Article

Bibliometric Analysis of Automated Assessment in Programming Education: A Deeper Insight into Feedback

José Carlos Paiva [1,2,*], Álvaro Figueira [1,2] and José Paulo Leal [1,2]

[1] CRACS – INESC TEC, 4169-007 Porto, Portugal
[2] Department of Computer Science, Faculty of Sciences, University of Porto, 4169-007 Porto, Portugal
* Correspondence: jose.c.paiva@inesctec.pt (J.C.P.)

Abstract: Learning to program requires diligent of practice and creates room for discovery, trial and error, debugging, and concept mapping. Learners must walk this long road themselves, supported by appropriate and timely feedback. Providing such feedback in programming exercises is not a humanly feasible task. Therefore, the early and steadily growing interest of computer science educators in automated assessment of programming exercises is not surprising. Automated assessment of programming assignments has been an active area of research for over a century, and interest in it continues to grow as it adapts to new developments in Computer Science and the resulting changes in educational requirements. It is therefore of paramount importance to understand the work that has been done, who has done it, its evolution over time, the relationships between publications, its hot topics, and open problems, among others. This paper presents a bibliometric study of the field with a particular focus on the issue of automatic feedback generation, using literature data from the Web of Science Core Collection. It includes a descriptive analysis using various bibliometric measures and data visualizations on authors, affiliations, citations, and topics. In addition, we perform a complementary analysis focusing only on the subset of publications on the specific topic of automatic feedback generation. The results are highlighted and discussed.

Keywords: automated assessment; programming education; programming exercises; computer science; bibliometrics; data visualizations; feedback

Citation: Paiva, J.C.; Figueira, Á.; Leal, J.P. Bibliometric Analysis of Automated Assessment in Programming Education: A Deeper Insight into Feedback. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Practice is the key to learning how to program. Practical programming skills can only be acquired through extensive and varied experience in solving programming challenges [1,2]. Such an experience should provide the learner with room for discovery, trial and error, debugging, and concept generation while supporting the learner with individualized, accurate, rich, and rapid feedback to unlock their progress. Obviously, feedback that meets these requirements cannot be guaranteed by a human instructor [3,4]. Automated assessment tools for programming tasks have emerged as a solution to this problem. They have been part of Computer Science (CS) education almost since learners began being asked to develop software, and their value is already unanimously recognized by practitioners. Nevertheless, interest in assessing various program properties (e.g., quality, behavior, readability, and security), in adapting feedback, and in developing better and more powerful tools has not waned since then [5].

Several studies have been carried out to synthesize the latest advancements in automated assessment for Computer Science (CS) education [4–7]. There are studies that focus on comparing the tools' features [7], exploring the methods and techniques applied in the different facets of the automated assessment tools [4,6], and other wider studies covering both [5]. To the best of the authors' knowledge, a single bibliometric study [8] has been conducted to analyze the quantitative aspects of scientific publications in the area and their relationshibibliops [9]. Such a study identifies the authors currently worth following and their affiliations, explores the evolution of publications and citations over time, establishes relationships between emerging topics in publications, computes the co-occurrence of topics and corresponding clusters, builds the citation networks, and elects the research trends.

