртуальній машині Java (JVM). Це означає, що використання Heritrix може зменшити такі проблеми, як керування конфігурацією, відстеження ефективності сканера та обмеження швидкості. Однак він також має деякі недоліки: архітектура Heritrix є складною та громіздкою, і не

призначена для додавання парсерів та розширюваності. Зазначимо, що Internet Archive запропонував та реалізував (у Heritrix) і створив стандартний формат для архівування веб-контенту, який називається WARC [113; 118; 132; 139].

Apache Nutch (Doug Cutting, 2003-теперішній час) – це розширюваний і масштабований проєкт кроулера з відкритим кодом. Він повністю реалізований мовою програмування Java, має високомодульну архітектуру, що дозволяє розробникам створювати плагіни для синтаксичного аналізу медіа-типу, пошуку даних, запитів і кластеризації. Nutch має повну бібліотеку для розподіленого режиму. Тому його використовують як інструмент для створення розподіленого сканера. Однак, час налаштування є тривалим і громіздким [113; 140].

BeautifulSoup (Leonard Richardson, 2004-теперішній час) – це потужна бібліотека Python, яка може бути дуже корисною для збору даних із веб-сторінок. Назва BeautifulSoup добре пояснює призначення цього пакету: його можна використовувати, щоб відокремити та вилучити дані, необхідні користувачеві, з файлів HTML і XML, шляхом створення дерева об’єктів Python. Він може отримувати дані за допомогою різних засобів, таких як теги та NavigableString, тобто для отримання даних необхідно буде встановити сторонні модулі. Також знадобиться інший модуль для сканування веб-сторінок, створених за допомогою JavaScript, оскільки BeautifulSoup дозволяє лише навігуватися між файлами HTML або XML [141; 142].

Selenium є інструментом для автоматизації браузера, що дозволяє керувати веб-браузером за допомогою коду. BeautifulSoup можна використовувати з більшою ефективністю завдяки використанню протоколу Selenium Webdriver для написання сценаріїв, які можна виконувати в таких популярних браузерах, як: Chrome, Internet Explorer, Firefox і Safari. Використання поєднання Selenium з BeautifulSoup дозволяє користувачам збирати більші набори даних під час автоматизованого кроулінгу веб-сайтів у великих масштабах [141; 142].

Scrapy (Zyte (колишня Scrapinghub), 2008-теперішній час) – це швидка високорівнева платформа веб-кроулінгу, яка використовується для сканування веб-сайтів, включаючи розподілений парсинг (HTML, XML та інших форматів даних), і вилучення структурованих даних із їх сторінок. Дану платформу можна використовувати для багатьох цілей – від аналізу даних до моніторингу й автоматизованого тестування. Scrapy реалізована мовою програмування Python; головним призначенням є сканування конкретного веб-сайту. На додаток до фактичного пакета Python, інсталяція Scrapy постачається з інструментом командного рядка. Кроулерами керують за допомогою спеціальної оболонки. Існуючі кроулери також можуть бути завантажені в Scrapy Cloud. Це означає, що навіть великі сайти можна сканувати без використання власного комп'ютера чи домашнього підключення до Інтернету. Крім того, можна налаштувати власний веб-сервер за допомогою програмного забезпечення з відкритим кодом Scrapy [108; 143].

Crawler4j (Yasser Ganjisaffar, 2010-2020) – це кроулер із відкритим кодом на Java, який забезпечує простий інтерфейс для сканування Інтернету. Використовуючи його, можна налаштувати багатопотоковий веб-сканер за кілька хвилин. Він може працювати в розподіленому режимі, тому легко сконфігурувати одночасну роботу декількох кроулерів. До недоліків Crawler4j можна віднести не оптимальне використання пам'яті через надмірне створення глобальних змінних, а також непотрібне дублювання даних. Значна кількість коду, написаного при реалізації кроулера, не дозволяє легко вносити зміни, усувати недоліки в роботі Crawler4j [107; 144].

Огляд існуючих кроулерів та історія розвитку кроулінгу наводять на думку, що кроулери – це спеціалізоване програмне забезпечення, яке, зазвичай, створювались в якості інструменту для вирішення конкретних завдань залежно від специфіки досліджень та функціональних вимог, а саме:

- пошук нових сторінок у веб-просторі (в ширину, в глибину),
- індексація веб-сторінок,

– створення звітів про структуру та вміст сайтів,
– збір інформації в конкретному домені тощо.

Оскільки більшість кроулерів на момент проведення досліджень були комерційними продуктами, то це унеможливлювало їх використання для проведення власних досліджень. Крім того, дуже важливим було об’єднання різних характеристик та алгоритмів пошуку, підлаштованих під наші дослідження, в одній системі. Тому ми розробили власне програмне забезпечення (кроулер), яке б вирішувало потрібні нам завдання, враховуючи загальні підходи до організації процесу кроулінгу, концепцію та його архітектуру.

2.3. Визначення та аналіз вимог до кроулера

При розробці якісного програмного забезпечення слід звернути увагу на кілька ключових етапів: визначення мети та вимог продукту, планування проєкту, вибір технології, проєктування й архітектуру та інше [145; 146]. Так, у роботі [147] зазначається, що ключовим фактором забезпечення ефективного застосування програмних продуктів та однією із основних потреб є досягнення високих значень показників якості програмного забезпечення. Дослідження, які проводились у роботах [148; 149], демонструють, що чинники якості сучасних програмних та інформаційних систем є менш залежними від написання програмного коду, але суттєво залежать від формування та формулювання вимог.

Розробка програмного забезпечення вимагає ґрунтовного підходу, який включає в себе вимоги до розроблюваного продукту [146].

Для розробки якісного програмного забезпечення, інформаційних систем та технологій потрібно визначити функціональні та нефункціональні вимоги, тобто визначити характеристики, якими повинен володіти програмний продукт, щоб відповідати потребам поставленої задачі [103].

Виокремлюють такі типи вимог (специфікацій) до програмного забезпечення:

– функціональні вимоги, які охоплюють поведінку системи та те, що система має робити, а що ні;
– нефункціональні вимоги, які описують, як система це буде робити.

Опишемо вимоги (специфікацію) до системи, яка розроблена, а саме: функціональні та нефункціональні вимоги [103].

Функціональні вимоги:

– задавати вхідні URL-адреси та глибину сканування;
– генерувати список посилань для завантаження;
– фільтрувати на основі URL-адрес, у якому шаблони визначають імена хостів та/або шляхи, що цікавлять;
– паралельно завантажувати веб-сторінки за гіперпосиланнями;
– парсити завантажені веб-сторінки та видобувати гіперпосилання;
– перетворювати отримані посилання відповідно до правил нормалізації посилань;
– забезпечувати збереження відповідної структури посилань;
– здійснювати проведення статистичного та кластерного аналізу різних зон веб-простору.

Нефункціональні вимоги:

Планування

Політика планування визначає порядок завантаження URL-адрес, що очікують у черзі. При широкому скануванні розмір черги може стати дуже великим. У нашому випадку система має підтримувати чергу з щонайменше 10 тисяч URL-адрес.

Політика відвідування сторінок

Потужні веб-кроулери можуть легко створити навантаження навіть на високоякісні веб-сервери. Тому дуже важливо, щоби кроулер мав змогу обмежити навантаження на веб-сервер при скануванні веб-простору.

Обробка веб-сторінок

Видобуток гіперпосилань є основною метою при обробці веб-сторінок. Тому веб-кроулер повинен дозволяти додавання модулів для роботи з Javascript, Flash та іншими медіа-типами, які мають вбудовані посилання.

Визначення дублікатів

Іншою важливою метою є усунення дублікатів. Кроулер повинен розпізнати під час сканування, що було виявлено дублікат, для вживання спеціальних кроків. Повинна бути можливість змінювати визначення «дубліката» (від дуже суворого до дуже вільного).

Також необхідно уникнути зберігання сторінок під час сканування, оскільки воно виконується лише для збору посилань або статистики. У цих випадках посилання та дані видобуваються кроулером і зберігаються безпосередньо у сховищі, що значно зменшує вимоги до дискового простору.

Інтерактивність

Мережа стає все більш інтерактивною, для отримання веб-сторінок користувачі повинні часто заповнювати форми, а браузери мають підтримувати автентифікацію та механізми cookie. Тому кроулер повинен підтримувати файли cookie та прості схеми паролів.

Стабільність

Кроулер повинен бути стійким до можливих перерв у роботі. Тривале сканування може бути тимчасово припинено з кількох причин: з’єднання WAN стає недоступним, виявлено помилку в програмному забезпеченні або потрібно змінити політику сканування. Для підтримки таких переривань має бути можливість перезапустити сканування з даних, що зберігаються у постійній пам'яті. В ідеалі такі перезапуски мають бути без втрат даних, наприклад, якщо документ завантажується, і завантаження не закінчено в момент зупинки кроулера, він має спробувати завантажити цей документ знову після перезапуску. Крім того, кроулер повинен забезпечити ведення

логування для моніторингу своєї поведінки та діагностики будь-яких проблем.

Протоколи

Має бути можливість легкого додавання нових протоколів завантаження до кроулеру.

Висока пропускна здатність

Один екземпляр кроулера повинен мати можливість завантажувати 100000 документів на день. Крім того, має бути реалізована можливість розподілу сканування між 5-10 такими екземплярами, щоби досягти 5-10-кратного покращення пропускної здатності сканування.

Портативність

Кроулер повинен стабільно працювати на Linux. Там, де це можливо, веб-кроулер також має бути здатний виконувати невеликі та тестові запуски на Windows.

2.4. Архітектура та структура побудованого програмного забезпечення

Кроулер складається з декількох модулів: Downloader, Content parser, Link extractor, Url resolver, Data access module, Url manager, Logging, Analytical module, Url queue, Download queue, Data store. Архітектуру написаного кроулера зображено на рис. 2.4.1 [103; 104].

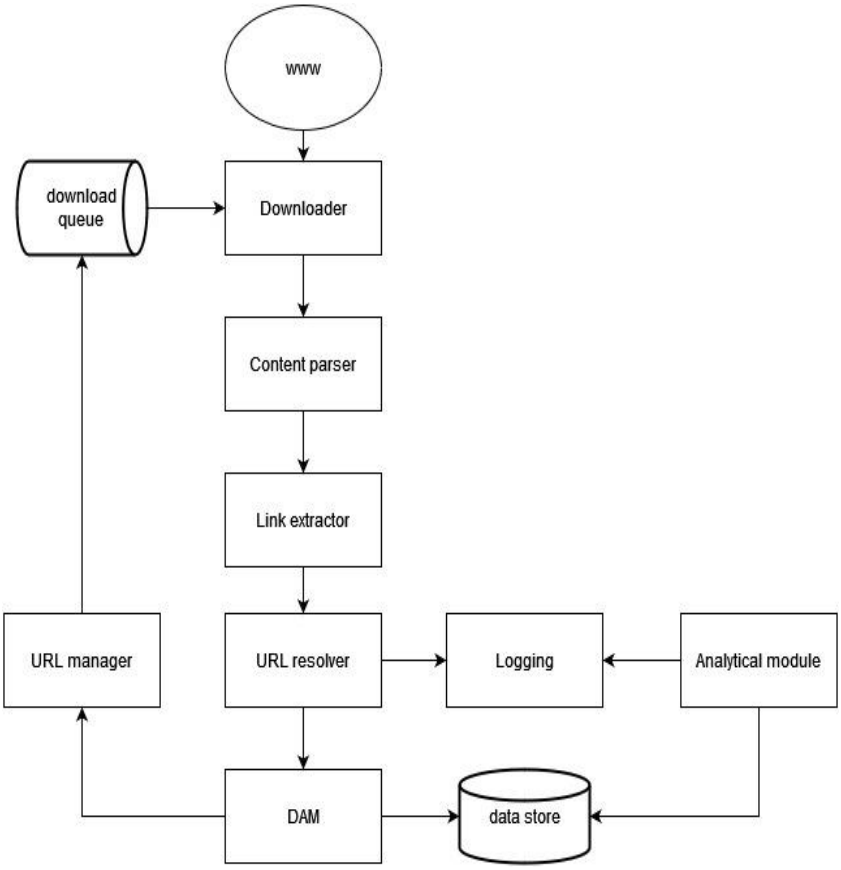


Рис. 2.4.1. Архітектура розробленого кроулера

Основною перевагою такої модульної архітектури нашого кроулера є те, що вона забезпечує гнучкість конфігурування та стабільну роботу під навантаженням. Наведемо короткий опис призначення модулів розробленого програмного забезпечення [103; 104].

Download queue (черга завантажень) – забезпечує збереження гіперпосилань для подальшого завантаження.

Downloader (заватажувач) – завантажує контент сторінки за вказаним URL, обробляє заголовок запиту до сторінки, дозволяє здійснювати завантаження контенту у багатопотоковому режимі.

Content parser (парсер контенту) – розбирає контент веб-сторінки, виділяючи структуру html-тегів та інформацію, яка знаходиться всередині тегів.

Link extractor (видобувач посилань) – знаходить та видобуває з контенту завантаженої сторінки посилання, враховуючи зовнішні та

внутрішні, а також посилання Flash-об’єктів. Для об’єктів HTML він повинен спробувати отримати посилання з Javascript та інших скриптових мов.

Url resolver (оброблювач посилань) – перетворює отримані посилання відповідно до правил нормалізації посилань. Такі перетворення запобігають завантаженню однакових ресурсів більш, ніж один раз.

Data access module (модуль доступу до даних) – забезпечує обробку та збереження відповідної структури посилань у базі даних.

Url manager (менеджер посилань) – генерує список посилань для завантаження. Цей модуль реалізує політику планування щодо порядку завантаження URL-адрес, відповідає за усунення дублікатів та визначає пріоритети при завантаженні сторінок.

Data store (сховище даних) – використовується для збереження веб-графу, тобто сторінок, їх структури посилань, різноманітної статистики.

Analytical module (аналітичний модуль) – надає інтерфейс для проведення статистичного та кластерного аналізу веб-графу.

Logging module (модуль логування) – агрегує всі лог-журнали, створені різними компонентами, щоб уніфікувати всі події в системі.

Проведемо детальний опис компонент системи.

1) ***Logging module*** *(модуль логування)*

Logging module є агрегатором усіх лог-журналів, створених компонентами системи. Усі компоненти записують події у свої лог-журнали, а модуль логування опитує їх і зберігає в центральному сховищі даних.

Таким чином, користувачі системи можуть отримати уніфіковане уявлення про всі події, що відбуваються в системі, і їм не потрібно переглядати різні файли лог-журналів у різних форматах для отримання узгодженого уявлення про певний потік або помилку.

Крім того, оскільки журнали зберігаються в сховищі даних із можливістю запиту, інші програми можуть отримати до нього доступ і проводити запити будь-яким способом, від простих запитів на основі REST

до інформаційних панелей із великою кількістю інформації. Дані є, і кожен може отримати до них доступ.

Архітектуру модуля логування зображено на рис. 2.4.2. Це класична багаторівнева архітектура, яка містить відмінність – верхній рівень є рівнем опитування в цій компоненті.

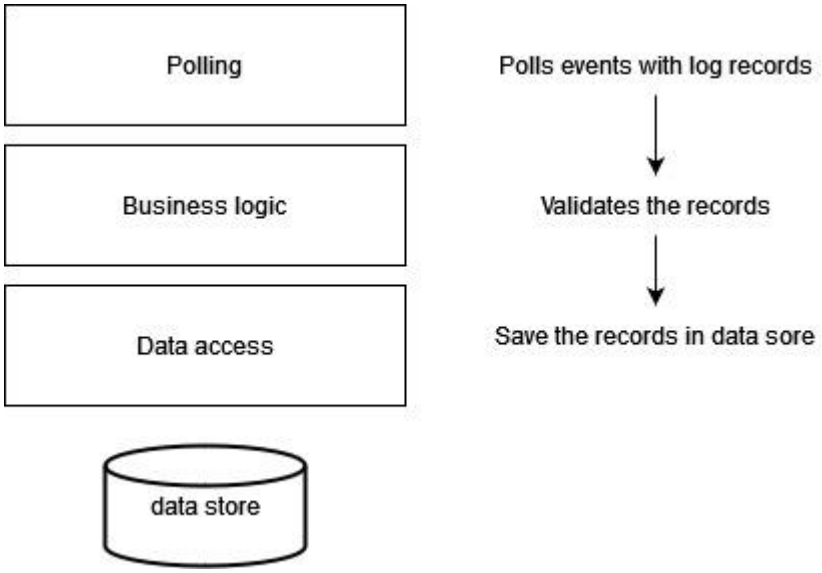


Рис. 2.4.2. Архітектура модуля логування

Причина внесення змін полягає в тому, що модуль логування не є веб-інтерфейсом або веб-додатком, він не надає жодного інтерфейсу, а натомість має рівень опитування, що керує роботою з подіями, які містять інформацію для логування.

Важливо, що кожен рівень має свою чітко визначену роль.

2) Download queue *(черга завантажень)*

Черга завантажень виконує роль буфера між завантажувачем та менеджером посилань. Це дозволяє даним компонентам працювати незалежно, як частина більшої системи. Зрештою, слабка зв'язність забезпечує очевидні переваги в розширюваності та масштабованості. Так, менеджер посилань має змогу планувати завдання для декількох екземплярів завантажувача одночасно.

3) ***Downloader*** *(завантажувач)*

Завантажувач використовується для завантаження веб-сторінок у багатопотоковому режимі. Він під'єднаний до черги, що містить посилання для завантаження, отримує їх та завантажує відповідні сторінки.

У випадку помилок модуль повинен повторити спробу завантаження. Якщо запит усе ще не вдається, модуль може зачекати та зробити ще одну спробу. Якщо необхідно, цей процес можна повторити зі збільшенням затримок між повторними спробами, доки не буде зроблено максимальну кількість запитів. Затримку можна збільшити поступово або експоненційно, залежно від типу збою та ймовірності того, що його буде виправлено протягом цього часу. Максимальна кількість запитів та час очікування між запитами задається в конфігураційному файлі модуля.

Завантажувач також відповідає за дотримання політики завантаження, що визначається менеджером посилань, а саме – за регулювання обсягу трафіку, спрямованого на будь-який конкретний веб-сервер.

Для ініціалізації процесу сканування модуль надає REST API, що дозволяє користувачу вказати початкові URL-адреси.

Детальний опис REST API завантажувача наведений в таблиці 2.4.1.

Таблиця 2.4.1. Опис REST API завантажувача

Ресурс	Шлях	Статус код
Почати сканування	POST /rest/crawler/start	200 OK

Наведемо приклад запиту для ініціалізації процесу сканування:

```
{
   "entry_points": [
      "http://chnu.edu.ua/index.php?page=ua"
   ]
}
```

Архітектуру завантажувача зображено на рис. 2.4.3.

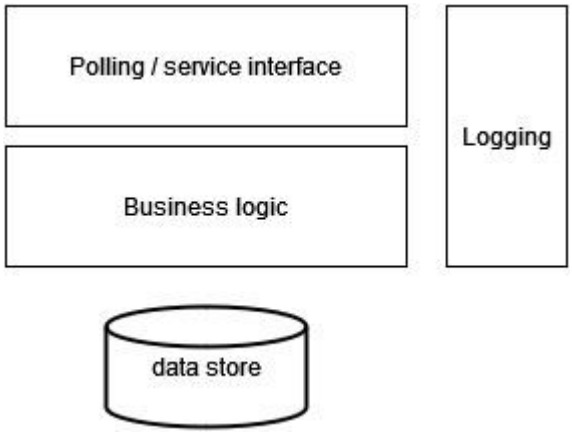


Рис. 2.4.3. Архітектура модуля Downloader

Polling / service interface – надає REST API для користувачів для старту процесу сканування, а також отримує посилання з черги та передає їх на рівень бізнес-логіки.

*Business logi*c – отримує посилання від рівня інтерфейсу, завантажує відповідні сторінки.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний на кожному рівні). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, котрі обробляються модулем логування.

4) Content parser *(парсер контенту)*

Парсер контенту розбирає вміст веб-сторінки, виділяючи структуру html-тегів та інформацію, яка знаходиться всередині тегів. Результатом роботи парсеру є об'єкт Document, що містить дерево всіх html-тегів на сторінці.

Основними етапами парсингу контенту є:

- препроцесінг вхідного потоку – декодування вхідного потоку байтів;
- токенізація – генерування маркерів початкового тегу, кінцевого тегу;
- побудова дерева – генерація об’єктної моделі документа (DOM) на основі маркерів.

Архітектуру парсера контенту зображено на рис. 2.4.4.

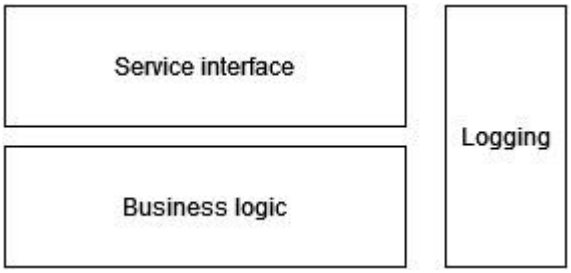


Рис. 2.4.4. Архітектура парсера

Service interface – надає API для завантажувача сторінок, яке дозволяє ініціалізувати процес парсингу, отримує контент сторінки у вигляді потоку байтів та перенаправляє його на рівень бізнес-логіки.

Business logic – здійснює препроцесінг вхідного потоку, токенізацію та побудову об’єктної моделі документа.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування.

Link extractor *(видобувач посилань)*

Видобувач посилань знаходить та видобуває з об’єктної моделі документа посилання. Link extractor повинен забезпечити коректну роботу з зовнішніми, внутрішніми посиланнями, також посиланнями, що згенеровані за допомогою Javascript.

Архітектуру видобувача посилань зображено на рис. 2.4.5.

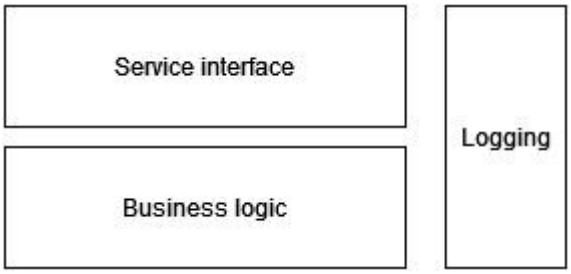


Рис. 2.4.5. Архітектура видобувача посилань

Service interface – надає API для парсера контенту, яке дозволяє ініціалізувати процес видобування посилань, отримує об’єктну модель документа та перенаправляє її на рівень бізнес-логіки.

Business logic – здійснює видобування посилань з об’єктної моделі документа.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування.

5) Url resolver *(оброблювач посилань)*

Оброблювач посилань перетворює отримані посилання відповідно до правил нормалізації посилань. Нормалізація посилань — це процес, за допомогою якого посилання модифікуються та стандартизуються послідовним чином. Метою процесу нормалізації є перетворення посилання на нормалізоване посилання з метою визначення, чи можуть два синтаксично різні посилання бути еквівалентними. Це дозволяє уникнути повторних завантажень однієї і тієї ж самої веб-сторінки.

Оброблювач посилань повинен підтримувати RFC 3986 стандарт.

Архітектуру оброблювача посилань зображено на рис. 2.4.6.

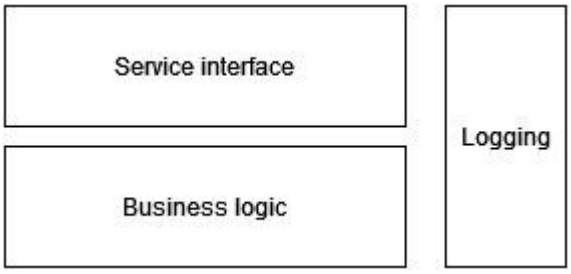


Рис. 2.4.6. Архітектура оброблювача посилань

Service interface – надає API для видобувача посилань, яке дозволяє ініціалізувати процес нормалізації посилань, отримує посилання та перенаправляє його на рівень бізнес-логіки.

Business logic – здійснює нормалізацію посилань відповідно до RFC 3986 стандарту.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування.

Data access module *(модуль доступу до даних)*

Модуль доступу до даних забезпечує обробку та збереження відповідної структури посилань у базі даних. Кожне посилання репрезентує веб-сторінку і повинне бути збережене в базі даних у вигляді вузла з відповідними зв'язками.

Архітектуру модуля доступу до даних зображено на рис. 2.4.7.

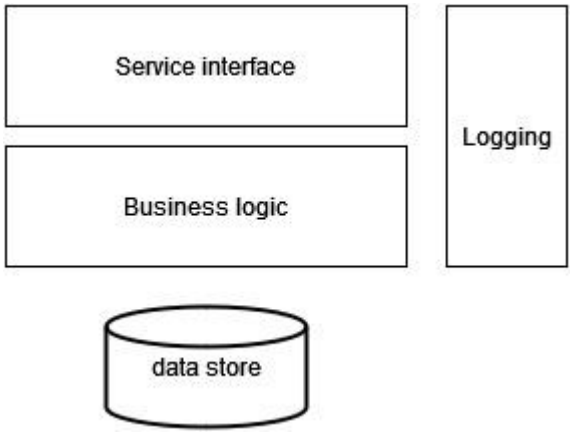


Рис. 2.4.7. Архітектура модуля доступу

Service interface – надає API для оброблювача посилань, яке дозволяє ініціалізувати процес збереження посилань, отримує посилання та перенаправляє його на рівень бізнес-логіки.

*Business logi*c – створює модель вузла для посилання, визначає чи існують у щойно створеного вузла зв'язки з іншими вузлами, зберігає вузол зі зв'язками в сховищі даних.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування. ***Url manager*** *(менеджер посилань)*

Менеджер посилань генерує список посилань для завантаження. Цей модуль визначає набір правил для кожного посилання стосовно регулювання обсягу трафіку, спрямованого на будь-який конкретний веб-сервер, повторних спроб завантаження контенту та ін.

Менеджер посилань здійснює планування порядку завантаження URL-адрес, відповідає за усунення дублікатів та визначає пріоритети при завантаженні сторінок.

Архітектуру менеджера посилань зображено на рис. 2.4.8.

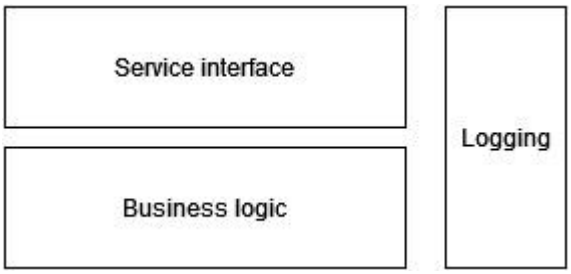


Рис. 2.4.8. Архітектура модуля доступу

Service interface – надає API для модуля доступу до даних, яке дозволяє ініціалізувати процес планування завантаження посилань, отримує посилання та перенаправляє його на рівень бізнес-логіки.

*Business logi*c – генерує конфігурацію для завантаження веб-сторінки за посиланням, визначає порядок та пріоритет завантаження.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування.

Data store *(сховище даних)*

Сховище даних використовується для збереження веб-графу. Головною вимогою до даного компоненту є можливість швидко додавати інформацію.

Стандартним методом зберігання ієрархічних даних є простий зв’язок «батько-нащадок». Кожен запис у базі даних містить батьківський ідентифікатор і за допомогою рекурсивного запиту можна отримати все

дерево. Для додавання нового запису до системи потрібен лише ідентифікатор батьківського запису без жодного іншого індексування.

Перевагами цього методу є простота і дешевизна введення нових записів. Якщо ми додаємо дочірній запис, нам просто потрібно знати батьківський ідентифікатор, і все інше буде коректно створено. Недоліком такого підходу є складність виконання рекурсивного запиту для отримання всього дерева. В нашому випадку перевага надається додаванню інформації, тому такий підхід є прийнятним.

В якості сховища даних було обрано найбільш популярну СКБД MySQL.

MySQL є рішенням для малих і середніх додатків. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або віддалені клієнти, проте є можливість будувати розподілені сховища і високонадійні конфігурації, що можуть забезпечити рівень доступності сервісу порядку 99.999% при виконанні вимог ACID до виконання транзакцій (атомарність, узгодженість, ізольованість, довговічність). Вихідні коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатопотоковості, що підвищує продуктивність системи в цілому [150-152].

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, які підтримують транзакції на рівні окремих записів. MySQL характеризується великою швидкістю, стійкістю і простотою використання [153].

Для некомерційного використання MySQL є безкоштовною. Можливості сервера MySQL [151-153]:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- кількість рядків у таблицях може досягати 50 млн;

– висока швидкість виконання команд;
– наявність простої та ефективної системи безпеки.

6) ***Analytical module (аналітичний модуль)***

Аналітичний модуль надає інтерфейс для проведення статистичного та кластерного аналізу веб-графу і може бути частиною інтелектуальної інформаційної системи [103]. А саме для:

– побудови розподілу ймовірності вузлів за ступенями по вхідних зв’язках (in degree) та розподілу ймовірності вузлів за ступенями по вихідних зв’язках (out degree);
– обчислення коефіцієнтів кластерності веб-графу;
– визначення середніх значень ступеня вузла для неорієнтованих графів;
– визначення оптимальної кількості кластерів методом k-core decomposition;
– знаходження центрів кластерів;
– розбиття досліджуваних мереж на кластери за допомогою алгоритму PIC (Power iteration clustering).

Архітектуру аналітичного модуля зображено на рис. 2.4.9.

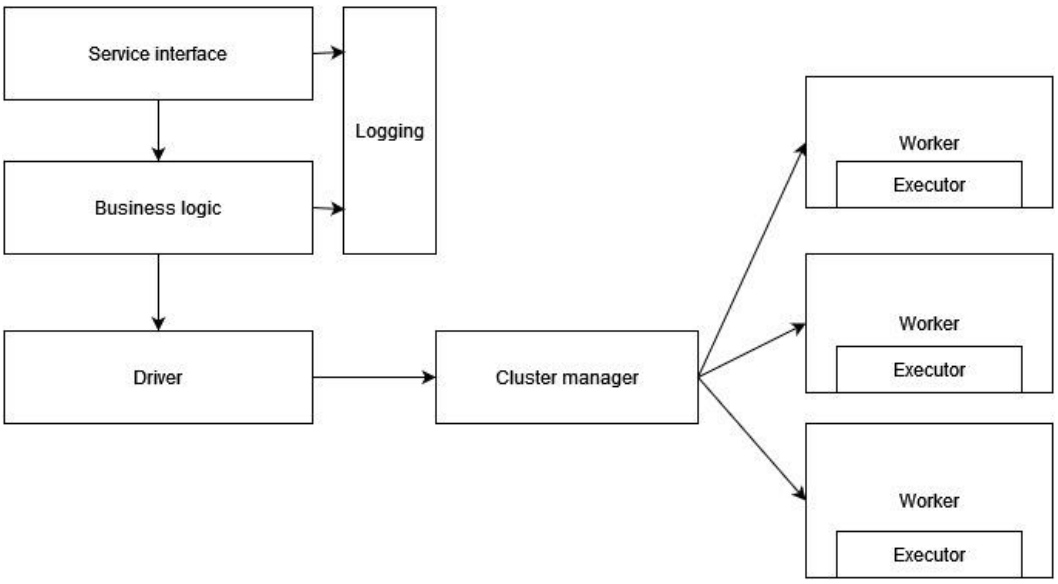


Рис. 2.4.9. Архітектура аналітичного модуля

Service interface – надає API для користувача, яке дозволяє ініціалізувати процес проведення статистичного та кластерного аналізу, отримує вхідні параметри та перенаправляє їх на рівень бізнес-логіки.

*Business logi*c – відповідно до вхідних параметрів витягує необхідні дані зі сховища даних, здійснює їх перетворення та передає на вхід драйверу для здійснення обчислень.

Driver – координує виконавців і контролює виконання завдань. Він складається з коду для виконання і відповідного контексту. Контекст отримує загальну задачу та ділить її на невеликі завдання, які обробляються виконавцями.

Workers – отримують від драйвера завдання та виконують їх. Виконавці утворюють кластер.

Cluster manager – відповідає за взаємодію з драйвером та виконавцями, здійснюючи такі завдання:

- керування виділенням ресурсів;
- керування поділом програми;
- керування виконанням програми.

Logging – це не фактичний рівень, а наскрізний компонент (тобто – він доступний для всіх рівнів). Цей компонент містить бібліотеку логування, яка використовується компонентом, і генерує події, які обробляються модулем логування.

Для проведення статистичного та кластерного аналізу веб-графу модуль надає REST API, що дозволяє користувачу вказати початкові URL-адреси.

Детальний опис REST API аналітичного модуля наведений в таблиці 2.4.2.

Таблиця 2.4.2.

Опис REST API аналітичного модуля

Ресурс	Шлях	Статус код
Розрахунок статистики	POST /rest/subnetwork/{subnetwork} /statistics	200 OK
Результат розрахунку статистики	GET /rest/subnetwork/{subnetwork} /statistics	200 OK
Побудова моделі алгоритму	POST /rest/subnetwork/{subnetwork} /{algo_name}	200 OK
Дані з моделі алгоритму	GET /rest/subnetwork/{subnetwork} /{algo_name}	200 OK

2.5. Засоби розробки програмного продукту

Для створення веб-кроулера, який би працював під операційною системою Windows та операційними системами сімейства Unix було обрано мову програмування Java, яка є потужним інструментом для розробки розподілених та багатопотокових систем, що завдяки своїм унікальним можливостям став незамінним для сучасної індустрії програмування. Ось деякі ключові аспекти, які роблять Java таким популярним вибором [154]:

– *Незалежність від платформи*: Java використовує віртуальну машину Java (JVM), що дозволяє виконувати програми на будь-якій платформі, яка підтримує JVM. Це робить Java ідеальним вибором для розробки розподілених систем, які можуть бути запускатись на різних операційних системах.

– *Механізми комунікації через мережу*: Java надає багато різних засобів для реалізації взаємодії між різними компонентами розподіленої системи. Наприклад, класи Socket та ServerSocket дозволяють реалізувати з'єднання між клієнтами та серверами за допомогою протоколів TCP/IP.

– *Розподілений об'єктний зв'язок*: Java RMI (Remote Method Invocation) дозволяє викликати методи об'єктів, які знаходяться на віддалених машинах. Цей механізм дозволяє створювати розподілені додатки зі зручним способом зв'язку між різними компонентами.

– *Підтримка веб-сервісів*: Java надає багато інструментів для реалізації веб-сервісів, таких як JAX-WS (Java API for XML Web Services) та JAX-RS (Java API for RESTful Web Services). Ці засоби дозволяють створювати розподілені системи, що взаємодіють за допомогою стандартних протоколів, таких як HTTP.

– *Безпека*: Java має вбудовану систему безпеки, яка дозволяє контролювати доступ до ресурсів та запобігати вразливостям. Це особливо важливо для розподілених систем, що зазвичай зазнають мережевих атак.

Для створення додатків на Java досить часто використовується Java Platform, Enterprise Edition (JEE). Перш за все, JEE забезпечує високий рівень портативності завдяки незалежності від платформи, дозволяючи розгортати додатки на різних сервлет-контейнерах без зміни коду [154].

Другою вагомою перевагою JEE є його висока здатність до розширення. Завдяки низці інтерфейсів та сервісів, розробники можуть додавати функціональність додатків, використовуючи різноманітні контейнери та бібліотеки. Це дозволяє створювати програми з великим спектром функціональних можливостей і динамічної поведінки.

Крім того, JEE пропонує механізм масштабування для розподілених систем. З використанням технологій віддаленого виклику методів, додатки можуть легко масштабуватися та забезпечувати високий рівень продуктивності, що робить JEE оптимальним вибором для великих корпорацій з великою кількістю користувачів.

Наступною значною перевагою JEE є його вбудована система безпеки. JEE забезпечує механізми автентифікації, авторизації та шифрування даних, що дозволяє підприємствам забезпечити надійний захист своїх додатків та конфіденційності користувачів [155].

Загалом, JEE є впливовою та сучасною платформою для розробки додатків корпоративного рівня, що надає розробникам різноманітні можливості для створення ефективних та масштабованих рішень у розподілених середовищах. Зокрема, JEE надає API, який дозволяє розробникам створювати робочі процеси та використовувати такі ресурси, як бази даних або веб-сервіси.

Додатки, створені за допомогою API JEE, запускаються на сервері додатків JEE, що дозволяє запускати одночасно кілька додатків на одному комп’ютері. Розробники можуть використовувати різні сервіси JEE серверу для запуску програм, створених за допомогою JEE API, а саме [155]:

- EJB-контейнер, який підтримує автоматичну синхронізацію Java об'єктів з базою даних (CMP – container managed persistence, BMP – bean managed persistence);
- JMS сервіс доставки повідомлень між компонентами і серверами;
- управління ресурсами (доступ до СУБД, файлової системи, поштового сервера і т. д.);
- безпека і захист даних;
- підтримка транзакцій (в тому числі і розподілених, двофазних);
- веб-сервер і сервлет-сервер;
- підтримка веб-сервісів;
- JavaServer Faces (JSF).

Додатки, створені за допомогою API JEE, як правило, підтримують багаторівневу розподілену архітектуру [155], що складається з клієнтського рівня, середнього рівня та внутрішнього рівня (рис. 2.5.1).

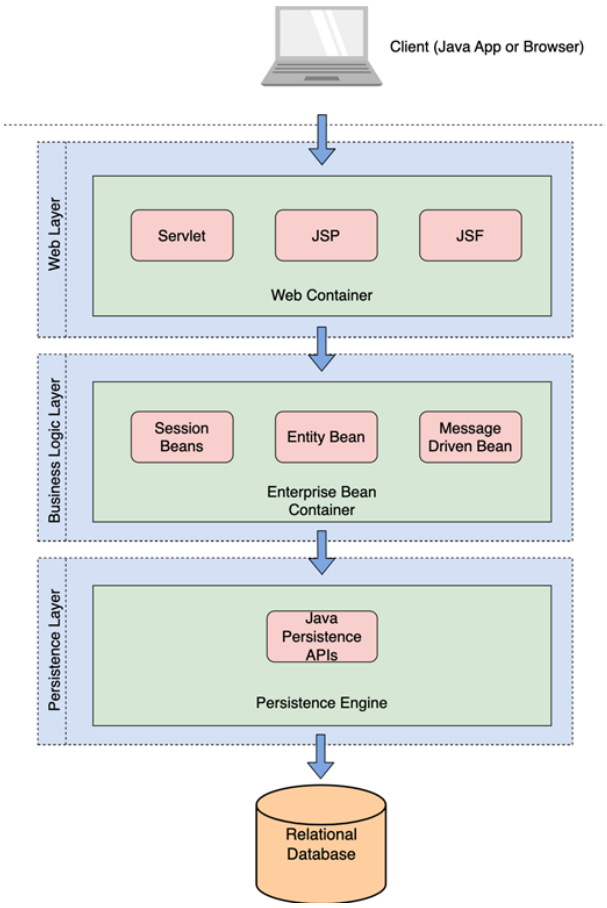


Рис. 2.5.1. Багаторівнева архітектура розподіленого додатку JEE

Компоненти клієнтського рівня працюють у клієнтських контейнерах. Клієнтський рівень можна реалізувати наступними способами [156]:

- Автономні програми Java-клієнти, які, зазвичай, мають графічний інтерфейс. Такі програми Java повинні бути встановлені на кожній клієнтській машині. Вони можуть отримати доступ до внутрішнього або середнього рівня через API, такі як JDBC.
- Статичні сторінки HTML надають обмежений графічний інтерфейс для програми.
- Динамічний HTML генерується сторінками JSP або сервлетами.

