Frontiers in Education

TYPE Review
PUBLISHED 28 March 2025
DOI 10.3389/feduc.2025.1525917

Teaching programming in higher education: a bibliometric analysis of trends, technologies, and pedagogical approaches

Mariuxi Vinueza-Morales[1], Jorge Rodas-Silva[2], Cristian Vidal-Silva[3]*, Jorge Córdova-Morán[1] and Edwin Cevallos-Ayón[1]

[1]Faculty of Sciences and Engineering, Universidad Estatal de Milagro, Milagro, Ecuador, [2]Innovation in Academic Processes, SofTech Research Group, Universidad Estatal de Milagro, Milagro, Ecuador, [3]Facultad de Ingeniería y Negocios, Universidad de Las Américas, Santiago, Chile

OPEN ACCESS

EDITED BY
Chayanika Uniyal,
University of Delhi, India

REVIEWED BY
Mohammed F. Farghally,
Virginia Tech, United States
Yasheshwar Yasheshwar,
University of Delhi, India
Nidhi Kesari,
University of Delhi, India

*CORRESPONDENCE
Cristian Vidal-Silva
cristian.vidal.silva@edu.udla.cl

RECEIVED 10 November 2024
ACCEPTED 10 March 2025
PUBLISHED 28 March 2025

CITATION
Vinueza-Morales M, Rodas-Silva J, Vidal-Silva C, Córdova-Morán J and Cevallos-Ayón E (2025) Teaching programming in higher education: a bibliometric analysis of trends, technologies, and pedagogical approaches.
Front. Educ. 10:1525917.
doi: 10.3389/feduc.2025.1525917

The continuous evolution of digital technologies has significantly reshaped the landscape of higher education, particularly in programming instruction. As programming skills become increasingly essential across multiple disciplines, analyzing the research trends, key contributors, and institutional impact on programming education is crucial. This study presents a bibliometric analysis to identify influential research works, leading authors, and major institutions that have shaped this field over the years. By systematically examining scholarly publications, this study explores the most recurrent pedagogical approaches, the role of emerging technologies, and the methodological frameworks used to assess programming competencies. Additionally, it highlights prevailing research trends, knowledge gaps, and future directions for improving programming education in higher education institutions. The findings of this work will support educators, curriculum designers, and policymakers in refining instructional strategies, fostering innovative learning environments, and aligning academic programs with industry demands.

KEYWORDS

programming education, bibliometric analysis, higher education, pedagogical strategies, emerging technologies, curriculum development, research trends

1 Introduction

The rapid evolution of technology has significantly reshaped various domains, including higher education (Froyd et al., 2012). One of the most crucial aspects of this transformation is the teaching and learning of programming, which has become an integral component of computer science curricula (Paiva et al., 2022). Understanding how students acquire programming competencies and identifying the factors influencing their success are critical concerns for educators and researchers. Numerous studies have explored diverse pedagogical approaches, assessment methodologies, and challenges associated with programming instruction (Medeiros et al., 2019). However, a consolidated understanding of these approaches' evolution, effectiveness, and future directions remains underdeveloped.

Programming education has experienced significant growth in recent years, with a surge of research examining innovative teaching strategies. Despite this expansion, gaps persist in consolidating insights on how programming education has evolved, which methodologies yield the highest impact, and where research efforts should be concentrated. This study aims to bridge these gaps through a bibliometric analysis identifying key trends, influential contributors, and emerging pedagogical methodologies.

In the digital era, programming has emerged as a fundamental competency across multiple disciplines, particularly within Science, Technology, Engineering, and Mathematics (STEM) fields (Abichandani et al., 2022). Traditionally, programming education was confined to technical and engineering domains; however, its role has expanded across diverse academic fields, recognizing its value not only as a technical skill but also as a means of fostering critical thinking and problem-solving abilities (González-Sanmamed et al., 2018; Jiménez-Toledo et al., 2019). Acquiring proficiency in programming extends beyond syntax and data structures—it requires logical reasoning, computational thinking, and adaptability to new paradigms (Lin et al., 2022).

1.1 Pedagogical approaches in programming education

Programming education relies on more than knowledge transmission; it fosters a computational mindset, enhances logical reasoning, and equips students with problem-solving strategies (Compañ-Rosique et al., 2015). As technology advances, programming education must continuously evolve to align with industry demands and student needs, ensuring long-term educational impact (Adamopoulos, 2020). Bloom's taxonomy provides a hierarchical framework for categorizing learning objectives and guiding instructional strategies in programming education (Masapanta-Carrion and Velázquez-Iturbide, 2018; Bloom, 1956). Additionally, research highlights the effectiveness of project-based learning (PBL) and pair programming in fostering collaboration, critical thinking, and real-world problem-solving skills (Younis et al., 2021; Shin et al., 2021).

Adaptability is a fundamental skill in programming, as languages, tools, and methodologies are constantly evolving (Merelli et al., 2016). Effective programming instruction must, therefore, emphasize autonomy and adaptability, equipping students with self-learning strategies, curiosity for emerging technologies, and resilience in overcoming challenges (Cheng et al., 2021; Vesin et al., 2018). Adaptive learning frameworks foster creative problem-solving and logical reasoning, preparing students for a rapidly changing technological landscape (Qadir et al., 2020).

1.2 Need for a bibliometric analysis

The extensive research on programming education underscores the growing recognition of its importance in academic and professional settings (Thuné and Eckerdal, 2018). Scholars, educators, and practitioners have contributed many studies, theories, and methodologies, shaping our understanding of effective teaching practices (Xie et al., 2019; Silva et al., 2020; Liargkovas et al., 2022; Yusuf and Noor, 2023). However, the sheer volume of publications and the fast pace of new developments make it challenging to stay informed about the most impactful trends and innovations.

