Preetha Mathew[1], Alphy Jose[2]

Leveraging Graph Theory for Advancements in Software Engineering: A Bibliometric Mapping of Research Trends and Themes.

Abstract: **-** This bibliometric study explores the role of graph theory in advancing software engineering by analyzing its applications in modeling, optimization, and system analysis. Utilizing Scopus as the bibliographic database, the study examines 1,597 documents published between 1976 and 2023. Tools like Biblioshiny and VOSviewer were employed to analyze key metrics, including annual scientific production, contributions of leading authors, and the most influential publication sources. The study highlights the dominance of the United States in global research, with significant contributions from China, Germany, and India. It also identifies trending topics such as program slicing, model checking, and graph transformation, showcasing their foundational and emerging significance. Thematic mapping categorizes research themes into motor, basic, niche, and emerging categories, reflecting their development and centrality. Bibliographic coupling and keyword co-occurrence networks reveal the diversity of research areas and interdisciplinary connections. Co-authorship analysis underscores the importance of global collaboration in advancing the field. Practical implications include leveraging graph-based models to address scalability and optimization challenges in software engineering. This analysis provides a comprehensive understanding of the research landscape and highlights opportunities for future innovation in the domain.

.***Keywords:*** Graph Theory, Software Engineering, Bibliometric Analysis, Biblioshiny, VOSviewer

1. INTRODUCTION

Graph theory, a branch of mathematics, plays a pivotal role in addressing complex problems across numerous disciplines, including software engineering [1], [2]. At its core, graph theory deals with the study of graphs, which are mathematical structures consisting of nodes (or vertices) connected by edges [3]. These graphs can model a wide array of relationships and structures, making them indispensable tools for solving real-world problems [4]. In software engineering, the versatility of graph theory has led to its application in areas such as system modeling, dependency analysis, and optimization tasks, demonstrating its relevance in the design, development, and maintenance of software systems [5].

In the realm of software design, graphs are often used to represent relationships among different components or modules [2], [6] For instance, directed acyclic graphs (DAGs) model dependencies in software projects, enabling effective management of build processes and understanding of module interactions. Similarly, control flow graphs (CFGs) are employed in program analysis to visualize the flow of execution in a program, aiding developers in debugging, performance optimization, and ensuring code quality [4]. The ability of graphs to abstract and simplify these relationships significantly enhances the efficiency and clarity of software development processes [7].

Graph theory also finds extensive use in algorithm design and optimization within software engineering [8]. Algorithms like Dijkstra's for shortest path, Kruskal's for minimum spanning tree, and Ford-Fulkerson for maximum flow are grounded in graph theory and are critical for solving problems in network analysis, database query optimization, and resource allocation [9]. Moreover, graph-based approaches are used to model and solve

[1]Department of Mathematics, Baselius College, Kottayam, Kerala, India

[2]Department of Mathematics, Little Flower College, Guruvayoor, Kerala, India

scheduling problems in software project management, ensuring optimal resource utilization and timely delivery of projects [10], [11].

The importance of graph theory extends to emerging areas like software testing and maintenance, where graphs are used to model test case dependencies, program slicing, and version control [12]. In software testing, graphs assist in identifying redundant test cases, optimizing test coverage, and detecting potential defects [13]. Additionally, graph-based models are employed in social network analysis, recommendation systems, and data mining, showcasing the interdisciplinary impact of graph theory in software engineering and beyond [14]. As software systems grow in complexity, the role of graph theory continues to expand, offering innovative solutions to modern engineering challenges [2].

Graph theory has profoundly influenced software engineering by providing tools to model and solve intricate problems in system design, analysis, and optimization [1], [3]. The application of graph theory in software engineering spans diverse domains, including dependency management, algorithm development, and system architecture [15]. Given its multidisciplinary impact and the increasing volume of scholarly output in this field, a bibliometric analysis is instrumental in exploring the trends, contributions, and collaborations that have shaped research on graph theory in software engineering.

Bibliometric analysis, facilitated by tools like Biblioshiny and VOSviewer, enables researchers to systematically examine large datasets of academic publications [16], [17], [18]. Biblioshiny, an interface for R's bibliometrix package, offers advanced visualization and statistical analysis capabilities [19], [20], [21], while VOSviewer specializes in network-based visualizations, including co-citation, co-authorship, and keyword co-occurrence networks [22], [23], [24]. By employing these tools, researchers can identify prolific authors, influential journals, and core research themes, as well as assess the evolution of the field over time. Such insights are valuable for understanding the academic landscape and guiding future research directions.

This study utilizes Biblioshiny and VOSviewer to conduct a comprehensive bibliometric analysis of graph theory applications in software engineering. The analysis highlights the temporal trends in publications, the geographical distribution of research contributions, and the collaborative networks that underpin advancements in the field. Moreover, keyword co-occurrence and thematic mapping offer a deeper understanding of the primary areas of focus, such as software testing, system modeling, and algorithmic optimization. By shedding light on the intellectual structure and emerging trends, this bibliometric analysis contributes to the strategic development of research in graph theory and its applications in software engineering.

2. MATERIALS AND METHODS

The primary source of bibliographic data for this study is Scopus, chosen for its inclusion of a broader array of high-quality journals compared to other databases [25], [26]. We retrieved publications using the query (TITLE-ABS-KEY ("Graph Theory") AND TITLE-ABS-KEY ("Software Engineering")). The search was not restricted to any particular language, and the data included articles from peer-reviewed journals, book chapters, and conference papers. We collected 961 articles from 573 different sources, spanning 1981 to 2023. To ensure accuracy, we screened the Scopus records to remove any duplicates. The results were saved as a "CSV" file, and we performed bibliometric analysis on the data using VOSviewer and Bibloshiny software.