Середній рівень складається з веб-рівня та бізнес-рівня. Компоненти веб-рівня працюють на веб-сервері JEE, який надає веб-контейнер. Компоненти бізнес-рівня працюють на сервері прикладних програм JEE, що надає контейнер EJB.

Компоненти веб-рівня складаються з сервлетів та сторінок JSP, які керують взаємодією з клієнтським рівнем, ізолюючи клієнтів від бізнес-рівня та внутрішнього рівня. Клієнти роблять запити до веб-рівня, котрий обробляє запити та повертає результати клієнту. Запити клієнтів до компонентів на веб-рівні зазвичай призводять до запитів на нижчі рівні додатку, включаючи внутрішній рівень.

Компонентами бізнес-рівня є EJB. Вони містять бізнес-логіку програми та роблять запити до внутрішнього рівня відповідно до бізнес-логіки, як правило, у відповідь на запит від веб-рівня.

Рівень сховища даних представляє збережені дані програми, часто у формі RDBMS [156].

Таким чином, використання багаторівневої архітектури дозволяє забезпечувати незалежну розробку і розвиток окремих частин додатку, ще більше збалансувати навантаження на різні вузли і мережу. Додаток складається з логічних і функціональних компонент, які можна використовувати повторно. Такий підхід значно спрощує розробку та підтримку розподілених систем.

З іншого боку, подібна архітектура дозволяє ізолювати додаток, повністю інкапсулюючи середовище виконання, і запускати його на віртуальній машині замість фізичного сервера, що надзвичайно важливо в епоху хмарних обчислень [157].

З метою оптимізації використання ресурсів віртуального або фізичного сервера була запроваджена технологія контейнеризації.

Контейнери – це абстракція на рівні програми, яка пакує код і залежності разом. Кілька контейнерів можуть працювати на одній віртуальній машині та спільно використовувати ядро ОС з іншими контейнерами, кожен із яких працює як ізольований процес у просторі користувача. Контейнери займають менше місця, ніж віртуальні машини, можуть працювати з більшою кількістю програм і потребують менше віртуальних машин і операційних систем [158; 159].

Найпопулярнішим інструментом для контейнеризації є Docker, який є відкритою платформою для розробки та експлуатації додатків. Використовуючи контейнери Docker, можна розгортати, копіювати, переносити додатки швидше та легше, ніж за допомогою віртуальної машини. У принципі Docker привносить хмароподібну гнучкість у будь-яку інфраструктуру, що може працювати на контейнерах, включаюючи і JEE [158-161].

У цілому JEE є оптимальним вибором для створення веб-кроулера оскільки підтримує:

- багаторівневу архітектуру,
- незалежність від платформи,
- роботу в багатопотоковому режимі,
- масштабованість,
- гнучкість,
- високу здатність до розширення,
- контейнеризацію,
- швидке розгортання.

Було проведено тестування розробленого програмного забезпечення. Основною метою тестування програмного забезпечення є оцінка відповідності програмного забезпечення зазначеним потребам.

Тестування програмного забезпечення зазвичай здійснюється з метою виявлення помилок для гарантування якості програмного забезпечення та виконання остаточної перевірки відповідності специфікаціям, дизайну.

Загалом, тестування поділяється на різні рівні. Тому, для оцінки якості розробленого програмного забезпечення використовувалися конкретні тестові сценарії та метрики, а саме коректність обробки структури веб-сайту, швидкості сканування, масштабованості. Функціонал компонентів програмного продукту покривався Unit тестами та інтеграційними тестами. Окрема увага приділялася приймальному тестуванню, яке засвідчило, що

розроблене програмне забезпечення повністю відповідає зазначеним вище (функціональним та нефункціональним) вимогам [103].

2.6. Результати роботи розробленого програмного забезпечення

Розроблений кроулер є ключовою компонентою системи для аналізу веб-даних. Він застосовувався для обробки та збору значних обсягів даних з різноманітних веб-сторінок. Для проведення подальших досліджень за допомогою нього була зібрана інформація з деяких сегментів WWW-простору: українського (edu.ua), ізраїльського (ac.il) та польського (edu.pl). Програма налаштовувалась перед початком роботи, тобто задавались: глибина індексації (2-5 рівнів), перелік точок входу – сайтів, які досліджуватимуться. Для кожного сегменту мережі вибиралася певна кількість точок входу (таблиця 2.6.1), яка використовувалися кроулером для зондування.

Таблиця 2.6.1.

Кількість точок входу для досліджуваних сегментів мережі

Назва зони	edu.ua	ac.il	edu.pl
Кількість точок входу	100	27	32

Програма, рухаючись по заданих веб-сторінках, збирала структуру посилань на інші сторінки та зберігала дані в базу даних. Для кожної зони було в середньому просканово більше 400 тисяч веб-сторінок. Загальна кількість таких сторінок для всіх зон, що досліджувались, налічує приблизно 2 000 000 веб-сторінок.

У нашому випадку, окрім збору інформації, реалізовані компоненти для проведення статистичного та кластерного аналізу різних зон веб-простору, тобто розроблено аналітичний модуль, який відповідає за:

- визначення ступеня кожного вузла;
- побудову розподілу ймовірності вузлів по вхідних та вихідних зв’язках;
- обчислення коефіцієнтів кластерності підмереж;

– визначення середнього значення ступеня вузла для неорієнтованих графів;
– визначення оптимальної кількості кластерів для кожного сегменту веб-простору.

Розроблений аналітичний модуль може бути використаний як частина відповідної інтелектуальної ситеми.

Аналіз, зібраної кроулером інформації, наведено у розділі 3.

ВИСНОВКИ ДО РОЗДІЛУ 2

У даному розділі подано аналітичний огляд існуючих програмних засобів для проведення збирання інформації у веб-просторі, наведено їх короткий опис, визначено їхні переваги та недоліки.

У результаті проведених досліджень:

- розроблено програмне забезпечення – кроулер для збору гіперпосилань з веб-сторінок;
- визначено та проаналізовано основні функціональні та нефункціональні вимоги до розробленого програмного забезпечення;
- обрано та обґрунтовано вибір засобів розробки програмного забезпечення;
- описано архітектуру та структуру розробленого кроулера;
- розроблено головний аналітичний модуль для проведення статистичного та кластерного аналізу отриманого веб-графу.

Розроблений програмний продукт (кроулер), на відміну від вже існуючих, володіє наступним функціоналом:

- сканування веб-простору та завантаження гіперпосилання з веб-сторінок у декілька потоків і використання черги для оптимальної навігації по вебу (уникнення повторного завантаження одного і того ж самого контенту);
- зберігання знайдених гіперпосилань у базі даних;
- повне керування процесом пошуку. Передбачено можливість роботи з множиною точок входу та задання глибини індексації;
- проведення розрахунків основних статистичних параметрів обраного сегменту веб-простору;
- здійснення кластерного аналізу зібраних даних;

– робота з системою як в режимі самостійного застосування, так і під керуванням JEE Application Server. Розробка проводилася з використанням JEE технологій.

Розроблений кроулер має модульну структуру та багаторівневу архітектуру, підтримує контейнеризацію і роботу в багатопотоковому режимі, легко масштабується, характеризується гнучкістю, високою здатністю до розширення та адаптивною спроможністю (можливість використовувати його для дослідження аналогічних графів іншого походження).

За допомогою кроулера була зібрана інформація з сегментів мережі WWW: українського (edu.ua), ізраїльського (ac.il) та польського (edu.pl). Проведено сканування і зібрано дані приблизно з 2 000 000 веб-сторінок.

Основні результати другого розділу опубліковані в роботах [103; 104].

РОЗДІЛ 3

СТАТИСТИЧНИЙ ТА КЛАСТЕРНИЙ АНАЛІЗ ВЕБ-ПРОСТОРУ МЕТОДАМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Сьогодні статистичному та кластерному аналізу великих об'ємів даних присвячено багато наукових публікацій. У значній кількості цих публікацій вивчають структуру та характеристики Web-простору [1; 7; 12; 32]. Найчастіше структуру таких даних подають за допомогою графа [8]. Такий підхід виявився дуже вдалим для дослідження характеристик складних мереж. Розглядаючи Web-простір як орієнтований граф, коли статичні сторінки приймали за вузли, а посилання між цими сторінками – за дуги, довели [8], що WWW-простір являє собою безмасштабну мережу. Аналогічні дослідження виконувалися у роботах [1; 7; 8; 18; 19; 192], які довели високий ступінь розвиненості всесвітньої мережі. Це підтвердили автори роботи [41], які вивчали статистичні характеристики національних веб-доменів Бразилії, Чилі, Греції, Кореї, Іспанії. Щодо українського сегменту веб-простору, то ці питання вивчені ще не досить повно [12; 46; 56; 162]. Тим не менше, за сукупністю результатів перелічених нами досліджень, вдалося зобразити структуру веб-простору [1; 41; 162-165].

Однак, зібрати та проаналізувати статистичні характеристики величезних наборів даних, якими й характеризується сучасний веб-простір, стає все складніше: непросто обробити мільйони вузлів з їхніми зв'язками та внутрішніми посиланнями, обчислити статистичні характеристики мережі. Найчастіше для обробки таких даних використовують процес кластеризації. Однією із задач кластеризації є зменшення розмірності даних за ознаками, за якими відбувається класифікація об'єктів. Ідея усіх методів кластеризації полягає у тому, щоб схожість об'єктів, які знаходяться в конкретному кластері, була максимальною за своєю семантикою. На основі процесу кластеризації побудовано велику кількість підходів до розбиття даних [166-168].

Виходячи з вищевикладеного, зрозуміло, що кластеризація великих об’ємів даних дозволяє сильно спростити їх аналіз для різних задач: виявлення структури множини об’єктів; спрощення подальшого аналізу масивів даних задля прийняття потрібних рішень; скорочення обсягу зберігання даних у разі надвеликої вибірки; виділення нетипових об’єктів, які за своїми ознаками не входять до жодного з кластерів тощо. Таким чином, поставлена задача може бути розглянута як задача кластеризації в Big Data, з використанням граничних теорем, що грунтуються на теорії випадкових матриць [169]. Такі уявлення також можуть бути використані для розбиття Smart Grid систем на частини.

В усіх таких випадках на основі вихідних даних будується матриця суміжності, яку потім досліджують за допомогою різних методів кластеризації. Наприклад, у роботах [88; 170-172] проводились такі дослідження із використанням методів спектральної кластеризації.

Однак, найбільш популярні методи кластеризації вимагають для своєї роботи уведення кількості кластерів, на яку треба розділити вихідний набір даних. У деяких задачах це не є проблемою, однак дуже часто зустрічаються випадки, коли така кількість заздалегідь невідома. Для визначення “оптимальної” кількості кластерів також існує низка методів, однак, постає питання порівняння цих методів за результативністю, перспективністю, зручністю у використанні та за іншими характеристиками.

У цьому розділі розглянуто деякі методи знаходження оптимальної кількості кластерів (методи “ліктя” та k-core decomposition) у наборі даних і проведено порівняння ефективності цих методів для визначення оптимальної кількості кластерів. Також запропоновано новий метод знаходження оптимальної кількості кластерів, який базується на спектральному розкладі матриці переходів для марковського процесу, що відповідає графу. Детальний опис цього методу буде наведено в розділі 4.

Результати проведених досліджень доповідались на конференціях та опубліковані в роботах [42-45; 47; 89; 96; 97; 103; 173-179].

3.1. Статистичні характеристики інформації у веб-просторі

Статистичні дослідження, які проводять науковці, в основному складаються з трьох етапів:

1. Підготовчого (розробка програми дослідження).

2. Основного (проведення емпіричного дослідження).

3. Завершального (обробка, аналіз даних, формування висновків і рекомендацій). Отже, для проведення статистичних досліджень потрібно перейти до етапу збору даних, які необхідні для обробки інформації та її подальшої інтерпретації.

Збір даних – це процес, за допомогою якого дослідники накопичують/агрегують необхідну їм інформацію, метою якого є проведення дослідження. Зібрані дані можуть бути як якісними так і кількісними. Найкращим методом збору кількісних даних є експериментальне дослідження. Такий тип досліджень може надати:

– достовірні дані про динаміку розвитку підмережі, що важливо для розуміння певних характеристик;

– аналіз зібраних даних, що допомагає отримати цінну інформацію, яку можна використати надалі для визначення типу мережі, розуміння її властивостей та подальшого її розвитку, зрозуміти еволюційні уявлення; отримати аналітичні висновки для досягнення конкретних результатів.

Активний розвиток такої області досліджень, як складні мережі, призвів до вивчення характеристик мережі, враховуючи не тільки її топологію, а й статистичні характеристики, які описують поведінку мережі при зміні структурних властивостей.

Для цього науковці вивчають та досліджують статистичні характеристики різноманітних мереж: енергетичних, транспортних, комп’ютерних, мереж авіаперевезень, мереж співавторства, соціальної мережі, всесвітньої мережі Інтернет та багато інших [1; 5; 8; 12; 15; 68].

Традиційно під мережею розуміють сукупність об'єктів, об'єднаних за допомогою зв'язків, причому кожен об'єкт повинен мати певні властивості та містити конкретну інформацію [1; 180]. Математичною моделлю мережі є граф. Він не містить ніякої інформації про елементи мережі, але передає структуру її зв'язків [1; 8]. Вершинами графа будуть вузли, а ребрами – зв'язки між вершинами, які можуть мати напрям, тоді утворюється орієнтований граф, і бути ненапрямленими, – в такому разі граф буде неорієнтованим. Тоді мережу можна описати за допомогою матриці суміжності A (adjacency matrix), де елементи матриці $a_{ij} = 1$, якщо існує ребро з i-ої в j-ту вершину, та $a_{ij} = 0$, якщо такого ребра немає. В залежності від того, є граф орієнтованим чи ні, матриця суміжності буде відповідно несиметричною або симетричною. Для неорієнтованих графів вважають $a_{ij} = a_{ji}$ та $a_{ii} = 0$. Зазначимо, що наведені вище поняття розглядаються для так званих незважених графів (тобто таких, у яких всі ребра рівноправні).

Кожен вузол мережі характеризується деякою величиною k_i, яку будемо називати ступенем вузла (node degree) [5]. Ступінь вузла – це кількість зв'язків у даного вузла або, з математичної точки зору, сума елементів в i-ому рядку (стовпці) матриці суміжності A. Для орієнтованого графу вводять поняття вхідного (in-degree) та вихідного (out-degree) ступеня вузла.

До важливих характеристик вузлів мережі відносять [1; 3; 10; 12; 15; 23; 181; 182]:

– вхідний ступінь вузла (in-degree) – кількість ребер графа, які входять у вузол;

– вихідний ступінь вузла (out-degree) – кількість ребер графа, які виходять з вузла;

– середній ступінь вузла $< k >$;

– відстань від одного вузла до інших;

– середня відстань від даного вузла до інших вузлів;

– ексцентричність (eccentricity) – найбільша з мінімальних відстаней від даного вузла до інших;

– посередництво (betweenness), що показує, скільки найкоротших шляхів проходить через даний вузол;

– центральність – загальна кількість зв'язків даного вузла по відношенню до інших;

– коефіцієнт кластеризації C_i вузла.

До важливих параметрів мережі відносять [1; 3; 10; 12; 15; 23; 181-182]:

– загальна кількість вузлів N;

– загальна кількість ребер;

– геодезична відстань між вузлами;

– діаметр мережі – найбільша геодезична відстань мережі;

– щільність – відношення кількості ребер в мережі до максимально можливої кількості ребер при заданій кількості вузлів;

– коефіцієнт кластеризації C .

Відстань між вузлами визначається як кількість кроків, котрі необхідно зробити, щоби за існуючими ребрами дістатися від одного вузла до іншого. Природно, вузли можуть бути з'єднані прямо або опосередковано, через інші вузли.

Найкоротшим шляхом між вузлами називається мінімальна відстань між ними. Для всієї мережі можна ввести поняття середнього найкоротшого шляху, як середнє по всім парам вузлів мінімальної відстані між ними, яке обчислюється за формулою (3.1.1):

$$l = \frac{2}{N(N+1)} \sum_{i \geq j} d_{ij}, \qquad (3.1.1)$$

де n – кількість вузлів, d_{ij} – найкоротша відстань між вузлами i та j .

Іноді зустрічаються мережі, у яких відстань між вузлами є нескінченою, відповідно і середня відстань у цьому разі також дорівнює

нескінченості. У такому випадку розглядається середній інверсний шлях між вузлами, який обчислюється за формулою (3.1.2) [23], [16]:

$$il = \frac{2}{N(N+1)} \sum_{i \geq j} \frac{1}{d_{ij}}, \tag{3.1.2}$$

Ще один параметр мережі – це діаметр мережі (ексцентричність), який дорівнює максимальному значенню серед всіх d_{ij} [10; 183; 184].

Такий параметр мережі, як посередництво (betweenness) показує, скільки найкоротших шляхів проходить через вузол [3; 185]. Головну роль у встановленні зв'язків між іншими вузлами в мережі відіграють вузли з найбільшим посередництвом. Посередництво вузла m визначається за формулою (3.1.3):

$$\sigma_m = \sum_{i \neq j} \frac{B(i,m,j)}{B(i,j)} \ , \tag{3.1.3}$$

де $B(i,j)$ – загальна кількість найкоротших шляхів між вузлами i та j; $B(i,m,j)$ – кількість найкоротших шляхів між вузлами i та j, що проходять через вузол m.

Статистичні дослідження такої характеристики мережі, як центральність проводили автори у роботах [186-189]. Центральність, як міру стійкості мережі, що вказує на скільки геодезичні шляхи (найкоротші шляхи від однієї вершини до іншої) стануть довшими, якщо вершину буде видалено з мережі, розглядали науковці [32; 181; 190; 191]. Автори [188] запропонували алгоритм знаходження центральності, який на практиці досить складно реалізувати.

Альтернативний спосіб для вимірювання загального зв’язку вузла в мережі запропонували автори роботи [184], увівши до розгляду таку характеристику мережі як ексцентричність. Інший алгоритм обчислення цього параметру наведений у роботі [183].

Однією з найважливіших характеристик складних мереж є розподіл вузлів за ступенями (degree distribution), тобто за кількістю зв’язків $P(k)$. Це

ймовірність того, що i-ий вузол мережі має ступінь $k_i = k$. Цей параметр мережі – розподіл вузлів за ступенями, демонструє поведінку мережі та в багатьох випадках знання цієї характеристики досить для розуміння властивостей цієї мережі та процесів, які в ній відбуваються [1; 10; 12; 15]. Найчастіше зустрічаються такі розподіли ступенів вузлів:

– розподіл Пуассона: $P(k) \approx e^{-<k>} \frac{<k>^k}{k!}$; (3.1.4)

– експоненційний розподіл: $P(k) \approx e^{\frac{-k}{<k>}}$; (3.1.5)

– степеневий розподіл: $P(k) \approx \frac{1}{k^\gamma}$, де $k \neq 0,\ \gamma > 0$. (3.1.6)

У формулах (3.1.4), (3.1.5), (3.1.6) використовується така характеристика мереж, як середнє число зв’язків на один вузол (3.1.7), тобто відношення всіх зв’язків у мережі до кількості вузлів:

$$< k >= \frac{1}{N} \sum_{i=1}^{N} k_i = \sum_{i=1}^{N} k_i \, P(k_i). \quad (3.1.7)$$

Особливості розподілу вузлів за ступенями наведено на рис. 3.1.1.

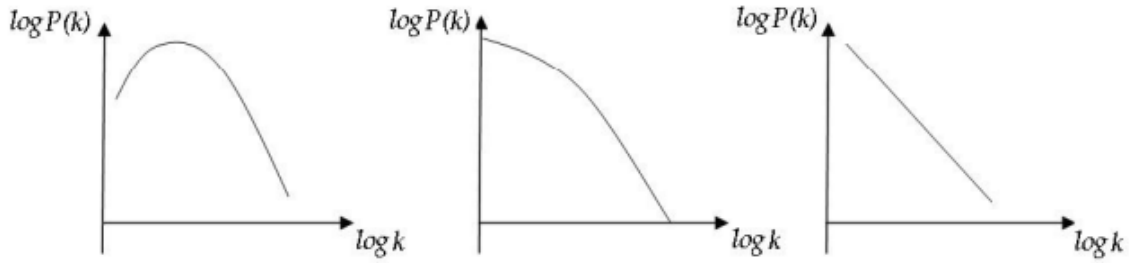


Рис. 3.1.1. Розподіл ступенів вузлів $P(k)$ в логорифмічному масштабі: а – розподіл Пуассона, б – експоненційний розподіл, в – степеневий розподіл

Якщо досліджуваний граф є випадковим, то розподіл ступенів вузлів буде описуватися розподілом Пуассона (3.1.4), і розподіл вузлів по числу зв'язків спадатиме досить швидко. Існують також мережі, розподіл вузлів по степенях яких підкоряється експоненційному закону (3.1.5) [2; 5]. Для них можна увести величину $< k >$, – середній степінь вузла, який певною мірою характеризує масштаб мережі. Для таких мереж існує лише формальне

поняття масштабу. Однак, за результатами багатьох досліджень встановлено, що більшість мереж реального світу (соціальних, технічних, біологічних, мережа Інтернет, WWW тощо) є безмасштабними мережами (scale-free) і описуються степеневим розподілом (3.1.6) і розподіл вузлів по числу зв'язків спадає досить швидко. Для безмасштабних мереж параметр γ є індивідуальною характеристикою мережі. Його значення, як правило, лежать в інтервалі $2 < \gamma \leq 3$ (але в рідкісних випадках може виходити за його межі). При такому розподілі в мережі можуть існувати вузли з дуже високим ступенем (величезною кількістю зв'язків), в той час як середня кількість зв'язків є відносно невеликою. Така ситуація практично не спостерігається в мережах, підпорядкованих розподілу Пуассона чи експоненційному розподілу.

Ще однією важливою характеристикою мережі є коефіцієнт кластеризації (D. Watts і S. Strogatz у 1998 році ввели у розгляд такий параметр мереж), який відповідає рівню зв'язності вузлів у мережі. Він характеризує тенденцію до утворення груп взаємопов'язаних вузлів, які називають кліками (clique) [1; 10; 16; 18]. Коефіцієнт кластеризації показує, скільки найближчих сусідів заданого вузла є також найближчими сусідами один для одного. Відношення реальної кількості зв'язків E_i, які з'єднують найближчих сусідів даного вузла i, до максимально можливої кількості (всі найближчі сусіди даного вузла були б з'єднані безпосередньо один з одним) (3.1.8):

$$C_i = \frac{2E_i}{k_i(k_i - 1)}. \qquad (3.1.8)$$

Альтернативне визначення коефіцієнта кластеризації, яке базується на трійках вузлів описані в роботах [192-194; 164], де його пропонується обчисляти за формулою (3.1.9):

$$C_i = \frac{\text{число трикутників з вершиною } i}{\text{число трійок із центром у вершині } i}. \qquad (3.1.9)$$

Кофіцієнт кластеризації визначають як для кожного вузла, так і для всієї мережі (3.1.10):

$$C = \frac{1}{N}\sum_{i=0}^{N} C_i \ . \qquad (3.1.10)$$

Для реально існуючих мереж типовими є висока скорельованість та відносно велике значення коефіцієнта кластеризації [42; 44; 45; 175].

3.2. Особливості застосування методів кластерного аналізу до сегментів веб-простору

На сьогоднішній день існує потреба аналізувати та отримувати інформацію з даних. Кластеризація є одним із таких аналітичних методів, що передбачає розподіл даних на групи ідентичних об'єктів. Кожна група відома як кластер, який складається з об'єктів, що мають спорідненість у межах кластера та відмінність від об'єктів в інших групах. Автори роботи [195] зазначають, що алгоритми кластеризації можна розділити на сім груп, а саме: ієрархічний алгоритм, алгоритм на основі щільності, алгоритм з розділенням, алгоритм на основі графів, алгоритм на основі сітки, алгоритм на основі моделі та алгоритм комбінованої кластеризації. Ці алгоритми дають різні результати відповідно до умов. Деякі методи кластеризації є кращими для великого набору даних, а деякі дають хороший результат для пошуку кластерів довільної форми.

Проведені дослідження та порівняння наступних алгоритмів: алгоритм k-means, k-Medoids, ієрархічний алгоритм кластеризації, алгоритм на основі сітки та алгоритм на основі щільності. Подібні дослідження проводились і у роботі [196]**,** де розглядаються декілька алгоритмів кластеризації: алгоритм k-means, алгоритм ієрархічної кластеризації, алгоритм на основі щільності, алгоритм самоорганізованої карти та алгоритм максимізації очікувань. Усі перелічені алгоритми пояснюються та аналізуються на основі таких факторів, як розмір набору даних, тип набору даних, кількість створених кластерів, якість, точність та продуктивність.

Великий клас алгоритмів кластеризації заснований на поданні вибірки у вигляді графа. Вершинам графа відповідають об'єкти вибірки, а ребрам – попарні відстані між об'єктами. Будується матриця суміжності, яку потім досліджують. Автори роботи [197] проводили дослідження над зв’язним неорієнтованим графом, застосовуючи декілька алгоритмів кластеризації (Марковської кластеризації (MCL), Iterative Conductance Cutting (ICC), геометрична MST кластеризація (GMC)), розглянули їх переваги та недоліки при проведенні кластеризації. Подібні дослідження проводились і в роботі [198], де автори розглянули п’ять алгоритмів кластеризації на основі графів, приділили увагу питанню вибору алгоритму кластеризації з точки зору ефективності, зробили порівняння алгоритмів за допомогою набору з шести індексів валідності [82], які зазвичай використовуються для оцінки якості алгоритму кластеризації.

У роботі [199] обговорюється питання якості кластеризації даних на графах та представлено новий спосіб оцінки якості кластеризації за двома параметрами, застосувавши його до алгоритмів спектральної кластеризації. Алгоритм кластеризації на графах, що включає ідею виконання випадкового блукання по графу для ідентифікації більш щільно зв'язаних підграфів, представлений у роботі [200], де індекс продуктивності вважають вимірюванням якості кластеризації графа. У роботі [201], також використовується ідея випадкових блукань для проведення кластеризації але для геометричних даних.

Для розбиття множини великої розмірності на кластери часто використовують спектральну кластеризацію. Цей метод вважають одним із прогресивних і популярних методів кластеризації завдяки універсальності (здатності працювати з багатовимірними даними й обробляти категоріальні ознаки) та можливості знаходити кластери довільної форми. За допомогою алгоритмів спектральної кластеризації зменшують розмірність даних. Наступним кроком є застосування деякого методу кластеризації (наприклад k-means) [44; 89; 202]. Слід зазначити, що інформація про кількість кластерів

та початкові центроїди на початковому етапі дослідження інформаційного простору, як правило, відсутня.

До спектральної кластеризації відносять всі методи, які розбивають множину на кластери за допомогою власних векторів матриці або інших матриць, отриманих з неї. Будь-який алгоритм спектральної кластеризації складається з трьох основних етапів [170]:

1. Попередня обробка.

Процес полягає в побудові та обробці матриці суміжності.

2. Спектральне відображення.

На цьому етапі знаходяться власні вектори для матриці суміжності. Для цього будується матриця Лапласа (різниця між діагональною матрицею, що містить ступінь кожного вузла і матрицею суміжності), яка далі розкладається на власні значення та власні вектори. Власні вектори, що відповідають найменшим ненульовим власним значенням (часто їх називають векторами Фідлера), використовуються для відображення вихідних точок даних у просторі нижчої вимірності.

3. Завершальна обробка даних.

Тут відбувається формування кластерів, тобто кластеризація в новому просторі меншої розмірності: перетворені точки даних (утворені вибраними власними векторами) потім кластеризуються за допомогою стандартного алгоритму кластеризації, наприклад k-means.

У роботах [98; 171; 172] досліджуються різноманітні алгоритми спектральної кластеризації, які групують дані за допомогою власних векторів матриці суміжності, отриманої з набору даних. Зокрема, автори прагнуть вирішити дві важливі проблеми: як автоматично визначати кількість кластерів і як виконати ефективну кластеризацію, враховуючи шуми та розрідженість даних. Проводиться аналіз характеристик простору, який показує, що не кожен власний вектор матриці суміжності даних є інформативним і релевантним для кластеризації; вибір власних векторів є критичним.

До алгоритмів спектральної кластеризації відноситься алгоритм PIC (Power Iteraition Clustering) [44; 89; 88]. Він являє собою простий, швидкий і масштабований метод кластеризації графів. На відміну від інших спектральних алгоритмів, обчислюється тільки один власний вектор (який насправді є лінійною комбінацією декількох власних векторів). В алгоритмі PIC пошук центроїдів кластерів здійснюється на основі власного вектора нормалізованої матриці W. Причому пошук даного власного вектора здійснюється на основі ітераційних методів. Псевдокод алгоритму PIC наведений нижче [88]:

Input: Normalized similarity matrix W, number of clusters k
Output: Clusters $C_1, C_2, ..., C_k$

1. Pick an initial vector $\mathbf{v}^0$.
2. $\mathbf{v}^{t+1} \leftarrow \frac{W\mathbf{v}^t}{\|W\mathbf{v}^t\|_1}$ and $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}^{t+1} - \mathbf{v}^t|$.
3. Increment t and repeat above step until $|\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}| \simeq \mathbf{0}$.
4. Use k-means on $\mathbf{v}^t$ and return clusters $C_1, C_2, ..., C_k$.

Рис. 3.2.1. Псевдокод алгоритму PIC

У результаті роботи алгоритму PIC отримали розклад зібраного набору даних на кластери, які формувались за алгоритмом k-means.

Завдання кластеризації полягає в побудові оптимального розбиття об'єктів на групи схожих об’єктів та відрізнялися від точок даних в інших групах. При цьому оптимальність може бути визначена як вимога мінімізації середньоквадратичної помилки розбиття (WSSE (Within set sum of squared errors)) – сумарне квадратичне відхилення точок кластерів від центрів цих кластерів) [84]:

$$e^2(X, L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2,$$

де c_j – центроїд (центр мас) кластера j, тобто точка із середніми значеннями характеристик для даного кластера.

Найпоширенішим алгоритмом цієї категорії є метод k-means. Висока схожість кластерів допомагає зменшити відстань між вузлами в межах

кластера, а також хороше розділення є більш важливим, щоб уникнути накладання кластерів. Вузол належить лише одному кластеру.

Алгоритм k-means широко використовується в кластерному аналізі, оскільки цей алгоритм має високу ефективність та масштабованість і швидко збігається при роботі з великими наборами даних [76; 203; 204; 219]. Його можна використовувати в різних галузях, таких як системи керування базами даних, спеціальні мережі, обробка зображень, бездротові сенсорні мережі та ін. Головною метою k-means є створення кластерів з мінімальною відстанню між усіма вузлами всередині кластерів.

Роботу алгоритму k-means можна описати наступними кроками:

Крок 1. Ввести число кластерів k**.**

Крок 2. Обрати початкові центроїди кластерів.

Крок 3. Віднести кожен об'єкт до кластера з найближчим центроїдом.

Крок 4. Зробити перерахунок центрів кластерів (центроїдів) згідно з їх поточним складом.

Крок 5. Повернутись до пунктів 3 та 4 за умови, що критерій зупинки алгоритму не задовольняється. Очевидно, що на кожній такій ітерації відбувається зміна меж кластерів та зміна їх центрів. У результаті мінімізується відстань між елементами всередині кластерів та збільшуються міжкластерні відстані.

Як критерій зупинки роботи алгоритму зазвичай вибирають мінімальну зміну середньоквадратичної помилки. Так само можна зупиняти роботу алгоритму, якщо на кроці 3 не було об'єктів, які перемістилися з кластера до кластера.

Алгоритм роботи k-means наведений на рис. 3.2.1. [203].

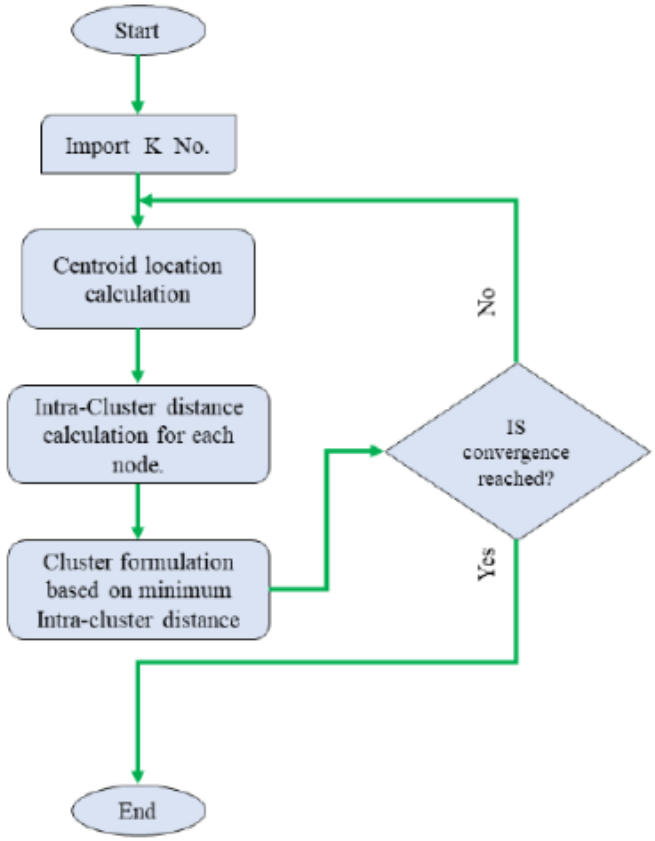


Рис. 3.2.1. Кроки алгоритму k-means

До недоліків алгоритму k-means відносять [76; 204; 219]:

1. Попередньо потрібно вказати кількість кластерів k.
2. Початкові центроїди вибираються довільним чином. Алгоритм k-means є чутливим до вибору початкових наближень центрів.
3. На алгоритм впливають точки шуму (викидів).

Отже, для ефективної роботи цього алгоритму та й інших алгоритмів кластеризації важливо знайти оптимальне значення кластерів k_{opt} та їх центроїди. Детальний опис найпоширеніших методів знаходження оптимальної кількості кластерів розглянуті в роботах [94-96; 205-209]. Наведемо опис декількох методів знаходження оптимальної кількості кластерів: метод «ліктя» та метод k-core decomposition.

Метод «ліктя» передбачає виконання наступних кроків [89]:

– Виконується кластеризація методом k-means, при цьому розраховується і записується значення функції штрафу (WSSE).

– Будується графік залежності функції штрафу (WSSE) від заданого числа кластерів.

– Вибираємо оптимальне число кластерів – це буде те число, де змінюється характер поведінки функції однорідності, тобто значення k,при якому відбувається найбільший перегин графіка.

Ще один спосіб визначення оптимальної кількості кластерів – це метод k-core decomposition [95-97]. Такий розклад дає змогу швидко знайти ядро графа, тобто максимально зв'язний підграф, в якому кожна вершина пов'язана принаймні з k вершинами в підграфі. Розклад k-core часто використовується у великомасштабному мережевому аналізі. Це алгоритм $O(m)$, де m – кількість потоків у непаралельних обчисленнях [95], [207]. Його головна мета – знайти сильну підгрупу, члени якої відіграють роль комунікаторів на графі. Кожен вузол у підграфі повинен мати принаймні ступінь k.

*k-c*ore decomposition має наступні властивості:

$$\forall u \in V\colon k\text{-}core\ (u) = k \leftrightarrow$$

$$\begin{cases} \text{Існує максимальний підграф } V_k \text{ такий, що } \forall v \in V_k\colon deg(v) \geq k, \\ \text{та} \\ \text{Не існує підграф } V_{k+1} \text{ такий, що } \forall v \in V_{k+1}\colon deg(v) \geq k+1. \end{cases}$$

Цей алгоритм призначений переважно для пошуку підграфа з найсильнішим зв'язком k. Це означає, що кожен член цього підграфа має принаймні k сусідів. Крім того, немає більшого підграфа, де кожен член має більше ніж k сусідів. Отже, якщо ми знайдемо вершину, яка має найвищий ступінь у цьому підграфі, то вона буде хорошим кандидатом для вибору її центром кластера.

Псевдокод методу k-core decomposition виглядає наступним чином:

Procedure k-core decomposition
1: **Input**
2: Graph: data in vertex is (degree, bool)
3: **Output**
4: Graph: data in vertex is K
5: **Pseudo Code**
6: **While**
7: **Initial the Pregel** send initial MSGs to all node
8: Graph.Vertex update (intitial MSGs)

```
9:              MSGs = message merge (all message sent)
10:             While messages.count > 0
11:                 Graph.Vertex update (messages)
12:                 MSGs = message merge (all message sent)
13:             End while
14:             K += 1
15:     End while
16:     Vertex update stage(messages)
17:              If (message.bool ) messages.degree =
max(origin, new degree)
18:             Message.bool = origin && new bool
19:     Message send stage
20:     If ( ! v1.bool || ! v2.bool)
21:             empty
22:     Else if(v1.degree == k && v2.degree > k)
23:             Send to v2 (1,true)
24:             Send to v1 (0,false)
25:     Else
26:             Iterator.empty
27:     Message merge stage
28:     (Sum, a.bool && b.bool)
```

Отже, проведення процесу кластеризації є нетривіальним. Процес кластеризації дозволяє розглядати великі набори даних та різко скорочувати їх, робити їх компактними та наочними. Результатом процесу кластеризації є набір кластерів, які містять в собі схожі елементи вхідної множини елементів.

Зауважимо, що:

– Не існує однозначно найкращого критерію якості кластеризації. Відомо багато алгоритмів, які не мають чітко вираженого критерію, але здійснюють досить розумну кластеризацію «з побудови». Всі вони можуть давати різні результати.

– Число кластерів, як правило, заздалегідь невідоме.

– Результат кластеризації істотно залежить від метрики ρ, вибір якої зазвичай також суб'єктивний і визначається дослідником.

3.3. Розробка інформаційної технології для збирання статистичної інформації та проведення кластерного аналізу у веб-просторі

Для проведення досліджень статистичних особливостей та кластерної структури складних мереж, а також доменів і зон веб-простору розроблена інформаційна технологія, основним функціональним призначенням якої є збирання, опрацювання, збереження та проведення статистичного та кластерного аналізу інформації у складних мережах [173; 178].

Роботу інформаційної системи можна розбити на наступні етапи:

1. Збирання та узагальнення інформації з веб-сторінок.

Для виконання цієї задачі нами було розроблене спеціальне програмне забезпечення – кроулер, детальний опис якого проведений у розділі 2. Користувач задає перелік адрес веб-сторінок – точок входу, і якщо потрібно, глибину індексації тощо.

2. Подання зібраної кроулером інформації у вигляді графу.

У результаті отримаємо граф веб-сторінок, статистичні характеристики якого й буде вивчатись. Для досліджуваної підмережі встановлюється ступінь кожного вузла, визначається коефіцієнт кластерності, будуються розподіли ймовірностей вузлів за вхідними та вихідними зв’язками. Об’єднуючи розраховані залежності для вхідних та вихідних підмереж, можемо отримати статистичні характеристики неорієнтованих графів веб-сторінок досліджених зон веб-простору. Ця задача виконується за допомогою фреймворку GrafX на кластерній системі обчислень з відкритим кодом Spark [210]. Обробка даних у Spark є швидкою та легкою завдяки оптимізованому механізму паралельних обчислень та гнучкому і уніфікованому API. Основна абстракція в Spark базується на концепції Resilient Distributed Dataset (RDD). Розширюючи можливості MapReduce фреймворку, Spark Core API спрощує написання аналітичних завдань.