The rapid proliferation of research in programming education necessitates a structured approach to identify effective teaching strategies, assess their impact, and map the intellectual landscape of the field (Cheah, 2020). This study seeks to provide a bibliometric analysis that examines publication trends, identifies the most cited sources, maps academic collaboration networks, and highlights key contributors to programming education at the higher education level. By synthesizing this information, our work offers a structured overview of the current state of programming education research and identifies areas that require further attention.

1.3 Objectives and research questions

This study aims to provide a systematic bibliometric analysis of programming education research. Specifically, it seeks to:

- Identify the most prevalent teaching and learning strategies in programming education.
- Recognize the most influential authors and seminal publications shaping the field.
- Determine the top contributing institutions and journals in programming education research.
- Highlight emerging trends and future research opportunities.

To achieve these objectives, we formulate the following research questions:

- **RQ1: What are the predominant trends in programming education methodologies in higher education?** This study addresses this question by analyzing research from the SCOPUS and WOS databases (2014–2023), identifying frequently cited methodologies such as project-based learning, flipped classrooms, and collaborative programming. The analysis examines the evolution of these strategies and their impact on programming competency development while acknowledging the potential contributions of non-indexed sources.
- **RQ2: Who are the most influential authors, and what are the seminal publications shaping programming education research?** This question is addressed by identifying the most cited authors and publications in SCOPUS and WOS. The study analyzes author networks, citation impact, and recurring themes in key publications while recognizing that other influential works may exist outside indexed databases.
- **RQ3: Which journals and institutions have made the most significant contributions to programming education research?** The study explores this question by examining journals and institutions with the highest publication and citation metrics between 2014 and 2023. The review highlights research centers that have consistently shaped programming

education discourse while acknowledging contributions from alternative publication platforms.

1.4 Paper organization

The remainder of this paper is organized as follows. Section 3 describes the research approach and bibliometric methodology, detailing the data collection process and analytical tools used. Section 4 presents the findings of the bibliometric analysis, identifying key contributors, influential institutions, and emerging trends. Section 5 discusses the implications of the findings, addressing gaps in research and potential future directions. Section 6 provides the conclusions, summarizing the study's contributions and suggesting avenues for further investigation.

2 Theoretical foundations

Computational thinking plays a critical role in the modern digital era, providing people with essential problem-solving skills that transcend the boundaries of computer science and programming, as noted by Lu et al. (2022). Based on principles from computer science, mathematics, and logic (Li et al., 2020), this approach enables people to solve complex problems, recognize patterns, and design algorithmic solutions (Srinivasa et al., 2022). As highlighted by Shen et al. (2022), computational thinking significantly influences daily life by helping people make informed decisions and solve problems efficiently. Whether optimizing daily routines or critically evaluating online information, this set of skills empowers people to navigate the complexities of the digital world (Kanaki and Kalogiannakis, 2022). Furthermore, incorporating it into educational curricula fosters essential competencies such as logical reasoning and creativity, preparing students for future challenges (Kwon et al., 2021). The Algorithm 1 presents an algorithm that describes the steps to master new topics through computational thinking (Vidal-Silva et al., 2024).

```
Require: A topic is selected
Ensure: A deep understanding is acquired
 1: Establish foundational knowledge
 2: Define the scope and key concepts
 3: Identify and collect credible resources
 4: Develop a structured learning plan
 5: Set measurable learning objectives
 6: while understanding is incomplete do
 7:    Review and refine selected materials
 8:    Expand research to gather diverse perspectives
 9:    Apply acquired knowledge through practice
10:    Validate comprehension via self-assessment
11:    Share insights through discussion or teaching
12: end while
13: Summarize key findings and document insights
```

Algorithm 1. Procedure for Effective Topic Mastery

Beyond everyday applications, the benefits of computational thinking extend to a wide range of disciplines (Li et al., 2020; Lai, 2021). Shin et al. (2022) demonstrate how this competency enhances scientific research by helping scientists analyze complex data, simulate experiments, and develop better models to understand the natural world. Computational thinking supports biology, physics, and social sciences research, improving decision-making by providing powerful tools for data-driven insights (Helbing et al., 2023). With the growing demand for digital literacy in the workforce, computational thinking equips individuals with the tools required to thrive in an evolving job market (Yadav and Berthelsen, 2021). This competency empowers individuals as students or professionals and as active participants in a technology-driven society.

2.1 Programming skills in engineering and technical education

Programming competencies are critically important in higher education, particularly within computer science and across disciplines of Science, Technology, Engineering, Arts, and Mathematics (STEAM) (Marín-Marín et al., 2021). The ability to think algorithmically, solve complex problems systematically, and develop automated solutions through coding has become an indispensable skill set for students, regardless of their field of study (Melro et al., 2023). Developing programming competencies in higher education equips students with fundamental skills that go beyond coding:

1. **Algorithmic thinking:** this allows students to break down complex problems into smaller, manageable tasks while constructing logical sequences of steps to address them (Lu et al., 2022).
2. **Problem-solving through programming:** programming fosters creativity and resilience, pushing students to iteratively refine their solutions until achieving the most effective outcome (Ju, 2024).
3. **Abstract thinking:** students can conceptualize real-world problems as abstract models, facilitating a deeper understanding of complex phenomena across various disciplines (Qian and Choi, 2023).
4. **Automation and efficiency:** programming enables students to streamline repetitive tasks, enhancing both productivity and efficiency (Selwyn et al., 2023).

The impact of programming competencies transcends computer science, offering substantial benefits in STEAM disciplines (Marín-Marín et al., 2021; Dúo-Terrón, 2023). These skills contribute to both the technical and creative dimensions of each field.

- **Science:** programming is a powerful tool for analyzing large datasets, modeling complex systems, and simulating experiments. In fields like biology, physics, and chemistry, the ability to automate data processing and perform statistical analyses improves scientific discoveries' precision and speed.