3. RESULTS AND FINDINGS

3.1. Main Information of the investigation

Table 1 summarizes the main information of a bibliometric investigation of graph theory in software engineering, covering the timespan from 1976 to 2023. A total of 1,597 documents were analyzed, sourced from 655 different journals, books, and conference proceedings. The field has experienced a steady annual growth rate of 3.89%, with an average document age of 18.1 years, indicating both the historical depth and continued relevance of the topic. Each document garners an average of 19.69 citations, reflecting its academic impact, supported by 27,788 references, showcasing the extensive research engagement.Regarding document content, the analysis identified 8,000 Keywords Plus and 3,007 Author's Keywords, highlighting the diverse and specialized research themes. The study involves 3,586 authors, with 205 contributing to single-authored documents. Collaboration is a hallmark

of this field, with an average of 2.75 co-authors per document and 15.4% of publications involving international co-authorships, showcasing a global and cooperative research environment.In terms of document types, conference papers dominate with 1,157 entries, highlighting the dynamic and fast-evolving nature of the field, followed by 436 journal articles, which provide in-depth studies. Book chapters, however, are minimal, with only 4 contributions. This data underscores the prominence of conferences in shaping research and the interdisciplinary appeal of graph theory applications in software engineering.

Table 1. Main Information of the Investigation

Description	Results
MAIN INFORMATION ABOUT DATA	
Timespan	1976:2023
Sources (Journals, Books, etc)	655
Documents	1597
Annual Growth Rate %	3.89
Document Average Age	18.1
Average citations per doc	19.69
References	27788
DOCUMENT CONTENTS	
Keywords Plus (ID)	8000
Author's Keywords (DE)	3007
AUTHORS	
Authors	3586
Authors of single-authored docs	205
AUTHORS COLLABORATION	
Single-authored docs	234
Co-Authors per Doc	2.75
International co-authorships %	15.4
DOCUMENT TYPES	
article	436
book chapter	4
conference paper	1157

3.2. Annual Scientific Production

Figure 1 presents the annual scientific production in graph theory in software engineering, which reflects a progressive increase over the decades, with noticeable fluctuations. In the early years (1976–1990), research activity was sparse, with fewer than 10 articles per year and several years with no publications. A significant growth phase began in the early 1990s, marked by a gradual rise from 15 articles in 1992 to 79 articles in 2003.he most prolific period occurred from 2004 to 2008, with peak output in 2004 at 135 articles, followed by consistently high publication rates of over 100 articles annually until 2006. This surge highlights heightened interest and advancements in applying graph theory to software engineering. However, after 2008, the publication rates displayed a fluctuating trend, stabilizing around 30–50 articles annually for much of the 2010s. Recent years (2021–2024) show a decline in annual scientific production, with 2021 reporting only 17 articles and 2022 marking a significant low at 6 articles. This could indicate a shift in research focus or saturation in specific subfields. Nonetheless, the gradual recovery in 2023 and 2024, with 12 articles each year, suggests renewed interest or exploration of emerging applications. Overall, the data illustrates the field's dynamic growth and evolving research trends.