Крім Core API, Spark пропонує інтегрований набір бібліотек високого рівня, які можна використовувати для спеціальних завдань, таких як обробка графів або машинне навчання. Зокрема, GraphX – це бібліотека для

виконання графо-паралельної обробки в Spark. На високому рівні GraphX розширює Spark RDD, уводячи нову абстракцію Graph: спрямований мультиграф із властивостями, прикріпленими до кожної вершини та ребра. Для забезпечення обчислення графів, GraphX надає набір фундаментальних операторів (наприклад, subgraph, joinVertices і aggregateMessages), а також оптимізований варіант Pregel API.

Графи за своєю природою є рекурсивними структурами даних, оскільки властивості вершин залежать від властивостей їхніх сусідів, які, у свою чергу, залежать від властивостей їхніх сусідів. Як наслідок, багато важливих алгоритмів графів ітеративно переобчислюють властивості кожної вершини, доки не буде досягнуто умови фіксованої точки. Для вираження цих ітераційних алгоритмів було запропоновано ряд графо-паралельних абстракцій. GraphX представляє варіант Pregel API.

На високому рівні оператор Pregel у GraphX – це абстракція масово-синхронного паралельного обміну повідомленнями, обмежена топологією графа. Оператор Pregel виконується в серії суперкроків, у яких вершини отримують суму своїх вхідних повідомлень з попереднього суперкроку, обчислюють нове значення для властивості вершини, а потім надсилають повідомлення сусіднім вершинам на наступному суперкроці. Вершини, які не отримують повідомлення, пропускаються в межах суперкроку. Оператор Pregel завершує ітерацію та повертає остаточний граф, коли не залишилося повідомлень.

Крім того, GraphX містить постійно колекцію спеціальних алгоритмів і конструкторів для здійснення аналітики графів. Зокрема, алгоритм зв’язних компонентів, алгоритм підрахунку кількості трикутників та обчислення коефіцієнту кластеризації в графі, PageRank.

Алгоритм зв’язних компонентів позначає кожен зв’язний компонент графа ідентифікатором його вершини з найменшим номером. У мережі два вузли з’єднані, якщо між ними на графі є шлях. Мережа називається з’єднаною, якщо всі пари вузлів з’єднані. Знайти зв’язні компоненти графа

легко в GraphX за допомогою методу connectedComponents. Наприклад, у соціальній мережі зв'язані компоненти можуть наближено ідентифікувати кластери.

Вершина є частиною трикутника, якщо вона має дві суміжні вершини з ребром між ними. GraphX реалізує алгоритм підрахунку, що проходять через кожну вершину, забезпечуючи міру кластеризації. Зокрема, трикутник підрахунку необхідний для обчислення коефіцієнтів кластеризації, які вимірюють локальну щільність для кожного вузла в мережі. Насправді коефіцієнт кластеризації даного вузла є оцінкою ймовірності того, що кожна пара його сусідів з'єднана. Таким чином, коефіцієнт можна розрахувати як відношення між сумарними зв'язками між сусідами вузла, яке також дорівнює кількості трикутників, які проходять через вузол, і кількістю всіх можливих пар сусідніх вузлів [211; 217].

3. *Визначення оптимальної кількості кластерів та їхніх центрів обраним методом*.

Після процесу побудови графа, ми за допомогою обраних методів (k-core decomposition [95] або методу "ліктя" [94]) визначаємо оптимальну кількість кластерів та центри кластерів для досліджуваного сегменту веб-простору.

Для реалізації методу "ліктя" використовували MLib (Machine Learning Library) з фреймворку Spark.

MLlib – це бібліотека машинного навчання, яка постачається з Apache Spark. Однією з головних переваг Spark є можливість масового масштабування обчислень, а це саме те, що потрібно для алгоритмів машинного навчання. Однак основне застереження полягає в тому, що всі алгоритми машинного навчання не можуть бути ефективно розпаралелені. У кожного алгоритму є свої проблеми з розпаралелюванням, будь то розпаралелювання завдань або даних. Враховуючи це, Spark стає де-факто платформою для створення алгоритмів і програм машинного навчання. Розробники, що працюють над Spark MLlib, реалізують все більше і більше

машинних алгоритмів у масштабованій і стислій формі в рамках Spark. Так, Apache Spark містить набір алгоритмів для вирішення проблеми кластеризації. Кластеризація – це проблема навчання без вчителя, за допомогою якої ми прагнемо згрупувати підмножини сутностей одна з одною на основі певного поняття подібності [212; 218].

Пакет spark.mllib підтримує наступні моделі:

- k-means,
- Gaussian mixture,
- Power iteration clustering (PIC),
- Latent Dirichlet allocation (LDA),
- Bisecting k-means,
- Streaming k-means.

Інший спосіб визначення оптимальної кількості кластерів, який застосовується**,** – це метод k-core decomposition, для якого використано GraphX на Spark.

4. *Проведення кластеризації.*

Цей етап виконується із використанням методу PIC-clastering (Power Iteraition Clustering) [88]. В результаті отримаємо розклад зібраного набору даних на кластери. Для проведення кластеризації методом PIC-clastering використовується MLib (Machine Learning Library) з фреймворку Spark [212; 218].

Графічно послідовність виконання вказаних завдань із використанням обраних методів подано на рис. 3.3.1 [103; 173; 178; 179].

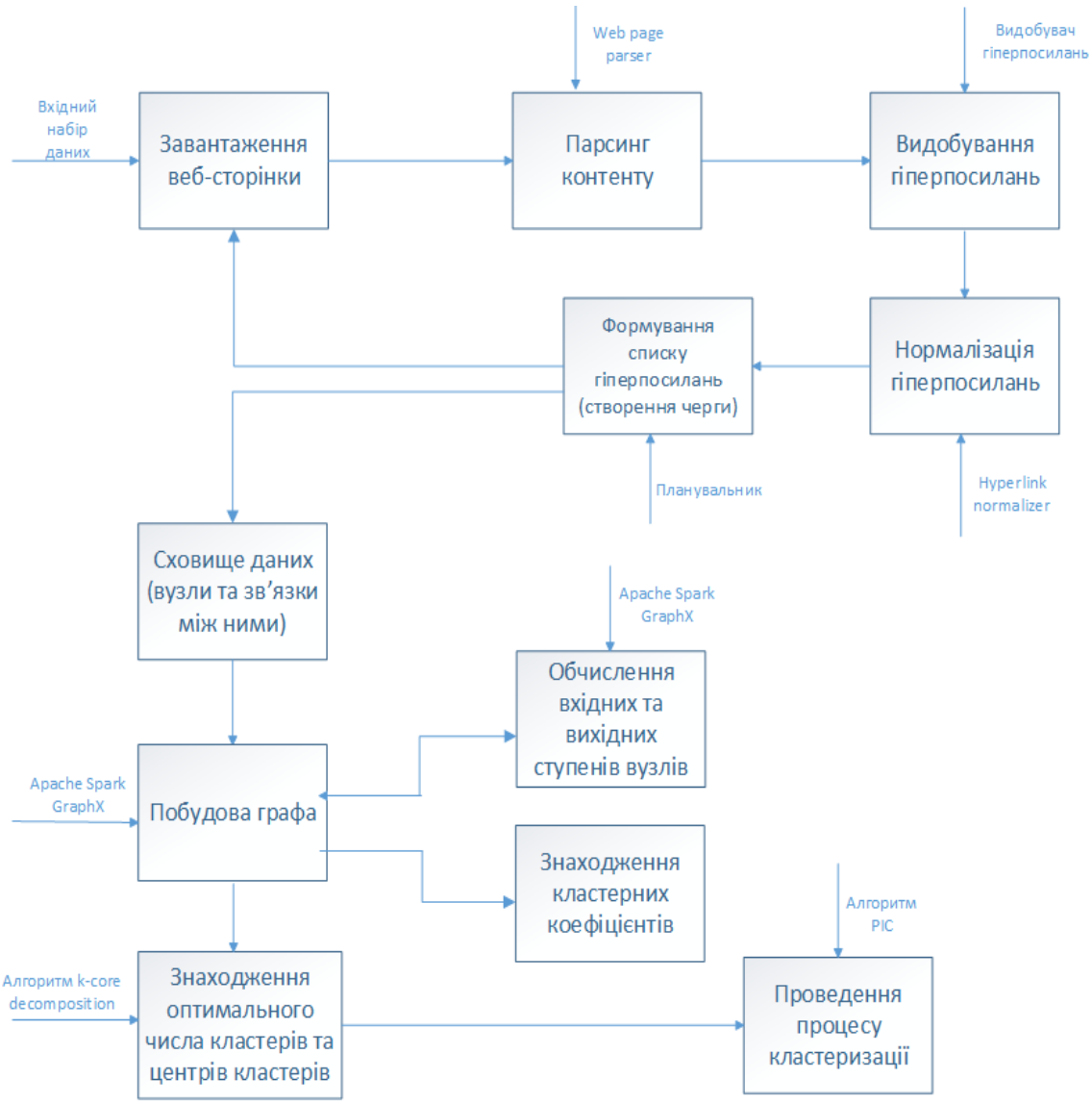


Рис. 3.3.1. Узагальнена схема проведення статистичного та кластерного аналізу у складних мережах [173]

Розроблена інформаційна технологія застосовувалась для проведення статистичного та кластерного аналізу різних зон веб-простору: польського сегменту веб-простору (edu.pl), ізраїльського сегменту (ac.il) та українського (edu.ua). Для кожного сегменту побудовано розподіл ймовірності вузлів по вхідних зв'язках (in degree) та розподіл ймовірності вузлів за ступенями по вихідних зв'язках (out degree). Обчислені коефіцієнти кластерності під мереж, визначено середні значення ступеня вузла для неорієнтованих графів та побудовані графи досліджуваних зон. Методом k-core decomposition

визначено оптимальну кількість кластерів для кожного сегменту веб-простору, знайдено центри кластерів і проведено розбиття досліджуваних сегментів на кластери за допомогою алгоритму PIC (Power iteration clustering) [103; 73; 178; 179].

Перевагою розробленої інформаційної технології є те, що за її допомогою маємо змогу працювати з великими наборами даних зібраних у мережі інтернет, досліджувати їх структуру та статистичні характеристики, проводити процес кластеризації [173].

3.4. Інтелектуальний аналіз статистичних даних сегментів веб-простору

Поняття мережі досить широко використовується у сучасній діяльності людства. Як правило, стрімкий розвиток мереж призводить до появи складних самоорганізованих систем, аналіз яких може надати дослідникам багато інформації про розвиток, сучасний стан та перспективи таких мереж. Тому дослідження комплексних мереж, особливо Word Wide Web (WWW), Інтернет, біологічних, соціальних та інших вважається актуальною науково-технічною задачею [42; 43; 45]. Дослідження загальної структури глобальної WWW-мережі проводились у працях [1; 8], а також різними науковцями проводились дослідження характеристик національних веб-доменів Бразилії, Чилі, Португалії, Греції, Іспанії, Південної Кореї. Однак, досліджень щодо характеристики українського веб-сегменту окремих доменів та зон проводилось мало. Такі дослідження дозволяють робити висновки про ступінь розвитку мереж, наявність чи відсутність в них спамерських структур тощо. Залишається нез’ясованим питання про те, які статистичні характеристики мають окремі зони національних доменів. Цікаво було б дослідити окремі сегменти веб-простору різних країн та порівняти їхні характеристики з характеристиками українського сегменту.

З використанням розробленої методики досліджуються статистичні характеристики та кластерна структура різних зон веб-простору, а саме академічних сегментів: України (edu.ua), Польщі (edu.pl) та Ізраїлю (ac.il).

Для дослідження вищенаведених зон веб-простору, що являють собою комплексні мережі, потрібно мати уявлення про їхню структуру. При проведенні своїх досліджень використаємо стандартну модель складної мережі [12; 162]. Для цього потрібно зібрати дані про кількість вхідних та вихідних вузлів, побудувати розподіл ймовірності вузлів за кількістю ребер $P(k)$ для вхідних (in degree) та вихідних (out degree) ступенів вузлів, визначити середнє значення ступеня вузла для неорієнтованих графів, обчислити кластерний коефіцієнт підмереж, який демонструє рівень зв'язності вузлів у мережі, визначити оптимальну кількість кластерів для кожного сегменту веб-простору двома методами – методом "ліктя" та методом k-core decomposition, провести розбиття досліджуваних мереж на кластери.

3.4.1. Статистичний та кластерний аналіз даних сегменту edu.ua

Не зважаючи на велику кількість публікацій, присвячених статистичному аналізу web-мережі, дослідження українського домену досить фрагментарне та неповне. Проведені дослідження покликані деякою мірою доповнити наші знання про український сегмент глобальної мережі [42; 45; 89; 174].

Для мережі edu.ua на вхід кроулеру подається 100 точок входження, а глибина зондування – 5 «стрибків». Програма, рухаючись заданими веб-сторінками, збирала структуру посилань на інші сторінки та зберігала дані в базу даних. Для зони edu.ua було в середньому просканово більше 400 тисяч веб-сторінок [42-45; 47; 89; 174; 175].

За допомогою розробленої інформаційної технології (описаної в пункті 3 розділу 3) одержали статистичні характеристики сегменту, який досліджуємо [42; 43; 45; 47; 174; 175; 177].

Результати статистичної обробки отриманих результатів по всіх точках входження для зони edu.ua по вихідних зв’язках (out degree) подано на рис. 3.4.1.1.

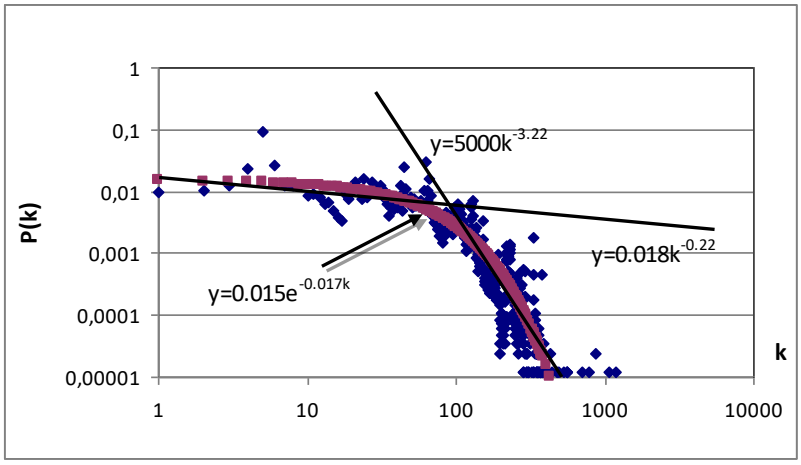


Рис. 3.4.1.1. Розподіл ймовірності вузлів за ступенями по вихідних зв’язках (out degree) для зони edu.ua

Як видно з рис. 3.4.1.1, мережу, яку досліджуємо, не можна описати одним степеневим законом, оскільки вона складається, в основному, з вузлів, які мають значну кількість вихідних зв’язків, тобто середній ступінь вузла зміщений у бік великих значень $\langle k \rangle = 55{,}8$. Такий підхід (апроксимацію різних ділянок різними степеневими законами) був продемонстрований у роботі [41], і ми вважаємо його дещо штучним [42; 43;45].

Більш природно виглядає апроксимація даних експоненційною залежністю. Як бачимо, розподіл імовірності вузлів за їх ступенями підпорядковується експоненційному закону $P(k) \approx A \cdot e^{(-\alpha k)}$, де $A = 0{,}015$ та $\alpha = 0{,}017$.

Статистику підмережі вхідних зв’язків мережі edu.ua подано на рис. 3.4.1.2.

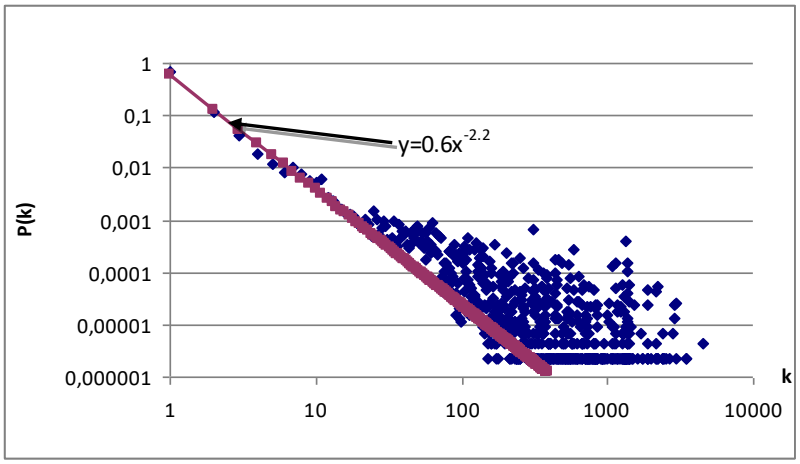


Рис. 3.4.1.2. Розподіл ймовірності вузлів для зони edu.ua по вхідних зв'язках (in degree)

Як видно з рис. 3.4.1.2, початкова ділянка графіка добре апроксимується прямою з показником ступеня (-2,2), що говорить про безмасштабність мережі, яку досліджують, по вхідних зв'язках. Якщо формально розрахувати середнє значення ступеня вузла ⟨k⟩, то воно, як і слід було очікувати, зсунуте в бік малих значень ⟨k⟩ = 9,6, яке в декілька разів менше за значення вихідних зв'язків для мережі edu.ua [42; 43; 45].

Об'єднуючи розраховані залежності для вхідних та вихідних підмереж, можна отримати статистичні характеристики неорієнтованого графу веб-сторінок зони edu.ua. Побудовані залежності наведені на рис. 3.4.1.3 [42; 43; 45; 47; 175; 177].

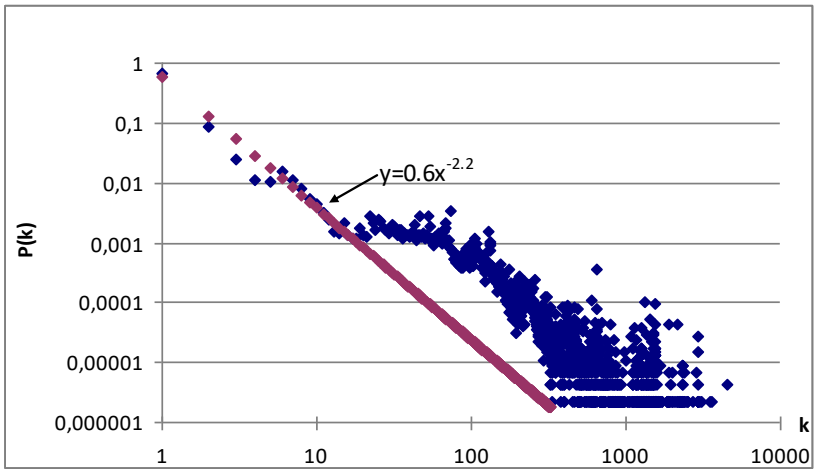


Рис. 3.4.1.3. Розподіл ймовірностей вузлів неорієнтованого графу для зони edu.ua

Аналізуючи отримані результати, можна стверджувати, що при малих значеннях k залежності мають усі властивості підмережі вхідних зв'язків (in degree), а у випадку великих значень k – домінують характеристики підмережі вихідних зв'язків (out degree), про що свідчить характерний «горб». Очевидно, що середні значення ступенів вузлів для таких мереж набувають проміжні значення між in degree та out degree. Зокрема, для зони edu.ua $\langle k \rangle = 19{,}8$. Відмітимо, що якраз у цих областях спостерігається перехід від характеристик in degree до out degree, що наочно проілюстровано на рис. 3.4.1.3.

У результаті проведених досліджень для підмережі edu.ua знайдений коефіцієнт кластерності $C = 0{,}11$, значення якого вказує на значну кількість перехресних посилань найближчих сусідів [44; 174].

Таким чином, аналізуючи статистичні характеристики неорієнтованого графа для зони, яку досліджуємо, можна зробити висновок, що вона не містить значних особливостей, є цілком розвиненою мережею і відповідає сучасним тенденціям розвитку глобальної мережі WWW [42; 43; 45; 175].

Проведемо дослідження кластерної структури отриманих наборів даних, для чого визначимо оптимальну кількість кластерів двома описаними у пункті 3.2 розділу 3 методами: методом "ліктя" та k-core decomposition.

На рис. 3.4.1.4 наведено графік функції штрафу для зони edu.ua веб-простору.

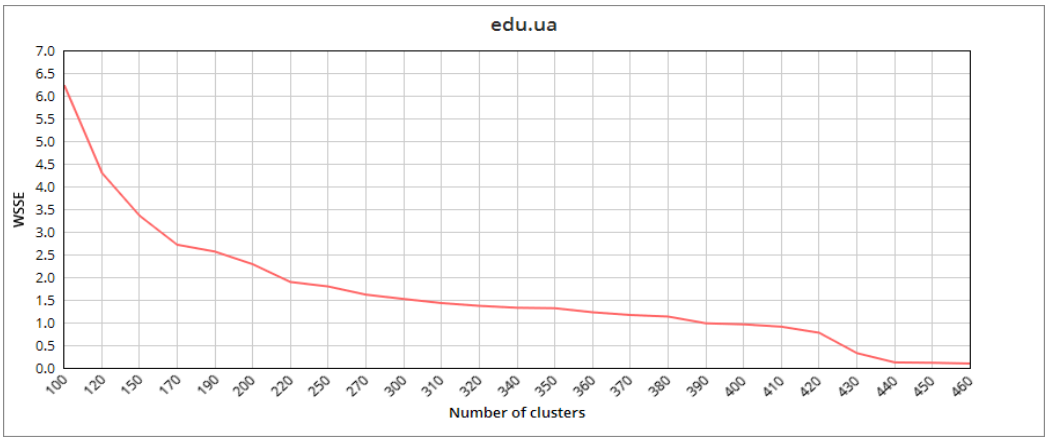


Рис. 3.4.1.4. Залежність функції штрафу від кількості кластерів для зони edu.ua

Визначивши з графіку точку найбільшого перегину, отримаємо оптимальну кількість кластерів: 220 – для edu.ua.

Застосувавши метод k-core decomposition, отримали, що оптимальна кількість кластерів дорівнює 229 [97].

Отже, отримано гарне узгодження оптимальної кількості кластерів для зони edu.ua веб-простору, виконане різними методами.

Залишилося виконати кластеризацію, для якої використовувався метод PIC. Результати наведені на рис. 3.4.1.5.

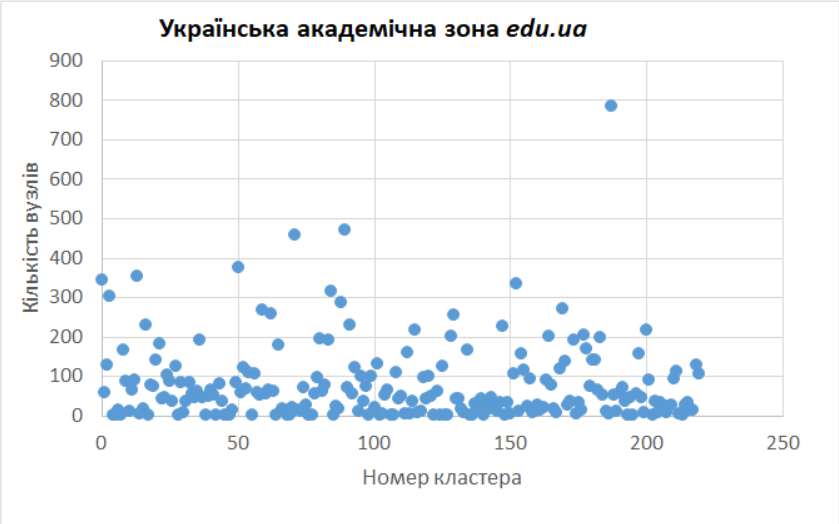


Рис. 3.4.1.5. Кластерна структура української академічної зони edu.ua (220 кластерів)

Структура українського сегменту edu.ua складається, як це видно з рисунку 3.4.1.5, з кластерів, основна кількість яких містить до 300 вузлів, а найбільший кластер – приблизно 800 вузлів.

3.4.2. Статистичний та кластерний аналіз даних сегменту edu.pl

Зацікавило питання, які статистичні характеристики та як поводять себе мережі академічних зон інших країн. За допомогою розробленої інформаційної технології [173], проведено дослідження статистичних характеристик зони академічного сегменту польського домену edu.pl. Для

цієї веб-зони було просканировано приблизно 400 тисяч веб-сторінок, при кількості вхідних вершин – 32 при глибині зондування – 5 «стрибків».

Опишемо отримані статистичні характеристики сегменту edu.pl.

Результати статистичної обробки отриманих результатів по всіх точках вихідних зв'язків (out degree) подано на рис. 3.4.2.1.

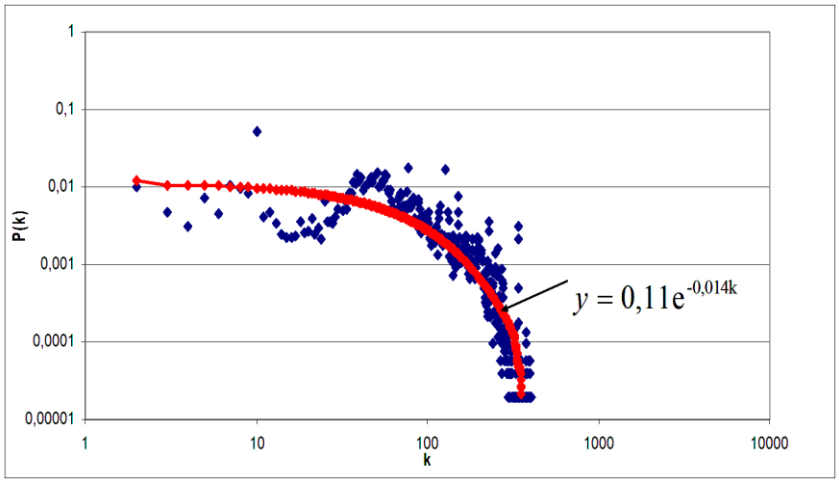


Рис. 3.4.2.1. Розподіл ймовірності вузлів за ступенями по вихідних зв'язках (out degree) для зони edu.pl

Мережа, яка досліджується, в основному складається з вузлів, які мають значну кількість вихідних зв'язків, тобто середній ступінь вузла зміщений в бік великих значень і $\langle k \rangle = 80{,}8$ [42; 176; 177]. Якщо провести апроксимацію даних за експоненційним законом, то для підмережі вихідних зв'язків отримано експоненційний розподіл ймовірності досліджених вузлів $P(k) \approx A \cdot e^{(-\alpha k)}$, з параметрами $A = 0{,}11$ та $\alpha = 0{,}014$.

Статистику підмережі вхідних зв'язків досліджених мереж подано на рис. 3.4.2.2.

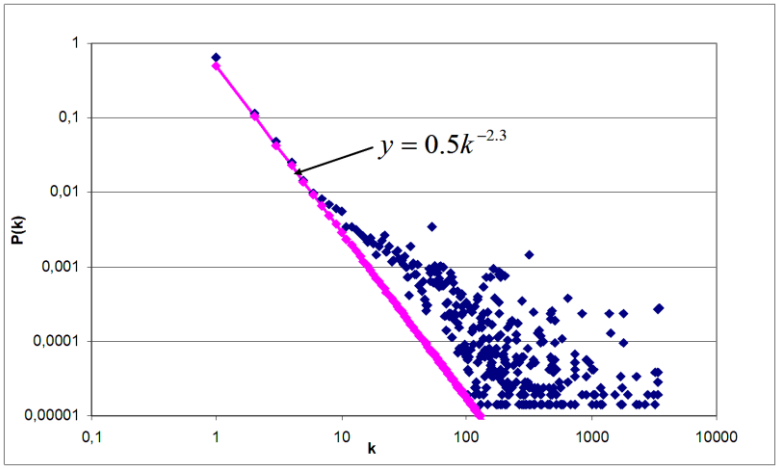


Рис. 3.4.2.2. Розподіл ймовірності вузлів за ступенями по вхідних зв'язках (in degree) для зони edu.pl

Значення показника ступеня дорівнює (-2,3), це свідчить про безмасштабність мережі edu.pl по вхідних зв'язках. Якщо формально розрахувати середнє значення ступеня вузла ⟨k⟩, то воно зміщене в бік малих значень: ⟨k⟩ = 16,1, яке в декілька разів менше за значення вихідних зв'язків для мережі edu.pl [42; 176; 177].

Провівши об'єднання розрахованих залежностей для вхідних та вихідних підмереж, отримали статистичні характеристики неорієнтованого графа веб-сторінок зони edu.pl. Побудовані залежності наведені на рис. 3.4.2.3.

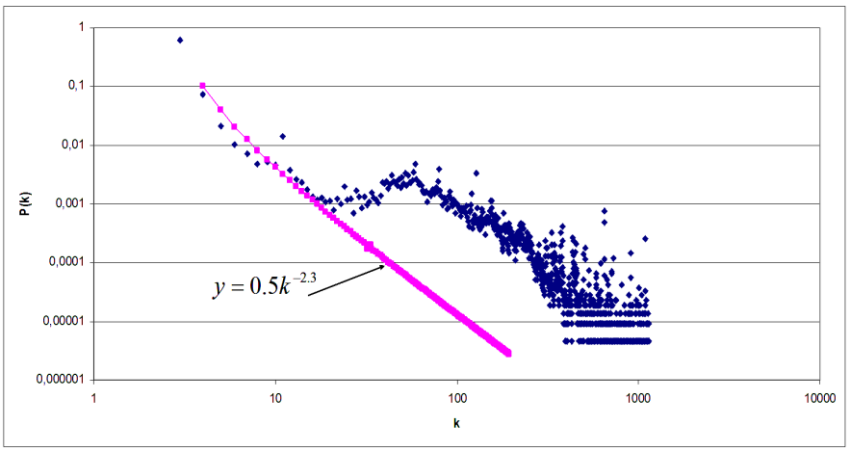


Рис. 3.4.2.3. Розподіл ймовірностей вузлів неорієнтованого графу для зони edu.pl

Характерний вигляд графіка свідчить, що властивості неорієнтованого графа залежать від підмереж вхідних та вихідних зв'язків, причому перша домінує при малих значеннях ступеня вузла, а друга – при великих значеннях [42; 176; 177]. Середнє значення ступеня вузла $\langle k \rangle$ дорівнює $\langle k \rangle = 34{,}6$.

Обчислене значення коефіцієнту кластерності мережі для зони edu.pl – 0,088 вказує на значну кількість перехресних посилань найближчих сусідів.

Для дослідження кластерної структури отриманих наборів даних визначимо оптимальну кількість кластерів двома методами: методом "ліктя" та k-core decomposition. Для польського сегменту веб-простору (edu.pl) оптимальна кількість кластерів методом "ліктя" – 210, методом k-core decomposition – 213 [89; 96; 97].

На рис. 3.4.2.4 подано графік функції штрафу для зони edu.pl веб-простору.

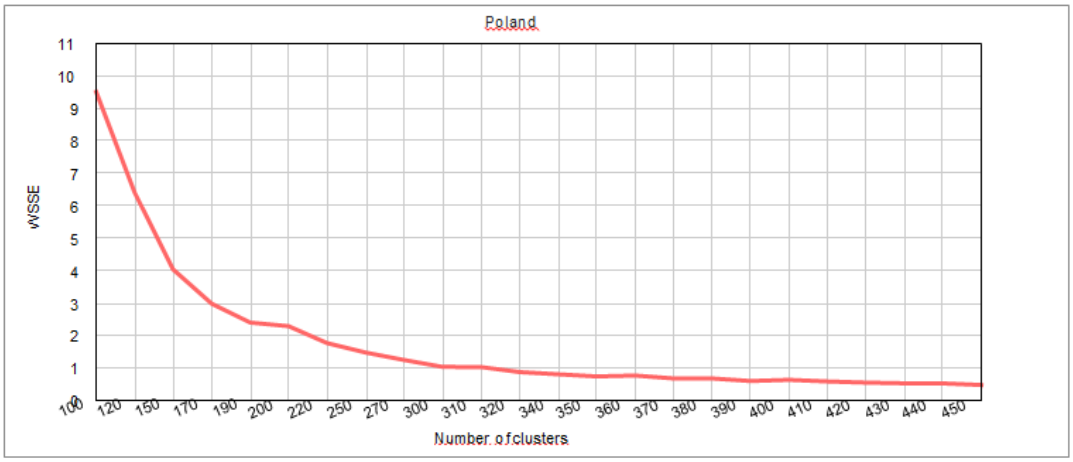


Рис. 3.4.2.4. Залежність функції штрафу від кількості кластерів для зони edu.pl

Виконаємо кластеризацію методом PIC. Розбиття мережі на кластери продемонстровано графіком (рис. 3.4.2.5) [96; 97].

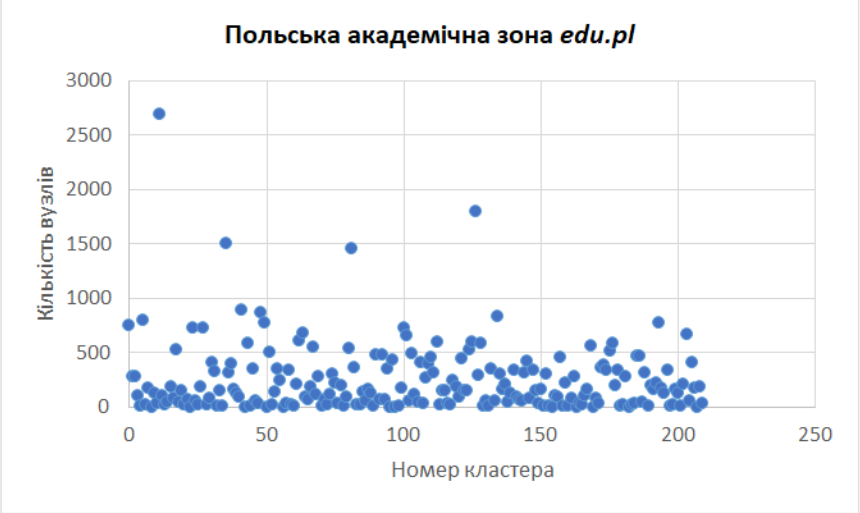


Рис. 3.4.2.5. Кластерна структура польської академічної зони edu.pl (210 кластерів)

Польська зона edu.pl складається, в основному, з кластерів, що містять до 1000 вузлів, але зустрічаються кластери з 1500-2700 вузлів.

3.4.3. Статистичний та кластерний аналіз даних сегменту ac.il

Розглянемо ще одну академічну веб-зону ac.il. Для проведення статистичного та кластерного аналізу академічної зони Ізраїлю (ac.il) інформаційній технології задавалось 27 точок входу та просканово більше 400 тисяч веб-сторінок.

Побудовано розподіл ймовірності вузлів по вихідних (рис. 3.4.3.1) та вхідних зв'язках (рис. 3.4.3.2).

Розподіл ймовірності вузлів за ступенями по вихідних зв'язках найкраще апроксимується експоненціальною залежністю $P(k) \approx A \cdot e^{(-\alpha k)}$, де $A = 0{,}015$ та $\alpha = 0{,}017$. На графіку подано апроксимуючі прямі для малих значень степеня вузлів і для великих його значень [42; 43; 45]. Середній ступінь вузла зміщений у бік великих значень $\langle k \rangle = 52{,}4$.

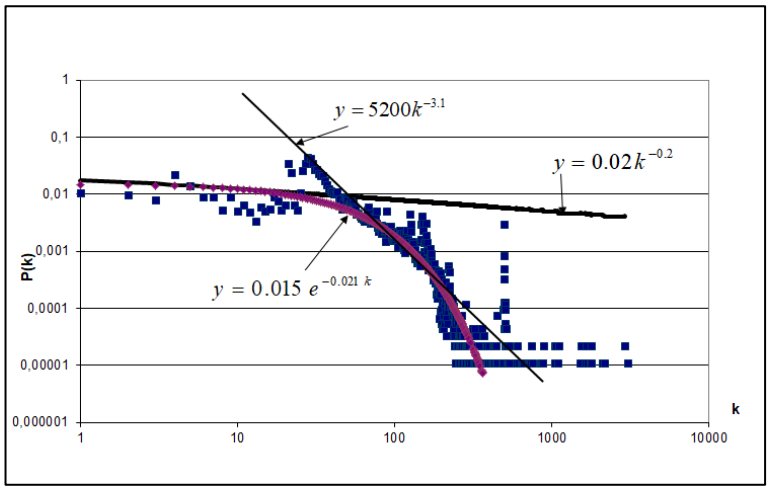


Рис. 3.4.3.1. Розподіл ймовірності вузлів за ступенями по вихідних зв'язках (out degree) для зони ac.il

Розподіл ймовірностей вузлів по вхідних зв'язках (in degree) для сегменту ac.il наведено на рис. 3.4.3.2.

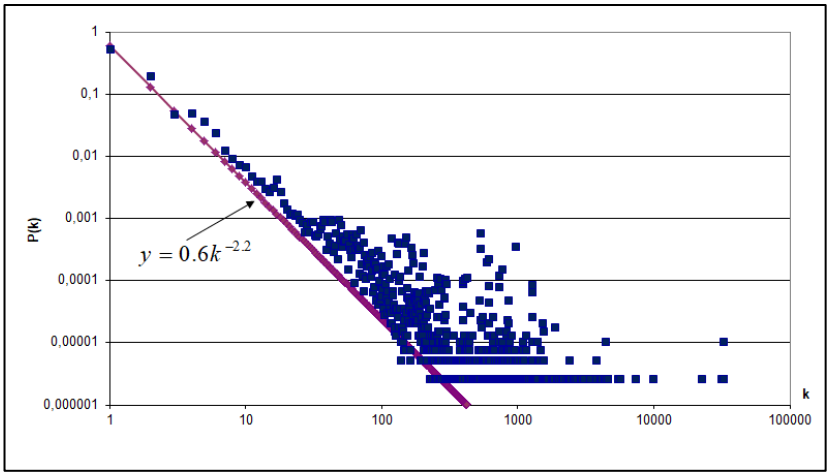


Рис. 3.4.3.2. Розподіл ймовірності вузлів за ступенями по вхідних зв'язках (in degree) для зони ac.il

Розподіл ймовірностей сегменту ac.il по вхідних зв'язках апроксимуються степеневим законом з показником степеня (-2,2), що свідчить про безмасштабність мережі по вхідних зв'язках [42; 43; 45]. При формальному розрахунку середнє значення степеня вузла <k> зміщене в бік малих значень: $\langle k \rangle = 10{,}1$.

Статистичні характеристики неорієнтованого графа веб-сторінок зони ac.il отримаємо після об’єднання розрахованих залежностей для вхідних та вихідних підмереж. Побудовані залежності наведені на рис. 3.4.3.3.