- **Technology:** programming drives a deeper understanding of technological systems, empowering students to innovate and develop new software tools.
- **Engineering:** programming is indispensable in engineering, whether used to model and simulate physical systems or optimize design processes. Coding equips students with tools that improve accuracy and efficiency in the mechanical, civil, and electrical engineering disciplines.
- **Art:** In the arts, programming opens new avenues for digital creativity.
- **Mathematics:** programming enhances the solving of mathematical problems by allowing the simulation of mathematical models and the resolution of large-scale calculations that would otherwise be impossible with manual methods.

3 Research approach and methodology

The rapid expansion of academic research and the increasing complexity of knowledge domains pose challenges for researchers seeking to stay updated within a specific field. A systematic literature review is crucial in synthesizing existing knowledge, identifying key trends, and highlighting gaps where further investigation is required (Briner and Denyer, 2012). Traditional literature reviews often rely on a single scientific database, which may lead to a fragmented view of research trends. To mitigate this limitation, scholars recommend employing multiple databases for a more comprehensive analysis (Echchakoui, 2020). This study used data from two well-established academic databases: Web of Science (WOS) and SCOPUS.

We ensure broad coverage of high-quality, peer-reviewed academic literature by incorporating both WOS and SCOPUS in our bibliometric analysis of teaching and learning strategies in programming education. These databases are recognized for indexing various scholarly outputs, including journal articles and conference proceedings in diverse disciplines such as computer science and education (Pranckutė, 2021). Their international scope (Valente et al., 2022) provides a global perspective on programming education methodologies, offering insights into different instructional approaches worldwide. Furthermore, using both databases enhances the robustness of the analysis by cross-referencing data, reducing bias, and improving the completeness of the literature review (Zhu and Liu, 2020). Consequently, selecting SCOPUS and WOS enables a comprehensive exploration of publication trends in programming education.

Records from both databases were merged into a unified dataset to conduct this analysis–the selection process adhered to well-defined inclusion and exclusion criteria, ensuring that only relevant studies were retained. Table 1 outlines the search parameters employed, while Figure 1 illustrates the trends in publication output from 2014 to 2023.

TABLE 1 Search criteria and data collection overview for the bibliometric analysis.

Criterion	Values
Time span	2014–2023
Date of query	June 2023
Document types	Journal articles and conference proceedings
Journal and conference types	Any type
Search fields	Title, abstract, and keywords
Search Terms	Programming AND Universities AND Strategies AND (Learning AND Teaching)—in English
Records	WOS: 1,464; SCOPUS: 361
Total records	1,697

3.1 Data collection and selection criteria

The data extraction process followed a rigorous methodology to ensure the relevance and quality of the selected records. Regarding database coverage, WOS contained more relevant records (1,464) than SCOPUS (361). The following exclusion criteria were applied to refine the dataset and maintain consistency in the analysis.

- **Removal of duplicate records:** studies indexed in both databases were identified and merged.
- **Relevance filtering:** articles that did not specifically address teaching and learning strategies for programming education in higher education were excluded.
- **Exclusion of peripheral studies:** papers that only mentioned programming education without analyzing its pedagogical effectiveness were removed.
- **Language constraints:** only English-language publications were considered to ensure comparability.
- **Full-text availability:** studies lacking accessible full-text versions were excluded.

Applying these criteria resulted in a refined dataset of **1,697** research articles, with an observed overlap of approximately 11%.

3.2 Publication trends and distribution

We analyzed the publication output over time to understand the evolution of research in programming education. Figure 1 presents the annual publication trends from 2014 to 2023, illustrating the growth and fluctuations in research activity within this domain.

3.3 Assessment tools and bibliometric analysis

This study employs bibliometric analysis as the primary methodological approach rather than survey-based or experimental methods. Bibliometric analysis allows for systematically evaluating

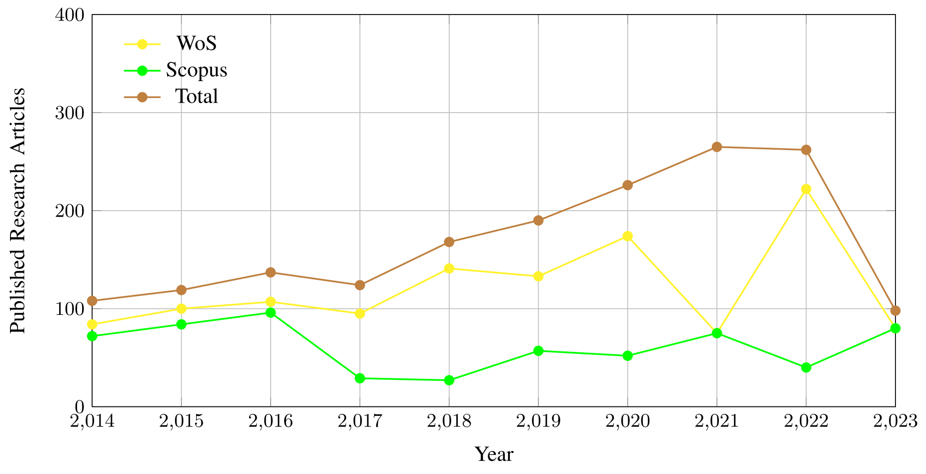

FIGURE 1
Total number of publications in WOS and SCOPUS, 2015–2023.

publication patterns, research impact, and intellectual structure in a given field.

The bibliometric analysis was conducted using **Bibliometrix**, an R-based open-source tool for statistical and visual exploration of academic literature. Bibliometrix facilitates comprehensive trend analysis, citation network mapping, and collaboration analysis across research fields (Chen et al., 2021). This tool was selected for its flexibility, integration with statistical software, and ability to generate interactive visual representations of bibliometric data.

Additionally, the **Biblioshiny** web-based module was employed to perform real-time data analysis (Moral-Munoz et al., 2020). Biblioshiny enables researchers to visualize co-citation networks, keyword co-occurrence patterns, and author collaborations, providing valuable insights into emerging trends and influential contributors in programming education research.