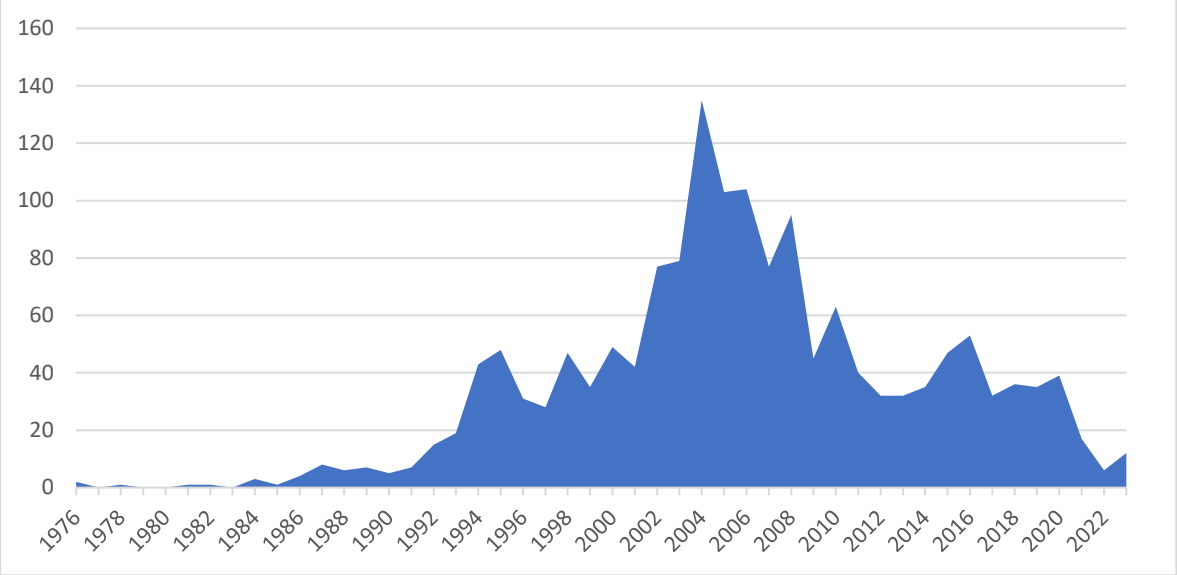

Figure 1. Annual Scientific Production

3.3. Most Relevant Authors

Table 2 presents the most relevant authors contributing to research on graph theory in software engineering, ranked by the number of articles published. Reiko Heckel and Gabriele Taentzer lead the list, each with 27 publications, indicating their significant influence and consistent contributions to the field. Hartmut Ehrig follows with 19 articles, while Zelimir F. Kurtanjek and Andy Schürr have 17 and 15 articles, respectively, demonstrating their prominent roles in advancing this area of research. Saket Saurabh (11 articles), Claudia Ermel (10 articles), and Bernhard Westfechtel (10 articles) also emerge as noteworthy contributors, showcasing their active participation in developing and disseminating research on graph theory applications. Luciano Baresi and Gregor Engels, with 8 publications each, round out the list, reflecting their valuable inputs to the academic discourse. Overall, this group of authors represents the core intellectual drivers of the field, whose work has likely shaped critical theories, methodologies, and applications of graph theory in software engineering.

Table 2. Most Relevant Authors

Authors	Articles
HECKEL, REIKO	27
TAENTZER, GABRIELE	27
EHRIG, HARTMUT	19
KURTANJEK, ZELIMIR F.	17
SCHÜRR, ANDY	15
SAURABH, SAKET	11
ERMEL, CLAUDIA	10
WESTFECHTEL, BERNHARD	10
BARESI, LUCIANO	8
ENGELS, GREGOR	8

3.4. Most Relevant Sources

Table 3 highlights the most relevant sources publishing research on graph theory in software engineering. The Lecture Notes in Computer Science (LNCS), including its subseries, is the most prolific source, contributing 213 articles, demonstrating its critical role in disseminating research in this domain. The Leibniz International Proceedings in Informatics (LIPICS) follows with 86 publications, indicating its importance in publishing

significant conference proceedings in theoretical and applied computer science. The Electronic Notes in Theoretical Computer Science (63 articles) and the Proceedings of the International Conference on Software Engineering (48 articles) further emphasize the strong connection between theoretical computer science and software engineering applications. Other notable sources include the IEEE Transactions on Software Engineering (21 articles) and the Proceedings of the Asia-Pacific Software Engineering Conference (APSEC) (19 articles), which focus on advancements in software engineering methodologies and practices. Lastly, sources like the ACM International Conference Proceeding Series, CEUR Workshop Proceedings, Information and Software Technology, and Reverse Engineering - Working Conference Proceedings, each with 15 articles, underline their contributions to niche and emerging areas within the field. Collectively, these sources represent a mix of journals, conference proceedings, and workshops that have significantly influenced the development and dissemination of graph theory research in software engineering.

Table 3. Most Relevant Sources

Sources	**Articles**
LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS)	213
LEIBNIZ INTERNATIONAL PROCEEDINGS IN INFORMATICS, LIPICS	86
ELECTRONIC NOTES IN THEORETICAL COMPUTER SCIENCE	63
PROCEEDINGS - INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING	48
IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	21
PROCEEDINGS - ASIA-PACIFIC SOFTWARE ENGINEERING CONFERENCE, APSEC	19
ACM INTERNATIONAL CONFERENCE PROCEEDING SERIES	15
CEUR WORKSHOP PROCEEDINGS	15
INFORMATION AND SOFTWARE TECHNOLOGY	15
REVERSE ENGINEERING - WORKING CONFERENCE PROCEEDINGS	15

3.5. Countries' Scientific Production

Table 4 highlights the scientific production by country in the realm of research. The USA leads with 935 documents, showcasing its dominance and substantial contributions to this research area. China follows with 541 documents, reflecting its growing investment and focus on computational and engineering research. Germany ranks third with 519 documents, demonstrating a strong academic presence, particularly in theoretical and applied computer science. European countries such as France (229), UK (222), and Italy (137) also make significant contributions, indicating robust research activities in these regions. Canada (214) and India (211) contribute comparably, highlighting their engagement in the global research network. Meanwhile, Japan (193) emphasizes its established role in technology and innovation, while Spain rounds out the list with 90 documents, indicating a modest but impactful presence. This distribution underscores the global interest and collaborative nature of graph theory research in software engineering, with key contributions from both developed and emerging economies.

Table 4. Countries scientific productions

Country	Documents
USA	935
CHINA	541
GERMANY	519
FRANCE	229
UK	222
CANADA	214
INDIA	211

JAPAN	193
ITALY	137
SPAIN	90

3.6. Trend Topics

Figure 2 illustrates the evolution of research topics, revealing shifts in focus and the emergence of new trends over time. During the 1990s and early 2000s, foundational topics such as "object-oriented design," "software development," and "specification" dominated, reflecting an emphasis on establishing robust software engineering methodologies. Concurrently, topics like "graph transformation," "Petri nets," and "graph algorithms" gained traction, signifying a growing interest in applying graph-theoretic concepts for system modeling and formal analysis. By the mid-2000s, the focus expanded to include advanced applications such as "model synchronization," "graph transformations," and "visualization," addressing challenges in maintaining consistency and improving software understanding. In the 2010s and beyond, research diversified further, with topics like "formal methods," "optimization," and "data mining" gaining prominence, reflecting the field's maturity and the application of graph theory to complex computational problems. Terms like "refactoring," "software testing," and "software metrics" highlight the practical use of graph-based approaches to enhance software quality and maintainability. Simultaneously, specialized areas such as "program slicing," "program dependence graphs," and "UML" demonstrate the integration of graph theory into specific processes within software engineering. Foundational topics like "graph theory" and "software engineering" remain consistently relevant across decades, underscoring their enduring significance, while recent years see a growing interest in applied areas like "graph drawing" and "visualization," showcasing the adaptability of graph theory to emerging challenges and applications in software engineering.