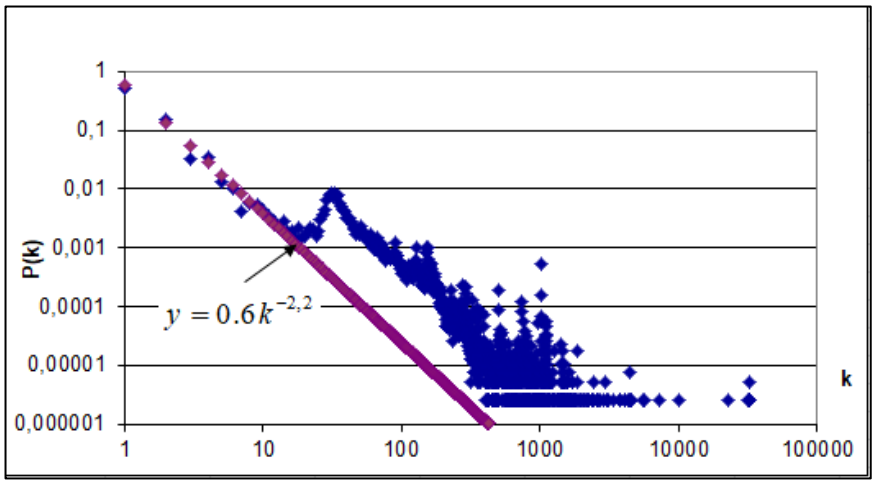


Рис. 3.4.3.3. Розподіл ймовірностей вузлів неорієнтованого графу для зони ac.il

З рис. 3.4.3.3. видно, що при малих значеннях степеня вузлів розподіл ймовірності для неорієнтованого графа визначається залежністю по вхідних зв’язках, а для великих – по вихідних [42; 43; 45; 175; 177]. Середні значення степеня вузла для таких мереж, зрозуміло, набувають проміжних значень: $\langle k \rangle = 22{,}6$.

Для підмережі ac.il обчислено коефіцієнти кластерності: $C = 0{,}104$,

Побудувавши графік функції штрафу для досліджуваної зони веб-простору (рис. 3.4.3.4) та використавши метод “ліктя”, визначили оптимальну кількість кластерів: $k_{opt} = 190$ [89].

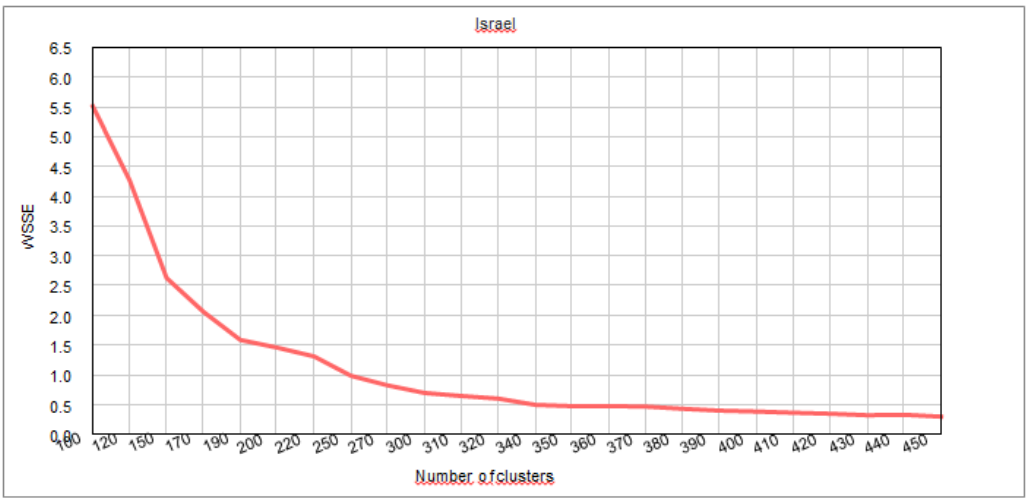


Рис. 3.4.3.5. Залежність функції штрафу від кількості кластерів для зони ac.il

Також визначили оптимальну кількість кластерів іншим методом – методом k-core decomposition: $k_{opt} = 196$ [96; 97].

Застосувавши метод PIC [44; 96], проведено кластеризацію даних. Кластерна структура ізраїльської академічної зони ac.il (190 кластерів) наведена на рис. 3.4.3.6.

Рис. 3.4.3.6. Кластерна структура ізраїльської академічної зони ac.il (190 кластерів)

Ізраїльський академічний сегмент складається з переважної кількості вузлів, у кластерах якого налічується до 2000 вузлів, хоча зустрічаються і кластери з 6000-11000 вузлів.

3.5. Обговорення отриманих результатів

Багато робіт присвячено вивченню структури WWW-простору, статистичні характеристики якої вивчають як граф, вузлами якого є web-сторінки, а ребрами – зв'язки між ними. Досліджуються як орієнтовані, так і неорієнтовані графи [1; 8]. З'ясовано, що всесвітня мережа WWW підкоряється статистичним законам комплексних мереж, причому встановлено, що розподіл вузлів графу, який відображає всесвітню павутину, підкоряється степеневому закону з показником, близьким до (-2,2) для вхідних зв'язків та (-2,7) – для вихідних. Це свідчить про безмасштабність такої мережі, тобто про високий ступінь розвиненості мережі в цілому [1; 8],

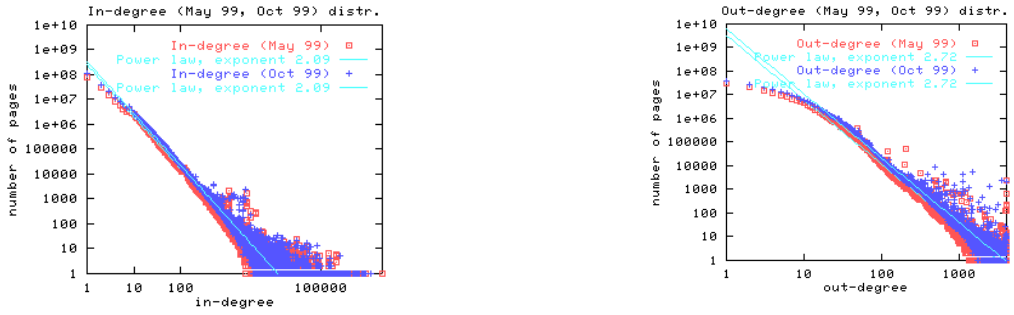


Рис. 3.5.1. Розподіл кількості веб-сторінок по вхідних (ліворуч) та вихідних (праворуч) зв'язках за даними роботи [8]

Через великий масштаб мережі та досить великий об'єм вибірки важко зробити висновки про характеристики Web. Також постає питання про репрезентативність такої вибірки. В літературі описані дослідження, які проводились на даних трьох типів: повне сканування одного веб-сайту (дані однорідні), випадкова вибірка з всього Інтернету (вибірка великого об'єму, тому дані менш повні і неоднорідні) і великі зразки даних конкретних спільнот. Якщо ж розглядати вибірки даних з певних спільнот, які мають спільний географічний, історичний чи культурний контекст, наприклад

національних доменів, то вони мають хороший баланс між різноманітністю та повнотою даних. Крім того, вони мають відносно помірний розмір, що забезпечує точність результатів досліджень. Так, у роботах [41; 50; 213-216] досліджувалися статистичні характеристики національних веб-доменів Бразилії, Чилі, Греції, Південної Кореї та Іспанії. Результати досліджень наведено на рис. 3.5.2 та 3.5.3, де відображено розподіл ймовірності веб-сторінок в залежності від їх степеня по вхідних та вихідних зв'язках.

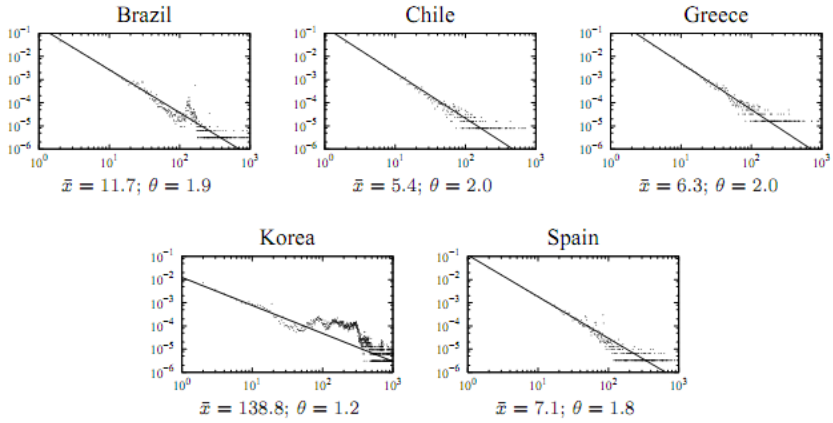


Рис. 3.5.2. Розподіл ймовірностей веб-сторінок в залежності від їх степеня для національних web-доменів по вхідних зв'язках

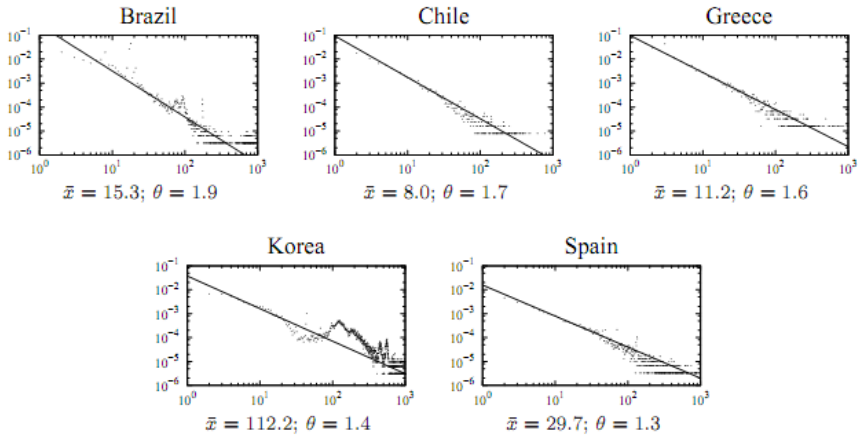


Рис. 3.5.3. Розподіл ймовірностей веб-сторінок в залежності від їх степеня для національних web-доменів по вихідних зв'язках

З рисунків видно, що південнокорейський домен має певні особливості порівняно з іншими: нестандартний вигляд кривої при середніх та великих

степенях вузлів, а також велике значення середнього степеня вузла як по вхідних, так і по вихідних зв'язках. Автори [41] ці особливості пов'язують з наявністю у цьому домені спамерських сайтів, які якраз і характеризуються сильними перехресними зв'язками.

Як видно з наведених результатів, статистичні характеристики веб-простору як усієї мережі WWW, так і національних веб-доменів можна описати за допомогою безмасштабних мереж, які підкоряються степеневому закону розподілу з показниками 1.8÷2.2 для вхідних зв'язків та 1.3÷2.7 – для вихідних.

Український сегмент Інтернет досліджувався в роботі [46], де також отримано аналогічні результати. Опорна мережа, що розглядалась, також виявилась безмасштабною, а залежність розподілу степенів її вузлів з високою точністю апроксимуються степеневою функцією. Обчислений коефіцієнт кластерності – 0,04, свідчить про те, що дана мережа (як і весь Інтернет [40]) є малим світом (Small World) [18]. Це дозволило авторам зробити висновок про досить високий рівень захищеності українського сегменту Інтернет від зовнішнього несистематичного «впливу» та про необхідність цілеспрямованого його підвищення для запобігання цілеспрямованих атак на основні вузли комутації.

Слід зазначити, що оскільки інтернет постійно змінюється, розширюється та росте, то відповідно й отримані характеристики також будуть змінюватись. Результати також будуть залежати від кроулерів чи сканерів, які збирають дані, та їх налаштувань. Веб-характеристики даних національних веб-доменів доповнюють дані**,** зібрані кроулерами та сканерами, і надають додаткову інформацію для пояснення поведінки та тенденції розвитку інтернету.

З використанням розробленої методики та інформаційної технології проведено статистичний та кластерний аналіз зон веб-простору: академічних сегментів України (edu.ua), Польщі (edu.pl) та Ізраїлю (ac.il). Для кожного сегменту мережі вибиралася певна кількість точок входу (таблиця 3.5.1), яка

використовувалися кроулером для зондування, із глибиною зондування 5 кроків.

Таблиця 3.5.1.

Кількість точок входу для досліджуваних сегментів мережі

Назва зони	edu.ua	ac.il	edu.pl
Кількість точок входу	100	27	32

Для кожного сегменту зібрані дані про кількість вхідних та вихідних вузлів, побудовано розподіл ймовірностей вузлів по вхідних та вихідних зв'язках, визначено середні значення ступеня вузла, побудовано неорієнтовані графи підмереж, що досліджуються, обчислено їхні коефіцієнти кластерності.

Як видно з рисунків (рис. 3.4.1.1, рис. 3.4.2.1, рис. 3.4.3.1), усі академічні зони веб-простору трьох країн по вхідних зв'язках мають дуже подібні статистичні характеристики [42; 45]. По вхідних зв'язках усі відповідають концепції "малого світу": розподіл ймовірності демонструє степеневу залежність з показниками (-2.2)÷(-2.3), що говорить про безмасштабність мереж по вхідних зв'язках. Зазначимо, що характер отриманих залежностей повністю відповідає отриманим раніше результатам сканування веб-сторінок Оксфордського університету (рис. 3.5.4), де було досліджено порядку 10 тисяч сторінок.

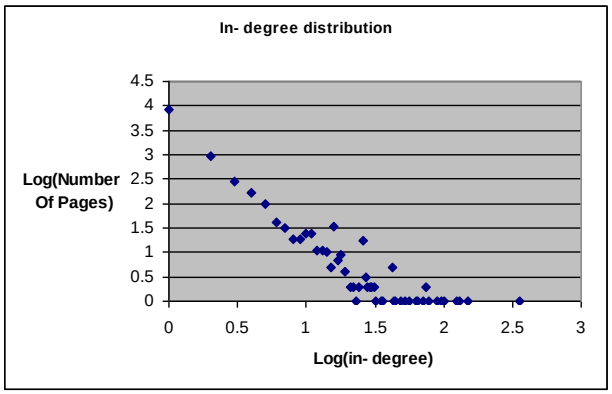


Рис. 3.5.5. Розподіл ймовірностей вузлів по вхідних зв'язках для Оксфордського університету

Якщо формально розрахувати середнє значення ступеня вузла $\langle k \rangle$, то воно зміщене в бік малих значень (таблиця 3.5.2).

Таблиця 3.5.2.

Значення середнього ступеня вузла (по вхідних зв'язках)

Назва зони	edu.ua	ac.il	edu.pl
Середнє значення вузла	9,6	10,1	16,1

Якщо вважати, що ізраїльський та польський академічні сегменти є цілком розвиненими мережами, то те ж саме можна стверджувати й про український академічний сегмент веб-простору, адже його статистичні характеристики цілком подібні до перших двох.

По вихідних зв’язках спостерігається аналогічна картина: усі три академічні зони, які досліджуються, демонструють схожі статистичні характеристики і найкраще апроксимуються експоненційними залежностями ($P(k) \approx A \cdot e^{(-\alpha k)}$) з досить близькими параметрами (таблиця 3.5.3).

Таблиця 3.5.3.

Значення параметрів розподілів ймовірностей вузлів

Назва зони	Коефіцієнт A	Коефіцієнт α
edu.ua	0,015	0,017
ac.il	0,015	0,021
edu.pl	0,11	0,014

Для порівняння на рисунку 3.5.5 подано аналогічні дані по вихідних зв’язках для Оксфордського університету, хід залежності яких аналогічний дослідженим нами. Причиною цього може бути те, що експоненційна залежність при малій та середній кількості вузлів переходить у степеневу при майже десятиразовому збільшенні вузлів, що відбувається при дослідженні глобального веб-простору в роботах [8; 43].

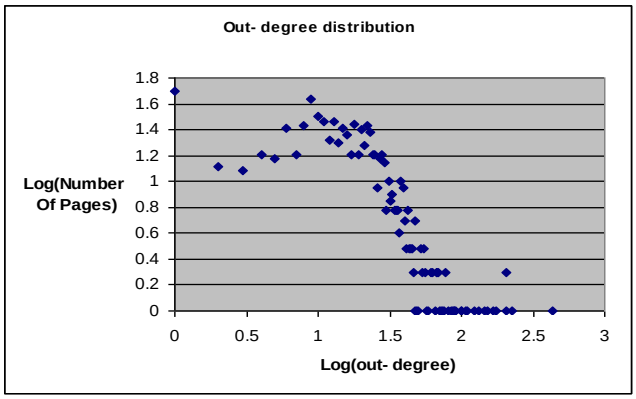


Рис. 3.5.5. Розподіл ймовірностей вузлів по вихідних зв’язках для Оксфордського університету

Як видно з рис. 3.4.1.1, рис. 3.4.2.1, рис. 3.4.3.1, ці мережі не можна описати одним степеневим законом. У крайньому разі, можна підібрати апроксимуючі прямі окремо для малих значень ступеня веб-сторінок та великих його значень. Такий підхід (апроксимацію різних ділянок різними степеневими законами) був продемонстрований у роботі [41], і його можна вважати дещо штучним [45]. Особливості статистичних характеристик по вихідних зв’язках, які ми спостерігаємо в наших дослідженнях, видно також на графіках, отриманих авторами в роботі [41] для національних веб-доменів Бразилії, Греції та Іспанії.

З графіків (рис. 3.4.1.1, рис. 3.4.2.1, рис. 3.4.3.1) видно, що досліджені мережі складаються, в основному, з вузлів, які мають значну кількість вихідних зв’язків, тобто середній ступінь вузла зміщений в бік великих значень (таблиця 3.5.4).

Таблиця 3.5.4.

Значення середнього ступеня вузла (по вихідних зв'язках)

Назва зони	edu.ua	ac.il	edu.pl
Середнє значення вузла	55,8	52,4	80,8

Можна зробити висновок, що українська академічна зона також цілком відповідає тенденціям розвитку веб-простору. Тим не менше, на відміну від

попередніх дослідників, ми не можемо стверджувати, що статистика за вихідними зв'язками також відповідає степеневому закону, що стверджувалося у роботах [12; 41; 162; 163].

Отримано статистичні характеристики неорієнтованого графу веб-сторінок зон edu.ua, edu.pl, ac.il [42; 43; 45]. Аналізуючи отримані результати, можна стверджувати, що при малих значеннях k залежності мають усі властивості підмережі вхідних зв'язків (in degree), а у випадку великих значень k – домінують характеристики підмережі вихідних зв'язків (out degree), про що свідчать характерні «горби» на графіках (рис. 3.4.1.3, рис. 3.4.2.3, рис. 3.4.3.3). Очевидно, що середні значення ступенів вузлів для таких мереж набувають проміжні значення між in degree та out degree (таблиця 3.5.5).

Таблиця 3.5.5.

Значення середнього ступеня вузла (неорієнтованого графу)

Назва зони	edu.ua	ac.il	edu.pl
Середнє значення вузла	19,8	22,6	34,6

Відмітимо, що якраз у цих областях спостерігається перехід від характеристик in degree до out degree.

Таким чином, аналізуючи статистичні характеристики неорієнтованих графів для зон, які досліджувались, можна зробити висновок, що вони є цілком розвиненими мережами і відповідають сучасним тенденціям розвитку глобальної мережі інтернет [42; 43; 45; 47; 175; 177].

Оскільки застосовувався метод зондування мережі, було також проведено дослідження, як змінюється статистика мереж від глибини зондування, тобто від кількості «стрибків» кроулера відносно точок входження в мережу.

З отриманих результатів можна зробити висновок, що глибина зондування, як це й очікувалося, дещо змінює статистику: зі збільшенням кількості «стрибків» незначно зростає коефіцієнт у показнику експоненти, а

самі залежності дещо «розтягуються» по осі ординат, що призводить також до зміни коефіцієнта перед експонентою. Цікавим є той факт, що статистика вузлів з малим ступенем змінюється незначно, тоді як при збільшенні глибини зондування значно збільшується кількість вузлів з великою кількістю вихідних зв'язків, що призводить також до зростання середнього ступеню вузлів [45]. Отже, збільшення глибини кроулінгу призводить до кількісних змін, а якісні показники змінюються незначно [175].

У результаті проведених досліджень отримано також коефіцієнти кластерності досліджуваних сегментів (таблиця 3.5.6). Дані вказують на велику кількість перехресних посилань найближчих сусідів [42; 44; 175; 177].

Таблиця 3.5.6.

Коефіцієнти кластерності для підмереж

Назва зони	edu.ua	ac.il	edu.pl
Коефіцієнт кластерності	0,11	0,104	0,088

Для реально існуючих мереж типовими є висока скорельованість та відносно велике значення коефіцієнта кластеризації (≈0.15) порівняно з відповідним його значенням для класичного випадкового графа (≈0,0002) Отримані значення свідчать про те, що інтернет-товариства українських навчальних закладів слабко пов'язані між собою і знаходяться в стадії формування та активного росту. Зазначимо, що для усіх мереж коефіцієнти кластерності коливаються біля значення 0.1, що також вказує на подібні статистичні характеристики усіх досліджених сегментів.

Визначено оптимальну кількість кластерів двома методами – методом "ліктя" та методом k-core decomposition [89; 96; 97]. Кількість кластерів, розрахована методом "ліктя", виявилася приблизно однаковою для усіх досліджених сегментів веб-простору. Результати визначення оптимальної кількості кластерів методом k-core decomposition та порівняння з методом "ліктя" наведено у таблиці 3.5.7.

Таблиця 3.5.7.

Оптимальна кількість кластерів методами
«ліктя» та k-core decomposition

Назва сегменту веб-простору	Метод «ліктя»	Метод k-core decomposition
Український сегмент веб-простору (edu.ua)	220	229
Польський сегмент веб-простору (edu.pl)	210	213
Ізраїльський сегмент веб-простору (ac.il)	190	196

З таблиці 3.5.7 видно, що отримано гарне узгодження оптимальної кількості кластерів для усіх досліджених зон веб-простору, виконане різними методами.

Застосувавши алгоритм PIC, проведено дослідження кластерної структури трьох сегментів веб-простору. Розбиття досліджуваних мереж на кластери продемонстровані графіками (рис. 3.4.1.5, рис. 3.4.2.5, рис. 3.4.3.5) [89; 96].

Аналіз графіків демонструє, що найбільш однорідну (розвинену) кластерну структуру має ізраїльський академічний сегмент, переважна кількість вузлів у кластерах якого – до 2000, хоча зустрічаються кластери з 6000-11000 вузлів. Польська зона edu.pl складається, переважно, з кластерів, що містять до 1000 вузлів, але зустрічаються кластери з 1500-2700 вузлів. Український сегмент edu.ua становить, як це видно з графіку, найменш розвинену структуру: основна кількість кластерів містить до 300 вузлів, а найбільший кластер – приблизно 800 вузлів, що значно менше за польську, а тим більше, за ізраїльську зони.

Очевидно, що такі результати свідчать про те, що український академічний сегмент веб-простору, хоча й демонструє статистичні характеристики, подібні до академічних мереж інших країн (Польщі та Ізраїлю), але має розвиватися в бік збільшення інтернет-представництва

навчальних закладів, а особливо, у формуванні зв’язків між веб-сайтами різних освітянських закладів, які формують простір edu.ua.

Проведені дослідження дозволяють зробити висновок про те, що академічна зона українського Інтернету не має значних особливостей, а рівень розвитку повністю відповідає сучасним тенденціям розвитку глобальної мережі Інтернет [42; 47; 97]. Тут відсутні нестандартні утворення, великі кластери, характерні, наприклад, для сайтів спамерів, які могли би змінити статистичний портрет мереж [177].

ВИСНОВКИ ДО РОЗДІЛУ 3

У цьому розділі проведено статистичний та кластерний аналіз зон веб-простору: українського (edu.ua), ізраїльського (ac.il) та польського (edu.pl). Наведено опис статистичних характеристик інформації у веб-просторі та методи кластерного аналізу для обробки цієї інформації.

У результаті проведених досліджень можна зробити наступні висновки:

1. Розроблено інформаційну технологію для проведення статистичного та кластерного аналізу інформації у складних мережах, яка складається з етапів: збирання та узагальнення інформації з веб-сторінок; подання зібраної кроулером інформації у вигляді графу; визначення оптимальної кількості кластерів та їхніх центрів; проведення процесу кластеризації.

2. За допомогою розробленої технології досліджено статистичні характеристики та кластерну структуру українського освітянського сегменту *edu.ua*, польської підмережі *edu.pl* та ізраїльської академічної зони *ac.il*. Для кожного сегменту побудовано розподіл ймовірності вузлів по вхідних (in degree) та вихідних (out degree)зв'язках, визначено середнє значення ступеня вузла для неорієнтованих графів. Проведено порівняльний аналіз статистичних характеристик досліджених мереж українського веб-простору з аналогічними сегментами розвинених країн. Показано, що вони є цілком розвиненими структурами та відповідають сучасним тенденціям розвитку глобальної мережі WWW, володіють властивостями безмасштабних графів. Українська підмережа *edu.ua* демонструє подібні до інших статистичні характеристики, але явно ненасичена вузлами освітянських закладів, що й продемонстрував кластерний аналіз. Український сегмент *edu.ua* становить найменш розвинену структуру, кількість вузлів у кластерах – найменша з усіх досліджених сегментів.

3. Проведено порівняння двох методів визначення оптимальної кількості кластерів у наборі даних: метода "ліктя" та *k-c*ore decomposition, які показали практично однакові результати, що свідчить про адекватність

підходів та зібраних статистичних даних. Тим не менше, зважаючи на те, що вже існують алгоритми, які дозволяють швидко змінювати *k-c*ore при динамічній зміні відповідного сегменту веб-простору без необхідності обходу цілого графу, на відміну від методу «ліктя», метод *k-c*ore decomposition ми вважаємо найбільш перспективним для проведення кластерного аналізу.

4. Обчислено кластерні коефіцієнти підмереж. Отримані значення вказують на велику кількість перехресних посилань найближчих сусідів.

5. Показано, що збільшення глибини зондування (кількості «стрибків» кроулера від точок входження) призводить лише до кількісних змін, а якісні показники мережі при цьому залишаються майже незмінними.

Основні результати третього розділу опубліковані в роботах [42-45; 47; 89; 96; 97; 103; 173-179].

РОЗДІЛ 4
ОЦІНКА ОПТИМАЛЬНОЇ КІЛЬКОСТІ КЛАСТЕРІВ СКЛАДНИХ МЕРЕЖ НА ОСНОВІ ВИПАДКОВИХ МАТРИЦЬ

4.1. Основні відомості та твердження

Теорія матриць має застосування у багатьох прикладних задачах, оскільки лаконічно описує взаємодію декількох процесів чи систем із багатьма факторами або числові характеристики багатовимірних процесів. У теорії випадкових [220; 221] процесів матриці використовуються як числові характеристики даних процесів, наприклад коваріаційна матриця. Крім того, теорія матриць широко використовується у прикладних задачах машинного навчання [222], в тому числі задачах класифікації та кластеризації.

Широкого вжитку набули задачі, в яких математичними моделями являються так звані випадкові матриці [223]. Теорія випадкових матриць розвивається як частина теорії граничних теорем та спектральної теорії матриць. Основна увага в теорії випадкових матриць приділяється граничним розподілам власних значень при $N \to \infty$, де N – розмірність матриці. Наведемо декілька основних означень та фактів із загальної теорії випадкових матриць, які будуть використані при доведенні основних тверджень. Спочатку розглянемо поняття випадкової матриці та ансамблів пов'язаних із конкретними розподілами.

Означення 4.1. Матриця $A = A^{(N)} = \left(A_{ij}\right)_{i=1,\dots,n,j=1,\dots,p}$ називається випадковою матрицею, якщо її елементи A_{ij} є випадковими величинами.

Означення 4.2. Сукупність матриць $A = A^{(N)}, N \geq 1$ називається гауссівським ансамблем, якщо всі елементи матриці A_{ij} мають нормальний розподіл, тобто $A_{ij} \sim \boldsymbol{N}(\mu, \sigma^2)$. Причому, гауссівський ансамбль комплекснозначних матриць називається

- ортогональним ансамблем (Gaussian Orthogonal Ensemble, GOE), якщо матриця A є симетричною; елементи

матриці вище головної діагоналі та елементи матриці на головній діагоналі є незалежними; елементи матриці мають наступний розподіл

$$A_{jk} \sim \boldsymbol{N}\left(0,\frac{1}{2}\right), k > j;$$

$$A_{jj} \sim \boldsymbol{N}(0,1);$$

- унітарним ансамблем (Gaussian Unitary Ensemble, GUE), якщо матриця A є ермітовою матрицею; елементи матриці вище головної діагоналі та елементи матриці на головній діагоналі є незалежними; елементи матриці вище головної діагоналі мають представлення

$$A_{jk} = U_{jk} + iV_{jk}, k > j,$$

$$U_{jk}, V_{jk} \sim i.i.d.\boldsymbol{N}\left(0,\frac{1}{4}\right);$$

$$A_{jj} \sim \boldsymbol{N}\left(0,\frac{1}{2}\right);$$

- симплектичним ансамблем (Gaussian Symplectic Ensemble, GSE), якщо матриця A є ермітовою матрицею; елементи матриці вище головної діагоналі та елементи матриці на головній діагоналі є незалежними; елементи матриці вище головної діагоналі мають представлення

$$A_{jk} = U_{jk} + iV_{jk}, k > j,$$

$$U_{jk}, V_{jk} \sim i.i.d.\boldsymbol{N}\left(0,\frac{1}{8}\right);$$

$$A_{jj} \sim \boldsymbol{N}\left(0,\frac{1}{4}\right).$$

Інколи три гауссівські ансамблі визначаються за допомогою індексу Дайсона [224]. В цьому випадку розподіл елементів матриці A, які належать до одного із ансамблів GUE, GOE і GSE, можна представити наступним чином:

- GOE. В цьому випадку сумісна щільність розподілу елементів задається співвідношенням

$$f(A) = \frac{1}{Z_{GOE}(N)} e^{-\frac{N}{2}tr(A^2)},$$

де $Z_{GOE}(N)$ – нормуюча константа, що задається співвідношенням

$$Z_{GOE}(N) = 2^{\frac{N}{2}} \pi^{\frac{N(N+1)}{4}};$$

- GUE. В цьому випадку сумісна щільність розподілу елементів задається співвідношенням

$$f(A) = \frac{1}{Z_{GUE}(N)} e^{-Ntr(A^2)},$$

де $Z_{GUE}(N)$ – нормуюча константа, що задається співвідношенням

$$Z_{GUE}(N) = 2^{-N} \pi^{\frac{3N}{2}};$$

- GSE. Тут сумісна щільність розподілу елементів має вигляд

$$f(A) = \frac{1}{Z_{GSE}(N)} e^{-2Ntr(A^2)},$$

де $Z_{GSE}(N)$ – нормуюча константа.

Ще одним важливим класом випадкових матриць є так звані матриці Вігнера, з яких і започатковано розгляд випадкових матриць та дослідження їх спектральних характеристик.

Означення 4.3. Випадкова матриця називається матрицею Вігнера, якщо вона має такі властивостями:

1. Матриця є симетричною комплекснозначною матрицею.
2. Елементи головної діагоналі та елементи вище головної діагоналі є незалежними в сукупності та мають розподіл Гаусса з нульовим математичним сподіванням.

3. Дисперсія елементів, що знаходяться вище головної діагоналі, дорівнює одиниці.

4. Дисперсія елементів, що знаходяться на головній діагоналі, скінченна.

Зауважимо, що основні результати для випадкових матриць стосуються розподілу власних значень останніх. Класичний результат щодо розподілу власних значень стосується розподілу власних значень матриці Вігнера. Надалі через $\mu_N(\cdot)$ будемо позначати розподіл власних значень матриці $\frac{1}{\sqrt{N}}A^{(N)}$, де основну роль буде саме відігравати залежність від розмірності матриці, тобто

$$\mu_N(B) = \frac{\#\left\{\lambda_j\left(\frac{1}{\sqrt{N}}A^{(N)}\right): \lambda_j\left(\frac{1}{\sqrt{N}}A^{(N)}\right) \in B\right\}}{N}. \qquad (4.1.1)$$

Найпростішим результатом щодо збіжності розподілу щільностей розподілу (4.1.1) у випадку несиметричних матриць наведено в роботах [169; 225].

Твердження 4.1. [225; 169]. Нехай $A = A^{(N)}, N \geq 1$ – сукупність випадкових матриць із незалежними елементами $A_{ij}, i, j = 1, \dots, N$, для яких $EA_{ij} = 0$, $DA_{ij} = \sigma^2 < \infty$. Тоді має місце слабка збіжність розподілу (4.1.1) до кругового розподілу

$$\mu_N(\cdot) \rightrightarrows \mu_C(\cdot),$$

де $\mu_C(\cdot)$ – розподіл на крузі комплексної множини із центром у початку координат та радіусом $r = 1$, що задається щільністю

$$f_C(z) = \begin{cases} \frac{1}{\pi}, & |z| \leq 1, \\ 0, & \text{в іншому випадку.} \end{cases}$$

Твердження 4.2. [223]. Нехай $A = A^{(N)}, N \geq 1$ – ансамбль із розподілом Вігнера та $\mu_N(\cdot)$ відповідний розподіл власних значень матриці $\frac{1}{\sqrt{N}}A^{(N)}$. Тоді має місце слабка збіжність

$$\mu_N(\cdot) \rightrightarrows \mu_{SC}(\cdot)$$

при $N \to \infty$, де $\mu_{SC}(\cdot)$ – напівколовий розподіл Вігнера зі щільністю

$$f_{SC}(x) = \frac{1}{2\pi}\sqrt{4 - x^2}, x \in [-2,2].$$

У роботі [169] розглянуте узагальнення даної теореми на випадок наявності доданків з обмеженим рангом.

Твердження 4.3. [169]. Нехай

1. $A = A^{(N)}, N \geq 1$ – ансамбль із розподілом Вігнера.
2. $B^{(N)}$ – послідовність детермінованих квадратних матриць розмірності $N \times N$, ранг яких рівний $o(N)$.
3. $\mu_N(\cdot)$ – розподіл власних значень матриці $\frac{1}{\sqrt{N}}\left(A^{(N)} + B^{(N)}\right)$.

Тоді має місце слабка збіжність

$$\mu_N(\cdot) \rightrightarrows \mu_{SC}(\cdot)$$

при $N \to \infty$.

У роботі [226] результати твердження 4.3 посилені до збіжності майже напевне у випадку додаткового обмеження на елементи детермінованих матриць $B^{(N)}$, а саме

$$\sup_{N \geq 1} \left\| B^{(N)} \right\|_F^2 < \infty,$$

де $\|\cdot\|_F$ – матрична норма Фробеніуса.

Узагальнення твердження 4.3 на випадок залежних випадкових елементів матриці $A^{(N)}$ розглянуті в роботі [227], де припущення про незалежність випадкових величин замінюються на припущення про скінченну кореляцію між ними. Розглянемо клас випадкових матриць, який узагальнює поняття матриць Вігнера.

Означення 4.4. Матриця $A = A^{(N)}$ задовольняє властивість C1, якщо виконуються наступні умови:

1. Матриця є дійснозначною матрицею.
2. Діагональні елементи матриці є незалежними та не залежать від елементів матриці поза головною діагоналлю.

3. Вектори $(A_{ij}, A_{ji}), i \neq j$ мають однаковий розподіл з $cor(A_{ij}, A_{ji}) = \rho \in [-1,1]$.

4. Математичне сподівання елементів матриці дорівнює нулю.

5. Дисперсія елементів головної діагоналі скінченна.

6. Дисперсія елементів поза головною діагоналлю дорівнює одиниці.

Твердження 4.4. [**227**]. Нехай

1. $A = A^{(N)}, N \geq 1$ – матриці, що задовольняють властивість С1 означення 4 з $\rho \in (-1,1)$.

2. $B^{(N)}$ – послідовність детермінованих квадратних матриць розмірності $N \times N$, ранг яких дорівнює $o(N)$.

3. $\mu_N(\cdot)$ – розподіл власних значень матриці $\frac{1}{\sqrt{N}}\left(A^{(N)} + B^{(N)}\right)$.

Тоді має місце слабка збіжність

$$\mu_N(\cdot) \rightrightarrows \mu_\rho(\cdot)$$

при $N \to \infty$, де $\mu_\rho(\cdot)$ – міра, що визначається функцією розподілу

$$F_\rho(x, y) = \mu_U(z \in \boldsymbol{C}: Re(z) < x, Im(z) < y),$$

μ_U – міра, що визначає рівномірний розподіл на еліпсоїді комплексної множини

$$\varepsilon_\rho = \left\{x + iy \in \boldsymbol{C}: \frac{x^2}{(1+\rho)^2} + \frac{y^2}{(1-\rho)^2} \leq 1\right\}.$$

Приклади еліпсів ε_ρ при різних значеннях ρ зображено на рис. 4.1.1.

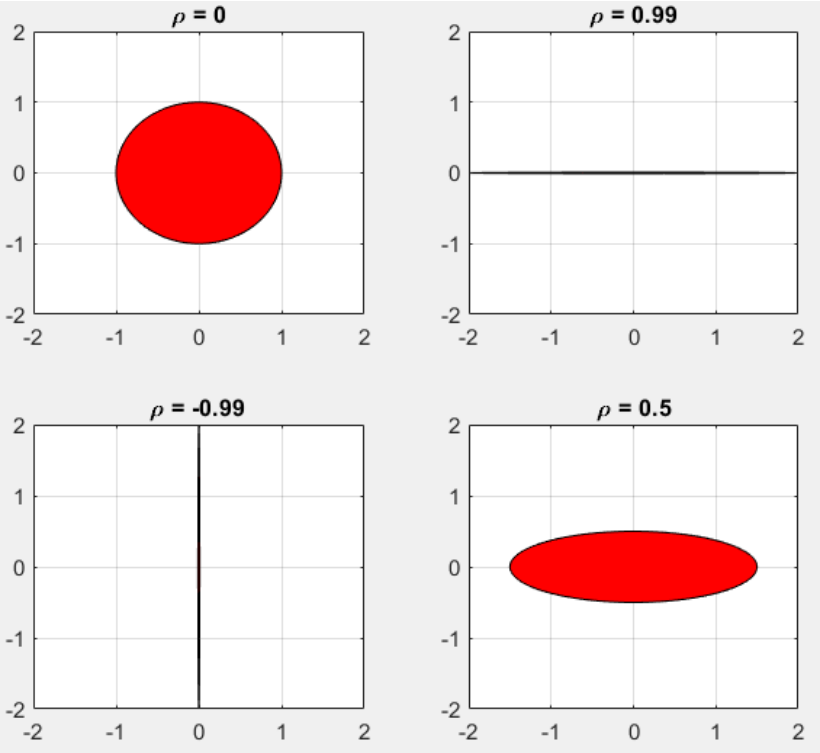


Рис. 4.1.1. Області концентрації власних значень граничного розподілу ε_ρ при різних значеннях ρ

Як ми можемо бачити з рис. 4.1.1, при $\rho \to 1$, матриці, що володіють властивістю C1 будуть збігатися до матриць Вігнера.

Поряд із класичними результатами збіжності розподілу власних значень, розглянемо також результат, який присвячено збіжності власних значень добутку випадкових матриць [228]. Цей загальний результат щодо асимптотичних спектральних властивостей випадкових матриць належить українським математикам В.О. Марченко та Л.А. Пастуру. Для подальшого розгляду скористаймося частинним випадком загальної теореми Марченко – Пастура, яка сформульована в роботі [169].

Твердження 4.5. [*169*]. Нехай $A = A_{D\times N}$ – комплекснозначна випадкова матриця з незалежними елементами, дисперсія кожного із яких дорівнює одиниці та визначена матриця

$$X_{D,N} = \frac{1}{N} AA^H,$$

де $A^H = \overline{A^T}$ – ермітово-спряжена матриця до матриці A.