3.4 Final considerations

This study provides a rigorous and data-driven approach to understanding programming education research trends by leveraging bibliometric methods and data from multiple sources. The systematic filtering of records, analysis of publication trends, and use of advanced bibliometric tools contribute to a comprehensive examination of the field, paving the way for further research into effective pedagogical strategies and technological innovations in programming education.

4 Findings and analysis

This section presents the results of the bibliometric analysis, highlighting key contributors, prominent institutions, highly cited works, and the most relevant terminologies in programming education research. These findings offer valuable insights into the evolution and impact of research in this Field.

4.1 Leading researchers in the field

The bibliometric analysis identified the most influential researchers in programming education based on their citation count, H-index, and publication record. Table 2 presents the top authors contributing significantly to the Field. Among them, **Li Yong, Liu Yonggang, Liu Yan, Liu Yuan, and Yong Wang** stand out as leading figures in programming education research. Their work has explored innovative methodologies, active learning strategies, and computational thinking frameworks, shaping the pedagogical approaches in programming instruction (Hirsch, 2005).

Analyzing the H-index and citation impact provides insights into these researchers' academic influence in promoting programming education. Their extensive contributions reinforce the importance of structured teaching strategies, computational thinking, and technology-enhanced learning in programming curricula.

4.2 Prominent institutions and their contributions

The bibliometric analysis also identified institutions that have played a pivotal role in programming education research. Table 3 lists the top ten universities with the highest publication output in the Field. These institutions, including the *University of California, Toronto University, and the University of Sydney*, have significantly contributed to advancing pedagogical models and digital learning frameworks.

These institutions are critical in advancing programming education through their research on student engagement, problem-solving techniques, and adaptive learning environments. The high volume of publications from these universities highlights the increasing academic focus on enhancing programming instruction methodologies.

TABLE 2 Top authors in programming education research in higher education.

Authors	WOS citations	WOS H-index	SCOPUS citations	SCOPUS H-index	Publications
Li Yong	2,058	23	7,591	42	11
Liu Yonggang	29	3	682	15	11
Liu Yan	2,072	22	9,997	50	11
Liu Yuan	1,151	18	938	16	11
Yong Wang	2,124	26	8,192	50	11

TABLE 3 Top institutions contributing to programming education research.

Institution	Publications	Country
University of California, San Francisco	74	USA
University of Toronto	69	Canada
University of Colorado Boulder	59	USA
University of Michigan	59	USA
University of Sydney	43	Australia
University of Calgary	36	Canada
McGill University	35	Canada
University of Pennsylvania	35	USA
National University of Singapore	34	Singapore
University of Minnesota	34	USA

TABLE 4 Top cited documents in programming education research.

References	Journal	Total citations (TC)	Average citations per year
Sung et al. (2016)	Computers and Education	620	77.5
McLaughlin et al. (2013)	Academic Medicine	603	60.3
Bers et al. (2014)	Computers and Education	378	37.8
Douzas et al. (2018)	Information Sciences	350	58.33
Chiu-Lin and Gwo-Jen (2016)	Computers and Education	319	39.88

4.3 Seminal works and references

We analyzed citation patterns and impact factors of highly cited documents in programming education to identify the most influential publications. Table 4 presents the top-cited papers, revealing the substantial influence of studies focusing on active learning, flipped classrooms, and computational thinking.

These documents reflect key advancements in programming education, such as:

- The impact of **active learning methodologies** on student performance (Sung et al., 2016).
- The role of **flipped classrooms** in improving engagement and retention (McLaughlin et al., 2013).
- The integration of **computational thinking** into early programming education (Bers et al., 2014).

4.4 Key terminology and concepts

We analyzed the most frequently used keywords to gain deeper insights into the thematic focus of programming education research. Figure 2 presents a word cloud visualization, highlighting dominant terms in the field.

Table 5 further categorizes the most frequently used keywords, demonstrating the emphasis on **learning strategies, student engagement, and educational technologies**.

These results confirm that programming education research primarily focuses on effective **teaching methodologies, student engagement strategies, and the integration of emerging technologies**. The findings suggest that future research should explore adaptive learning systems, AI-driven programming education, and interdisciplinary learning approaches.

4.5 Final remarks

The bibliometric analysis of programming education research has revealed significant trends, key contributors, and influential publications. The findings emphasize the growing academic focus on student-centered learning models, the integration of AI in programming education, and the role of global collaboration in shaping programming instruction methodologies. By leveraging these insights, educators and researchers can refine instructional strategies and enhance the effectiveness of programming education in higher education institutions.

5 Discussion

This study provides a comprehensive bibliometric analysis of programming education research, identifying key trends, influential contributors, and emerging pedagogical methodologies. Analyzing data from SCOPUS and Web of Science (WOS) offers valuable insights into how programming education has evolved over the past decade and highlights opportunities for future advancements.