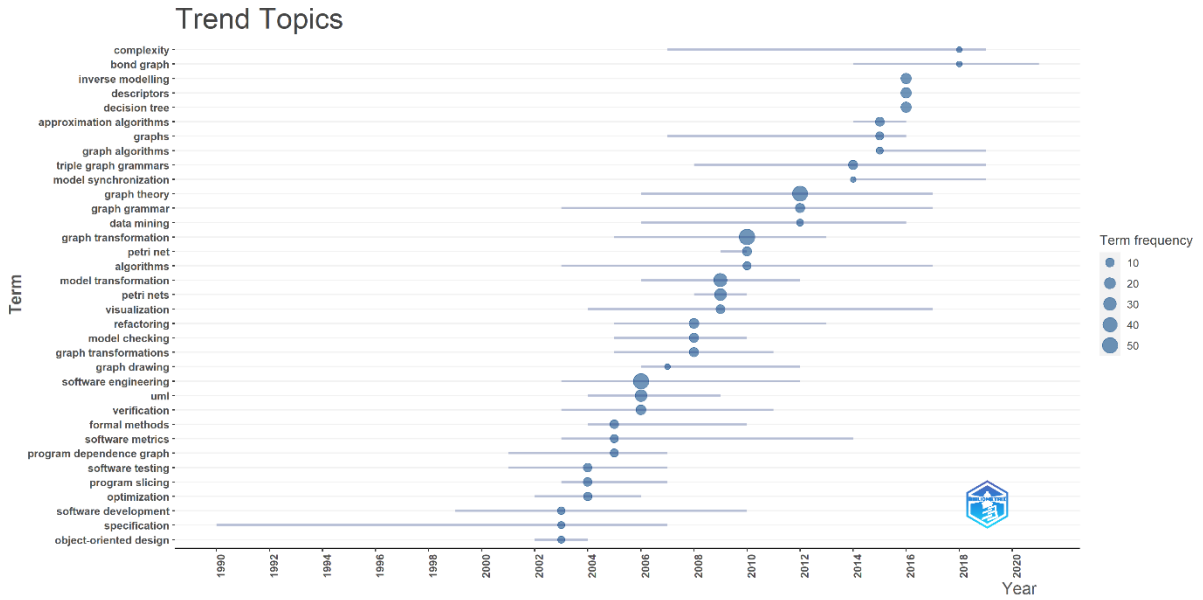

Figure 2. Trending topics in role of graph theory in network security

3.7. Thematic Map

Figure 3 presents the thematic visualization of keywords, categorising themes based on their development degree (density) and relevance (centrality). In the Motor Themes quadrant, highly central and well-developed topics such as program slicing and program dependence graph dominate. These themes play a crucial role in software engineering by enabling effective code analysis, debugging, and optimization. Their high density indicates substantial research and application, positioning them as the drivers of innovation in the field.

The Basic Themes quadrant houses foundational but less-developed areas such as software engineering, graph theory, refactoring, verification, and model checking. These themes are essential to the broader domain and serve as a basis for further advancements. Additionally, tools and methodologies like graph transformation, UML, and Petri nets are fundamental for modeling and analyzing complex systems but still require focused efforts to reach higher levels of development. These themes are central to the field and have the potential to evolve with the influence of motor themes.

Conversely, the Niche Themes quadrant includes specialized topics like decision tree, descriptors, and inverse modeling, which are highly developed but lack widespread application in software engineering. Meanwhile, the Emerging or Declining Themes quadrant highlights less-developed and peripheral topics such as approximation algorithms and parameterized complexity. These areas, while currently less central, hold promise for future growth as computational challenges evolve. The map underscores the interconnectedness of these themes, suggesting that advancements in motor themes could bolster basic and emerging themes, creating a robust ecosystem for software engineering innovation.