Тоді при виконанні умов

$$\lim_{D\to\infty} D = \infty, \qquad \lim_{D\to\infty} \frac{D}{N} = c \in (0,1]$$

має місце слабка збіжність розподілу власних значень

$$\mu_{N,D}(\cdot) \rightrightarrows \mu_{MP}(\cdot)$$

при $N \to \infty$, де $\mu_{D,N}(\cdot)$ – розподіл власних значень матриці $X_{D,N}$, $\mu_{MP}(\cdot)$ – розподіл Марченко – Пастура, що визначається щільністю

$$f_{MP}(x) = \frac{1}{2\pi c x}\sqrt{(x-a)(b-x)}, x \in [a,b],$$

де

$$a = \left(1-\sqrt{c}\right)^2, \quad b = \left(1+\sqrt{c}\right)^2.$$

У випадку,

$$\lim_{D\to\infty} D = \infty, \qquad \lim_{D\to\infty} \frac{D}{N} = c > 1$$

має місце слабка збіжність розподілу власних значень

$$\mu_{N,D}(\cdot) \rightrightarrows \mu_{MP2}(\cdot)$$

при $N \to \infty$, де $\mu_{MP2}(\cdot)$ визначається із наступного співвідношення

$$\mu_{MP2}(B) = \mu_{MP}(B) + \left(1-\frac{1}{c}\right) I_{\{0\in B\}}$$

для $B \in \beta_{\mathbb{C}}$.

Поряд із випадковими матрицями, важливим класом матриць являються стохастичні матриці, які визначаються як матриці перехідних ймовірностей в теорії дискретних ланцюгів Маркова **[**229**].** Крім того, важливою властивістю стохастичних матриць є розміщення власних значень.

Означення 4.5. Матриця $P = P_{N\times N}$ називається стохастичною, якщо

- всі елементи матриці P є невід'ємними;
- сума елементів по рядках (або стовпцях) дорівнює одиниці.

Стохастичні випадкові матриці і будуть основним математичним апаратом при дослідженні кластерної структури веб-просторів. Вибір саме цього математичного апарату базується на особливостях спектральних

властивостей стохастичної матриці P. Надалі наведемо декілька тверджень, які будуть використані нами при доведені основних фактів дисертаційної роботи. Один із основних фактів буде ґрунтуватися на теоремі Перрона – Фробеніуса [230].

Твердження 4.6. **[***230***]**. Нехай матриця $A = A^{(N)}$ – нерозкладна матриця із невід'ємними елементами. Тоді існує додатне власне значення $\lambda_1(A)$ матриці A, для якого виконуються нерівності:

$$|\lambda_i(A)| \le \lambda_1(A), i = 2, \dots, N;$$

$$\min_{j=1,..,N} \sum_{i=1}^{N} A_{ij} \le \lambda_1(A) \le \max_{j=1,..,N} \sum_{i=1}^{N} A_{ij}\,; \tag{4.1.2}$$

$$\min_{i=1,..,N} \sum_{j=1}^{N} A_{ij} \le \lambda_1(A) \le \max_{i=1,..,N} \sum_{j=1}^{N} A_{ij}\,. \tag{4.1.3}$$

Власне значення $\lambda_1(A)$ є простим коренем характеристичного рівняння та даному власному значенню відповідає власний вектор $e_1(A)$ із невід'ємними координатами.

Враховуючи нерівності для стохастичної випадкової матриці P, визначеної в означенні 4.5, отримаємо наслідок із теореми Перрона – Фробеніуса про рівність одиниці власного значення $\lambda_1(P)$, тобто

$$\lambda_1(P) = 1 \tag{4.1.4}$$

для довільної стохастичної матриці.

Інший важливий результат буде стосуватися теорії ланцюгів Маркова та кількості незвідних класів у множині станів $S = \{1,\ 2, \dots,\ N\}$. Для цього визначимо дискретний ланцюг Маркова $\xi_n, n \ge 0$ [229**]** із значеннями в множині S та матрицею перехідних ймовірностей P за 1 крок.

Означення 4.6. Послідовність випадкових величин $\xi_n, n \ge 0$ називається дискретним ланцюгом Маркова, якщо для довільних станів $i, j, i_1, \dots, i_m \in S$ та довільного $m \in \mathbb{N}$ виконується співвідношення

$$\Pr\left(X_{n+1} = j | X_n = i, X_{n_1} = i_1, \dots, X_{n_m} = i_m\right) =$$
$$= \Pr(X_{n+1} = j | X_n = i),$$

де $n_1, \dots, n_m \in \{0, \dots, n-1\}$. Причому ймовірності переходу за один крок виражаються через елементи матриці P,

$$\Pr(\xi_{n+1} = j | \xi_n = i) = P_{ij}, n \geq 0.$$

Означення 4.7. Сукупність станів $S_p \subset S$ називається незвідним класом для дискретного ланцюга Маркова $\xi_n, n \geq 0$, якщо для довільних станів $i, j \in S_p$ існує $n > 0$, таке що

$$\Pr(\xi_n = i | \xi_0 = j) > 0.$$

Твердження 4.7. [229]. Кількість незвідних класів дорівнює кратності $\lambda(P) = 1$ матриці перехідних ймовірностей P, тобто

$$\#\{\lambda_i(P), i = 1, \dots, N: \lambda_i(P) = 1\} = \#\{S_p: S_p - \text{незвідний клас}\}.$$

4.2. Застосування теорії випадкових матриць

Використання теорії випадкових матриць в прикладних задачах зосереджено для систем великих розмірів із складною взаємодією, в яких враховується наявність випадковостей. В даний час теорія випадкових матриць набула широкого використання у різних прикладних задачах, в тому числі задачах машинного навчання, задачах кодування та декодування, задачах криптографії тощо.

Так у роботі [231] розглянуто використання теорії випадкових матриць в задачах кодування та декодування зображень на основі так званого методу стеганографії. Основною ідеєю цієї роботи є факт моделювання випадкових матриць кодування та декодування зображень на основі так званих блочних матриць. При цьому показано, що розроблений алгоритм кодування та декодування має вищу криптостійкість у порівнянні із відомими класичними алгоритмами.

Теорія випадкових матриць використовується і при аналізі безпеки об’єкту на основі перевірки змін показників датчиків по периметру даного об’єкту [232]. Основним технічним засобом при аналізі порушень периметру об’єкту в роботі розглянуто лазери з неперервним розподілом інтенсивності

та припущенням про розподіл Релея інтенсивності лазерів. Результатами цієї роботи є знаходження оптимальної інтенсивності лазерів для максимальної чутливості при виявленні порушень периметру об’єкту. Використання випадкових матриць із елементами з розподілом Релея дозволило крім знаходження оптимальної інтенсивності кожного лазера, вказати взаємодію між ними та оптимальні режими перемикання.

Фінансова математика є також однією із областей, де вплив випадковостей являється найбільш важливим фактором і оцінка фінансових ризиків однією із найбільш важливих задач. У роботі [233] для оцінки рівня фінансових ризиків використано теорію великих матриць, причому основним математичним об’єктом дослідження є так звані випадкові кореляційні матриці. Таким чином дослідження роботи присвячені аналіз випадкових матриць, які мають наступні властивості:

- Матриця R є симетричною.
- Матриця R є додатньовизначеною.
- Діагональні елементи матриці R дорівнюють одиниці.

На основі властивостей випадкових матриць вдалося визначити зв'язок між власними значеннями випадкових матриць та кореляційними коефіцієнтами випадкової кореляційної матриці, а також визначити рівень фінансового ризику в задачах оптимізації портфеля цінних паперів.

У роботі [234] розглянуто задачу відслідковування об’єктів на основі випадкових матриць, причому через випадкові матриці розглядаються відстань між об’єктами. Крім того, випадковість з’являється за рахунок зміни розміру цілей та наявності хибних цілей. Використовуючи теорію випадкових матриць, авторам роботи вдалося розробити алгоритм для виявлення викидів та згладжування шумів у системі.

Ще один криптографічний алгоритм на основі випадкових матриць, розроблено в роботі [235]. У даній роботі розглянуто новий метод побудови випадкової матриці, що має обернену. Слід зауважити, що задача перевірки існування оберненої матриці має поліноміальну складність поряду $O(n^2)$, де

n – розмірність матриці. Тому алгоритм генерації матриць, що мають обернену є вкрай важливий для симетричних алгоритмів шифрування на основі матричних перетворень.

4.3. Застосування до оцінки оптимальної кількості кластерів

Розглянемо основні позначення, які будуть використовуватися у цьому розділі роботи. Основними математичними об'єктами, які будуть використані нижче є граф та ланцюг Маркова [237].

Граф будемо позначати через $G = (V, E)$, де $V = \{v_1, v_2, \dots, v_N\}$ – множина вершин графа, $E = \{e_1, \dots, e_m\}$ – множина ребер графа, що з'єднують вершини із V. Припустимо, що граф задається матрицею суміжності A:

$$A = A_{N \times N},$$

де елемент A_{ij} рівний ваговому коефіцієнту між вершинами i та j.

Стохастична матриця

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^{N} A_{ik}}, \tag{4.3.1}$$

що відповідає графу G та задається матрицею суміжності A, задається співвідношенням (4.3.1). Основним предметом дослідження буде матриця P.

Надалі будемо впорядковувати власні значення матриці A, тобто

$$|\lambda_1(A)| \geq |\lambda_2(A)| \geq \cdots \geq |\lambda_N(A)|.$$

Скористаємось фактом, що одним із власних значень стохастичної матриці, що відповідає незвідному ланцюгу Маркова, є одиниця та всі власні значення стохастичної матриці не перевищують її за абсолютним значенням.

Згідно з твердженням 4.6 вірні наступні співвідношення

$$\begin{cases} \lambda_1(P) = 1, \\ |\lambda_i(A)| \leq 1, i \geq 1. \end{cases}$$

Теорема 4.3.1. Нехай виконуються наступні умови:

1. Всі елементи матриці $A = A_{N\times N}$ є незалежними та мають однаковий розподіл

$$A_{ij} \sim Distr,$$

де розподіл $Distr$, для якого виконуються умови

$$P(A_{ij} < 1) = 0, \;\; EA_{ij} < \infty.$$

2. Щільність розподілу $Distr$ задовольняє умову

$$f_A(x) \le \frac{M}{x^{\alpha}}, \qquad \alpha \in (2,3],$$

де $M > 0$.

3. Елементи матриці переходу дискретного ланцюга Маркова за один крок P_{ij} (4.3.1) рівні нормованим по рядкам елементам матриці A.

Тоді буде мати місце наступна збіжність

$$\lim_{N\to\infty} \max(|\lambda_2(P)|, .., |\lambda_N(P)|) = 0.$$

Доведення. Слід зауважити, що елементи матриці

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^{N} A_{ik}}$$

набувають значень із інтервалу $[0,1]$. Крім того, на основі однакового розподілу випадкових величин A_{ij} та їх незалежності, отримаємо наступні співвідношення

$$E(P_{ij}) = E\left(\frac{A_{ij}}{\sum_{k=1}^{N} A_{ik}}\right) = \frac{1}{N},$$

$$E(P_{ij}^2) \le O\left(\frac{1}{N}\right).$$

Для доведення справедливості останнього співвідношення, припустимо що $f_A(x), x \in R$ – щільність розподілу елемента матриці. Тоді щільність розподілу елемента матриці P_{ij} буде наступною

$$f_{P_{ij}}(x) =$$

$$\int\limits_{R_+^{N-1}} f_A\left(\frac{x(x_2+\cdots+x_N)}{1-x}\right) f_A(x_2)\dots f_A(x_N)\frac{(x_2+\cdots+x_N)}{(1-x)^2}dx_2\dots dx_N =$$

$$= (N-1)\int\limits_{R_+^{N-1}} f_A\left(\frac{x(x_2+\cdots+x_N)}{1-x}\right) f_A(x_2)\dots f_A(x_N)\frac{x_2}{(1-x)^2}dx_2\dots dx_N \leq$$

$$\leq M(N-1)\int\limits_{R_+^{N-1}} \left(\frac{x(x_2+\cdots+x_N)}{1-x}\right)^{-\alpha} f_A(x_2)\dots f_A(x_N)\frac{x_2}{(1-x)^2}dx_2\dots dx_N \leq$$

$$\leq M * EA_{ij} * (N-1)^{1-\alpha}\frac{(1-x)^{\alpha-2}}{x^\alpha} = \frac{K}{(N-1)^{\alpha-1}}\frac{(1-x)^{\alpha-2}}{x^\alpha} \leq$$

$$\leq \frac{K}{N-1}\frac{(1-x)^{\alpha-2}}{x^\alpha} = O\left(\frac{1}{N}\right)\frac{(1-x)^{\alpha-2}}{x^\alpha},$$

де $K = M * EA_{ij}$.

Аналогічно із теоремою [236] розглянемо матрицю що, складається із усереднених значень матриці $P = P_{N\times N}$, тобто квадратну матрицю $\tilde{P}$, для якої виконується співвідношення

$$\tilde{P}_{ij} = \frac{1}{N}.$$

Власними значеннями даної матриці будуть

$$\lambda_1(\tilde{P}) = 1, \lambda_2(\tilde{P}) = \cdots = \lambda_N(\tilde{P}) = 0.$$

Для доведення теореми, скористаємося наступною нерівністю

$$\sum_{i=1}^{N}\left(\lambda_i(A) - \lambda_i(B)\right)^2 \leq Tr(A-B)^2,$$

Враховуючи той факт, що $\alpha \in (2,3]$ та взявши в якості матриць A, B відповідно матриці $P, \tilde{P}$, отримаємо

$$\sum_{i=2}^{N} E\left(\lambda_i(P) - \lambda_i(\tilde{P})\right)^2 = \sum_{i=2}^{N} E\lambda_i^2(P) \leq$$

$$\leq TrE\left(P-\tilde{P}\right)^2.$$

Враховуючи незалежність елементів матриці A, отримаємо

$$\sum_{i=2}^{N} E\left(\lambda_i(P) - \lambda_i\left(\tilde{P}\right)\right)^2 \le \sum_{i=1}^{N} D(P_{ii}) = O\left(\frac{N}{N^{\alpha-1}}\right) = O\left(\frac{1}{N^{\alpha-2}}\right),$$

що і доводить твердження теореми.

Зауваження 4.3.1. У роботі [236] розглядається вужчий клас розподілів $Distr$, оскільки вимагається наявність моменту порядку $2+\delta, \delta > 0$. Ця умова необхідна у доведенні для використання центральної граничної теореми у формі Лінденберга чи Ляпунова. В нашому випадку умова послаблюється і вимагається існування першого моменту та обмеженість щільності розподілу степеневою функцією. Таким чином, у розглянутій теоремі вдалося розширити клас випадкових матриць A, для яких може бути застосована збіжність власних значень.

Зауваження 4.3.2. Для спрощення викладок теореми 4.3.1 було зроблено припущення про носій розподілу $Distr$, а саме накладено умову $P\left(A_{ij} < 1\right) = 0$. Цю умову можна послабити до $P\left(A_{ij} < 0\right) = 0$, припустивши що $P\left(A_{ij} > 1\right) > 0$. В даному випадку

$$f_{P_{ij}}(x) =$$

$$= (N-1) \int\limits_{R_+^{N-1}} f_A\left(\frac{x(x_2 + \cdots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2}{(1-x)^2} dx_2 \dots dx_N =$$

$$= (N-1) \int\limits_{B^{(N)}} f_A\left(\frac{x(x_2 + \cdots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2}{(1-x)^2} dx_2 \dots dx_N +$$

$$+(N-1) \int\limits_{\overline{B^{(N)}}} f_A\left(\frac{x(x_2 + \cdots + x_N)}{1-x}\right) f_A(x_2) \dots f_A(x_N) \frac{x_2}{(1-x)^2} dx_2 \dots dx_N,$$

де множина $B^{(N)}$ визначається співвідношенням

$$B^{(N)} = \{x_2 + x_3 + \cdots + x_N \le N-1\}.$$

Тоді

$$f_{P_{ij}}(x) \le$$

$$\leq (N-1)\int\limits_{B^{(N)}} f_A\left(\frac{x(x_2+\cdots+x_N)}{1-x}\right) f_A(x_2)\ldots f_A(x_N)\frac{x_2}{(1-x)^2}\,dx_2\ldots dx_N +$$

$$+M(N-1)\int\limits_{R_+^{N-1}} \left(\frac{x(x_2+\cdots+x_N)}{1-x}\right)^{-\alpha} f_A(x_2)\ldots f_A(x_N)\frac{x_2}{(1-x)^2}\,dx_2\ldots dx_N.$$

Другий випадок аналогічно із доведеною теоремою рівний $O\left(\frac{1}{N^{\alpha-1}}\right)$. Згідно наших припущень щодо щільності $f_A(x)$, отримаємо що

$$(N-1)\int\limits_{B^{(N)}} f_A\left(\frac{x(x_2+\cdots+x_N)}{1-x}\right) f_A(x_2)\ldots f_A(x_N)\frac{x_2}{(1-x)^2}\,dx_2\ldots dx_N \leq$$

$$\leq K(N-1)\int\limits_{B^{(N)}} f_A(x_2)\ldots f_A(x_N)\frac{x_2}{(1-x)^2}\,dx_2\ldots dx_N \leq$$

$$\leq K_1(N-1)a^{N-2} \leq \frac{K_1}{N^2}$$

для великих N, де

$$K_1 = (N-1)EA_{ij}, \qquad a = \int\limits_0^1 f_A(x)dx < 1.$$

Методи знаходження оптимальної кількості кластерів k-core decomposition [95] та «метод ліктя» [94] були розглянуті у розділі 3. Цими методами визначались оптимальна кількість кластерів та їхні центри кластерів для різних сегментів веб-простору. Розглянемо тепер задачу визначення оптимальної кількості кластерів на основі стохастичної матриці P, що задає переходи в деякій реальній системі [240; 241]. Також будемо припускати, що стохастична матриця P має блочний вигляд, тобто має місце співвідношення

$$P = \begin{pmatrix} P^{(1)} & Err^{(1,2)} & \cdots & Err^{(1,k_{opt})} \\ Err^{(2,1)} & P^{(2)} & \cdots & Err^{(2,k_{opt})} \\ \vdots & \vdots & \ddots & \vdots \\ Err^{(k_{opt},1)} & Err^{(k_{opt},2)} & \cdots & P^{(k_{opt})} \end{pmatrix}, \qquad (4.3.2)$$

де $P^{(i)}$ задають ймовірності переходу в i–му кластері та $Err^{(i,j)}$ – ймовірності переходу з i–го кластеру в j–ий кластер. Матриці $Err^{(i,j)}$ будемо характеризувати як “помилки” переходів між сусідніми кластерами. Нехай зв’язки всередині кластера є набагато сильнішими ніж зв’язки між кластерами. Дане припущення опишемо наступним чином

$$\min_{i\in 1,\dots,k_{opt}} \min_{m\in 1,\dots,n_i} \sum_{j=1}^{n_i} P_{mj}^{(i)} \geq$$

$$\geq \alpha \max_{i\in 1,\dots,k_{opt}} \max_{m\in 1,\dots,n_i} \left(1 - \sum_{j=1}^{n_i} P_{mj}^{(i)}\right),$$

де n_i – розмірність матриці $P^{(i)}$, $\alpha \in (1,\infty)$ – параметр подібності. Зауважимо, що при $\alpha \to \infty$ всі помилки $Err_{k,m}^{(i,j)} \to 0$. В цьому випадку матриця P' перетворюється в блочно-діагональну матрицю вигляду

$$P' = \begin{pmatrix} P^{(1)} & 0 & \cdots & 0 \\ 0 & P^{(2)} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & P^{(k_{opt})} \end{pmatrix}. \qquad (4.3.3)$$

Тобто, матриця P' у цьому випадку задає чітко визначені k_{opt} кластери даних. Крім того, для матриці P', заданої співвідношенням (4.3.3),

$$\lambda_1(P') = \cdots = \lambda_{k_{opt}}(P') = 1$$

та

$$|\lambda_i(P') - 1| > \varepsilon, \;\; i > k_{opt},$$

де параметр ε залежить від розмірності системи (кількості об’єктів кластеризації) та параметру α. Для визначення оптимальної кількості кластерів ми будемо користуватися наступним співвідношенням

$$k_{opt} = \#\left\{\lambda_i : |\lambda_i(P) - 1| \leq \frac{1}{\alpha\sqrt{N}}\right\}. \qquad (4.3.4)$$

Дане співвідношення є наслідком теореми 4.3.1 та асимптотичної поведінки власних значень $\lambda_i(P)$, $i \leq k_{opt}$.

Відзначимо, що отримані результати узгоджуються із теоремою Марченко–Пастура [228] про асимптотичний розподіл власних значень випадкових матриць із незалежними елементами [238; 239]. Також, слід зауважити, що результати теореми можуть трактуватися як закон великих чисел. Крім того, визначення оптимальної кількості кластерів будується на асимптотиці власних значень матриці P при $N \to \infty$. Одержані результати можуть бути застосовані для визначення оптимальної кількості кластерів Grid system, складних мереж, при дослідженнях структури веб-простору тощо [240; 241].

4.4. Аналіз кластерної структури інтернет-мереж на основі випадкових матриць

Розглянемо задачу розбиття на підгрупи, тобто задачу кластеризації. Вище припускалось, що всі зв'язки в одному кластері є однотипними. Дане припущення можна формалізувати наступним чином: елементи матриці суміжності A мають однаковий розподіл, тобто

$$A_{ij} \sim Distr,$$

де $Distr$ – розподіл з носієм $S \subset (0,\infty)$ та скінченним другим моментом та ненульовим середнім:

$$\mu_1 = EA_{ij}; \quad \mu_2 = EA_{ij}^2 < \infty.$$

Тоді матрицю перехідних ймовірностей будемо визначати із наступного співвідношення:

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^{N} A_{ik}}. \qquad (4.4.1)$$

Матриця P_{ij} буде стохастичною; крім того, усі елементи будуть мати той же розподіл, носієм якого є множина Для доведення основного твердження даного розділу будемо використовувати твердження 4.1.

Слід зауважити, що матриця вигляду (4.4.1) є “незручною” для дослідження, оскільки елементи в рядках не є незалежними за рахунок $\sum_{j=1}^{N} A_{ij}$ у знаменнику. Даний факт означає, що елементи матриці P не є незалежними, що порушує основні умови твердження 4.1 для визначення асимптотики розподілу власних значень матриці P. Ще однією проблемою є той факт, що елементи матриці P мають носієм множину [0,1] та

$$E(P_{ij}) = \frac{1}{N} \neq 0.$$

Таким чином, порушується і друга умова твердження 4.1.

Для подолання цих проблем використаємо закон великих чисел. Для цього поряд із матрицею P розглянемо матрицю $\tilde{P}$

$$\tilde{P}_{ij} = \frac{A_{ij}}{N\mu_1}.$$

Зауважимо, що елементи матриці $\tilde{P}_{ij}$ є незалежними випадковими величинами; крім того, асимптотика власних значень матриць P та $\tilde{P}$ є однаковою. Для цього розглянемо наступну лему [242].

Лема 4.4.1. Спектр матриць P та $\tilde{P}$ є асимптотично еквівалентним, тобто

$$\lim_{N\to\infty} \max_{i=1,\ldots,N} \left|\lambda_i\left(P - \tilde{P}\right)\right| = 0.$$

Доведення. Згідно з існуванням скінченного другого моменту для A_{ij}, отримаємо, що

$$\frac{1}{N}\sum_{j=1}^{N} A_{ij} \to \mu_1$$

у середньому квадратичному. Використовуючи даний факт, ми отримаємо, що

$$P_{ij} - \tilde{P}_{ij} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} - \frac{A_{ij}}{N\mu_1} =$$

$$= \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} \frac{N\mu_1 - \sum_{j=1}^{N} A_{ij}}{N} = \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} \left(\mu_1 - \frac{1}{N} \sum_{j=1}^{N} A_{ij} \right).$$

Таким чином, елементи різниці двох матриць будуть прямувати до 0. Використовуючи той факт, що $\frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}}$ мають однаковий розподіл для довільних i, отримаємо наступне співвідношення:

$$\sum_{i=1}^{N} \left| \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} \left(\mu_1 - \frac{1}{N} \sum_{j=1}^{N} A_{ij} \right) \right| = \sum_{i=1}^{N} \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} \left| \frac{1}{N} \sum_{j=1}^{N} A_{ij} - \mu_1 \right| =$$

$$= o(1) \sum_{i=1}^{N} \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} = o(1) \to 0$$

у середньому квадратичному.

Лема 4.4.1 доведена.

Зауваження 4.4.1. Можна посилити твердження леми 4.4.1 та довести, що

$$\lim_{N \to \infty} N^{\alpha} \max_{i=1,\ldots,N} \left| \lambda_i \left(P - \tilde{P} \right) \right| = 0$$

для $\alpha \in [0,1)$.

Отже, ґрунтуючись на лемі 4.4.1, надалі можемо зосередити увагу на дослідженні власних значень матриці $\tilde{P}$, які задовольняють умову незалежності [242].

Теорема 4.4.1. Розподіл власних значень матриці $\frac{\sqrt{N}}{\sigma}\tilde{P}$ має асимптотичний розподіл, який зосереджений в одиничному крузі $\left\{z \in C : |z| \le 1\right\}$ та володіє одним викидом $\frac{\sqrt{N}}{\sigma} + o(1)$.

***Доведення*.** Поряд із матрицею $\tilde{P}$ розглянемо центровану матрицю

$$\bar{P} = \left(\frac{\mu}{N}\right)_{i,j=1,\dots,N}.$$

Використаємо наступне співвідношення для $\tilde{P}$:

$$\tilde{P} = \tilde{P} - \bar{P} + \bar{P} = P^C + \bar{P}.$$

Основним фактом даного представлення є те, що матриця P^C має вигляд

$$P^C = \begin{pmatrix} \frac{A_{11} - \mu}{N} & \dots & \frac{A_{1N} - \mu}{N} \\ \vdots & \ddots & \vdots \\ \frac{A_{N1} - \mu}{N} & \dots & \frac{A_{NN} - \mu}{N} \end{pmatrix}.$$

Отже, елементи матриці P^C задовольняють умовам твердження 4.1 і можна стверджувати, що власні значення матриці

$$\frac{\sqrt{N}}{\sigma}P^C = \begin{pmatrix} \frac{A_{11} - \mu}{\sigma\sqrt{N}} & \dots & \frac{A_{1N} - \mu}{\sigma\sqrt{N}} \\ \vdots & \ddots & \vdots \\ \frac{A_{N1} - \mu}{\sigma\sqrt{N}} & \dots & \frac{A_{NN} - \mu}{\sigma\sqrt{N}} \end{pmatrix}$$

будуть мати асимптотичний рівномірний розподіл у крузі $\left\{z \in C : |z| \le 1\right\}$.

Для остаточного доведення теореми залишилося знайти розподіл власних значень матриці $\bar{P}$ та її зв'язок із матрицею P^C. Скористаємося поняттям нормальної матриці. Враховуючи той факт, що елементи матриці $\frac{\sqrt{N}}{\sigma}P^C$ є незалежними випадковими величинами, отримаємо, що

$$\left(\frac{\sqrt{N}}{\sigma}P^C * \left[\frac{\sqrt{N}}{\sigma}P^C\right]'\right)_{ij} = \frac{N}{\sigma^2}\sum_{k=1}^{N}\left(A_{ik}-\mu\right)\left(A_{jk}-\mu\right) =$$

$$= \left(\frac{\sqrt{N}}{\sigma}P^C * \left[\frac{\sqrt{N}}{\sigma}P^C\right]'\right)_{ji}.$$

З іншого боку, матриця $\frac{\sqrt{N}}{\sigma}\bar{P}$ також є нормальною, оскільки

$$\left(\frac{\sqrt{N}}{\sigma}\bar{P} * \left[\frac{\sqrt{N}}{\sigma}\bar{P}\right]'\right)_{ij} = \frac{N}{\sigma^2}\sum_{k=1}^{N}\frac{\mu^2}{N^2} = \frac{\mu^2}{N\sigma^2} =$$

$$= \left(\frac{\sqrt{N}}{\sigma}P^C * \left[\frac{\sqrt{N}}{\sigma}P^C\right]'\right)_{ji}.$$

Крім того, матриці $\frac{\sqrt{N}}{\sigma}P^C$ та $\frac{\sqrt{N}}{\sigma}\bar{P}$ є переставними, тобто

$$\frac{\sqrt{N}}{\sigma}P^C\frac{\sqrt{N}}{\sigma}\bar{P} = \frac{N}{\sigma^2}P^C\bar{P}$$

рівна за розподілом матриці

$$\frac{\sqrt{N}}{\sigma}\bar{P}\frac{\sqrt{N}}{\sigma}P^C = \frac{N}{\sigma^2}\bar{P}P^C.$$

Таким чином, можна вказати унітарну випадкову матрицю U, для якої

$$\frac{\sqrt{N}}{\sigma}P^C = UD_{P^C}U^H, \quad \frac{\sqrt{N}}{\sigma}\bar{P} = UD_{\bar{P}}U^H,$$

де D_{P^C} та $D_{\bar{P}}$ – діагональні матриці. У цьому випадку власні значення матриці

$$\frac{\sqrt{N}}{\sigma}\left(P^C + \bar{P}\right)$$

визначаються як сума власних значень $\frac{\sqrt{N}}{\sigma}P^C$ та $\frac{\sqrt{N}}{\sigma}\bar{P}$. Власні значення матриці $\frac{\sqrt{N}}{\sigma}\bar{P}$ рівні

$$\lambda_1\left(\frac{\sqrt{N}}{\sigma}\bar{P}\right)=\frac{\sqrt{N}}{\sigma},\ \lambda_2\left(\frac{\sqrt{N}}{\sigma}\bar{P}\right)=\ldots=\lambda_N\left(\frac{\sqrt{N}}{\sigma}\bar{P}\right)=0. \qquad (4.4.1)$$

Використовуючи властивості нормальних переставних матриць, приходимо до висновку, що власні значення матриці $\frac{\sqrt{N}}{\sigma}\left(P^C+\bar{P}\right)$ будуть сумою власних значень матриці $\frac{\sqrt{N}}{\sigma}P^C$ та $\frac{\sqrt{N}}{\sigma}\bar{P}$. Крім того, нас не цікавить порядок додавання на основі (4.4.1). Таким чином, власні значення матриці $\frac{\sqrt{N}}{\sigma}\left(P^C+\bar{P}\right)$ будуть мати асимптотичний рівномірний розподіл в одиничному крузі

$$\left\{z\in C:|z|\le 1\right\},$$

крім одного значення, яке дорівнює $\frac{\sqrt{N}}{\sigma}+o(1)$. Використовуючи даний факт, отримаємо, що власні значення $\tilde{P}$ асимптотично розміщені в крузі

$$\left\{z\in C:|z|\le\frac{\sigma}{\sqrt{N}}\right\}$$

та одне з власних значень даної матриці дорівнює $1+o\left(\frac{1}{\sqrt{N}}\right)$.

Теорема 4.4.1 доведена.

4.5. Про один клас стохастичних випадкових матриць

Цей пункт присвячений методам кластерного аналізу на графах, як одного із основних підрозділів штучного інтелекту, зокрема, методам перевірки гіпотези про входження вершин графу до одного кластеру при проведенні дослідження ділянок вебпростору. При цьому основним

математичним апаратом є випадкові матриці великої розмірності з додатковими обмеженнями на їхні елементи. Асимптотичні властивості власних значень випадкових матриць, які використовують для опису функціонування ділянок вебпростору, відрізняються від класичних властивостей тим, що елементи матриці невід'ємні та залежні по рядках або стовпчиках. У цьому випадку порушується основне припущення теорії випадкових матриць про незалежність елементів матриці [243].

Класичні та удосконалені методи кластеризації на графах описані в [197; 244-247]. Наприклад, у роботі Chen Y., Sanghavi S. та Xu H. [244] наведено новий алгоритм кластеризації незважених графів, який є удосконаленням стохастичної блочної моделі, а саме – опуклу версію методу максимальної правдоподібності. Застосування цього методу дає кращі результати, порівняно з методами, що використовують поліноміальні коефіцієнти у випадку масштабування розміру кластера. Цей алгоритм застосовний до відновлення багатьох класичних генеративних моделей для проведення кластеризації графів.

У статті [245] S. E. Schaeffer розглядаються основні методи кластеризації графів, основні означення та міри якості кластерів. Представлено глобальні алгоритми кластеризації для всього набору вершин графу і розглянуто задачу визначення кластера для конкретної вершини. А також зазначено області застосування алгоритмів кластеризації графів. Аналогічна задача визначення кластера для конкретної вершини буде розглядатися в цьому розділі, вирішення якої буде базуватись на статистичних тестах.

У [246] застосовано алгоритм графових нейронних мереж для кластеризації графів і отримано кращі результати для проведення класифікації вузлів порівняно з методами k-means (DGI), SBM, MinCut та ін.

Ще один підхід до проведення процесу кластеризації графів полягає у використанні поняття внутрішньокластерної щільності [197] на противагу міжкластерній розрідженості. В [197] було проведено експериментальне

оцінювання різних підходів до кластеризації графів, а саме розглянуто алгоритми марковської кластеризації, Iterative Conductance Cutting, геометрична MST-кластеризація, які базуються на властивостях матриць великої розмірності. В [247] запропоновано більш повно і системно аналізувати дані в прикладних задачах та проводити кластеризацію за різними показниками подібності для тих самих даних, а також досліджувати різні типи зв'язків між ними.

У роботах [241; 242] основним математичним апаратом дослідження оптимальної кількості кластерів при дослідженні ділянок веб-простору є випадкові матриці із додатковими обмеженнями на елементи. В якості обмежень розглядаються квадратні стохастичні матриці, тобто матриці, в яких сума елементів по рядках (або по стовпцях) дорівнює 1 та всі елементи є невід'ємними. Особлива увага при дослідженні таких матриць приділяється гауссівським ансамблям (Gaussian Ensemble), які є унітарними, ортогональними або симплектичними ансамблями. Враховуючи результати, отримані у роботах з даного напрямку, можна стверджувати, що сумісний розподіл власних значень, при розгляді нормування $\frac{1}{\sqrt{N}}$ (де N – розмірність відповідної матриці) для вказаних вище ансамблів, добре вивчений. Основна перевага даних ансамблів ґрунтується на припущенні про нормальний розподіл елементів матриці та їх незалежність.

Проте, у випадку опису функціонування ділянок веб-простору, елементи матриці суміжності графа повинні бути невід'ємними з ймовірністю 1, проте умова незалежності для елементів матриці відсутня. Тому асимптотичні властивості власних значень такої матриці будуть відрізнятися від класичних властивостей.

Як і в роботах [241; 242], припустимо, що функціонування веб-простору описується графом із матрицею суміжності

$$A = A_{ij};\ i, j = 1,..., N,$$

де N – кількість розглянутих об’єктів (веб-сторінок), а A_{ij} – відображає кількість переходів із веб-сторінки i на веб-сторінку j, причому як і в роботах припускаємо що A_{ij} є випадковими величинами. Для спрощення всіх наступних викладок припустимо, що матриця суміжності є симетричною, тобто $A_{ij} = A_{ji}$ та всі елементи, які знаходяться вище головної діагоналі, є незалежними в сукупності випадковими величинами.

Розглянемо випадок, при якому елементи матриці суміжності підпорядковані експоненціальному закону розподілу,

$$A_{ij} \sim i.i.d.Exp(\lambda);\ j \geq i,\ i = 1,...,N, \tag{4.5.1}$$

тобто елементи матриці суміжності вище головної діагоналі є незалежними.

Основним об’єктом дослідження будуть стохастичні матриці із випадковими величинами, що визначаються наступними співвідношеннями

$$P_{ij} = \frac{A_{ij}}{\sum_{k=1}^{N} A_{ik}}. \tag{4.5.2}$$

Розглянемо спочатку розподіл елементів випадкової матриці P_{ij} із відповідними моментами першого та другого порядків.

Лема 4.5.1. Нехай виконується умова (4.5.1). Тоді розподіл кожного елемента матриці P має вигляд

$$f_{P_{ij}}(y) = (N-1)(1-y)^{N-2},\ y \in (0,1).$$

Доведення. Розглянемо наступне перетворення

$$y_1 = \frac{x_1}{x_1 + ... + x_N},\ \ y_2 = x_2,\ \ ...\ ,\ \ y_N = x_N.$$

Відповідне обернене перетворення має вигляд

$$x_1 = y_1 \frac{y_2 + ... + y_N}{1 - y_1},\ \ x_2 = y_2,\ \ ...\ ,\ \ x_N = y_N.$$

Якобіан оберненого перетворення дорівнює

$$\frac{\partial x}{\partial y}=\begin{pmatrix} \frac{y_2+...+y_N}{(1-y_1)^2} & \frac{y_1}{1-y_1} & ... & \frac{y_1}{1-y_1} \\ 0 & 1 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & 1 \end{pmatrix}$$

з визначником

$$\left|\det\left(\frac{\partial x}{\partial y}\right)\right|=\frac{y_2+...+y_N}{(1-y_1)^2}$$

за умови $y_1\in(0,1)$ та $y_2>0,...,y_N>0.$ Враховуючи незалежність випадкових величин $A_{i1},...,A_{iN}$, отримаємо

$$f_{P_{ij}}(y)=\lambda^N\int\limits_{R_+^{N-1}} e^{-\lambda\left(y_1\frac{y_2+...+y_N}{1-y_1}+y_2+...+y_N\right)}\frac{y_2+...+y_N}{(1-y_1)^2}dy_2...dy_N=$$

$$=\frac{\lambda^N(N-1)}{(1-y_1)^2}\int\limits_{R_+^{N-1}} e^{-\lambda\frac{1}{1-y_1}(y_2+...+y_N)}y_N dy_2...dy_N=$$

$$=(N-1)(1-y_1)^{N-2}\int\limits_0^\infty e^{-y_N}y_N dy_N=$$

$$=(N-1)(1-y_1)^{N-2},\ y_1\in(0,1).$$

Лема 4.5.1 доведена.

Враховуючи розподіл елемента матриці P із леми 4.5.1, отримаємо, що середнє значення та значення другого моменту кожного елемента матриці P дорівнює

$$EP_{ij}=\int\limits_0^1(N-1)(1-y)^{N-2}ydy=\frac{1}{N};$$

$$EP_{ij}^2=\int\limits_0^1(N-1)(1-y)^{N-2}y^2dy=\frac{2}{N(N+1)}\approx\frac{2}{N^2}.$$

Дані результати узгоджуються із відповідними припущеннями, зробленими в роботах [236; 241]. Також умови леми 4.5.1 відповідають

припущенням щодо однорідності переходів усередині кластера, а саме той факт, що розподіл переходу на довільну веб-сторінку із того ж кластеру є сталим. Дана умова порушується у випадку розгляду системи із більше ніж одним кластером. У цьому випадку, перехід між кластерами є зазвичай нижчий ніж у самих кластерах, тобто

$$\lambda_{between} \ll \lambda_{within}$$

де λ_{within} – інтенсивність переходу в кластері, $\lambda_{between}$ – інтенсивність переходу між кластерами. Отже, враховуючи найбільш загальний випадок про довільну кількість кластерів, припустимо, що

$$A_{ij} \sim i.i.d.Exp(\lambda_j),\ j \geq i,\ i = 1,...,N, \tag{4.5.3}$$

Лема 4.5.2. Нехай виконується умова (4.5.3). Тоді розподіл елемента матриці P_{ij} має вигляд

$$f_{P_{ij}}(y) = \Lambda(1-y_1)^{N-2} \sum_{\substack{k=1,\\ k\neq j}}^{N} \left((\lambda_j y + \lambda_k - \lambda_k y)^{-2} \prod_{\substack{u=1,\\ u\neq k,j}}^{N} (\lambda_j y + \lambda_u - \lambda_u y)^{-1} \right) \tag{4.5.4}$$

для $y \in (0,1)$.