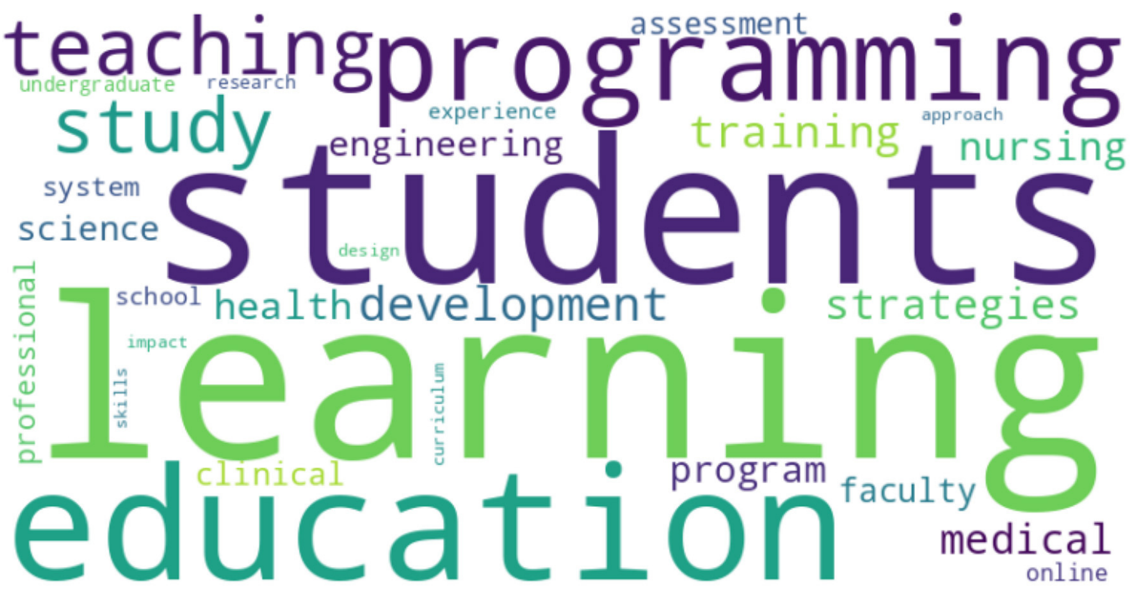

FIGURE 2
Word cloud highlighting the most frequent terms in the analyzed articles.

5.1 Emerging trends and directions

The results of this study directly address **RQ1**, revealing that the most effective teaching approaches in programming education are *project-based learning, flipped classrooms, and collaborative programming*. These methodologies have been consistently cited as effective strategies for enhancing student engagement, problem-solving abilities, and knowledge retention. The increased adoption of these techniques aligns with the growing emphasis on active and experiential learning models.

Additionally, a growing interest has been observed in leveraging *emerging technologies* such as **augmented reality (AR), virtual reality (VR), and artificial intelligence (AI)** to enhance programming instruction. These technologies are being explored to create immersive learning environments, offering students interactive and personalized experiences.

Addressing **RQ2**, this study identified the most influential researchers and seminal publications that have shaped programming education. The most cited works emphasize the importance of interdisciplinary teaching approaches, computational thinking development, and technology-enhanced learning. The bibliometric analysis also revealed extensive collaboration networks among leading researchers, demonstrating a global commitment to improving programming education methodologies.

A key trend highlighted in the analysis is the shift toward *collaborative programming techniques*, such as **pair programming and peer learning**, which have shown positive effects on code quality and student learning outcomes. Furthermore, as software development practices evolve, educational frameworks increasingly align with industry demands by incorporating teamwork-based learning experiences.

TABLE 5 Top ten keywords by frequency and percentage of appearance.

Keyword	Frequency	Percentage
Learning	1,127	2.50%
Programming	911	2.02%
Students	898	1.99%
Teaching	594	1.32%
Education	500	1.11%
University	345	0.76%
Computer	339	0.75%
Course	332	0.74%
Based	299	0.66%
Strategies	274	0.61%

5.2 Major contributions of this study

This study makes several contributions to the field of programming education by systematically analyzing its evolution and identifying dominant pedagogical strategies. Unlike traditional literature reviews that focus on isolated methodologies, the bibliometric approach offers a macro-level perspective, allowing for the identification of **knowledge gaps, research directions, and emerging trends**.

The key contributions of this study include:

- **Comprehensive mapping of programming education research:** by synthesizing findings from two major academic databases, this study presents a structured overview of the most influential research works, authors, and institutions contributing to programming education.

- **Identification of technological advancements:** the findings highlight the increasing role of AI, AR, and VR in programming education, underscoring their potential to transform instructional strategies.
- **Insights into pedagogical effectiveness:** the study confirms the dominance of active learning methodologies, such as project-based learning and flipped classrooms, in improving student learning outcomes.
- **Recognition of global collaboration trends:** the bibliometric analysis reveals strong academic networks, demonstrating the interconnected nature of programming education research across institutions and countries.

5.3 Gaps and challenges in existing studies

Despite the advancements in programming education, several critical gaps remain that require further exploration:

- **Lack of longitudinal studies:** most research in programming education focuses on short-term outcomes, such as student performance in individual courses. However, there is a significant gap in studies that assess the *long-term impact* of programming education strategies on professional career development, skill retention, and adaptability to technological advancements.
- **Limited research in diverse educational contexts:** most studies analyzed originate from developed countries, particularly the United States and China. There is a pressing need to explore programming education in *underrepresented regions*, including developing countries, rural communities, and alternative learning environments.
- **Need for research on non-traditional learning pathways:** with the rise of *coding boot camps, online platforms, and self-directed learning*, there is limited research on how these alternative educational models compare to traditional classroom-based programming instruction.
- **Challenges in integrating AI and adaptive learning tools:** while AI-driven educational platforms have shown promise, more research is required to evaluate their *long-term efficacy*, scalability, and ethical implications in programming education.

5.4 Potential biases and limitations

As with any bibliometric study, there are inherent limitations that must be acknowledged:

- **Selection bias:** this study relies on SCOPUS and WOS, which, despite their extensive coverage, may exclude relevant research from non-indexed sources or regional academic journals.
- **Temporal bias:** the analysis is limited to publications from 2014 to 2023, potentially overlooking foundational studies that continue to influence programming education.
- **Reliance on citation-based metrics:** while bibliometric indicators such as the H-index and citation count provide valuable insights, they may not fully capture the *qualitative impact* of a study or its effectiveness in real-world educational settings.