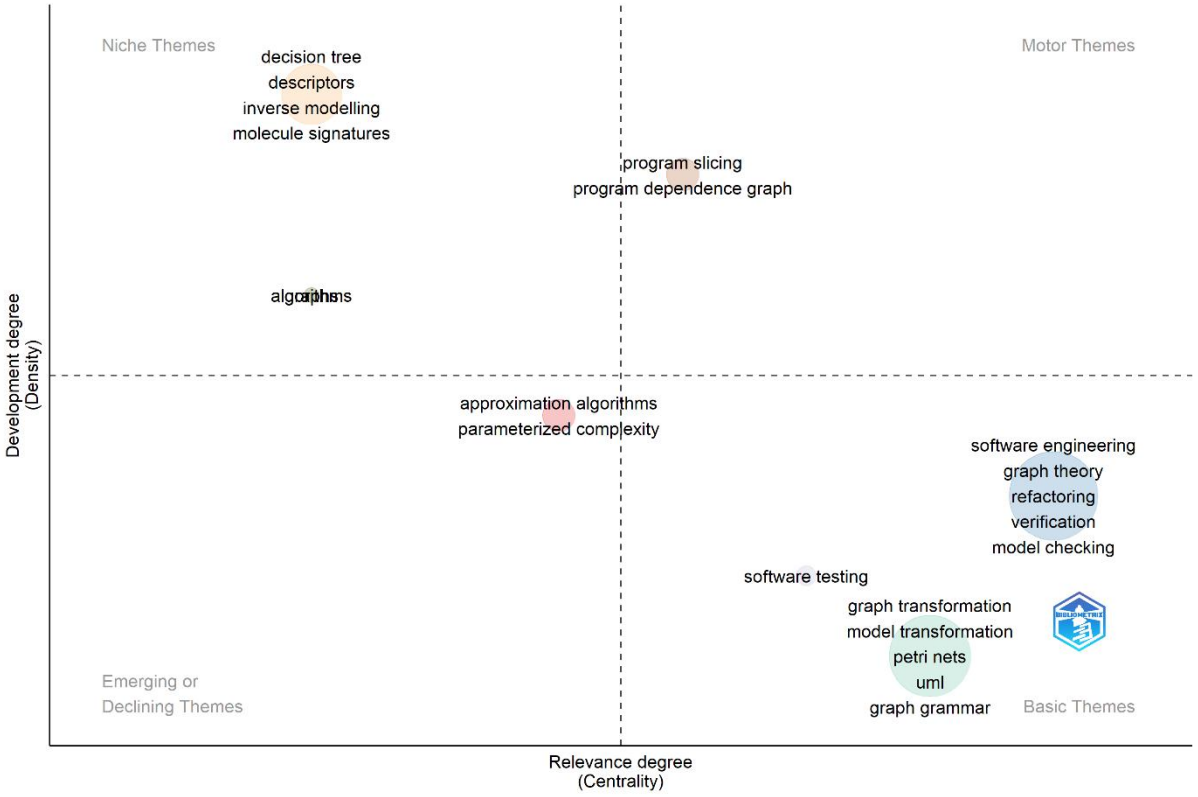

Figure 3. Thematic visualisation of keywords

3.8. Bibliographic Coupling of Sources

Figure 4 presents a bibliographic coupling network of sources filtered with a minimum citation threshold of 5. It reveals a dense network of scholarly discourse in the field of Graph Theory for Advancements in Software Engineering. From 654 sources, 45 met the citation threshold, and the resulting network comprises 40 key items divided into 8 clusters, demonstrating the interconnectedness and diversity of research contributions. The prominent nodes, such as Lecture Notes in Computer Science, Electronic Notes in Theoretical Computer Science, and IEEE Transactions on Software, indicate influential publications that serve as critical reference points for the domain. These larger nodes suggest sources with high citation frequency and relevance, forming the backbone of the field's academic ecosystem. Connections between nodes signify overlapping research topics or shared references, highlighting areas of collaboration and thematic convergence. Clusters represent distinct thematic areas, with each cluster aggregating sources that share related research interests. For instance, one cluster might focus on theoretical advancements in algorithms and graph models, while another could center around practical

applications in software engineering and system optimization. The visualization illustrates not only the diversity of research but also the interdisciplinary nature of graph theory's application, where it intersects with artificial intelligence, software testing, and computational complexity. This map highlights the significant role of bibliographic coupling in identifying key sources and emerging areas of research within the field.