Доведення. Аналогічно із твердженням леми 4.5.1, розглянемо перетворення

$$y_1 = \frac{x_1}{x_1 + x_2 + ... + x_N}, y_2 = x_2, ..., y_N = x_N.$$

Обернене перетворення та Якобіан будуть визначатися як у лемі 4.5.1. Враховуючи незалежність випадкових величин $A_{i1}, ..., A_{iN}$, отримаємо

$$f_{P_{i1}}(y) = \Lambda \int_{R_+^{N-1}} e^{-\left(\lambda_1 y_1 \frac{y_2+...+y_N}{1-y_1} + \lambda_2 y_2 + ... + \lambda_N y_N\right)} \frac{y_2 + ... + y_N}{(1-y_1)^2} dy_2 ... dy_N =$$

$$= \frac{\Lambda}{(1-y_1)^2} \int_{R_+^{N-1}} e^{-\sum_{j=2}^{N} y_j \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_j\right)} (y_2 + ... + y_N) dy_2 ... dy_N,$$

де

$$\Lambda = \lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_N .$$

Обчислимо окремо кожен доданок

$$\frac{\Lambda}{(1-y_1)^2} \int\limits_{R_+^{N-1}} e^{-\sum\limits_{j=2}^{N} y_j \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_j \right)} y_k dy_2 \ldots dy_N =$$

$$= \frac{\Lambda}{(1-y_1)^2} \prod_{\substack{j=2, \\ j \neq k}}^{N} \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_j \right)^{-1} \int\limits_0^{\infty} e^{-y_k \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_k \right)} y_k dy_2 \ldots dy_N =$$

$$= \frac{\Lambda}{(1-y_1)^2} \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_k \right)^{-2} \prod_{\substack{u=2, \\ u \neq k}}^{N} \left(\frac{\lambda_1 y_1}{1-y_1} + \lambda_u \right)^{-1} =$$

$$= \Lambda (1-y_1)^{N-2} \left(\lambda_1 y_1 + \lambda_k - \lambda_k y_1 \right)^{-2} \prod_{\substack{u=2, \\ u \neq k}}^{N} \left(\lambda_1 y_1 + \lambda_u - \lambda_u y_1 \right)^{-1} .$$

Таким чином,

$$f_{P_{i1}}(y) = \Lambda (1-y_1)^{N-2} \sum_{k=2}^{N} \left((\lambda_1 y_1 + \lambda_k - \lambda_k y_1)^{-2} \prod_{\substack{u=2, \\ u \neq k}}^{N} (\lambda_1 y_1 + \lambda_u - \lambda_u y_1)^{-1} \right).$$

Аналогічно доводиться вигляд щільності $f_{P_{ij}}(y)$ для довільного j з врахуванням перетворення

$$y_j = \frac{x_j}{x_1 + x_2 + \ldots + x_N},\ y_1 = x_1, \ldots,\ y_{j-1} = x_{j-1},\ y_{j+1} = x_{j+1}, \ldots,\ y_N = x_N .$$

Дане зауваження і доводить (4.5.4) леми 4.5.2.

Як бачимо із леми 4.5.2, щільність розподілу значно ускладниться в порівнянні з аналогічним однорідним випадком, розглянутим у лемі 4.5.1. Крім того, відношення між щільностями P_{ij} та P_{im} буде залежати від відношення між інтенсивностями та не буде залежати від спільного множника у щільностях.

Справді

$$\frac{f_{P_{ij}}(y)}{f_{P_{im}}(y)} = \frac{\sum\limits_{k\neq j}(\lambda_j y+\lambda_k-\lambda_k y)^{-2}\prod\limits_{\substack{u=1,\\ u\neq k,j}}^{N}(\lambda_j y+\lambda_u-\lambda_u y)^{-1}}{\sum\limits_{k\neq j}(\lambda_m y+\lambda_k-\lambda_k y)^{-2}\prod\limits_{\substack{u=1,\\ u\neq k,m}}^{N}(\lambda_m y+\lambda_u-\lambda_u y)^{-1}} =$$

$$= \frac{\sum\limits_{k\neq j}(t\lambda_j y+t\lambda_k-t\lambda_k y)^{-2}\prod\limits_{\substack{u=1,\\ u\neq k,j}}^{N}(t\lambda_j y+t\lambda_u-t\lambda_u y)^{-1}}{\sum\limits_{k\neq j}(t\lambda_m y+t\lambda_k-t\lambda_k y)^{-2}\prod\limits_{\substack{u=1,\\ u\neq k,m}}^{N}(t\lambda_m y+t\lambda_u-t\lambda_u y)^{-1}}.$$

Таким чином, параметри $(\lambda_1,\lambda_2,...,\lambda_N)$ будуть інваріантними відносно множення на той же множник. Отже, ми можемо завжди припускати що

$$\hat{\lambda}_{i1}=1.$$

Крім того, побудована в лемі 4.5.2 щільність дозволяє як оцінювати параметри моделі для кожного конкретного вузла веб-простору, зокрема, так і перевіряти умову однорідності переходів, яка була використана як припущення у лемі 4.5.1. Для оцінки параметрів моделі можна користуватися методом максимальної правдоподібності для розподілу (4.5.3), при цьому функція правдоподібності для деякого стану буде мати вигляд

$$L(\lambda)=L(\lambda_1,\lambda_2,...,\lambda_N)=$$

$$=\prod_{j=1}^{N}\prod_{t=1}^{T}\Lambda(1-y_{tj})^{N-2}\sum_{k\neq j}(\lambda_j y_{tj}+\lambda_k-\lambda_k y_{tj})^{-2}\prod_{\substack{u=1,\\ u\neq k,j}}^{N}(\lambda_j y_{tj}+\lambda_u-\lambda_u y_{tj})^{-1}$$

або логарифмічної функції правдоподібності

$$l(\lambda)=NT\ln(\Lambda)+$$

$$+\sum_{j=1}^{N}\sum_{t=1}^{T}\left[\begin{array}{c}(N-2)\ln(1-y_{tj})+\\ \ln\left(\sum\limits_{k\neq j}(\lambda_j y_{tj}+\lambda_k-\lambda_k y_{tj})^{-2}\prod\limits_{\substack{u=1,\\ u\neq k,j}}^{N}(\lambda_j y_{tj}+\lambda_u-\lambda_u y_{tj})^{-1}\right)\end{array}\right],$$

де через $y_t = (y_{t1},...,y_{tN})$ позначаємо вектор спостережень (пропорцій) переходу зі стану i у стани $1,2,\ldots,N$, де час спостережень змінюється від 1 до T.

У роботах [239; 241; 242] розглянуто методи вибору оптимальної кількості кластерів, проте дані методи не дають зробити висновки про належність веб-сторінок до одного чи до різних кластерів. Оскільки перевірка однорідності вибірок буде перевірятися на різних вибірках по різних веб-сторінках, то для визначення приналежності до одного кластера двох веб-сторінок можна використати 3 підходи.

Перший підхід – використання тесту Колмогорова-Смірнова. Недоліком цього підходу є те, що тест є непараметричним з невисокою потужністю.

Другий підхід – побудова інтервалів надійності із використанням інформації за Фішером. Такий підхід ґрунтується на використанні статистичних методів із більш високою потужністю, зокрема, на можливості використання асимптотичних властивостей оцінок параметрів розподілу та побудову інтервалів надійності із використанням інформації за Фішером [248; 249].

Загальний алгоритм перевірки гіпотези про приналежність i-го та j-го станів до одного кластера, де параметри для двох станів $\lambda^{(i)}, \lambda^{(j)} \in R_+^N$, можна визначити за допомогою наступних гіпотез:

$$H_0 : \lambda^{(i)} = \lambda^{(j)}$$

проти альтернативної гіпотези

$$H_A : \lambda^{(i)} \neq \lambda^{(j)}$$

щодо параметрів розподілу двох станів з сімейства (4) можна сформулювати наступним чином.

Крок 1. Оцінка параметрів для i-го стану. Позначимо дану оцінку через $\hat{\lambda}^{(i)}$.

Крок 2. Обчислення інформації за Фішером $I_T(\hat{\lambda}^{(i)})$ для розподілу (4.5.4) та побудова інтервалу надійності $CI_\alpha^{(i)}$ для розподілу $\mathbf{N}(\hat{\lambda}^{(i)}; I_T^{-1}(\hat{\lambda}^{(i)}))$ для першої вибірки, де через α будемо позначати рівень значущості.

Крок 3. Оцінка параметрів для j-го стану. Позначимо дану оцінку через $\hat{\lambda}^{(j)}$.

Крок 4. Перевірка належності оцінки $\hat{\lambda}^{(j)}$ до інтервалу надійності $CI_\alpha^{(i)}$ та висновок про приналежність i-го та j-го станів до одного кластера.

Третій підхід – використання bootstrap методу моделювання інтервалу надійності $CI_\alpha^{(i)}$, запропонованого на основі робіт [250; 251]. Згідно з цим методом інтервал надійності $CI_\alpha^{(i)}$ будується наступним чином: на основі оцінки $\hat{\lambda}^{(i)}$ генеруються вибірки $y_t^{(i)} = (y_{t1}^{(i)}, ..., y_{tN}^{(i)})$, $t = 1, ..., B$; за вибірками $y_t^{(i)}$ здійснюється оцінка параметру λ для розподілу (4.5.4) (де B – кількість ітерацій у bootstrap методі); на основі отриманих оцінок, будується інтервал надійності $CI_\alpha^{(i)}$.

4.6. Моделювання та порівняння з класичними результатами

Модельний приклад 4.6.1.

У роботі [242] розглянуто задачу визначення оптимальної кількості кластерів для мережі, зв'язки між вузлами в одному кластері будемо моделювати за допомогою розподілу Пуассона з параметром $\lambda_1 + \lambda_2$. Крім того, шуми, які будуть відображати міжкластерні зв'язки, будемо моделювати за допомогою розподілу Пуассона з параметром λ_2, кількість кластерів при моделюванні рівна k. Таким чином, розподіл елемента матриці суміжності A буде мати вигляд

$$A_{ij} \sim \begin{cases} Poiss(\lambda_1 + \lambda_2), & i, j \text{ в одному кластері}, \\ Poiss(\lambda_2) \text{ – в іншому випадку}. \end{cases}$$

Для визначення оптимального значення кількості кластерів k_{opt}, як було зазначено в теоремі 4.4.1, скористаємось власними значеннями, близькими до одиниці (тобто викидами). Точніше кажучи,

$$k_{opt} = \#\left\{ \lambda_i(P) : |\lambda_i(P)| \geq \frac{2\max\left(\sigma_1,...,\sigma_{k_{opt}}\right)}{\sqrt{\min\left(N_1,...,N_{k_{opt}}\right)}} \right\},$$

де N_i – кількість елементів у i-му кластері, σ_i^2 – дисперсія елементів у i-му кластері. Для розв'язання проблеми різних розмірностей будемо використовувати наступний критерій визначення викидів:

$$k_{opt} = \#\{\lambda_i : \mathrm{Re}(\lambda_i) > \max_{i=1,...,N} |\mathrm{Im}(\lambda_i)|\}.$$

Результати проведеної кластеризації наведено в таблиці 4.6.1. З таблиці видно, що невірна оцінка оптимальної кількості кластерів може виникати у трьох випадках.

- Кількість кластерів велика, тобто $k = O(N)$.
- Розміри кластерів дуже різняться, тобто значення співвідношення

$$r = \frac{\max(N_i)}{\min(N_i)}$$

є великим, де висновок про r слід робити з врахуванням N та k.

- Середнє значення шуму, за яким будуються міжкластерні зв'язки, має однаковий порядок із усередненими зв'язками всередині кластерів, тобто

$$E(A_{ii}) \approx E(A_{ij}).$$

Дані три критичні ситуації призводять до заниження значення оптимальної кількості кластерів k_{opt}. Виконання двох сценаріїв ми можемо бачити в останньому випадку таблиці 4.6.1. По-перше, в цьому випадку

$$r = \frac{\max(N_i)}{\min(N_i)} = \frac{800}{20} = 40,$$

що свідчить про незбалансованість між розмірами кластерів. По-друге, середнє значення міжкластерного зв'язку рівне 8, в той же час середній

зв'язок у кластері рівний $\lambda_1+\lambda_2=8+10=18$, тобто дані зв'язки задовольняють співвідношення $E(A_{ii})\approx E(A_{ij})$. З рис. 4.6.1 (власні значення матриці P для випадку $\lambda=(\lambda_1,\lambda_2)=(10,8)$ та $N=(500,20,300,800,100,400)$, за виконання цих двох сценаріїв), викиди "стягуються" до центру. В цьому випадку відбувається заниження оптимальної кількості кластерів k_{opt}.

Таблиця 4.6.1.

Приклади проведення кластеризації при різних параметрах

$\lambda=(\lambda_1,\lambda_2)$	$N=(N_1,...,N_k)$	k	k_{opt}
$(20, 0.5)$	500, 200, 300, 800, 100, 400	6	6
$(20,1)$	500, 200, 300, 800, 100, 400	6	6
$(20,1)$	1000, 200, 20	3	3
$(20, 4)$	200, 200, 200	3	3
$(20,10)$	1000, 200, 200, 200, 200	5	5
$(10, 0.5)$	500, 200, 300, 800, 100, 400	6	6
$(10,8)$	500, 20, 300, 800, 100, 400	6	5

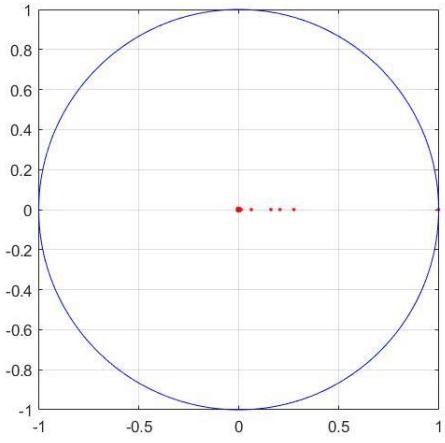


Рис. 4.6.1. Власні значення матриці P для випадку $\lambda=(\lambda_1,\lambda_2)=(10,8)$ та $N=(500,20,300,800,100,400)$

Якщо ж жодна з критичних ситуацій не має місця, то всі викиди будуть близькими до $\lambda=1$, як і зазначено в теоремі 4.4.1. Даний результат

проілюстровано на рис. 4.6.2 при власних значеннях $\lambda = (\lambda_1, \lambda_2) = (10, 0.5)$ матриці P та $N = (500,\ 200,\ 300,\ 800,\ 100,\ 400)$.

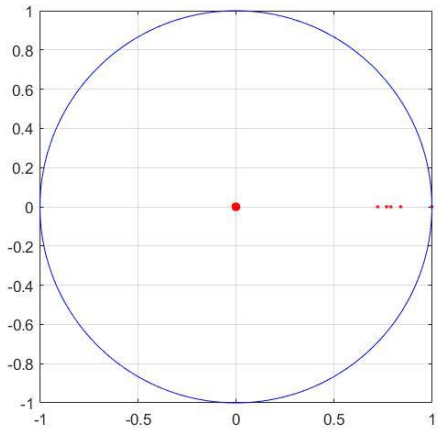


Рис. 4.6.2. Власні значення матриці P для випадку $\lambda = (\lambda_1, \lambda_2) = (10, 0.5)$ та $N = (500,\ 200,\ 300,\ 800,\ 100,\ 400)$

Як бачимо з проведеного моделювання, для розподілу Пуассона розроблений метод оцінки кластерів дає достатньо точні результати оцінки оптимальної кількості кластерів [242].

Модельний приклад 4.6.2.

Розглянемо тестування розглянутого нового методу на основі моделювання методом Монте Карло із $M = 10^4$ реалізаціями. При моделюванні будемо використовувати $N = 10^2, \dots, 10^5$ вершин з $S_1, \dots, S_k$ кластерами. Для аналізу нами буде використано декілька основних параметрів:

1. λ_1 – параметр Пуассонівського розподілу, на основі якого моделюється кількість зав'язків для вершин графу, що знаходяться в різних кластерах,

$$A_{ij} \sim Pois(\lambda_1), i \in S_m, j \in S_l, m \neq l.$$

2. λ_2 – параметр Пуассонівського розподілу, на основі якого моделюється кількість зав'язків для вершин графу, що знаходяться в одному кластері, тобто

$$A_{ij} \sim Pois(\lambda_1 + \lambda_2), i, j \in S_m.$$

3. (L, H) – мінімальна та максимальна кількість об'єктів у кластері.

4. k – кількість кластерів, використана при моделюванні.

Оптимальну кількість кластерів будемо шукати чотирма методами: методом «ліктя» [206], методом k-core decomposition [95], методом силуету [93] та розробленим методом.

Результати проведеного моделювання, тобто приклади знаходження оптимальної кількості кластерів методами «ліктя», k-core decomposition, силуету [93; 252] можна побачити в таблиці 4.6.2.

Таблиця 4.6.2.

Результати моделювання середньої кількості кластерів та середньоквадратичного відхилення від середньої кількості $\mu(\sigma)$

Гіперпараметри моделювання $(\lambda_1, \lambda_2, L, H, \boldsymbol{k})$	Метод «ліктя»	k-core decomposition	Метод силуету	Новий метод
$(10, 1{,}200{,}300, \mathbf{20})$	17.4 (4.25)	16.1 (7.3)	16.4 (5.3)	10 (8)
$(10, 2{,}200{,}300, \mathbf{20})$	18.5 (2.31)	18.2 (3.74)	17.4 (5)	19 (5)
$(10, 5{,}200{,}300, \mathbf{20})$	19.1 (2.21)	19.3 (2.35)	18.3 (4.5)	20 (0.5)
$(10, 10{,}200{,}300, \mathbf{20})$	19.4 (1.85)	19.5 (1.35)	19 (2.83)	20 (0.23)
$(10, 20{,}200{,}300, \mathbf{20})$	19.5 (1.11)	19.8 (0.98)	19.4 (1.95)	20 (0.16)
$(10, 40{,}200{,}300, \mathbf{20})$	19.6 (0.65)	19.86 (0.43)	19.71 (1.03)	20 (0.09)
$(10, 100{,}200{,}300, \mathbf{20})$	19.98 (0.115)	19.99 (0.09)	19.87 (0.26)	20 (0.02)
$(10, 1{,}200{,}300, \mathbf{150})$	173 (25.7)	164 (12.4)	171 (25.4)	78 (53)
$(10, 5{,}200{,}300, \mathbf{150})$	161 (15.3)	157 (9.21)	163 (18.4)	144 (7.9)
$(10, 20{,}200{,}300, \mathbf{150})$	156 (7.89)	153.4 (7.31)	158.3 (13.23)	148 (4.28)
$(10, 40{,}200{,}300, \mathbf{150})$	153 (4.26)	152.1 (4.16)	154.5 (9.31)	149.3 (2.11)
$(10, 40{,}20{,}300, \mathbf{150})$	156 (5.73)	154.3 (10.23)	145.5 (15.18)	144.3 (13.7)

Як ми бачимо із даних обчислень, для малих значень відношення $\frac{\lambda_2}{\lambda_1}$ оцінка оптимальної кількості кластерів на основі методу ліктя, k-core decomposition та методу силуету є точнішим ніж за новим методом. Проте при $\frac{\lambda_2}{\lambda_1} > 30\%$ запропонований метод показує меншу помилку визначення точної кількості кластерів, а також менше стандартне відхилення порівняно із іншими методами. Ті самі результати ми можемо бачити на двох наступних рисунках (рис. 4.6.3 та рис. 4.6.4) . На рис. 4.6.3 видно, що ширина інтервалу надійності значно зменшується для нового методу із ростом співвідношення $\frac{\lambda_2}{\lambda_1}$. Для інших трьох методів звуження ширини інтервалу надійності відбуваються більш плавно, тобто помилка для даних методів є вищою.

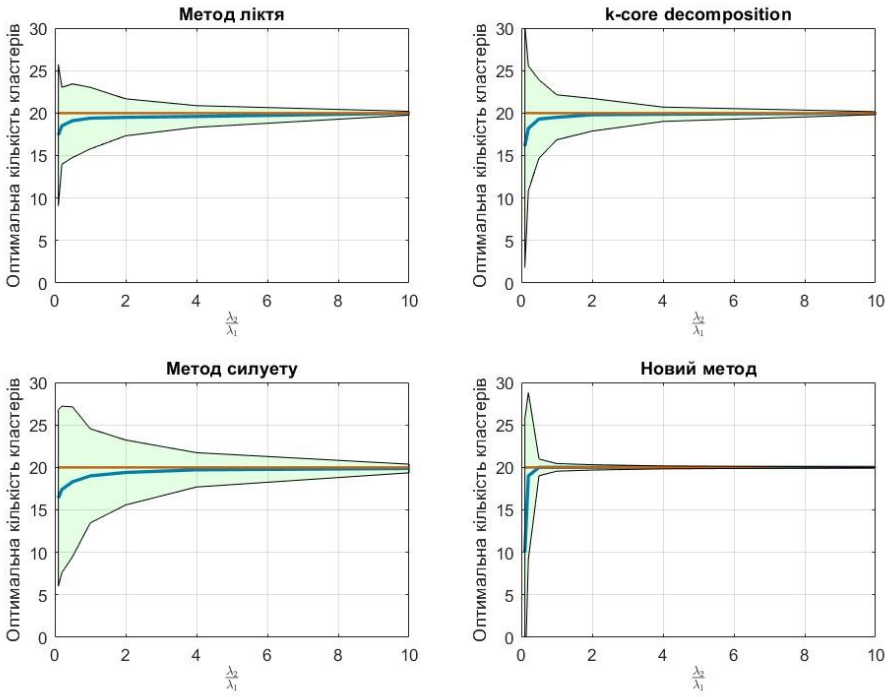


Рис. 4.6.3. Оптимальна кількість кластерів, визначена за чотирма методами разом із 95% інтервалом надійності

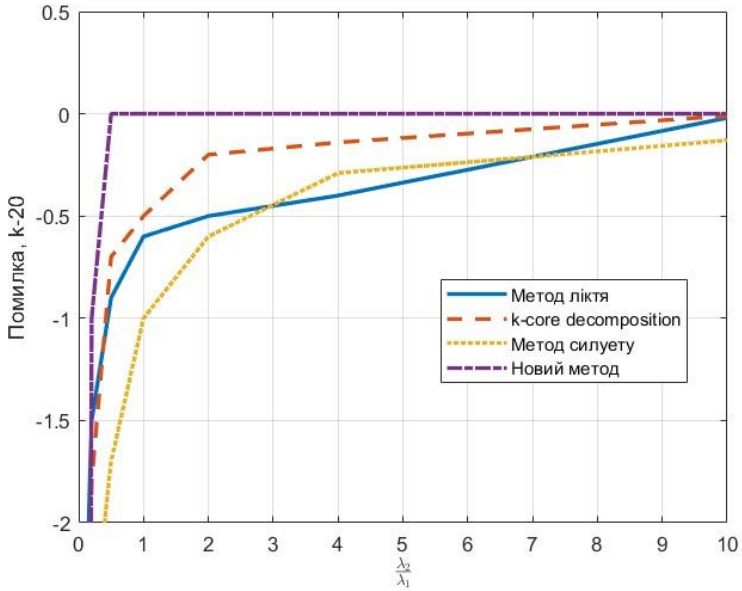


Рис. 4.6.4. Помилка кластеризації для чотирьох методів кластеризації при k=20

З рис. 4.6.4 видно збіжність оптимальної кількості кластерів, отриманих за новим методом, відбувається набагато швидше із ростом відношення $\frac{\lambda_2}{\lambda_1}$ і при $\frac{\lambda_2}{\lambda_1} > 4$ всі чотири методи в середньому будуть давати вірний результат. Згідно наших припущень це означає, що при середній кількості переходів в кластерах вп'ятеро більшій, ніж між кластерами, всі чотири методи будуть давати однаковий результат. Проте в діапазоні $\frac{\lambda_2}{\lambda_1} \in (0.3; 1)$ запропонований новий метод показує кращі результати.

Зауваження 4.6.1. До недоліків запропонованого методу слід віднести те, що значна варіація величини кластерів може призвести до зменшення середнього значення k_{opt}. Цей результат відслідковується із останнього рядка таблиці 4.6.2. З цього можна зробити висновок, що новий запропонований метод є чутливий до наявності кластерів малої розмірності.

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі на основі аналізу основних теорем та тверджень теорії випадкових матриць, розглянуто їх застосування до задач кластеризації на графах.

Проаналізовано основні властивості спектру стохастичної матриці графу. Отримані результати дозволили провести спектральний аналіз власних значень матриці переходу для випадку однотипного зв'язку для всіх елементів.

Проаналізовано граничний розподіл власних значень матриці переходу на основі зв'язку викидів серед власних значень та оптимальної кількості кластерів k_{opt}. На основі припущень про однотипність зв'язків у кластері, зроблено висновок про оптимальну кількість кластерів для різних прикладних задач. Отримані теоретичні результати про асимптотичні властивості спектру стохастичної матриці.

Проведено моделювання мережі зв'язків, що розподілені за законом Пуассона, та знайдено оптимальну кількість кластерів. Проведено порівняльний аналіз нового методу знаходження оптимальної кількості кластерів з відомими класичними методами (метод «ліктя», k-core decomposition, силуету). Результати моделювання вказують на високу точність визначення оптимальної кількості кластерів новим методом.

Основні наукові результати розділу опубліковані в працях [238-242].

ЗАГАЛЬНІ ВИСНОВКИ

У дисертаційній роботі поставлено і розв’язано актуальну науково-прикладну задачу розробки спеціалізованого програмного забезпечення для збирання статистичної інформації у веб-просторі та її обробки методами інтелектуального аналізу даних. На основі аналізу основних теорем та тверджень теорії випадкових матриць, розглянуто їх застосування до задач кластеризації на графах. Отримані результати впроваджено у освітній процес вищого навчального закладу та роботу підприємств.

Основні результати дисертаційної роботи:

1. Проведено аналітичний огляд існуючих програмних засобів для збирання інформації у веб-просторі, наведено їх короткий опис, визначено їхні переваги та недоліки. Продемонстровано складність аналізу комплексних мереж, що призводить до розбіжностей у інтерпретації результатів різних авторів.

2. Вперше на основі сучасних методів розробки програмного забезпечення створено кроулер, який, на відміну від існуючих, має модульну структуру та багаторівневу архітектуру, підтримує контейнеризацію і роботу в багатопотоковому режимі, легко масштабується, характеризується гнучкістю, високою здатністю до розширення та можливістю аналізу аналогічних графів відмінного від веб походження, і на його основі створено інформаційну технологію збирання та обробки інформації у складних мережах. Особливістю даного кроулера є наявність вбудованого аналітичного модуля для проведення статистичного та кластерного аналізу складних мереж.

3. За допомогою розробленої технології вперше досліджено статистичні характеристики та кластерну структуру українського освітянського сегменту *edu.ua*, польської підмережі *edu.pl* та ізраїльської академічної зони *ac.il*. Порівняльний кластерний аналіз показав, що українська підмережа *edu.ua* демонструє подібні до інших статистичні характеристики, але ненасичена вузлами освітянських закладів. Український сегмент *edu.ua* становить,

найменш розвинену структуру, кількість вузлів у кластерах – найменша з усіх досліджених сегментів.

4. Показано, що збільшення глибини сканування до 5 рівнів від точок входження не змінює якісні характеристики мережі, що демонструє адекватність отриманих результатів.

5. Проаналізовано граничний розподіл власних значень матриці переходу P на основі зв'язку викидів серед власних значень та оптимальної кількості кластерів k_{opt}. На основі припущень про однотипність зв'язків у кластері, зроблено висновок про оціночне значення оптимальної кількості кластерів для різних випадків. У результаті дослідження побудовано критерій оцінки оптимальної кількості кластерів k_{opt}, обчислення якого ґрунтується на власних значеннях стохастичної матриці P, а саме

$$\hat{k}_{opt} = \#\{\lambda_i(P): Re(\lambda_i(P)) > \max|Im(\lambda_i(P))|\}.$$

6. Набула подальшого розвитку теорія дослідження графових структур: сформульовано і доведено твердження, які дозволяють оцінювати розподіли власних значень випадкових матриць та переносити класичні результати на матриці із слабко корельованими елементами; дані твердження дозволяють розробити якісний алгоритм перевірки належності елементів до одного кластеру.

7. Проведено моделювання на основі методу Монте - Карло мережі зв'язків (графів), зв'язки в яких будуються за законом Пуассона, оцінено оптимальну кількість кластерів та проведено порівняльний аналіз із класичними методами кластеризації (метод «ліктя», k-core decomposition, метод силуету). Результати моделювання вказують на високу точність визначення оптимальної кількості кластерів новим методом при зменшенні відношення середньої величини зв'язків між кластерами до середньої величини в кластерах, а також меншу чутливість до різного розміру кластерів порівняно з класичними методами.

8. Результати дисертаційної роботи впроваджено у компаніях «Qlicks B.V.», ТОВ «Квант Азимут» і використовуються в освітньому процесі відділу комп'ютерних технологій Навчально-наукового інституту фізико-технічних та комп’ютерних наук, Чернівецького національного університету імені Юрія Федьковича.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Newman M. E. J. The Structure and Function of Complex Networks. *SIAM Review*. 2003. Vol. 45, no. 2. P. 167–256.

2. Complex networks: Structure and dynamics / S. Boccaletti et al. *Physics Reports*. 2006. Vol. 424, no. 4–5. P. 175–308.

3. Ланде Д. В., Субач І. Ю., Бояринова Ю. Є. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки : навч. посібн. Київ : ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. 300 с.

4. Complex networks, communities and clustering: A survey / B. Saha et al. *arXiv preprint arXiv:1503.06277*, 2015.

URL: https://doi.org/10.48550/arXiv.1503.06277

5. Barabasi A., Réka A. Statistical mechanics of complex networks. *Rev. Mod. Phys*. 2002. Vol. 74, no. 1. P. 47–97. URL: https://doi.org/10.1103/RevModPhys.74.47

6. Network Motifs: Simple Building Blocks of Complex Networks / R. Milo et al. *Science*. 2002. Vol. 298, no. 5594. P. 824–827.

7. Strogatz S. Exploring complex networks. *Nature*. 2001. Vol. 410. P. 268–276.

8. Graph structure in the Web / A. Broder et al. *Computer networks* : Proceedings of the 9th World Wide Web Conference. 2000. Vol. 33, no. 1– 6. P. 309–320.

9. Using graph theory to analyze biological networks / G. A. Pavlopoulos et al. *BioData Mining*. 2011. Vol. 4, no. 10, 27 p.

10. Newman M. E. J. Networks: An Introduction. Oxford : Oxford University Press, 2018. 789 p.

11. A Dynamic Model for On-Line Social Networks / A. Bonato et al. *Algorithms and Models for the Web-Graph*. Berlin, Heidelberg : Springer, 2009. P. 127–142.

12. Складні мережі / Ю. Головач та ін. *Журнал фізичних досліджень*. 2006. Т. 10, № 4. С. 247–289.

13. Erdös P., Rényi A. On Random Graphs I. *Publicationes Mathematicae Debrecen*. 1959. Vol. 6. P. 290–297.

14. Erdös P., Rényi A. On the Evolution of Random Graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*. 1960. Vol. 5. P. 17–61.

15. Dorogovtsev S., Mendes J. F. Evolution of networks. *Advances In Physics.* 2002. Vol. 51, no. 4. P. 1079–1187.

16. Ланде Д. В., Субач І. Ю. Візуалізація та аналіз мережевих структур : навч. посібн. Київ : «Політехніка», 2021. 80 с.

17. Dorogovtsev S. N., Mendes J. F. F. The Nature of Complex Networks (online edn). Oxford : Oxford Academic, 2022. 456 p.

18. Watts D. J., Strogatz S. H. Collective dynamics of ‘small-world’ networks. *Nature*. 1998. Vol. 393. P. 440–442.

19. Classes of small-world networks / L. A. N. Amaral et al. *Proceedings of the National Academy of Sciences of the United States of America*. 2000. Vol. 97, no. 21. P. 11149–11152.

20. Alchalabi A. E. On Small-World Networks: Survey and Properties Analysis. *arXiv preprint arXiv: abs/2101.11191*, 2021. URL: https://doi.org/10.48550/arXiv.2101.11191

21. Bharali A., Doley A. On Small-World and Scale-Free Properties of Complex Network. *International Journal of Mathematics and Statistics Invention (IJMSI)*. 2017. Vol. 5, no. 7. P. 11–16.

22. Barabási A.-L., Albert R., Jeong H. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*. 1999. Vol. 272, no. 1-2. P. 173–187.

23.Зубок В. Ю., Дармохвал О. Т. Огляд використання математичних параметрів складної мережі для аналізу топології Інтернет. *Збірник наукових праць ІПМЕ ім. Г.Є. Пухова НАН України*. 2010. №55. С. 19–28.

24. Li A. Review of Scale Free Networks and its variances. Massachusetts Institute of Technology, Department of Physics. 2011. URL:

https://web.mit.edu/8.592/www/grades/projects/Projects(2011)/AoxiLi.pdf (Last accessed: 02.11.2017).

25. Adamic L., Buyukkokten O., Adar E. A social network caught in the Web. *First Monday*. 2003. Vol. 8, no. 6.

26. Kumar R., Novak J., Tomkins A. Structure and Evolution of Online Social Networks. Link Mining: Models, Algorithms, and Applications. New York, NY, 2010. P. 337–357.

27. Golder S. A., Wilkinson D. M., Huberman B. A. Rhythms of Social Interaction: Messaging Within a Massive Online Network. In: Communities and Technologies / ed. by C. Steinfield et al. London : Springer, 2007. P. 41–42.

28. Analysis of topological characteristics of huge online social networking services / Y.-Y. Ahn et al. *The 16th International Conference*, Banff, Alberta, Canada, 8–12 May 2007. New York, New York, USA, 2007. P. 835–844.

29. Measurement and analysis of online social networks / A. Mislove et al. *The 7th ACM SIGCOMM Conference*, San Diego, California, USA, 24–26 October 2007. New York, New York, USA, 2007. P. 29–42.

30. Newman M. E. J. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*. 2006. Vol. 103, no. 23. P. 8577–8582.

31. On the evolution of user interaction in Facebook / B. Viswanath et al. *The 2nd ACM Workshop*, Barcelona, Spain, 17 August 2009. New York, New York, USA, 2009. P. 37–42.

32. Newman M. E. J. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*. 2001. Vol. 98, no. 2. P. 404–409.

33. Arqué N., Nettleton D. Analysis of On-Line Social Networks Represented as Graphs - Extraction of an Approximation of Community Structure Using Sampling. In: *Modeling Decisions for Artificial Intelligence (MDAI) 2012*. *Lecture Notes in Computer Science*, vol 7647. Girona, Catalunya, Spain: Springer, 2012. P. 149–160.

34. Nettleton D. Data Mining of Social Networks Represented as Graphs. *Computer Science Review*. 2013. Vol. 7, no. 1. P. 1–34.

35. Ахрамович В. М. Моделювання і візуалізація соціальних мереж. *Зв'язок*. 2020. № 2(144). С. 28–33.

36. Evolution of the social network of scientific collaborations / A. L. Barabási et al. *Physica A: Statistical Mechanics and its Applications*. 2002. Vol. 311, no. 3–4. P. 590–614.

37. Newman M. E. J. Coauthorship networks and patterns of scientific collaboration. *Proc. Nat. Acad. Sci.* 2004. vol. 101. P. 5200-5205.

38. Does country-level R&D efficiency benefit from the collaboration network structure? / J. Guan et al. *Research Policy*. 2016. Vol. 45, no. 4. P. 770–784.

39. Vinayak R. A., Kshitij A. Signatures of capacity development through research collaborations in artificial intelligence and machine learning. *Journal of Informetrics*. 2023. Vol. 17, no. 1, article 101358.

40. Ландэ Д., Снарский А. Попытки объять необъятное, или World Wide Web под прицелом. *Сети и бизнес.* 2007, № 4 (35). С. 18–24.

41. Baeza-Yates R., Castillo C., Efthimiadis E. N. Characterization of national Web domains. *ACM Trans. Intern. Tech.* 2007. Vol. 7, no. 2, Article 9. 32 pages.

42. Kyrychenko O.**,** Ostapov S., Kanovsky I. Comparison of Statistical Characteristics of Certain Internet Subdomains. Chapter 1. Monograph. *Internet in the information society. Insights on the information systems, structures and applications* / ed. by M. Rostanski, P. Pikiewicz. Scientific Publishing of the Academy of Business in Dabrowa Gornicza : Wydawnictwo Naukowe, 2014. 138 p. ISBN: 978-83-62897-91-9.

43. Кириченко О. Л., Kanovsky I., Остапов С. Е. Складні мережі та їх статистичні характеристики: аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2013)* : тези

доповідей Всеукр. наук.-практ. конф., м. Чернівці, 27–31 травня 2013 р. Чернівці : Видавничий дім «Родовід», 2013. С. 16–22.

44. Кириченко О. Л., Остапов С. Е. Дослідження структури Веб-простору за допомогою кластерного аналізу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2016)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 21–24 травня 2016 р. Чернівці : Видавничий дім «Родовід», 2016. С. 112–114.

45. Kyrychenko O., Ostapov S., Kanovsky I. Investigation of the certain internet domain statistical characteristics / Статистичні характеристики деяких зон інтернету та їх дослідження. *Eastern-European Journal of Enterprise Technologies*. 2013. Vol. 6, no. 12(66). P. 91–96.

46. Фурашев В., Зубок В., Ландэ Д. Параметры украинского сегмента Интернет как сложной сети. *Открытые информационные и компьютерные технологии.* 2008. Том 40. С. 235–242.

47. Кириченко О. Л., Остапов С. Е., Kanovsky I. Статистичні характеристики українських доменів edu.ua, net.ua глобальної www-мережі та їх дослідження. *Актуальні проблеми інформаційних технологій, економіки та права (ІТЕП–2013)* : тези доповідей Міжнар. наук.-практ. конф., м. Чернівці, 3–5 квітня 2013 р. Чернівці : Книги-ХХІ, 2013. С.84–85.

48. Dorogovtsev S. N., Mendes J. F. F., Samukhin A. N. Giant strongly connected component of directed networks. *Physical Review E*. 2001. Vol. 64, no. 2, article 025101(R).

49. Leung S.-T. A., Perl S. E., Stata R., Wiener J. L. Towards Web-scale Web archeology. Paolo Alto CA: Compaq Research Center, 2001. 21 p. (Tech. rep. 174, Sept.)

50. Gomes D., Silva M. J. Characterizing a national community Web. *ACM Transactions on Internet Technology*. 2005. Vol. 5, no. 3. P. 508–531.

51. Tsallis C., de Albuquerque M. P. Are citations of scientific papers a case of nonextensivity?. *European Physical Journal B*. 2000. vol. 13, no. 4. P. 777–780.