5.5 Opportunities for further investigation

Based on the identified gaps, this study proposes several avenues for future research:

- **Longitudinal studies on programming education outcomes:** more research is needed to understand how different teaching methodologies influence students' career progression and long-term skills development.
- **Interdisciplinary approaches to programming instruction:** future research should explore how integrating programming with other fields (e.g., cognitive science, psychology, and educational technology) can enhance learning outcomes.
- **Expanding research to diverse geographic regions:** more studies should focus on programming education in developing countries and alternative learning environments to provide a globally representative understanding of effective teaching practices.
- **Evaluation of AI-driven teaching methods:** as AI continues to shape education, empirical studies should assess the effectiveness of adaptive learning systems, automated code feedback, and AI-powered tutoring in programming instruction.
- **Bridging the gap between academia and industry needs:** future research should investigate how programming curricula can better align with industry requirements to ensure students graduate with relevant, in-demand skills.

5.6 Final remarks

The findings of this study emphasize the dynamic nature of programming education, reflecting continuous advancements in pedagogical approaches and technological integration. While significant progress has been made in refining instructional strategies, considerable opportunities remain to enhance accessibility, personalization, and real-world applicability in programming education.

Future studies can address the challenges and research gaps outlined in this discussion and contribute to the development of more inclusive, effective, and technologically enhanced programming education models. These insights will be invaluable for educators, curriculum designers, and policymakers seeking to optimize programming instruction for the next generation of learners.

6 Conclusions

This bibliometric analysis provides a comprehensive overview of the evolution of programming education research in higher education. The findings underscore the growing academic interest in programming instruction, revealing significant contributions from researchers and institutions worldwide. Over the past decade, programming education has evolved from a technical skill confined to computer science disciplines to a critical competency across various domains, including Science, Technology, Engineering, Arts, and Mathematics (STEAM).

The analysis highlights three predominant pedagogical strategies: **project-based learning, flipped classrooms, and collaborative programming**, which have proven effective in enhancing students' problem-solving abilities, engagement, and knowledge retention. These approaches align with the increasing demand for industry-relevant programming skills, suggesting that interactive and experiential learning methodologies play a crucial role in student success.

6.1 Key contributions and implications

The study's findings have several important implications for educators, curriculum designers, and policymakers:

- **Enhanced teaching methodologies:** Adopting student-centered approaches such as *project-based learning, pair programming, and active learning* has reshaped programming education. These strategies have demonstrated measurable benefits in improving students' comprehension, engagement, and long-term retention of programming concepts.
- **Integration of emerging technologies:** The use of *artificial intelligence (AI), virtual reality (VR), and adaptive learning technologies* in programming instruction has gained momentum. These tools offer personalized learning experiences, real-time feedback, and interactive coding environments, presenting new opportunities for improving educational outcomes.
- **Collaboration networks in research:** The bibliometric review indicates that leading institutions and researchers in the United States, China, and Europe are driving advancements in programming education. However, *limited representation from developing regions* highlights the need for more inclusive global collaboration to ensure equitable access to high-quality programming education.
- **Impact of institutional support:** Universities with strong research output in programming education play a crucial role in shaping curriculum design and pedagogical innovations. Encouraging *cross-institutional collaborations* can lead to better resource-sharing and knowledge dissemination across diverse educational contexts.

6.2 Challenges and areas for future research

Despite these advancements, several challenges persist that require further investigation:

- **Need for longitudinal studies:** Most existing research focuses on short-term outcomes, such as student grades or performance in isolated courses. Future studies should explore *long-term impacts* of different teaching methodologies on *students' professional careers, adaptability to new technologies, and retention of programming skills* beyond academia.
- **Limited research in underrepresented regions:** The majority of studies analyzed originate from technologically advanced countries. *There is a pressing need for research that examines programming education in developing countries, rural areas, and non-traditional learning environments.* Expanding research to diverse educational contexts will provide a more comprehensive understanding of global programming education challenges and best practices.
- **Adapting pedagogical strategies to non-traditional learning paths:** With the *rise of online learning platforms, coding boot camps, and self-directed learning*, traditional classroom-based strategies may not always be practical. Future research should explore *how programming education can be optimized for learners outside formal university settings*.
- **Assessing the role of AI in personalized learning:** While AI-based learning tools are being increasingly adopted, their long-term efficacy remains unclear. Future studies should focus on *how AI-driven adaptive learning environments* influence student motivation, engagement, and mastery of programming concepts.
- **Bridging the gap between academia and industry needs:** Employers often report skill gaps among graduates entering the tech industry. Research should investigate *how academic institutions can better align their programming curricula with real-world industry demands* to ensure students are equipped with the necessary competencies for professional success.

6.3 Final thoughts

This study provides a *structured roadmap for future research* by identifying **key trends, influential contributors, and emerging areas of programming education**. By addressing the identified gaps and leveraging new technologies, educators and researchers can enhance programming instruction, making it more effective, inclusive, and aligned with the evolving demands of the digital economy.

The findings reinforce the idea that **programming is no longer a niche skill but a fundamental competency across disciplines**, requiring continuous adaptation of teaching methodologies. As technology advances, *collaborative efforts among academia, industry, and policymakers* will be crucial in shaping the future of programming education, ensuring that students worldwide receive high-quality, relevant, and accessible instruction.

Author contributions

MV-M: Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing, Conceptualization, Data curation, Formal

analysis. JR-S: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. CV-S: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. JC-M: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. EC-A: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that no financial support was received for the research and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Gen AI was used in the creation of this manuscript.

Publisher's note

References

Abichandani, P., Sivakumar, V., Lobo, D., Iaboni, C., and Shekhar, P. (2022). Internet-of-things curriculum, pedagogy, and assessment for stem education: a review of literature. *IEEE Access* 10, 38351–38369. doi: 10.1109/ACCESS.2022.3164709

Adamopoulos, F. A. (2020). *Learning Programming, Student Motivation*. Cham: Springer International Publishing.

Bers, M. U., Flannery, L., Kazakoff, E. R., and Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Comp. Educ.* 72, 145–157. doi: 10.1016/j.compedu.2013.10.020

Bloom, B. (1956). *Taxonomy of Educational Objectives*. London: Longman.