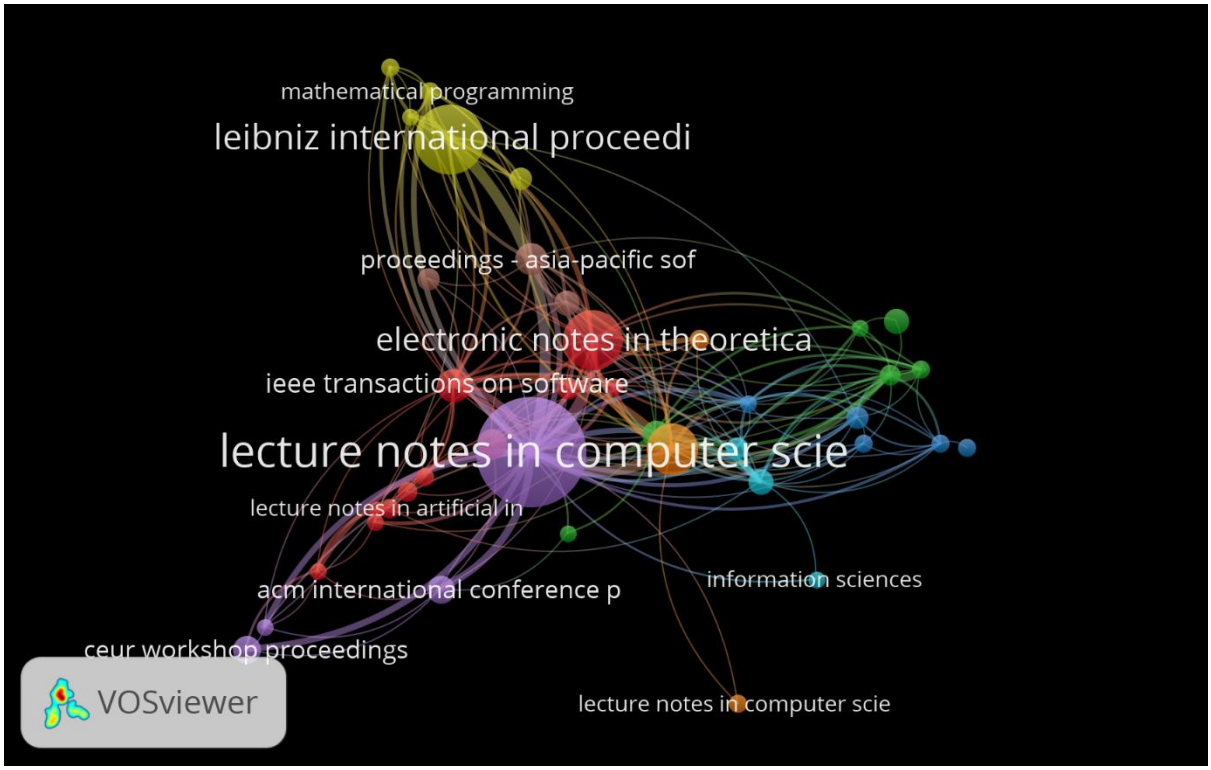

Figure 4. Bibliographic coupling of sources

3.9. Co-occurrence of Keywords

Figure 5 illustrates a co-occurrence network of keywords that highlights the interrelation of 70 keywords in the field of Graph Theory for Advancements in Software Engineering, categorized into 63 items and grouped into 10 clusters. Cluster 1, with 14 keywords, is the largest and represents core themes like software engineering, graph transformation, and model checking, which serve as foundational concepts in the field. Cluster 2, with 11 keywords, focuses on practical applications such as software testing, refactoring, and program slicing, emphasizing their relevance to optimizing and enhancing software development processes. Smaller clusters, such as Cluster 9 (3 keywords) and Cluster 10 (2 keywords), showcase niche or emerging areas, like triple graph grammars, which reflect evolving research trends and specialized applications.

The network demonstrates the thematic diversity within the field, with larger clusters bridging theoretical concepts like graph algorithms and parameterized complexity to practical methodologies such as model-driven engineering and software reuse. Central keywords, including software engineering, graph transformation, and model transformation, act as hubs, connecting multiple clusters and illustrating their foundational importance. The map reveals the interconnectedness of research areas, highlighting established topics while also identifying underexplored and emerging themes. This visualization provides a valuable framework for navigating the complex research landscape and fostering interdisciplinary collaboration.

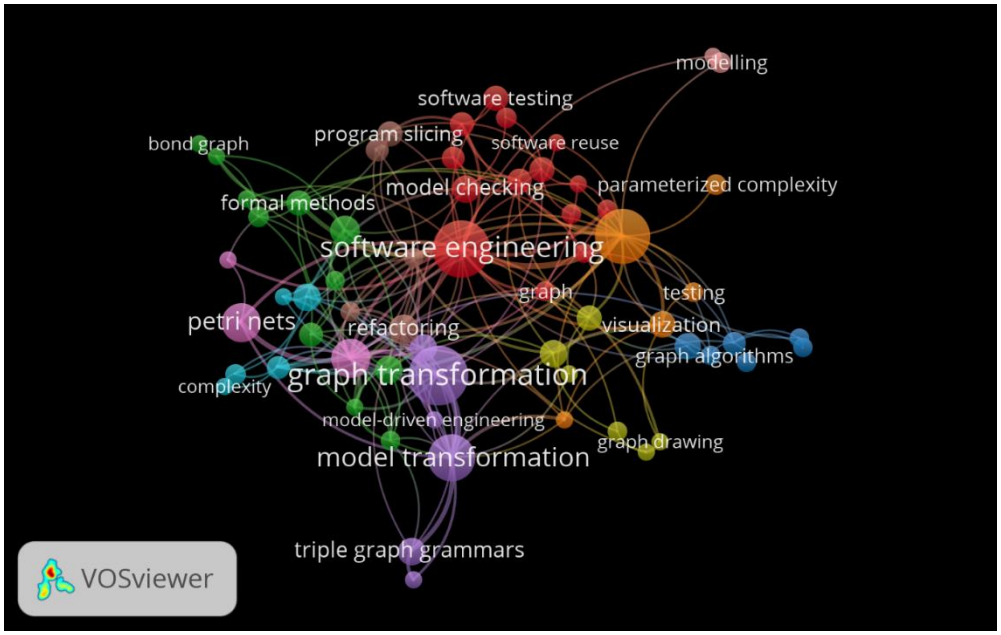

Figure 5. Co-occurrence of all keywords

3.10. Co-authorship of Countries Collaborations

Figure 6 illustrates the co-authorship network of country collaboration patterns among countries, with the United States emerging as the central hub, maintaining strong connections with key countries like China, the United Kingdom, Germany, and Canada. Regional clusters highlight the dynamics of collaboration, such as robust partnerships within Europe (e.g., Germany, France, and Italy) and growing contributions from Asia, with countries like India and Taiwan connecting to larger nodes. Smaller nodes, such as South Korea and Croatia, represent niche contributors that are linked to the global network through selective partnerships. This visualization emphasizes the interconnectedness of research efforts while showcasing opportunities for emerging research economies to strengthen their global influence.