52. Vazquez A. Statistics of citation networks. *arXiv preprint* arXiv:cond-mat/0105031v1, 2001. URL: https://doi.org/10.48550/arXiv.cond-mat/0105031

53. Jeong H., N′eda Z., Barabasi A.-L. Measuring preferential attachment for evolving networks. *EPL (Europhysics Letters)*. 2003. Vol. 61, Issue 4. P. 567–572.

54. Зубок В. Ю., Головін А. Ю. Огляд і порівняльний аналіз методик та засобів дослідження топології Інтернету. *Інформація і право*. 2011. №2. С. 78-84.

55. Zhou S., Mondragón R. J. Accurately modeling the internet topology. *Physical Review E*. 2004. Vol. 70, no. 6, article 066108.

56. Пасічник В. В., Іванущак Н. М. Імітаційне моделювання процесу росту локальних комп'ютерних мереж. *Вісник Хмельницького національного університету (технічні науки)*. 2010. № 5. С. 238–245.

57. Пасічник В. В., Іванущак Н. М. Еволюційне моделювання і візуалізація комп'ютерних мереж у середовищі PROCESSING. *Вісник Національного університету «Львівська політехніка» (Комп'ютерні науки та інформаційні технології)*. 2012. Вип. 732. С. 147–156.

58. Зубок В. Ю. Аналіз впливу топології на ефективність та вразливість в глобальних комп'ютерних мережах. *Інформаційні технології. Безпека.* 2012. № 2(2). С. 17–25.

59. The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles / R. Guimera et al. *Proceedings of the National Academy of Sciences*. 2005. Vol. 102, no. 22. P. 7794–7799.

60. Li W., Cai X. Statistical analysis of airport network of China. *Phys. Rev.E.* 2004. Vol. 69. P. 461–466.

61. Guida M., Maria F. Topology of the Italian airport network: A scale-free small-world network with a fractal structure?. *Chaos Solitons & Fractals*. 2007. Vol. 31, no. 3. P. 527–536.

62. Latora V., Marchiori M. Is the Boston subway a small-world network?. *Physica A: Statistical Mechanics and its Applications*. 2002. Vol. 314, no. 1-4. P. 109–113.

63. Small-world properties of the Indian railway network / P. Sen et al. *Phys. Rev. E.* 2003. Vol. 67. P. 125–129.

64. Xu X., Hu J., Liu F., Liu L. (або Xinping X., Junhui H., Feng L., Lianshou L. - не розумію, де ім'я, де прізвище) Scaling and correlations in three bus-transport networks of China. *Physica A: Statistical Mechanics and its Applications*. 2007. Vol. 374, no. 1. P. 441–448.

65. von Ferber C., Holovatch Y., Palchykov V. Scaling in public transport networks. *Condensed Matter Physics.* 2005. Vol. 8, no. 1. P. 225–234.

66. Sienkiewicz J., Holyst J. A. Public transport systems in Poland: From Bialystok to Zielona Gora by bus and tram using universal statistics of complex networks. *Acta Phys. Polonica.* 2005. Vol. 36, no. 5. P. 1771–1778.

67. Public transport networks: empirical analysis and modeling / von Ferber C. et al. *Physica A.* 2007. no. 380. P. 585–591.

68. Пасічник В. В., Іванущак Н. М. Дослідження статистики мереж громадського транспорту найбільших міст західного регіону України. *Східно-Європейський журнал передових технологій*. 2011. Том 6/4, № 54. С. 13–17.

69. Voigt A., Almaas E. Complex Network Analysis in Microbial Systems: Theory and Examples. *Microbial Systems Biology*. New York, NY, 2021. P. 167–191.

70. The large-scale organization of metabolic networks / H. Jeong et al. *Nature*. 2000. Vol. 407, no. 6804. P. 651–654.

71. Berg J., Lässig M., Wagner A. A. Structure and evolution of protein interaction networks: a statistical model for link dynamics and gene duplications. *BMC Evolutionary Biology*. 2004. Vol. 4, no. 1. P. 51.

72. Najma F. A. Biological Networks in Human Health and Disease / ed. by R. Ishrat. Singapore : Springer Nature Singapore, 2023. URL: https://doi.org/10.1007/978-981-99-4242-8

73. Saraiya P. R., Ganage Y. Study of clustering techniques in the data mining domain. *International Journal of Computer Science and Mobile Computing*. 2018. Vol. 7, no. 11. P. 31–37.

74. Machine Learning and Data Mining Methods in Diabetes Research / I. Kavakiotis et al. *Computational and Structural Biotechnology Journal*. 2017. Vol. 15. P. 104–116.

75. Clustering and Classification Algorithms in Data Mining / M. V. Ahamad et al. *International Journal of Advance Research in Science and Engineering*. 2017. Vol. 6, no. 7.

76. Kasthuri M., Kanchana S., Hemalatha R. A Comparative Study on Various Clustering Techniques in Data Mining. *International Journal of Computer Sciences and Engineering*. 2018. Vol.06, Issue.11. P. 1–8.

77. Everitt B., Landau S., Leese M. Cluster Analysis (4th edition). London : Arnold, 2001. 248 p.

78. Niknam T., Amiri B. An efficient hybrid approach based on PSO, ACO and k-means for cluster analysis. *Applied Soft Computing*. 2010. Vol. 10, no. 1. P. 183–197.

79. Graaff A., Engelbrecht A. Using sequential deviation to dynamically determine the number of clusters found by a local network neighbourhood artificial immune system. *Applied Soft Computing*. 2011. Vol. 11, no. 2. P. 2698–2713.

80. Saha I., Maulik U., Plewczynski D. A new multi-objective technique for differential fuzzy clustering. *Applied Soft Computing*. 2011. Vol. 11, no. 2. P. 2765–2776.

81. Saha S., Bandyopadhyay S. Some connectivity based cluster validity indices. *Applied Soft Computing*. 2012. Vol. 12, no. 5. P. 1555–1565.

82. Maulik U., Bandyopadhyay S. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002. Vol. 24, no. 12. P. 1650–1654.

83. Anitha A. An Efficient Agglomerative Clustering Algorithm for Web Navigation Pattern Identification. *Circuits and Systems*. 2016. Vol. 7, no. 9. P. 2349–2356.

84. Волосюк Ю. В. Аналіз алгоритмів кластеризації для задач інтелектуального аналізу. *Збірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка*. 2014. Вип. 47. С. 112–119.

85. Sajana T., Sheela Rani C. M., Narayana K. V. A Survey on Clustering Techniques for Big Data Mining. *Indian Journal of Science and Technology*. 2016. Vol. 9, no. 3. P. 1–12.

86. Kankal S. S., Dhakne A. R., Tayade Y. R. A Brief Survey on Clustering Algorithms in Data Mining. *International Journal for Scientific Research and Development (IJSRD)*. 2017. Vol. 4, no. 11. P. 494-497.

87. Мелешко Є. Методи кластеризації графів соціальних мереж для побудови рекомендаційних систем. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2019. Том 2, № 54. С. 129–134.

88. Lin F., Cohen W. W. Power iteration clustering. *Proceedings of the 27th International Conference on Machine Learning*, Haifa Israel, June 21–24, 2010. Madison WI, USA : Omnipress, 2010. P. 655–662.

89. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Проведення оптимальної кластеризації структури веб-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2017)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 05–08 жовтня 2017 р. Чернівці : Видавничий дім «Родовід», 2017. С. 68–70.

90. Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing / Yan Jun, Zhang Benyu et al. *IEEE transactions on Knowledge and Data Engineering*. 2006. Vol. 18, no. 3. P. 320-333.

91. Napoleon D., Pavalakodi S. A New Method for Dimensionality Reduction Using K-Means Clustering Algorithm for High Dimensional Data Set. *International Journal of Computer Applications*. 2011. Vol. 13, no. 7. P. 41–46.

92. A Decision Criterion for the Optimal Number of Clusters in Hierarchical Clustering / Y. Jung et al. *Journal of Global Optimization*. 2003. Vol. 25. P. 91–111.

93. Lenssen L., Schubert E. Clustering by Direct Optimization of the Medoid Silhouette. *15th International Conference, SISAP 2022* : Proceedings, Bologna, 5–7 October 2022. Berlin, Heidelberg, 2022. P. 190–204.

94. Kuraria A., Jharbade N., Soni M. Centroid Selection Process Using WCSS and Elbow Method for K-Mean Clustering Algorithm in Data Mining. *International Journal of Scientific Research in Science, Engineering and Technology*. 2018. Vol. 4, no. 11. P. 190–195.

95. Cheng S.-T., Chen Y.-C., Tsai M.-S. Using k-Core Decomposition to Find Cluster Centers for k-Means Algorithm in GraphX on Spark. *CLOUD COMPUTING 2017: The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization* : Proceedings, Athens. 2017. P. 93–98.

96. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Проведення оптимальної кластеризації структури деяких зон веб-простору за допомогою методу K-Core Decomposition. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2018)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 11–14 жовтня 2018 р. Чернівці : Видавничий дім «Родовід», 2018. С. 48–50.

97. Кириченко О. Л., Остапов С. Е. Застосування методу k-core Decomposition для проведення оптимальної кластеризації. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXVII Міжнар. наук.-практ. конф. MicroCAD-2019 (м. Харків, 15–17 травня 2019 р.) : у 4 ч. Ч. IV. / за ред. проф. Сокола Є. І. Харків : НТУ «ХПІ». С. 155.

98. Lucińska M., Wierzchon S. Clustering Based on Eigenvectors of the Adjacency Matrix. *International Journal of Applied Mathematics and Computer Science*. 2018. Vol. 28, no. 4. P. 771–786.

99. Lutkevich B. What is a search engine? | Definition from TechTarget. *WhatIs.com* : website. URL:

https://www.techtarget.com/whatis/definition/search-engine

100. Survey on research of themed crawling technique. / X. Pan et al. *Application Research of Computers*. 2019. Vol. 37, no. 5.

101. Chang Z. A Survey of Modern Crawler Methods. *The 6th International Conference on Control Engineering and Artificial Intelligence* (*CCEAI 2022*) : Proceedings, New York, NY, USA, 2022. P. 21–28.

102. Summary of web crawler technology research / Yu Linxuan et al. 2020 J. Phys.: Conf. Ser. 2020. Vol. 1449. Article 012036. 7 p.

103. Кириченко О. Особливості архітектури програмного забезпечення для збору та аналізу статистичної інформації в глобальній мережі. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2023. № 2. С. 107–112

104. Кириченко О.Л., Kanovsky I., Остапов С.Е. Програмне забезпечення для дослідження статистичних характеристик глобальної мережі WWW. *Системи обробки інформації.* 2013. Том 2, вип. 3. С. 99-104.

105. Najork M., Heydon A. High-Performance Web Crawling. *Handbook of Massive Data Sets. Massive Computing, vol. 4* / ed. by J. Abello, P. Pardalos, M. Resende. Boston, MA, 2002. P. 25–45.

106. Najork M., Heydon A. Web Crawler Architecture. Encyclopedia of Database Systems / ed. by Liu L., Özsu M. New York, NY, 20017. P. 1–4. URL: https://doi.org/10.1007/978-1-4899-7993-3_457-3

107. GitHub - yasserg/crawler4j: Open Source Web Crawler for Java. *GitHub* : website. URL: https://github.com/yasserg/crawler4j

108. GitHub - scrapy/scrapy: Scrapy, a fast high-level web crawling & scraping framework for Python. *GitHub* : website. URL:https://github.com/scrapy/scrapy

109. UbiCrawler: a scalable fully distributed web crawler / P. Boldi et al. *Softw. Pract. Exper*. 2004. Vol. 34, no. 8. P. 711–726.

110. Seymour T., Frantsvog D., Kumar S. History Of Search Engines. *International Journal of Management & Information Systems (IJMIS)*. 2011. Vol. 15, no. 4. P. 47–58.

111. Short History of Early Search Engines. *Thehistoryofseo* : website. URL: www.thehistoryofseo.com/The-Industry/Short_History_of_Early_Search_Engines

112. Web Archiving: Techniques, Challenges, and Solutions / A. Anthony et al. *International Journal of Management &Amp; Information Technology*. 2013. Vol. 5, no. 3. P. 598–603.

113. Lee D. Web Archiving: Review of Web Crawler. *International Research Journal of Engineering and Technology (IRJET)*. 2021. Vol. 08, no. 02. P. 1318–1321.

114. World Wide Web Wanderer. *Wikipedia* : website. URL: https://en.wikipedia.org/wiki/World_Wide_Web_Wanderer (date of access: 21.11.2023).

115. History of Search Engines: From 1945 to Google Today. *Search Engines History* : website. URL: http://www.searchenginehistory.com

116. Patel H. The Top Search Engine List for 2022. *Iceconnect* : website.

URL: https://www.iceconnect.com/blog/tech-tips/the-ultimate-top-12-search-engine-list/#Excite

117. Eichmann D. The RBSE Spider-Balancing Effective Search Against Web Load. *Computer networks and ISDN systems*. 1994. Vol.27, no. 2. P. 308.

118. Ahuja M., Singh J., Nica V. Web Crawler: Extracting the Web Data. *International Journal of Computer Trends and Technology*. 2014. Vol. 13, no. 3. P. 132–137.

119. A Smart Itsy Bitsy Spider for the Web / Hsinchun Chen et al. *Journal of the American Society for Information Science*. 1998. Vol. 49, no. 7. P. 604–618.

120. Patel H. The Top Search Engine List for 2022. *Iceconnect* : website. URL: https://www.iceconnect.com/blog/tech-tips/the-ultimate-top-12-search-engine-list/#Yahoo

121. Pinkerton B. Webcrawler: finding what people want : Ph. D. Seattle, WA, 2000. 105 p.

122. WebCrawler Timeline. *Iceconnect* : website. URL:

http://www.thinkpink.com/bp/WebCrawler/History.html

123. Menczer F,. Pant G., Srinivasan P. Topical Web Crawlers: Evaluating Adaptive Algorithms. *ACM Transactions on Internet Technology*. 2004. Vol. V, no. N. P. 1–38.

124. What was the First Search Engine? *Wisegeek* : website. URL: https://web.archive.org/web/20070427154159/http://www.wisegeek.com/what-was-the-first-search-engine.htm

125. Patel H. The Top Search Engine List for 2022. *Iceconnect* : website. URL: https://www.iceconnect.com/blog/tech-tips/the-ultimate-top-12-search-engine-list/#Lycos

126. Sullivan D. Search & Real Time Madness. *SearchEngineLand* : website. URL: https://searchengineland.com/search-real-time-madness-31668

127. Burner M. Crawling towards Eternity. *New Architect* : website. URL: https://people.apache.org/~jim/NewArchitect/webtech/1997/05/burner/index.html

128. Bergman M. K. The deep Web: surfacing hidden value. Journal of Electronic Publishing. *International Journal of Computer Trends and Technology*. 2014. Vol. 13, no. 3. P. 132–137.

129. Miller R., Bharat K. SPHINX: A Framework for Creating Personal, Site-Specific Web Crawlers. *Computer Network and ISDN Systems.* 1998. Vol. 30, Issues 1–7. P. 119-130.

130. WebSPHINX: A Personal, Customizable Web Crawler. *Cs.cmu* : website. URL: http://www.cs.cmu.edu/~rcm/websphinx

131. Brin S., Page L. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer Networks*. 2012. Vol. 56, no. 18. P. 3825–3833.

132. A brief history of web crawlers / S. M. Mirtaheri et al. *Conference of the Center for Advanced Studies on Collaborative Research (University of Ottawa in collaboration with IBM)*: Proceedings, Ottawa, 2013. P. 40–54.

133. CoBWeb-a crawler for the Brazilian Web / A. S. da Silva et al. *6th International Symposium on String Processing and Information Retrieval. 5th International Workshop on Groupware*, Cancun, Mexico, 1999. P 184–191.

134. Heydon A., Najork M. Mercator: A scalable, extensible Web crawler. *World Wide Web*. 1999. Vol. 2, no. 4. P. 219–229.

135. Hunyadi L. Coordinating and controlling distributed web robots : Master's Thesis. Universitas Budapest University of Technology and Economics, Budapest, 2007. 87 p.

136. Zeinalipour-Yazti D., Dikaiakos M. Design and Implementation of a Distributed Crawler and Filtering Processor. In: Halevy, A., Gal, A. (eds) *Next Generation Information Technologies and Systems. NGITS 2002. Lecture Notes in Computer Science*, vol 2382. Berlin, Heidelberg : Springer, 2002. P. 58–74.

137. Risvik K.M., Michelsen R. Search engines and Web dynamics. *Computer Networks*. 2002. Vol. 39, no. 3. P. 289–302.

138. Castillo C., Baeza-Yates R. WIRE: an Open-Source Web Information Retrieval Environment. *Workshop on Open Source Web Information Retrieval (OSWIR)*. Compiegne, France, 2005. P. 27–30.

139. Jack P., Levitt N. Heritrix. *Webarchive.jira* : website. URL: https://archive.ph/OCKIh

140. Nutch Wiki. *Cwiki*: website. URL:
https://cwiki.apache.org/confluence/display/nutch

141. Pandey S. Web Scraping using Beautiful Soup. *Browserstack*: website. URL: https://www.browserstack.com/guide/web-scraping-using-beautiful-soup

142. Selenium vs. BeautifulSoup in 2023: Which Is Better? *Zenrows*: website. URL: https://www.zenrows.com/blog/selenium-vs-beautifulsoup#what-are-the-advantages-of-beautifulsoup

143. Faruque A. Python Web Scraping Tools: Advantages And Disadvantages Simply Explained. *Wscraper*: website. URL: https://wscraper.com/python-web-scraping-tools-advantages-and-disadvantages-simply-explained

144. Optimizing Crawler4j using MapReduce Programming Model / G. M. Siddesh et al. Journal of The Institution of Engineers (India): Series B. 2016. Vol. 98, no. 3. P. 329–336.

145. Wiegers K., Beatty J. Software Requirements. Developer Best Practices (3rd Edition). Redmond, WA, USA : Microsoft Press, 2013. 672 p.

146. The role of the requirements in software development. *ASD* : website. URL: https://asd.team/blog/the-role-of-the-requirements-in-software-development

147. Говорущенко Т.О., Поморова О.В. Моделювання процесу оцінювання достатності інформації щодо якості у специфікаціях вимог до програмного забезпечення. *Вісник Хмельницького національного університету. Серія «Технічні науки».* 2017. № 6. С. 70–80.

148. Говорущенко Т.О., Боднар М.А., Кушнір В.О. Сучасні проблеми формування та аналізу вимог до програмного забезпечення. *Вимірювальна та обчислювальна техніка в технологічних процесах*. 2019. № 1. С. 45–53.

149. Hovorushchenko T., Pavlova O., Bodnar M. Development of an Intelligent Agent for Analysis of Nonfunctional Characteristics in Specifications of Software Requirements. *Eastern-European Journal of Enterprise Technologies*. 2019. Vol. 1, no. 2 (97), P. 6–17.

150. Bell Ch., Kindahl M., Thalmann L. MySQL High Availability: Tools for Building Robust Data Centers (2nd Edition). Sebastopol, California, USA : O'Reilly Media, 2014. 760 p.

151. Daniel Nichter. Efficient MySQL Performance: Best Practices and Techniques (1st Edition). Sebastopol, California, USA : O'Reilly Media, 2022. 335 p.

152. MySQL, Cluster. *Wikipedia* : website. URL: https://asd.team/blog/the-role-of-the-requirements-in-software-development https://uk.wikipedia.org/wiki /MySQL_Cluster

153. *MySQL*: website. URL: https://www.mysql.com

154. Harold E.R. Java Network Programming (4th Edition). Sebastopol, California, USA : O'Reilly Media, 2014. 504 p.

155. Telang T. What is J2EE? *Educative*: website. URL: https://www.educative.io/answers/what-is-j2ee

156. Concept: Java 2 Platform Enterprise Edition (J2EE) Overview. *Swi* : website. URL:

https://swi.cs.vsb.cz/RUPLarge/tech.j2ee/guidances/concepts/java_2_platform_enterprise_edition_j2ee_overview_9A95BA45.html

157. Віртуальні машини. *Compbest* : website. URL:

https://compbest.com.ua/ua/virtualnye-mashiny

158. Cook J. Docker for Data Science. uilding Scalable and Extensible Data Infrastructure Around the Jupyter Notebook Server. Apress Berkeley, CA, 2017. 257 p.

159. Use containers to Build, Share and Run your applications. *Docker* : website. URL: https://www.docker.com/resources/what-container

160. Що таке Docker та технологія контейнерів Linux. *Блог VPS* : website. URL: https://vps.ua/blog/ukr/docker-and-linux-containers

161. Scanzani A. Java EE Clustering in Docker. *Medium*: website. URL: https://medium.com/digital-software-architecture/java-ee-clustering-in-docker-6fc35ac609c4

162. Пасічник В.В., Іванущак Н.М. Дослідження та моделювання складних мереж. *Східно-Європейський журнал передових технологій*. 2010. Том 2/3, № 44. С. 43–48.

163. Kleinberg J. M. Navigation in a small world. *Nature.* 2000. Vol. 406. P. 845.

164. Newman M. E. J., Strogatz S. H. and Watts D. J. Random graphs with arbitrary degree distributions and their applications. *The Structure and Dynamics of Networks*, Princeton: Princeton University Press, 2006, P. 269–285.

165. Пасічник В. В., Іванущак Н. М. Моделювання складних мереж. *Вісник Національного університету «Львівська політехніка» (Комп'ютерні науки та інформаційні технології)*. 2010. Вип. 672. С. 120–128.

166. Clustering Social Networks / N. Mishra et al. *Models for the Web-Graph, volume 4863 of Lecture Notes in Computer Science, chapter 5* / ed. by A. Bonato, F. R. K. Chung. Berlin, Heidelberg, 2007. P. 56–67.

167. Domingos P., Richardson M. Mining the network value of customers. *International conference on Knowledge discovery and data mining*: Proceedings of the seventh ACM SIGKDD international conference, New York, USA, 2001. NY, USA, 2001. p. 57–66.

168. Fortunato S. Community detection in graphs. *Physics Reports*. 2010. Vol. 486, no. 3–5. P. 75–174.

169. Qiu R. C., Antonik P. Smart Grid using Big Data Analytics: A Random Matrix Theory Approach. Wiley, 2017. 632 p.

170. von Luxburg U. A tutorial on spectral clustering. *Statistics and Computing*. 2007. Vol. 17, no. 4. P. 395–416.

171. Xiang T., Gong Sh. Spectral clustering with eigenvector selection. *Pattern Recognition*. 2008. Vol. 41, no. 3. P. 1012–1029

172. Ng A. Y., Jordan M., Weiss Y. On spectral clustering: Analysis and an algorithm. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. Cambridge, MA, USA : MIT Press, 2002. P. 849–856.

173. Kyrychenko O. Information technology for statistical cluster analysis of information in complex networks. *Computer Systems and Information Technologies*. 2022. No 4. P. 47–51.

174. Кириченко О. Л., Остапов С. Е. Статистичні характеристики деякої частини українського домену глобальної мережі WWW. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2012)* : тези доповідей Всеукр. наук.-практ. конф., м. Чернівці, 3–5 травня 2012 р. Чернівці : Золоті литаври, 2012. С. 93–94.

175. Кириченко О. Л., Kanovsky I. Y., Остапов С. Е. Дослідження статистики складних мереж на прикладі українських та ізраїльського доменів www-простору. *Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано- та мікроелектроніки* : матеріали ІІІ-ої Міжнар. наук.-практ. конф., м. Чернівці, 24–26 жовтня 2013 р. Чернівці, 2013. С. 125–126.

176. Кириченко О. Л., Остапов С. Е. Статистичні характеристики польського домену Інтернет edu.pl. *Інтелектуальні технології в системному програмуванні (ІТСП-2014)* : збірн. наук. праць ІІІ Всеукр. наук.-практ. конф. молодих учених та студентів, м. Хмельницький, 23–25 квітня 2014 р. Хмельницький : Гонта А.С., 2014. С. 269–270.

177. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Моделювання, дослідження та аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2014)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 27–30 травня 2014 р. Чернівці : Видавничий дім «Родовід», 2014. С. 174–176.

178. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Розробка інформаційної технології проведення статистично-кластерного аналізу інформації у складних мережах. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2019)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 03–06 жовтня 2019 р. Чернівці : ЧНУ, 2019. С. 92–94.

179. Кириченко О. Л., Остапов С. Е. Інформаційна технологія для проведення досліджень у складних мережах. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXVIII Міжнар. наук.-практ. конф. MicroCAD-2020 (м. Харків, 28–30 жовтня 2020 р.) : у 5 ч. Ч. І. / за ред. проф. Сокола Є.І. Харків : НТУ «ХПІ». С. 324.

180. de Nooy W., Mrvar A., Batagelj V. Exploratory Social Network Analysis with Pajek. Expanded edition. New York, USA : Cambridge University Press, 2011. 442 p.

181. Latora V., Marchiori M. Efficient behavior of small-world networks. *Phys. Rev. Lett*. 2001. Vol. 87, no. 19. Article 198701.

182. Снарский А. А., Ландэ Д. В. Моделирование сложных сетей : учебное пособие. Киев : НТУУ «КПИ», 2015. 212 с.

183. Takes F. W., Kosters W. A. Computing the Eccentricity Distribution of Large Graphs. *Algorithms*. 2013. Vol. 6, no. 1. P. 100–118.

184. Gyan R., Zhi-Li Zh. On random eccentricity in complex networks. Twin Cities, USA, 2011. 17 p. (Preprint. University of Minnesota; Preprint Technical Report).

185. Manoj B. S., Chakraborty A., Singh R. Complex Networks: A Networking and Signal Processing Perspective (1st Edition). New York : Pearson, 2018. 576 p.

186. Classification of scale-free networks / K.-I. Goh et al. *Proc. Natl. Acad. Sci. USA*. 2002. Vol. 99, no. 20. P. 12583–12588.

187. Scott J. Social Network Analysis: A Handbook (2nd ed.). London : Sage Publications, 2000. 224 p.

188. Estrada E., Rodríguez-Velázquez J.A. Subgraph centrality in complex networks. *Phys. Rev. E*. 2005. Vol. 71, no. 5. Article 056103.

189. Borgatti S. P., Everett M. G. A graph-theoretic perspective on centrality. *Soc. Netw*. 2006. Vol. 28, no. 4. P. 466–484.

190. Attack vulnerability of complex networks / P. Holme et al. *Phys. Rev. E*. 2002. Vol. 65, no. 5. Article 056109.

191. Latora V., Marchiori M. Economic small-world behavior in weighted networks. *Eur. Phys. J. B.* 2003. Vol. 32. P. 249–263.

192. Barrat A., Weigt M. On the properties of small-world network models. *The European Physical Journal B –Condensed Matter and Complex Systems.* 2000. Vol. 13. P. 547–560.

193. Ebel H., Davidsen J., Bornholdt S. Dynamics of Social Networks. *Complexity*. 2003. Vol. 8, no. 2. P. 24–27.

194. Newman M. E. J. Random Graphs with Clustering. *Physical review letters*. 2009. Vol. 103, no. 5. Article 058701.

195. Patel K. M. A., Thakral P. The best clustering algorithms in data mining. *International Conference on Communication and Signal Processing (ICCSP)*. Melmaruvathur, India, 2016. P. 2042–2046.

196. Ahalya G., Pandey H. M. Data clustering approaches survey and analysis. *International Conference on Futuristic Trends on Computational*

Analysis and Knowledge Management (ABLAZE). Greater Noida, India, 2015. P. 532–537.

197. Brandes U., Gaertler M., Wagner D. Experiments on Graph Clustering Algorithms. *ESA 2003. Lecture Notes in Computer Science, vol. 2832*/ ed. by G. Di Battista, U. Zwick. Berlin, Heidelberg : Springer, 2003. P. 568–579.

198. Benchmarking graph-based clustering algorithms / P. Foggia et al. *Image and Vision Computing.* 2009. Vol. 27, no. 7. P. 979–988

199. Kannan R., Vempala S., Vetta A. On Clusterings: Good, Bad and Spectral. *Journal of the ACM*. 2001. Vol. 51, no. 3. P. 497–515.

200. van Dongen S. Graph Clustering by Flow Simulation : PhD thesis. University of Utrecht, 2000. 173 p.

201. Harel D., Koren Y. On Clustering Using Random Walks. *Foundations of Software Technology and Theoretical Computer Science. FSTTCS 2001. Lecture Notes in Computer Science, vol. 2245* / ed. by R. Hariharan, V. Vinay, M. Mukund. Springer, Berlin, Heidelberg, 2001. P. 18–41.

202. Han J., Kamber M. Data Mining: Concepts and Techniques (2nd Edition). San Francisco, CA, USA : Morgan Kaufmann, 2006. 800 p.

203. A New Unsupervised Validation Index Model Suitable for Energy-Efficient Clustering Techniques / H. Abdulrazzak et al. *IEEE Access.* 2023. Vol. 11. P. 67540-67555.

204. Wang J., Su X. An improved K-Means clustering algorithm. *2011 IEEE 3rd International Conference on Communication Software and Networks*. Xi'an, China, 2011. P. 44–46.

205. A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm / Shi C. et al. *J Wireless Com Network*. 2021. Vol. 2021, no. 1. P. 16.

206. Indonesian provincial clustering using elbow method for the national food security during pandemic / R. Trimanto et al. *Bulletin of Applied Mathematics and Mathematics Education.* 2022. Vol. 2, no. 2. P. 51–58.

207. Streaming Algorithms for k-core Decomposition / A. E. Sariyuce et al. *39th International Conference on Very Large Data Bases* : Proceedings of the VLDB Endowment, Riva del Garda, Trento, 26–30 August 2013. Vol. 6, no. 6. P. 433–444.

208. Shutaywi M., Kachouie N. Silhouette analysis for performance evaluation in machine learning with applications to clustering. *Entropy*. 2021. Vol. 23, no. 6. P. 759.

209. An Analysis of the Application of Simplified Silhouette to the Evaluation of k-means Clustering Validity / H. Franco-Penya et al. *13th International Conference on Machine Learning and Data Mining MLDM 2017*. July 15–20, 2017, New York, USA.

210. GraphX: a resilient distributed graph system on spark / R. Xin et al. *First International Workshop on Graph Data Management Experiences and Systems (GRADES '13)*. New York, NY, USA : Association for Computing Machinery, 2013. P. 1–6.

211. Ramamonjison R. Apache Spark Graph Processing. Packt Publishing, 2015. 148 p.

212. Sankar K., Karau H. Fast Data Processing with Spark (2nd Edition). Packt Publishing, 2015. 184 p.

213. Baeza-Yates R., Castillo C. La Web Chilena 2004. *Tech. rep., Center for Web Research.* Chilie : University of Chile, 2005. 33 p.

214. Baeza-Yates R., Castillo C., López V. Características de la Web de España. *El profesional de la información*. 2006. Vol. 15, no. 1. P. 6–17.

215. Baeza-Yates R., Lalanne F. Characteristics of the Korean Web. *Informe técnico, Korea-Chile IT Cooperation Center ITCC*. 2004.

216. Efthimiadis E., Castillo C. Charting the Greek Web. *Researchgate.net* : website. URL:

https://www.researchgate.net/profile/Carlos-Castillo/publication/200111085_Charting_the_Greek_Web/links/00b7d5342acd2137e9000000/Charting-the-Greek-Web.pdf

217. GraphX – Spark 3.5.0 Documentation. *Spark.apache* : website. URL: https://spark.apache.org/docs/latest/graphx-programming-guide.html

218. MLlib: Main Guide. *Spark.apache* : website.URL: https://spark.apache.org/docs/latest/ml-guide.html

219. Chitra K., Maheswari D. A Comparative Study of Various Clustering Algorithms in Data Mining. *International Journal of Computer Science and Mobile Computing*. 2017. Vol. 6, no. 8. P. 109–115.

220. Schilling R.L. Brownian Motion: A Guide to Random Processes and Stochastic Calculus. Berlin, Boston : De Gruyter, 2021. 519 p.

221. Girardin V., Limnios N. Applied Probability: From Random Sequences to Stochastic. Springer, 2019. 273 p.

222. Murphy K.P. Machine Learning A Probabilistic Perspective. The MIT Press, 2012. 1104 p.

223. Couillet R., Liao Z. Random Matrix Methods for Machine Learning. Cambridge : Cambridge University Press, 2022. 408 p.

224. Adler M., van Moerbeke P. PDEs for the Joint Distributions of the Dyson, Airy and Sine Processes. *The Annals of Probability.* 2005. Vol. 33, no. 4. P. 1326–1361.

225. Wigner E.P. Characteristic Vectors of Bordered Matrices With Infinite Dimensions. *The Annals of Mathematics, Second Series*. 1955. Vol. 62, no. 3. P. 548–564

226. Tao T., Vu V., Krishnapur M. Random matrices: Universality of ESDs and the circular law. *Ann. Probab*. 2010. Vol. 38, no. 5. P. 2023–2065.

227. Nguyen H., O'Rourke S. The elliptic law. *International Mathematics Research Notices*. 2015. Vol. 2015, no. 17. P. 7620–7689.

228. Silverstein J.W., Bai Z.D. On the Empirical Distribution of Eigenvalues of a Class of Large Dimensional Random Matrices. *Journal of Multivariate Analysis*. 1995. Vol. 54, no 2. P. 175-192.

229. Privault N. Understanding Markov Chains: Examples and Applications. Singapore : Springer, 2018. 372 p.

230. Bapat R.B., Raghavan T.E.S. Nonnegative matrices and applications. Cambridge : Cambridge University Press, 1997. 356 p.

231. Wang J., Tan F. S., Yuan Y. Random Matrix Transformation and Its Application in Image Hiding. *Sensors*. 2023. 23. P. 1017.

232. Intrusion Monitoring Based on High Dimensional Random Matrix by Using Ultra-Weak Fiber Bragg Grating Array / Gu Hongcan, Huang Junbing et al. *Photonics*. 2023. Vol. 10, no. 7. P. 733.

233. Guo Shiqin, Xu Jilin. Art Financial Risk Prediction Algorithm Based on Random Matrix. *Mathematical Problems in Engineering*. 2022. P. 1–10.

234. Lan Jian. Extended Object Tracking Using Random Matrix With Extension-Dependent Measurement Numbers. *IEEE Transactions on Aerospace and Electronic Systems*. 2023. Vol. 59, no. 4. P. 4464–4477.

235. Generation of high-order random key matrix for Hill Cipher encryption using the modular multiplicative inverse of triangular matrices / Chen Yuehong, Xie Rong et al. *Wireless Networks*. 2023. P. 1–11. URL: https://doi.org/10.1007/s11276-023-03330-8

236. Кнігніцька Т.В., Малик І.В., Горбатенко М.Ю. Кластеризація: марковський алгоритм. *Буковинський математичний журнал*. 2020. Том 7, № 2. С. 59–75.

237. Kemeny J. G., Snell J. L., Finite Markov chains. — The University Series in Undergraduate Mathematics. Princeton : Van Nostrand, 1960.

238. Кириченко О. Л., Малик І. В., Остапов С. Е. Кластеризація великих даних на основі спектрального аналізу матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2020)* : праці IX Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2020 р. Чернівці : ЧНУ, 2020. С. 83–84.

239. Кириченко О. Л., Малик І. В., Остапов С. Е. Асимптотичний розподіл власних значень матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2021)* : праці X Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2021 р. Чернівці : ЧНУ, 2021. С. 22–24.

240. Kyrychenko O. L., Knignitska T. V., Ostapov S. E. Stochastic models in artificial intelligence development. *Modern stochastics: theory and applications V* : Materials of the Intern. Conf., Kyiv, Ukraine, June 1–4, 2021. Kyiv, 2021. P. 35.

241. Кириченко О.Л., Малик І.В., Остапов С.Е. Стохастичні моделі в задачах штучного інтелекту. *Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки*. 2021. № 2. С. 53–57.

242. Кириченко О. Л., Малик І. В., Остапов С. Е. Аналіз кластерної структури Інтернет-мереж на основі випадкових матриць. *Проблеми керування та інформатики.* 2022. №1. С. 37–46.

243. Tao T. Topics in Random Matrix Theory. American Mathematical Society, 2023. 282 p.

244. Chen Y., Sanghavi S., Xu H. Improved Graph Clustering. *IEEE Transactions on Information Theory*. 2014. Vol. 60, no. 10. P. 6440–6455.

245. Schaeffer S.E. Graph Clustering. *Computer Science Review*. 2007. Vol. 1, no. 1. P. 27–64.

246. Tsitsulin A., Palowitch J., Perozzi B., Müller E. Graph Clustering with Graph Neural Networks. *Journal of Machine Learning Research*. 2020. Vol. 24, no. 127. P. 1–21.

247. Kondruk N.E., Malyar M.M. Analysis of Cluster Structures by Different Similarity Measures. *Cybernetics and Systems Analysis*. 2021. Vol. 57. P. 436–441.

248. Robert C.P. The Bayesian Choice. New York : Springer, 2007. 602 p.

249. Taylor S. Clustering Financial Return Distributions Using the Fisher Information Metric. *Entropy*. 2019. Vol. 21, no. 2. P. 1–16.

250. High-Dimensional Data Bootstrap / V. Chernozhukov et al. *Annual Review of Statistics and Its Application*. 2023. Vol. 10. P. 427–449

251. Binhimd S., Kalantan Z. Bootstrap approach for clustering method with applications. *International Journal of Advanced and Applied Sciences*. 2023. Vol. 10, no. 3. P. 189–195.

252. Batool F., Hennig C. Clustering with the Average Silhouette Width. *Computational Statistics & Data Analysis*. 2021. Vol. 158. Article 107190.

ДОДАТКИ

ДОДАТОК А
ДІАГРАМА ОСНОВНИХ КЛАСІВ КРОУЛЕРА ТА ЇХ ОПИС

У додатку А зображено діаграму основних класів програми (кроулера), що описує інтерфейс, внутрішню структуру та взаємозв’язки основних компонентів системи. Наведемо опис основних класів та їх призначення.

Page – клас, який являє собою модель веб-сторінки, містить властивості що відображають основні характеристики сторінки та методи роботи з ними.

PageDAOLocal – інтерфейс, який містить набір базових операцій для роботи з класом Page, а саме: збереження, пошук, вилучення об’єктів класу Page в базі.

PageDAOImpl – реалізація інтерфейсу PageDAOLocal.

PageStatProperty – клас містить властивості, що відображають основні статистичні характеристики сторінки, які нас цікавлять, та методи роботи з ними.

PageStatPropertyDAOLocal – інтерфейс містить набір базових операцій для роботи з класом PageStatProperty, а саме: збереження, пошук, видалення об'єктів класу PageStatProperty в базі.

PageStatPropertyDAOImpl – реалізація інтерфейсу PageStatPropertyDAOLocal.

StatisticalService – інтерфейс являє собою набір бізнес-правил для розрахунку статистичних характеристик веб-сторінки.

StatisticalServiceImpl – реалізація інтерфейсу StatisticalService.