Briner, R., and Denyer, D. (2012). *Systematic Review and Evidence Synthesis as a Practice and Scholarship Tool*. Oxford: Oxford University Press, 112–129.

Cheah, C.-S. (2020). factors-contributing-to-the-difficulties-in-teaching-and-learning-of-computer-programming-a-literature-review. *Contemp. Educ. Technol.* 12:ep272. doi: 10.30935/cedtech/8247

Chen, X., Zou, D., Xie, H., and Wang, F. L. (2021). Past, present, and future of smart learning: a topic-based bibliometric analysis. *Int. J. Educ. Technol. Higher Educ.* 18:2. doi: 10.1186/s41239-020-00239-6

Cheng, Q., Benton, D., and Quinn, A. (2021). "Building a motivating and autonomy environment to support adaptive learning," in *2021 IEEE Frontiers in Education Conference (FIE)* (Lincoln, NE: IEEE), 1–7.

Chiu-Lin, L., and Gwo-Jen, H. (2016). A self-regulated flipped classroom approach to improving students' learning performance in a mathematics course. *Comput. Educ.* 100, 126–140. doi: 10.1016/j.compedu.2016.05.006

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., and Molina-Carmona, R. (2015). Ense nando a programar: un camino directo para desarrollar el pensamiento computacional. *Revista de Educación a Distancia (RED)* 46:11. doi: 10.6018/red/46/11

Douzas, G., Bacao, F., and Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Inf. Sci.* 465, 1–20. doi: 10.1016/j.ins.2018.06.056

Dúo-Terrón, P. (2023). Analysis of scratch software in scientific production for 20 years: Programming in education to develop computational thinking and steam disciplines. *Educ. Sci.* 13:4. doi: 10.3390/educsci13040404

Echchakoui, S. (2020). Why and how to merge scopus and web of science during bibliometric analysis: the case of sales force literature from 1912 to 2019. *J. Market. Analyt.* 8:9. doi: 10.1057/s41270-020-00081-9

Froyd, J. E., Wankat, P. C., and Smith, K. A. (2012). Five major shifts in 100 years of engineering education. *Proc. IEEE* 100, 1344–1360. doi: 10.1109/JPROC.2012.2190167

González-Sanmamed, M., Sangrá, A., Souto-Seijo, A., and Estévez Blanco, I. (2018). Ecolog'ıas de aprendizaje en la era digital: desaf'ıos para la educación superior. *Publicaciones* 48, 25–45. doi: 10.30827/publicaciones.v48i1.7329

Helbing, D., Mahajan, S., Fricker, R. H., Musso, A., Hausladen, C. I., Carissimo, C., et al. (2023). Democracy by design: Perspectives for digitally assisted, participatory upgrades of society. *J. Comput. Sci.* 71:102061. doi: 10.1016/j.jocs.2023.102061

Hirsch, J. (2005). An index to quantify an individual's scientific research output. *Proc. National Acad. Sci.* 102, 16569–16572. doi: 10.1073/pnas.0507655102

Jiménez-Toledo, J., Collazos, C., and Revelo-Sánchez, O. (2019). Consideraciones en los procesos de ense nanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura. *TecnoLógicas* 22:83–117. doi: 10.22430/22565337.1520

Ju, Z. (2024). *Computational Thinking Through Programming: A Meta-Analysis of Collaborative Versus Solo Problem Solving*. Sydney: Sydney School of Education and Social Work, University of Sydney.

Kanaki, K., and Kalogiannakis, M. (2022). Assessing algorithmic thinking skills in relation to age in early childhood stem education. *Educ. Sci.* 12:380. doi: 10.3390/educsci12060380

Kwon, K., Ottenbreit-Leftwich, A. T., Brush, T. A., Jeon, M., and Yan, G. (2021). Integration of problem-based learning in elementary computer science education: effects on computational thinking and attitudes. *Educ. Technol. Res. Dev.* 69, 2761–2787. doi: 10.1007/s11423-021-10034-3

Lai, R. P. (2021). Beyond programming: a computer-based assessment of computational thinking competency. *ACM Trans. Comp. Educ.* (*TOCE)* 22, 1–27. doi: 10.1145/3486598

Li, Y., Schoenfeld, A. H., di Sessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., et al. (2020). *Computational Thinking is more about Thinking than Computing*. New York, NY: Springer.

Liargkovas, G., Papadopoulou, A., Kotti, Z., and Spinellis, D. (2022). Software engineering education knowledge versus industrial needs. *IEEE Trans. Educ.* 65, 419–427. doi: 10.1109/TE.2021.3123889

Lin, Y.-T., Yeh, M. K.-C., and Tan, S.-R. (2022). Teaching programming by revealing thinking process: Watching experts' live coding videos with reflection annotations. *IEEE Trans. Educ.* 65, 617–627. doi: 10.1109/TE.2022.3155884

Lu, C., Macdonald, R., Odell, B., Kokhan, V., Demmans Epp, C., and Cutumisu, M. (2022). A scoping review of computational thinking assessments in higher education. *J. Comp. Higher Educ.* 34, 416–461. doi: 10.1007/s12528-021-09305-y

Marín-Marín, J.-A., Moreno-Guerrero, A.-J., Dúo-Terrón, P., and López-Belmonte, J. (2021). Steam in education: a bibliometric analysis of performance and co-words in web of science. *Int. J. STEM Educ.* 8:41. doi: 10.1186/s40594-021-00296-x

Masapanta-Carrion, S., and Velázquez-Iturbide, J. (2018). "A systematic review of the use of bloom's taxonomy in computer science education," in *SIGCSE '18* (New York City: Association for Computing Machinery), 441–446.