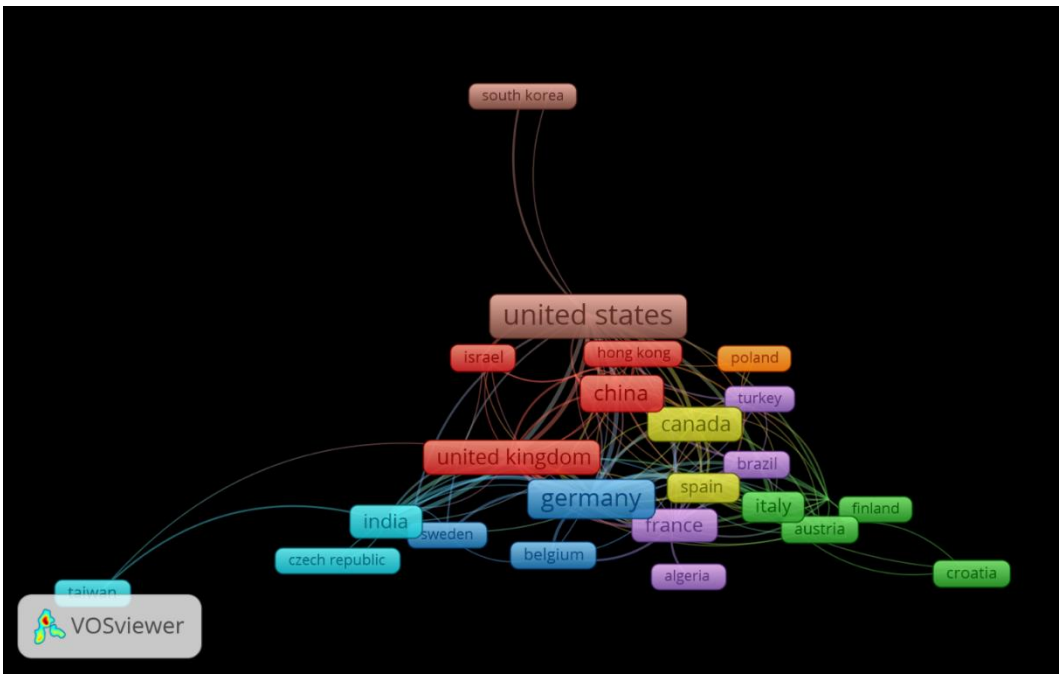

Figure 6. Co-authorship of countries

DISCUSSIONS

The analysis of graph theory applications in software engineering, based on bibliometric mapping and thematic trends, provides valuable insights into the field's research landscape. The thematic map highlights the central role of motor themes such as program slicing and program dependence graph, which dominate due to their high relevance and development. These themes drive innovation in areas like code analysis and debugging. Conversely, foundational yet less-developed basic themes like software engineering, graph theory, and model checking represent critical areas requiring further exploration. Emerging topics such as approximation algorithms and niche areas like triple graph grammars suggest opportunities for future research in computational complexity and specialized modeling techniques.

Bibliographic coupling and keyword co-occurrence analyses reveal the diversity and interdisciplinary nature of the field. Influential sources like the Lecture Notes in Computer Science and IEEE Transactions on Software Engineering serve as hubs of scholarly discourse, facilitating the dissemination of research on both theoretical advancements and practical applications. Core themes, such as graph transformation, bridge theoretical concepts with practical methodologies like model-driven engineering and software testing, underscoring the interplay between foundational and application-driven research. At the same time, niche areas remain underexplored, presenting opportunities for interdisciplinary applications and novel insights.

The analysis also identifies significant trends in global research collaboration. The United States stands out as the central hub in international co-authorship networks, with strong links to countries like China, Germany, and India, reflecting its leadership in advancing the field. European countries demonstrate robust intra-regional collaborations, while emerging economies such as India and Taiwan are gradually strengthening their contributions. These patterns underscore the importance of fostering inclusive global partnerships to expand the research ecosystem and address challenges across diverse contexts.

Despite the significant advancements, gaps remain in integrating graph theory into emerging technologies like AI-driven software development and real-time system optimization. Practical implications include leveraging graph-based models to enhance scalability, efficiency, and reliability in software engineering processes. Addressing these gaps through interdisciplinary collaboration and targeted funding can drive innovation in addressing modern challenges, ensuring the sustained growth and relevance of graph theory in software engineering. This analysis provides a roadmap for future research while emphasizing the need for inclusivity and innovation in expanding the field's boundaries.

CONCLUSION

This bibliometric analysis highlights the significant role of graph theory in advancing software engineering, particularly in areas such as program slicing, graph transformation, and model checking, which are pivotal for improving software reliability and optimization. The study underscores the global nature of research, with the United States, China, and Germany emerging as major contributors, while collaboration patterns reveal opportunities for more inclusive partnerships with emerging economies like India and Taiwan. However, the field requires further exploration of underdeveloped themes such as approximation algorithms and niche areas like triple graph grammars. To advance the field, researchers should focus on integrating graph theory into cutting-edge domains like AI-driven software development and real-time systems optimization. Institutions should prioritize interdisciplinary collaborations to leverage expertise across fields, addressing complex challenges effectively. Additionally, funding agencies and academic platforms should encourage research in underexplored themes, fostering innovation and inclusivity. By addressing these gaps, the field can continue to drive impactful advancements in software engineering and beyond.

REFERENCES

[1] F. Fagnani and P. Frasca, "Graph theory," in *Lecture Notes in Control and Information Sciences*, vol. 472, 2018, pp. 1–30. doi: 10.1007/978-3-319-68022-4_1.

[2] A. Chatzigeorgiou, N. Tsantalis, and G. Stephanides, "Application of graph theory to OO software engineering," presented at the Proceedings - International Conference on Software Engineering, 2006, pp. 29–35. doi: 10.1145/1137661.1137669.

[3] A. Majeed and I. Rauf, "Graph theory: A comprehensive survey about graph theory applications in computer science and social networks," *Inventions*, vol. 5, no. 1, p. 10, 2020.

[4] S. Markstrum and G. M. Haggard, "Using graph theory visualization to motivate Software Engineering concepts," presented at the ASEE Annual Conference and Exposition, Conference Proceedings, 2011. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85029087501&partnerID=40&md5=27fbb05a58da2d965d9eec7cebc21f09

[5] A. E. T. Wahyuni, R. Adawiyah, and I. I. Makhfudloh, "Graph Applications in Software Engineering".

[6] L. Baresi and M. Pezzè, "From graph transformation to software engineering and back," *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.*, vol. 3393 LNCS, pp. 24–37, 2005, doi: 10.1007/978-3-540-31847-7_2.