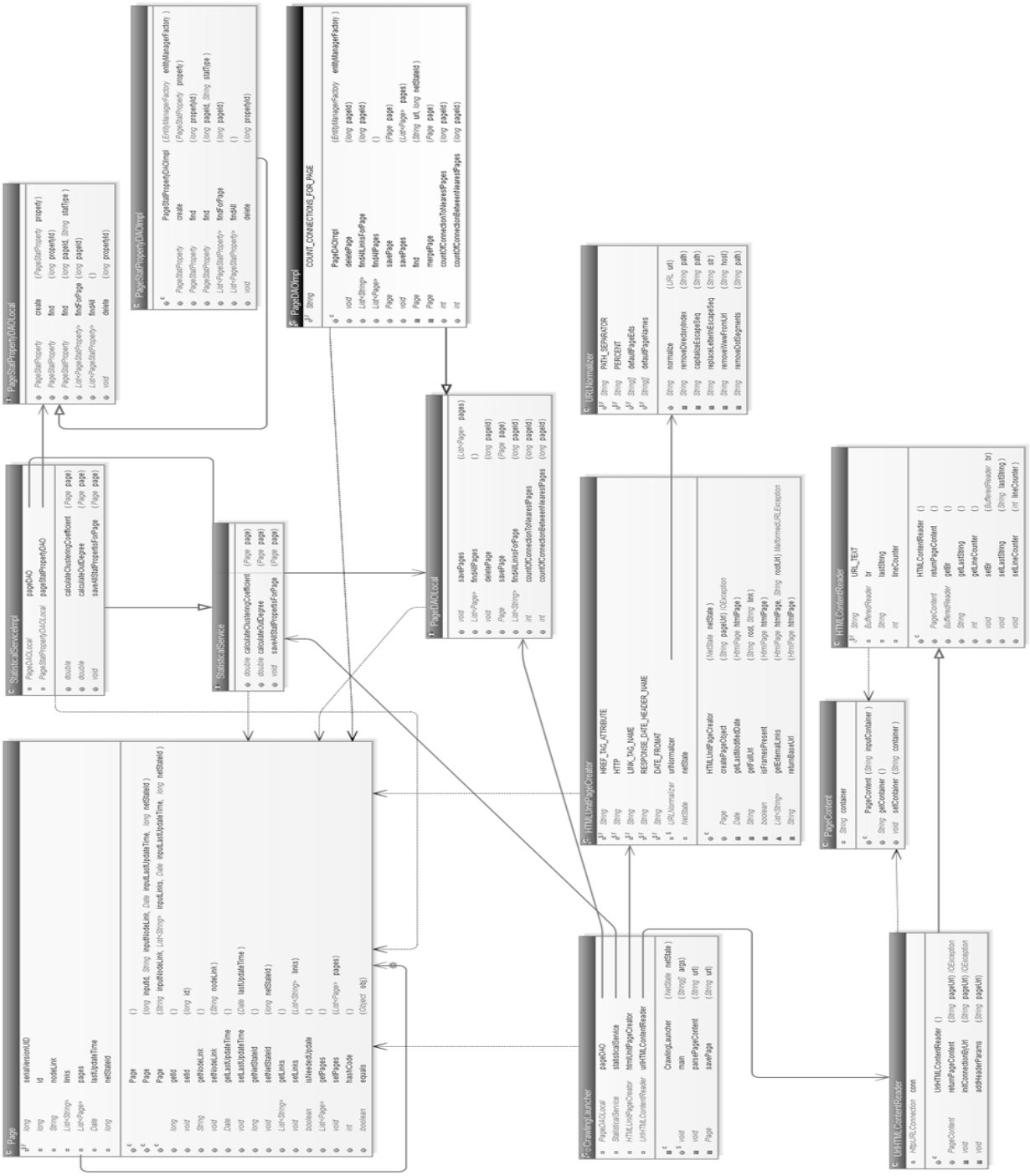
HTMLContentReader – клас, який представляє загальну логіку для читання та обробки вмісту веб-сторінки.

URLHTMLContentReader – клас, який успадковує HTMLContentReader і представляє логіку для читання та обробки вмісту веб-сторінки за допомогою класу URL.

PoiContent – клас, який містить контент веб-сторінки та функції для роботи з цим контентом.

URLNormalizer – допоміжний клас для обробки URL і приведення їх до стандартного вигляду.

HTMLUnitPageCreator – клас містить загальну логіку з обробки вмісту веб-сторінки, а також вилучення посилань.



ДОДАТОК Б

ЛІСТИНГ ЧАСТИНИ КОДУ ПРОГРАМИ

```
//вміст файлу PageCreator.java
public class PageCreator {

  private static final String A_TAG_PATTERN = "<[Aa].*?>";
  private static final String FORM_TAG_PATTERN = "<[fF][oO][rR][mM].*?>";

  private static final String URL_PATTERN = "((https?):(((//)|(\\\\\\\\))+[\\w\\d:#@%/;$()~_?\\+-=\\\\\\.&]*)";
  private static final String SHORT_URL_PATTERN = "(../../[\\w\\d:#@%/;$()~_?\\+-=\\\\\\.&]*)";
  private static final String HTML_A_HREF_TAG_PATTERN =
"\\s*(?i)href\\s*=\\s*(\"([^\"]*\")|'[^']*'|([^'\">\\s]+))";
  private static final String MODIFIED_DATE_PATTERN = "Last-Modified:.*?\\n";

  private List < PageContent > contents;
  private static URLNormalizer urlNormalizer = new URLNormalizer();
  private NetState netState;

  public PageCreator(NetState netState) {
    this.netState = netState;
  }

  public PageCreator(NetState netState, List < PageContent > inputContents) {
    this(netState);
    this.contents = inputContents;
  }

  public List < Page > createPageObjects() {
    List < Page > result = new ArrayList < Page > ();

    for (PageContent content: contents) {
      Page page = this.createPageObject(content);
      if (page != null) {
        result.add(page);
      }
    }

    return result;
  }

  public Page createPageObject(PageContent content) {
    String contentString = content.getContainer();

    String nodeLink = this.returnLinkString(contentString);
    List < String > externalLinks = new ArrayList < String > ();

    if (nodeLink != null) {
      String rootUrl = this.getRootUrl(nodeLink);
      externalLinks.addAll(this.returnLinkStrings(this.getMatcher(A_TAG_PATTERN, contentString), rootUrl));
      externalLinks
        .addAll(this.returnLinkStrings(this.getMatcher(FORM_TAG_PATTERN, contentString), rootUrl));
      Date modifiedDate = this.getLastModifiedDate(this.getMatcher(MODIFIED_DATE_PATTERN,
contentString));
      return new Page(nodeLink, externalLinks, modifiedDate, this.netState.getId());
    }
    return null;

  }
```

```
private List < String > returnLinkStrings(Matcher matcher, String rootUrl) {
  List < String > result = new ArrayList < String > ();

  while (matcher.find()) {
    String createdLink = this.returnLinkString(matcher.group());
    if (createdLink == null) {
      createdLink = this.returnLinkStringByNodeLink(matcher.group(), rootUrl);
    }
    if (createdLink != null) {
      result.add(createdLink);
    }
  }
  return result;
}

private Matcher getMatcher(String patternString, String content) {
  Pattern pattern = Pattern.compile(patternString);
  return pattern.matcher(content);
}

private String returnLinkString(String linkTag) {
  Pattern pattern = Pattern.compile(URL_PATTERN);
  Matcher matcher = pattern.matcher(linkTag);
  if (matcher.find()) {
    String result = null;
    try {
      result = urlNormalizer.normalize(new URL(matcher.group()));
    } catch (MalformedURLException e) {

    }
    return result;
  }
  return null;
}

private String returnLinkStringByNodeLink(String linkTag, String rootUrl) {
  Pattern pattern = Pattern.compile(SHORT_URL_PATTERN);
  Matcher matcher = pattern.matcher(linkTag);
  if (matcher.find()) {
    String urlString = matcher.group();
    int startPos = urlString.indexOf("../../") + 6;
    if (startPos != -1) {
      int endPos = urlString.length();
      String fullUrl = rootUrl + urlString.substring(startPos, endPos);
      try {
        fullUrl = urlNormalizer.normalize(new URL(fullUrl));
      } catch (MalformedURLException e) {}
      return fullUrl;
    }
  }
  return this.returnLinkToSelfNode(linkTag, rootUrl);
}

private String returnLinkToSelfNode(String linkTag, String rootUrl) {
  Pattern pattern = Pattern.compile(HTML_A_HREF_TAG_PATTERN);
  Matcher matcher = pattern.matcher(linkTag);
  if (matcher.find()) {
    String partOfUrl = matcher.group(1);
    if (!partOfUrl.contains("javascript")) {
      String fullUrl = rootUrl;
      if (partOfUrl.length() > 1) {
        System.out.println("Part of URL!!!!! " + partOfUrl);
```

```
          fullUrl = fullUrl + partOfUrl.substring(1, partOfUrl.length() - 1);
        }
        try {
          fullUrl = urlNormalizer.normalize(new URL(fullUrl));
        } catch (MalformedURLException e) {

        }
        return fullUrl;
      }
    }
    return null;
  }

  private String getRootUrl(String url) {
    String[] arr = url.split("/");
    String result = "";
    for (int i = 0; i < 3; i++) {
      if (i < arr.length) {
        result = result + arr[i] + "/";
      }
    }
    return result;
  }

  private Date getLastModifiedDate(Matcher matcher) {
    if (matcher.find()) {
      String fullString = matcher.group();
      if (fullString.indexOf("Last-Modified:") != -1) {
        String dateString = fullString.substring(15, fullString.length());
        DateFormat dateFormat = new SimpleDateFormat("EEE, dd MMM yyyy HH:mm:ss zzz", Locale.US);
        try {
          return dateFormat.parse(dateString);
        } catch (ParseException e) {
          return null;
        }
      }
    }
    return null;
  }
}

//вміст файлу MyCrawler.java
public class MyCrawler extends WebCrawler {

  private PageWrapper pageWrapper = new PageWrapper();

  private static Pattern filters = Pattern.compile(".*(\\.(css|js|bmp|gif|jpe?g" +
    "|png|tiff?|mid|mp2|mp3|mp4" +
    "|wav|avi|mov|mpeg|ram|m4v|pdf" +
    "|rm|smil|wmv|swf|wma|zip|rar|gz))$");

  /**
   * You should implement this function to specify
   * whether the given URL should be visited or not.
   */
  public boolean shouldVisit(WebURL url) {
    String href = url.getURL().toLowerCase();
    if (filters.matcher(href).matches()) {
      return false;
    }
    return true;
  }
```

```
  /**
   * This function is called when a page is fetched
   * and ready to be processed by your program
   */
  public void visit(Page page) {
    String nodeLink = page.getWebURL().getURL();
    List < WebURL > links = page.getURLs();
    List < String > externalLinks = new ArrayList < String > ();
    for (WebURL link: links) {
      externalLinks.add(link.getURL());
    }
    this.pageWrapper.processPage(nodeLink, externalLinks);
  }

}

//вміст файлу PageWrapper.java

public class PageWrapper {

  private static URLNormalizer urlNormalizer = new URLNormalizer();

  private static Pattern filters = Pattern.compile(".*(\\.(css|js|bmp|gif|jpe?g" +
    "|png|tiff?|mid|mp2|mp3|mp4" +
    "|wav|avi|mov|mpeg|ram|m4v|pdf" +
    "|rm|smil|wmv|swf|wma|zip|rar|gz))$");

  private PageDAOLocal pageDAO = DAOService.getPageDAO();
  private StatisticalService statisticalService = ServiceFactory.getStatisticalService();
  private NetStateDAOLocal netStateDAO = DAOService.getNetStateDAO();
  private NetState netState;

  public PageWrapper() {
    NetState netState = new NetState("crawler", new Date());
    netState.setId(28);
    this.netState = this.netStateDAO.create(netState);
  }

  public void processPage(String nodeLink, List < String > externalLinks) {
    Page newPage = new Page(this.normalizeUrl(nodeLink), this.normalizeUrls(externalLinks), null,
this.netState.getId());
    Page savedPage = this.pageDAO.savePage(newPage);
    if (savedPage != null) {
      if (savedPage.getPages() != null) {
        Set < String > links = new HashSet < String > ();
        for (Page page: savedPage.getPages()) {
          links.add(page.getNodeLink());
        }
      }
    }
  }

  private List < String > normalizeUrls(List < String > urls) {
    List < String > result = new ArrayList < String > ();
    for (String url: urls) {
      if (!filters.matcher(url).matches()) {
        result.add(this.normalizeUrl(url));
      }
    }
    return result;
  }
```

```
  private String normalizeUrl(String urlStr) {
    try {
      return urlNormalizer.normalize(new URL(urlStr));
    } catch (MalformedURLException e) {
      return "";
    }
  }
}

// вміст файлу URLNormalizer.java

public class URLNormalizer {

  public static final String PATH_SEPARATOR = "/";
  public static final String PERCENT = "%";
  public static final String[] defaultPageExts = {
    "asp",
    "aspx",
    "html",
    "htm",
    "shtml",
    "php"
  };
  public static final String[] defaultPageNames = {
    "default",
    "index"
  };

  public String normalize(URL url) {
    String protocol = url.getProtocol().toLowerCase();
    String host = removeWwwFromUrl(url.getHost().toLowerCase());
    String path = url.getPath();
    removeDirectoryIndex(url.getPath());
    path = capitalizeEscapeSeq(path);
    path = removeDotSegments(path);
    String query = (StringUtils.isNotBlank(url.getQuery())) ? "?" + url.getQuery() : "";

    return protocol + ":" + PATH_SEPARATOR + PATH_SEPARATOR + host + PATH_SEPARATOR + path +
query;
  }

  private String removeDirectoryIndex(String path) {
    String result = path;
    for (String pageExt: defaultPageExts) {
      for (String pageName: defaultPageNames) {
        String pageNameWithExt = pageName + "." + pageExt;
        if (StringUtils.contains(path, pageNameWithExt)) {
          result = StringUtils.remove(result, pageNameWithExt);
        }
      }
    }
    return result;
  }

  private String capitalizeEscapeSeq(String path) {
    List < String > resultParts = new ArrayList < String > ();
    String[] pathParts = path.split(PATH_SEPARATOR);
    for (String part: pathParts) {
      if (StringUtils.isNotBlank(part)) {
        if (StringUtils.contains(part, PERCENT)) {
          part = replaceLetterInEscapeSeq(part);
```

```
        }
        resultParts.add(part);
      }
    }
    String result = StringUtils.join(resultParts.iterator(), PATH_SEPARATOR);
    if (path.endsWith(PATH_SEPARATOR) && path.length() > 1) {
      result = result + PATH_SEPARATOR;
    }
    return result;
  }

  private String replaceLetterInEscapeSeq(String str) {
    List < String > resultStrs = new ArrayList < String > ();
    String[] strs = str.split(PERCENT);
    boolean isMoreThanFirst = false;
    for (String sword: strs) {
      if (isMoreThanFirst) {
        resultStrs.add(StringUtils.capitalize(sword));
      } else {
        resultStrs.add(sword);
      }
      isMoreThanFirst = true;
    }
    return StringUtils.join(resultStrs.iterator(), PERCENT);
  }

  private String removeWwwFromUrl(String host) {
    int index = host.indexOf("www.");
    if (index == 0) {
      return host.substring(index + 4);
    }

    return host;
  }

  private String removeDotSegments(String path) {
    if (StringUtils.isBlank(path) || (path.indexOf("/.") == -1)) {
      return path;
    }
    String[] inputSegments = path.split(PATH_SEPARATOR);
    List < String > outputSegments = new ArrayList < String > ();
    for (String segment: inputSegments) {
      if (!segment.equals(".") && !segment.equals("..")) {
        outputSegments.add(segment);
      }
    }
    return StringUtils.join(outputSegments.iterator(), PATH_SEPARATOR);
  }

}

//вміст файлу StatisticalServiceImpl.java
public class StatisticalServiceImpl implements StatisticalService {

  private PageDAOLocal pageDAO = DAOService.getPageDAO();
  private PageStatPropertyDAOLocal pageStatPropertyDAO = DAOService.getPageStatPropertyDAO();

  @Override
  public double calculateClusteringCoefficient(long pageId) {
    double countOfConnection = this.pageDAO.countOfConnectionBetweenNearestPages(pageId);
    double countOfNearestPages = this.pageDAO.countOfConnectionToNearestPages(pageId);
    if (countOfNearestPages <= 1) {
```

```
      return 0;
    }
    return 2 * countOfConnection / (countOfNearestPages * (countOfNearestPages - 1));
  }

  @Override
  public double calculateOutDegree(long pageId) {
    return this.pageDAO.countOfConnectionToNearestPages(pageId);
  }

  @Override
  public void saveAllStatPropertisForPage(long pageId) {
    PageStatProperty clusteringCoefficientProperty = new PageStatProperty(pageId,
      "clusteringCoefficient", StringUtils.EMPTY + this.calculateClusteringCoefficient(pageId));

    PageStatProperty outDegreeProperty = new PageStatProperty(pageId,
      "outDegree", StringUtils.EMPTY + this.calculateOutDegree(pageId));

    this.pageStatPropertyDAO.create(clusteringCoefficientProperty);
    this.pageStatPropertyDAO.create(outDegreeProperty);

  }

  @Override
  public void saveAllStatProperties(List < Long > pageIds) {
    EntityTransaction tx = this.pageStatPropertyDAO.getTransaction();
    tx.begin();
    for (Long pageId: pageIds) {
      PageStatProperty clusteringCoefficientProperty = new PageStatProperty(pageId,
        "clusteringCoefficient", StringUtils.EMPTY + this.calculateClusteringCoefficient(pageId));

      PageStatProperty outDegreeProperty = new PageStatProperty(pageId,
        "outDegree", StringUtils.EMPTY + this.calculateOutDegree(pageId));

      PageStatProperty inDegreeProperty = new PageStatProperty(pageId,
        "inDegree", StringUtils.EMPTY + this.pageDAO.countOfIncomingConnectionToPage(pageId));

      this.pageStatPropertyDAO.merge(clusteringCoefficientProperty);
      this.pageStatPropertyDAO.merge(outDegreeProperty);
      this.pageStatPropertyDAO.merge(inDegreeProperty);
    }
    this.pageStatPropertyDAO.flush();
    this.pageStatPropertyDAO.clean();
    tx.commit();
  }
}

// вміст файлу PicAlgo.java

public class PicAlgo {
  private static Logger logger = Logger.getLogger(PicAlgo.class.getName());

  private static String fileName = "Synthetic_15_ clusters";
  private static int clusterQuantity = 150;

  public static void runPic() throws IOException {
    String logFile = "D:" + File.separator + "Data" + File.separator + "pic_data" + File.separator + fileName + ".csv";
    SparkConf conf = new SparkConf().setAppName("Simple Application").setMaster("local[2]");
    JavaSparkContext sc = new JavaSparkContext(conf);

    JavaRDD < String > data = sc.textFile(logFile);
```

```
  JavaRDD < Tuple3 < Long, Long, Double >> similarities = data.map(
    line -> new UrlDataPreparator().prepareData(line.split(","))
  );

  PowerIterationClusteringModel model = new PowerIterationClustering().setK(clusterQuantity)
    .setMaxIterations(10)
    .run(similarities);

  Map < Integer, List < Long >> clusters = new HashMap < > ();

  List < PowerIterationClustering.Assignment > assignments = model.assignments().toJavaRDD().collect();
  for (PowerIterationClustering.Assignment assignment: assignments) {
    List < Long > ids = clusters.get(assignment.cluster());
    if (ids == null) {
      ids = new ArrayList();
    }
    ids.add(assignment.id());
    clusters.put(assignment.cluster(), ids);
  }

  Map < Integer, List < Long >> treeClusters = new TreeMap(clusters);

  writeToFile(treeClusters);
  writeClustersToFile(treeClusters);

  for (Map.Entry < Integer, List < Long >> entry: treeClusters.entrySet()) {
    if (entry.getValue().size() == 23) {
      loadIntoGraphDB(entry.getValue(), similarities.collect());
    }
  }

  printCenteresOfGraph(treeClusters.get(2));

  sc.stop();
}

private static void writeToFile(Map < Integer, List < Long >> treeClusters) throws IOException {
  String file = "D:" + File.separator + "Data" + File.separator + "pic_data" + File.separator + "output" +
    File.separator + fileName + "_output_" + clusterQuantity + ".csv";
  File dictionaryFile = new File(file);
  BufferedWriter writer = new BufferedWriter(new FileWriter(dictionaryFile));

  for (Map.Entry < Integer, List < Long >> entry: treeClusters.entrySet()) {
    for (Long id: entry.getValue()) {
      writer.write(id + "," + entry.getKey());
      writer.newLine();
    }
  }
  writer.close();
}

private static void writeClustersToFile(Map < Integer, List < Long >> treeClusters) throws IOException {
  String file = "D:" + File.separator + "Data" + File.separator + "pic_data" + File.separator + "output" +
    File.separator + fileName + "_clusters_output_" + clusterQuantity + ".csv";
  File dictionaryFile = new File(file);
  BufferedWriter writer = new BufferedWriter(new FileWriter(dictionaryFile));

  for (Map.Entry < Integer, List < Long >> entry: treeClusters.entrySet()) {
    writer.write(entry.getKey() + " -> " + entry.getValue().size());
    writer.newLine();
  }
  writer.close();
```

```
  }

  private static void loadIntoGraphDB(List < Long > nodeIds, List < Tuple3 < Long, Long, Double >> rawData) {
    logger.info("Nodes in cluster: " + nodeIds.size());
    for (Long nodeId: nodeIds) {
      for (Tuple3 < Long, Long, Double > tuple: rawData) {
        if (nodeId.equals(tuple._1()) || nodeId.equals(tuple._2())) {
          try {
            Neo4jRestHelper.addNodesToGraph(tuple._1(), tuple._2(), tuple._3());
          } catch (JSONException e) {
            e.printStackTrace();
          } catch (URISyntaxException e) {
            e.printStackTrace();
          }
        }
      }
    }
  }

  private static void printCenteresOfGraph(List < Long > nodeIds) {
    logger.info("Nodes in cluster: " + nodeIds.size());
    Map < Long, Integer > maxDistances = new HashMap < > ();
    for (Long nodeId: nodeIds) {
      Integer maxDistance = nodeIds.stream()
        .mapToInt(nestedNodeId -> nodeId.equals(nestedNodeId) ? 0 : Neo4jRestHelper.getDistance(nodeId,
nestedNodeId))
        .max()
        .getAsInt();
      maxDistances.put(nodeId, maxDistance);
    }
    Integer diameter = maxDistances.values().stream()
      .mapToInt(Integer::intValue)
      .min()
      .getAsInt();
    logger.info("Diameter: " + diameter);

    maxDistances.entrySet().stream()
      .filter(entry -> entry.getValue().equals(diameter))
      .forEach(entry -> logger.info("Centre: " + entry.getKey()));
  }

}

//вміст файлу Neo4jRestHelper.java
public class Neo4jRestHelper {

  private static final String SERVER_ROOT_URI = "http://localhost:7474/db/data/";

  public static void main(String[] args) throws JSONException, URISyntaxException {

    WebResource resource = getWebResource(SERVER_ROOT_URI);
    ClientResponse response = resource.get(ClientResponse.class);

    System.out.println(String.format("GET on [%s], status code [%d]",
      SERVER_ROOT_URI, response.getStatus()));
    response.close();

    for (Long nodeId: Arrays.asList(111 L, 112 L)) {
      URI nodeLocation = getNode(nodeId);
      if (nodeLocation == null) {
        nodeLocation = createNode();
        addLabelToNode(nodeLocation, nodeId);
```

```
    }
   }
  }

  /**
   * @param firstNode
   * @param secondNode
   * @return
   * @throws JSONException
   * @throws URISyntaxException
   */
  public static int getDistance(Long firstNode, Long secondNode) {
    try {
      //http://localhost:7474/db/data/node/0/path
      URI firstNodeLocation = getNode(firstNode);
      final String nodeEntryPointUri = firstNodeLocation + "/path";
      // POST JSON to the relationships URI
      ClientResponse response = makePostRequest(nodeEntryPointUri,
generateJsonGraphAlgorithm(getNode(secondNode)));
      JSONObject obj = new JSONObject(response.getEntity(String.class));
      if (obj.has("exception")) {
        return -1;
      }
      return obj.getInt("length");

    } catch (Exception e) {
      return -1;
    }
  }

  private static String generateJsonGraphAlgorithm(URI endNode) throws JSONException {
    JSONObject relationships = new JSONObject();
    relationships.put("type", "knows");
    relationships.put("direction", "out");

    JSONObject result = new JSONObject();
    result.put("to", endNode.toString());
    result.put("cost_property", "similarity");
    result.put("relationships", relationships);
    result.put("algorithm", "dijkstra");
    return result.toString();
  }

  /**
   * @param parentNode
   * @param childNode
   * @param similarity
   * @throws JSONException
   * @throws URISyntaxException
   */
  public static void addNodesToGraph(Long parentNode, Long childNode, Double similarity) throws
JSONException, URISyntaxException {
    List < URI > uries = new ArrayList(2);
    for (Long nodeId: Arrays.asList(parentNode, childNode)) {
      URI nodeLocation = getNode(nodeId);
      if (nodeLocation == null) {
        nodeLocation = createNode();
        addLabelToNode(nodeLocation, nodeId);
      }
      uries.add(nodeLocation);
    }
```

```
  addRelationship(uries.get(0), uries.get(1), similarity);
}

/**
 * @param nodeId
 * @return
 * @throws JSONException
 * @throws URISyntaxException
 */
public static URI getNode(Long nodeId) throws JSONException, URISyntaxException {
  //http://localhost:7474/db/data/label/115/nodes
  final String uri = SERVER_ROOT_URI + "label/" + nodeId + "/nodes";
  ClientResponse response = makeGetRequest(uri);
  JSONArray array = new JSONArray(response.getEntity(String.class));
  response.close();
  if (array.length() == 0) {
    return null;
  }
  return new URI(((JSONObject) array.get(0)).getString("self"));
}

private static URI createNode() {
  // http://localhost:7474/db/data/node
  final String nodeEntryPointUri = SERVER_ROOT_URI + "node";

  // POST {} to the node entry point URI
  ClientResponse response = makePostRequest(nodeEntryPointUri, "{}");

  final URI location = response.getLocation();
  response.close();
  return location;
}

private static void addLabelToNode(URI location, Long nodeId) {
  //http://localhost:7474/db/data/node/10251/labels
  final String nodeEntryPointUri = location + "/labels";

  // POST {} to the node entry point URI
  ClientResponse response = makePostRequest(nodeEntryPointUri, "\"" + nodeId + "\"");
  response.close();
}

private static URI addRelationship(URI startNode, URI endNode, Double similarity)
throws URISyntaxException, JSONException {
  URI fromUri = new URI(startNode.toString() + "/relationships");
  String relationshipJson = generateJsonRelationship(endNode, similarity);

  // POST JSON to the relationships URI
  ClientResponse response = makePostRequest(fromUri.toString(), relationshipJson);

  final URI location = response.getLocation();
  System.out.println(String.format(
    "POST to [%s], status code [%d], location header [%s]",
    fromUri, response.getStatus(), location.toString()));

  response.close();
  return location;
}

private static String generateJsonRelationship(URI endNode, Double similarity) throws JSONException {
  JSONObject similarityObj = new JSONObject();
  similarityObj.put("similarity", similarity);
```

```
    JSONObject result = new JSONObject();
    result.put("to", endNode.toString());
    result.put("type", "knows");
    result.put("data", similarityObj);
    return result.toString();
  }

  private static ClientResponse makePostRequest(String entryPointUri, String entity) {
    return getWebResource(entryPointUri).accept(MediaType.APPLICATION_JSON)
      .type(MediaType.APPLICATION_JSON)
      .entity(entity)
      .post(ClientResponse.class);
  }

  private static ClientResponse makeGetRequest(String entryPointUri) {
    return getWebResource(entryPointUri).accept(MediaType.APPLICATION_JSON)
      .type(MediaType.APPLICATION_JSON)
      .get(ClientResponse.class);
  }

  private static WebResource getWebResource(final String entryPointUri) {
    Client client = Client.create();
    client.addFilter(new HTTPBasicAuthFilter("neo4j"));
    return client.resource(entryPointUri);
  }

}
```

ДОДАТОК В
АКТИ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

QLICKS

Bezoekadres
Koos Postemalaan 2 (Gateway C) Mediapark
1217 ZC Hilversum Nederland

Contactinformatie
+31 35 677 6666
info@qlicks.nl

Act

implementation of the results of the dissertation work
Oksana Leonidivna Kyrychenko
on the topic "Research of statistical characteristics of complex networks methods of intelligent data analysis", submitted to the defense for obtaining a scientific degree Doctor of Philosophy
in the field of knowledge 12 - "Information technologies"
in specialty 121 - "Software engineering"

This act to confirm that Qlicks B.V. uses the results implemented in the dissertation research of Oksana Kyrychenko, a Ph.D. candidate of the Yuriy Fedkovich Chernivtsi National, in particular some functionality have been developed on the basis of the proposed algorithm in the software product:

- customers are categorized into different segments, which are then effectively used for personalized marketing campaigns and strategies;

- customer behavior is predicted based on the analysis of purchase history, search history or profiles in social networks.

The main advantage of the algorithm developed by the acquirer is the high quality of the obtained results and the low cost of computing resources.

30.10.2023  A Gielkens (Qlicks B.V.)

Акт
впровадження результатів дисертаційної роботи
Кириченко Оксани Леонідівни
на тему «Дослідження статистичних характеристик складних мереж
методами інтелектуального аналізу даних»,
представленої до захисту на здобуття наукового ступеня
доктора філософії з галузі знань 12 – «Інформаційні технології»
за спеціальністю 121 – «Інженерія програмного забезпечення»

Цей акт складено на підтвердження того, що ТОВ «Квант Азимут» використовує у своїй діяльності результати, які реалізовано в дисертаційному дослідженні здобувача ступеня доктора філософії Чернівецького національного університету імені Юрія Федьковича Кириченко Оксани Леонідівни, а саме:

- запропонований підхід до формування архітектури аналітичного модуля дозволяє створювати масштабоване і гнучке спеціалізоване програмне забезпечення, що підтримує контейнеризацію, роботу в багатопотоковому режимі, з високою здатністю до розширення та легко інтегрується в різноманітні власні програмні продукти.

Використання зазначеного підходу дозволило покращити ефективність та точність аналітичних процесів у наших розробках.

Директор ТОВ “Квант Азимут” Валь О. Д.

«Затверджую»
Директор Навчально-наукового інституту фізико-технічних та комп'ютерних наук Чернівецького національного університету імені Юрія Федьковича
д.ф.-м.н., професор

_____ О.В. Ангельський
«9» жовтня 2023 р.

АКТ

впровадження результатів дисертаційної роботи Кириченко Оксани Леонідівни «Дослідження статистичних характеристик складних мереж методами інтелектуального аналізу даних» до захисту на здобуття наукового ступеня доктора філософії за спеціальністю «Інженерія програмного забезпечення» в навчальному процесі

Комісія Чернівецького національного університету імені Юрія Федьковича в складі:

заступник директора ННІФТКН з наукової роботи, д.ф.-м.н., професор Дуболазов О.В.;

завідувача кафедри математичних проблем управління і кібернетики, д.ф.-м.н., професора Дріня Я.М.;

завідувача кафедри програмного забезпечення комп'ютерних систем, д.ф.-м.н., професора Остапова С.Е.

склали цей акт про те, що результати дисертаційної роботи здобувача наукового ступеня доктора філософії за спеціальністю «Інженерія програмного забезпечення» Кириченко О.Л. впроваджені в навчальний

процес кафедр МПУіК та ПЗКС, зокрема, в дисциплінах «Технології машинного навчання», «Методи та засоби кластерного аналізу», «Проектування інформаційних систем».

При викладанні цих дисциплін використовуються наступні матеріали досліджень, які отримані в дисертаційній роботі:

- особливості процесу збору інформації в мережі;
- розробка спеціалізованого програмного забезпечення для збору інформації в мережі (кроулера);
- застосування алгоритмів кластеризації до обробки великих даних;
- обчислення статистичних характеристик мережі;
- використання розробленого спектрального методу оцінки оптимальної кількості кластерів у задачах кластеризації на графах;
- тестування різних методів знаходження оптимальної кількості кластерів на основі моделювання методом Монте - Карло;
- моделювання мережі зв'язків, що розподілені за законом Пуассона.

Використання зазначених результатів дозволило підвищити якість навчального процесу із згаданих дисциплін.

Заступник директора ННІФТКН з
наукової роботи д.ф.-м.н., професор О.В. Дуболазов

Завідувач кафедри МПУіК,
д.ф.-м.н., професор Я.М. Дрінь

Завідувач кафедри ПЗКС,
д.ф.-м.н., професора С.Е. Остапов

ДОДАТОК Г

СПИСОК ПУБЛІКАЦІЙ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

Наукові праці у періодичних наукових виданнях, проіндексованих у наукометричних базах даних Scopus:

1. Kyrychenko O., Ostapov S., Kanovsky I. Investigation of the certain internet domain statistical characteristics / Статистичні характеристики деяких зон інтернету та їх дослідження. *Eastern-European Journal of Enterprise Technologies*. 2013. Vol. 6, no. 12(66). P. 91–96. (Scopus)

Наукові праці у виданнях, включених до переліку наукових фахових видань України:

2. Кириченко О.Л., Малик І.В., Остапов С.Е. Стохастичні моделі в задачах штучного інтелекту. *Вісник Київського національного університету імені Тараса Шевченка. Серія фізико-математичні науки*. 2021. № 2. С. 53–57.
3. Kyrychenko O. Information technology for statistical cluster analysis of information in complex networks. *Computer Systems and Information Technologies*. 2022. No 4. P. 47–51.
4. Кириченко О. Особливості архітектури програмного забезпечення для збору та аналізу статистичної інформації в глобальній мережі. *Information Technology: Computer Science, Software Engineering and Cyber Security*. 2023. № 2. С. 107–112.

Наукові праці, які додатково відображають наукові результати дисертації:

5. Kyrychenko O.**,** Ostapov S., Kanovsky I. Comparison of Statistical Characteristics of Certain Internet Subdomains. Chapter 1. Monograph. *Internet in the information society. Insights on the information systems, structures and applications* / ed. by M. Rostanski, P. Pikiewicz. Scientific Publishing of the Academy of Business in Dabrowa Gornicza : Wydawnictwo Naukowe, 2014. 138 p. ISBN: 978-83-62897-91-9.

6. Кириченко О.Л., Малик І.В., Остапов С.Е. Аналіз кластерної структури Інтернет-мереж на основі випадкових матриць. *Проблеми керування та інформатики.* 2022. №1. С. 37–46.
7. Кириченко О. Л., Kanovsky I., Остапов С. Е. Програмне забезпечення для дослідження статистичних характеристик глобальної мережі WWW. *Системи обробки інформації*. 2013. Том 2. Вип. 3. С. 99–104.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

8. Кириченко О. Л., Остапов С. Е. Статистичні характеристики деякої частини українського домену глобальної мережі WWW. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2012)* : тези доповідей Всеукр. наук.-практ. конф., м. Чернівці, 3–5 травня 2012 р. Чернівці : Золоті литаври, 2012. С. 93–94.
9. Кириченко О. Л., Остапов С. Е., Kanovsky I. Статистичні характеристики українських доменів edu.ua, net.ua глобальної www-мережі та їх дослідження. *Актуальні проблеми інформаційних технологій, економіки та права (ІТЕП–2013)* : тези доповідей Міжнар. наук.-практ. конф., м. Чернівці, 3–5 квітня 2013 р. Чернівці : Книги-ХХІ, 2013. С.84–85.
10. Кириченко О. Л., Kanovsky I., Остапов С. Е. Складні мережі та їх статистичні характеристики: аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2013)* : тези доповідей Всеукр. наук.-практ. конф., м. Чернівці, 27–31 травня 2013 р. Чернівці : Видавничий дім «Родовід», 2013. С. 16–22.
11. Кириченко О. Л., Kanovsky I. Y., Остапов С. Е. Дослідження статистики складних мереж на прикладі українських та ізраїльського доменів www-простору. *Фізико-технологічні проблеми радіотехнічних пристроїв, засобів телекомунікацій, нано- та мікроелектроніки* : матеріали ІІІ-ої Міжнар. наук.-практ. конф., м. Чернівці, 24–26 жовтня 2013 р. Чернівці, 2013. С. 125–126.

12. Кириченко О. Л., Остапов С. Е. Статистичні характеристики польського домену Інтернет edu.pl. *Інтелектуальні технології в системному програмуванні (ІТСП-2014)* : збірн. наук. праць ІІІ Всеукр. наук.-практ. конф. молодих учених та студентів, м. Хмельницький, 23–25 квітня 2014 р. Хмельницький : Гонта А.С., 2014. С. 269–270.
13. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Моделювання, дослідження та аналіз деяких сегментів web-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2014)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 27–30 травня 2014 р. Чернівці : Видавничий дім «Родовід», 2014. С. 174–176.
14. Кириченко О. Л., Остапов С. Е. Дослідження структури Веб-простору за допомогою кластерного аналізу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2016)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 21–24 травня 2016 р. Чернівці : Видавничий дім «Родовід», 2016. С. 112–114.
15. Кириченко О.Л., ОстаповС.Е., Кановський І.Я. Проведення оптимальної кластеризації структури веб-простору. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2017)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 05–08 жовтня 2017 р. Чернівці : Видавничий дім «Родовід», 2017. С. 68–70.
16. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Проведення оптимальної кластеризації структури деяких зон веб-простору за допомогою методу K-Core Decomposition. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2018)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 11–14 жовтня 2018 р. Чернівці : Видавничий дім «Родовід», 2018. С. 48–50.
17. Кириченко О. Л., Остапов С. Е. Застосування методу k-Core Decomposition для проведення оптимальної кластеризації. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXVII Міжнар. наук.-практ. конф. MicroCAD-2019 (м. Харків, 15–17

травня 2019 р.) : у 4 ч. Ч. IV. / за ред. проф. Сокола Є. І. Харків : НТУ «ХПІ». С. 155.

18. Кириченко О. Л., Остапов С. Е., Кановський І. Я. Розробка інформаційної технології проведення статистично-кластерного аналізу інформації у складних мережах. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2019)* : праці Міжнар. наук.-практ. конф., м. Чернівці, 03–06 жовтня 2019 р. Чернівці : ЧНУ, 2019. С. 92–94.
19. Кириченко О. Л., Остапов С. Е. Інформаційна технологія для проведення досліджень у складних мережах. *Інформаційні технології: наука, техніка, технологія, освіта, здоров'я* : тези доп. XXVIII Міжнар. наук.-практ. конф. MicroCAD-2020 (м. Харків, 28–30 жовтня 2020 р.) : у 5 ч. Ч. І. / за ред. проф. Сокола Є.І. Харків : НТУ «ХПІ». С. 324.
20. Кириченко О. Л., Малик І. В., Остапов С. Е. Кластеризація великих даних на основі спектрального аналізу матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2020)* : праці IX Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2020 р. Чернівці : ЧНУ, 2020. С. 83–84.
21. Кириченко О. Л., Малик І. В., Остапов С. Е. Асимптотичний розподіл власних значень матриці переходу. *Проблеми інформатики та комп'ютерної техніки (ПІКТ–2021)* : праці X Міжнар. наук.-практ. конф., м. Чернівці, 28–31 жовтня 2021 р. Чернівці : ЧНУ, 2021. С. 22–24.
22. Kyrychenko O. L., Knignitska T. V., Ostapov S. E. Stochastic models in artificial intelligence development. *Modern stochastics: theory and applications V* : Materials of the Intern. Conf., Kyiv, Ukraine, June 1–4, 2021. Kyiv, 2021. P. 35.