McLaughlin, J., Roth, M., Glatt, D., Gharkholonarehe, N., Davidson, C., Griffin, L., et al. (2013). The flipped classroom: a course redesign to foster learning and engagement in a health professions school. *Acad. Med.* 89:86. doi: 10.1097/ACM.0000000000000086

Medeiros, R. P., Ramalhomand, G. L., and Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Trans. Educ.* 62, 77–90. doi: 10.1109/TE.2018.2864133

Melro, A., Tarling, G., Fujita, T., and Staarman, J. K. (2023). What else can be learned when coding? A configurative literature review of learning opportunities through computational thinking. *J. Educ. Comp. Res.* 61, 901–924. doi: 10.1177/07356331221133822

Merelli, E., Paoletti, N., and Tesei, L. (2016). Adaptability checking in complex systems. *Sci. Comp. Prog.* 115–116, 23–46. doi: 10.1016/j.scico.2015.03.004

Moral-Munoz, J., Herrera-Viedma, E., Espejo, A., and Cobo, M. (2020). Software tools for conducting bibliometric analysis in science: an up-to-date review. *El Profesional de la Información* 29:3. doi: 10.3145/epi.2020.ene.03

Paiva, J. C., Leal, J. P., and Figueira, A. (2022). Automated assessment in computer science education: a state-of-the-art review. *ACM Trans. Comput. Educ.* 22:3513140. doi: 10.1145/3513140

Pranckutė, R. (2021). Web of science (wos) and scopus: the titans of bibliographic information in today's academic world. *Publications* 9:1. doi: 10.3390/publications9010012

Qadir, J., Yau, K.-L. A., Ali Imran, M., and Al-Fuqaha, A. (2020). "Engineering education, moving into 2020s: Essential competencies for effective 21st century electrical & computer engineers," in *2020 IEEE Frontiers in Education Conference (FIE)* (Uppsala: IEEE), 1–9.

Qian, Y., and Choi, I. (2023). Tracing the essence: ways to develop abstraction in computational thinking. *Educ. Technol. Res. Dev.* 71, 1055–1078. doi: 10.1007/s11423-022-10182-0

Selwyn, N., Hillman, T., Bergviken-Rensfeldt, A., and Perrotta, C. (2023). Making sense of the digital automation of education. *Postdigit. Sci. Educ.* 5, 1–14. doi: 10.1007/s42438-022-00362-9

Shen, J., Chen, G., Barth-Cohen, L., Jiang, S., and Eltoukhy, M. (2022). Connecting computational thinking in everyday reasoning and programming for elementary school students. *J. Res. Technol. Educ.* 54, 205–225. doi: 10.1080/15391523.2020.1834474

Shin, N., Bowers, J., Krajcik, J., and Damelin, D. (2021). Promoting computational thinking through project-based learning. *Disciplinary Interdiscipl. Sci. Educ. Res.* 3:7. doi: 10.1186/s43031-021-00033-y

Shin, N., Bowers, J., Roderick, S., McIntyre, C., Stephens, A. L., Eidin, E., et al. (2022). A framework for supporting systems thinking and computational thinking through constructing models. *Instruct. Sci.* 50, 933–960. doi: 10.1007/s11251-022-09590-9

Silva, L., Mendes, A. J., and Gomes, A. (2020). "Computer-supported collaborative learning in programming education: a systematic literature review," in *2020 IEEE Global Engineering Education Conference (EDUCON)* (Porto: IEEE), 1086–1095.

Srinivasa, K., Kurni, M., and Saritha, K. (2022). "Computational thinking," in *Learning, Teaching, and Assessment Methods for Contemporary Learners: Pedagogy for the Digital Generation* (Cham: Springer), 117–146.

Sung, Y.-T., Chang, K.-E., and Liu, T.-C. (2016). The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Comp. Educ.* 94, 252–275. doi: 10.1016/j.compedu.2015.11.008

Thuné, M., and Eckerdal, A. (2018). Analysis of students' learning of computer programming in a computer laboratory context. *Eur. J. Eng. Educ.* 44, 1–18. doi: 10.1080/03043797.2018.1544609

Valente, A., Holanda, M., Mariano, A. M., Furuta, R., and Da Silva, D. (2022). "Analysis of academic databases for literature review in the computer science education field," in *2022 IEEE Frontiers in Education Conference (FIE)* (Uppsala: IEEE), 1–7.

Vesin, B., Mangaroska, K., and Giannakos, M. (2018). Learning in smart environments: user-centered design and analytics of an adaptive learning system. *Smart Learn. Environm.* 5:24. doi: 10.1186/s40561-018-0071-0

Vidal-Silva, C., Cárdenas-Cobo, J., Tupac-Yupanqui, M., Serrano-Malebrán, J., and Sánchez Ortiz, A. (2024). Developing programming competencies in school-students with block-based tools in chile, ecuador, and peru. *IEEE Access* 12, 118924–118936. doi: 10.1109/ACCESS.2024.3449228

Xie, B., Loksa, D., Nelson, G., Davidson, M., Dong, D., Kwik, H., et al. (2019). A theory of instruction for introductory programming skills. *Com. Sci. Educ.* 29, 205–253. doi: 10.1080/08993408.2019.1565235

Yadav, A., and Berthelsen, U. (2021). *Computational Thinking in Education: A Pedagogical Perspective*. London: Routledge.

Younis, A., Sunderraman, R., Metzler, M., and Bourgeois, A. (2021). Developing parallel programming and soft skills: A project-based learning approach. *J. Parallel Distrib. Comp.* 158, 151–163. doi: 10.1016/j.jpdc.2021.07.015

Yusuf, A., and Noor, N. M. (2023). Research trends on learning computer programming with program animation: a systematic mapping study. *Comp. Appl. Eng. Educ.* 31, 1552–1582. doi: 10.1002/cae.22659

Zhu, J., and Liu, W. (2020). A tale of two databases: the use of web of science and scopus in academic papers. *Scientometrics* 123:8. doi: 10.1007/s11192-020-03387-8