[7] A. X. Zheng, J. Dunagan, and A. Kapoor, "Active graph reachability reduction for network security and software engineering," presented at the IJCAI International Joint Conference on Artificial Intelligence, 2011, pp. 1750–1756. doi: 10.5591/978-1-57735-516-8/IJCAI11-293.

[8] A. Striuk, O. Rybalchenko, and S. Bilashenko, "Development and using of a virtual laboratory to study the graph algorithms for bachelors of software engineering," presented at the CEUR Workshop Proceedings, 2020, pp. 974–983. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85096146260&partnerID=40&md5=d849301d1f5f02dee1c4f4d1df16dd2f

[9] V. Rajappa, A. Biradar, and S. Panda, "Efficient software test case generation using genetic algorithm based graph theory," presented at the 2008 First International Conference on Emerging Trends in Engineering and Technology, IEEE, 2008, pp. 298–303.

[10] K. Sreejil and R. Balakumar, "Analysis of fuzzy graph theory and its application," *Adv. Math. Sci. J.*, vol. 9, no. 3, pp. 1239–1245, 2020, doi: 10.37418/amsj.9.3.49.

[11] E. Cachia and M. Micallef, "An event-driven cartographic approach to modelling software engineering knowledge," presented at the KMIS 2011 - Proceedings of the International Conference on Knowledge Management and Information Sharing, 2011, pp. 18–27. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84862175141&partnerID=40&md5=c98aa3bf111a568d233ca4ebaa7b4b03

[12] R. Arora and S. Goel, "Javarelationshipgraphs (jrg): Transforming Java projects into graphs using neo4j graph databases," presented at the ACM International Conference Proceeding Series, 2019, pp. 80–84. doi: 10.1145/3305160.3305173.

[13] C. Elasri, N. Kharmoum, F. Saoiabi, M. Boukhlif, S. Ziti, and W. Rhalem, "Applying Graph Theory to Enhance Software Testing in Medical Applications: A Comparative Study," presented at the Lecture Notes in Networks and Systems, 2024, pp. 70–78. doi: 10.1007/978-3-031-52388-5_7.

[14] K. Karnavel and J. Santhoshkumar, "Automated software testing for application maintenance by using bee colony optimization algorithms (BCO)," presented at the 2013 International Conference on Information Communication and Embedded Systems, ICICES 2013, 2013, pp. 327–330. doi: 10.1109/ICICES.2013.6508211.

[15] W. K. Harrison, "The role of graph theory in system of systems engineering," *IEEE Access*, vol. 4, pp. 1716–1742, 2016.

[16] F. Husain and M. S. Mustafa, "A Decade of Islamic Banking Research: Bibliometric Review with Biblioshiny and Vosviewer," *Jambura Sci. Manag.*, vol. 5, no. 2, pp. 67–85, 2023.

[17] Z. Guofang, M. S. Rasul, and M. Omar, "A Bibliometric Analysis of Publications on University-Industry Collaboration Using VOSviewer and R-biblioshiny," *Multidiscip. J. Educ. Soc. Technol. Sci.*, vol. 11, no. 2, pp. 26–50, 2024.

[18] S. K. Patra, V. D. P. Kotni, S. Kumar, and M. G. Chintaluri, "Charting the brand love landscape: a comprehensive bibliometric analysis with biblioshiny and VOSviewer to evaluate present developments, and future research directions," *Int. J. Serv. Stand.*, vol. 14, no. 1, pp. 65–93, 2024.

[19] R. Komperda, "Likert-type survey data analysis with R and RStudio," *ACS Symposium Series*, vol. 1260. pp. 91–116, 2017. doi: 10.1021/bk-2017-1260.ch007.

[20] E. Souza de Cursi, “Some Tips to Use R and RStudio,” in *Uncertainty Quantification using R*, E. Souza de Cursi, Ed., Cham: Springer International Publishing, 2023, pp. 1–108. doi: 10.1007/978-3-031-17785-9_1.

[21] D. Kumar, A. K. Shandilya, and S. Choudhuri, “Artificial Intelligence-Enabled Bibliometric Analysis in Tourism and Hospitality Using Biblioshiny and VOSviewer Software,” in *AI-Centric Modeling and Analytics*, CRC Press, 2023, pp. 260–291.

[22] N. J. Van Eck and L. Waltman, “Software survey: VOSviewer, a computer program for bibliometric mapping,” *Scientometrics*, vol. 84, no. 2, pp. 523–538, Aug. 2010, doi: 10.1007/s11192-009-0146-3.

[23] R. Jumansyah, E. S. Soegoto, and C. N. Albar, “COMPUTATIONAL BIBLIOMETRIC ANALYSIS OF EVOLUTIONARY GAME THEORY (EGT) RESEARCH USING VOSVIEWER,” vol. 18, 2023.

[24] J. T. McAllister, L. Lennertz, and Z. A. Mojica, “Mapping A Discipline: A Guide to Using VOSviewer for Bibliometric and Visual Analysis,” *Sci. Technol. Libr.*, vol. 41, no. 3, pp. 319–348, 2022, doi: 10.1080/0194262X.2021.1991547.

[25] A.-W. Harzing and S. Alakangas, “Google Scholar, Scopus and the Web of Science: a longitudinal and cross-disciplinary comparison,” *Scientometrics*, vol. 106, pp. 787–804, 2016.

[26] J. Baas, M. Schotten, A. Plume, G. Côté, and R. Karimi, “Scopus as a curated, high-quality bibliometric data source for academic research in quantitative science studies,” *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 377–386, Feb. 2020, doi: 10.1162/qss_a_00019.