This paper is an extension of the bibliometric study of Paiva et al. [8], previously introduced. This extension is twofold. Firstly, we extend the time span of the bibliometric research up to the end of the past year (2022). Lastly, we present a complementary bibliometric analysis focusing on a specific topic within the area, automatic feedback generation. Our goal is to answer the following six groups of research questions, considering the interval 2010-2022. The first group of research questions, **RQ1**, aims to summarize the collected data, including the annual scientific production (**RQ1-1**); the average time interval for a new publication to get the first citation (**RQ1-2**); and the main journals/conferences to find literature in the area (**RQ1-3**). The second group (**RQ2**) concerns authors. It aims to find out the common team size per publication (**RQ2-1**), the most productive, active, and impactful authors during the time span (**RQ2-2**), whether those authors publish alone or in groups (**RQ2-3**), the most evident collaboration partnerships (**RQ2-4**), and the main affiliations (**RQ2-5**). The third group (**RQ3**) targets citations, in particular, we want to identify the most influential (**RQ3-1**) and the most relevant (**RQ3-2**) co-citations. The fourth group (**RQ4**) investigates the topics being discussed in publications, including which are the basic, niche, motor, and emerging (**RQ4-1**), their evolution during the analyzed time span (**RQ4-2**), the frequent terms being used (**RQ4-3**), and whether those vary during the years (**RQ4-4**). The fifth group (**RQ5**) concentrates on feedback-specific questions, particularly, we want to know what are the active sub-topics (**RQ5-1**) and research lines (**RQ5-2**). Table 1 lists the research questions addressed in this study.

Table 1. List of research questions addressed in this study

Group	No.	Question
RQ1	1	What is the annual scientific production?
	2	What is the average time interval for a new publication to get the first citation?
	3	Which are the main journals/conferences to find literature in the area?
RQ2	1	What's the common team size per publication?
	2	Which are the most productive, active, and impactful authors?
	3	Do those authors publish alone or in group?
	4	Which are the most evident author collaboration partnerships?
	5	What are the authors' main affiliations?
RQ3	1	Which are the most influential citations?
	2	Which are the most relevant co-citations?
RQ4	1	Which are the basic, niche, motor, and emerging topics?
	2	How did topics evolved during the analyzed time span?
	3	Which are the frequent terms being used?
	4	How is the yearly frequency of the most frequent terms?
RQ5	1	Which are the active sub-topics within feedback?
	2	Which are the active research lines within feedback?

The remainder of this paper is organized as follows. Section 2 presents the methodology used to conduct this study. Section 3 presents the results of the bibliometric analysis and answers each of the research questions. Section 4 discusses the results, and compares them to recent literature review [5]. Finally, Section 5 summarizes the major contributions of this study.

2. Methodology

The data for this study has been collected from the Web of Science (WoS) Core Collection, during the third week of March 2023. To this end, a query [10] has been built to search all fields of a publication for the following combination of keywords: (automatic OR automated) AND (assessment OR evaluation OR grading OR marking) AND (programming OR computer science OR program).

The query includes a filter to limit results to those published from 2010 up to end of 2022. In addition to that, two refinements were needed. The first to narrow down search results to the adequate WoS categories for this area, namely: Computer Science Information Systems, Computer Science Artificial Intelligence, Computer Science Interdisciplinary Applications, Computer Science Software Engineering, Education Educational Research, Education Scientific Disciplines, Multidisciplinary Sciences, and Education Special. Even though some of these categories may still include out-of-scope publications, excluding them could result in the loss of important publications. The result was a set of 16471 publications.

For these publications, we have extracted their full record, including cited references. A total of thirty-three BibTeX exports were necessary to obtain the data from all the 16471 publications, due to the limitations of WoS on the number of records allowed to be exported in a single request (in these conditions, the limit is 500). Finally, the BibTeX files obtained have been merged into a single BibTeX file. Having this set in a single file, we have imported it into R using bibliometrix [11] – an open-source R-tool for quantitative research in scientometrics and bibliometrics, and proceeded with a pre-processing phase. This phase aims to identify the relevant publications for analysis by applying inclusion/exclusion criteria identical to that used by Paiva et al. [5], after carefully reading the titles and abstracts of each paper. The result from this phase is a set of 779 publications, which we used for further analysis.

After filtering, we extract the 2-grams terms from both the abstract and the title into two new columns of the dataset, reducing a few synonyms to a single term (e.g., “grading tool”, “assessment tool”, and “marking tool” to “aa tool”) and eliminating general terms (e.g., “automated assessment” or “computer science”). Then, the resulting dataset is analyzed using R and traditional data processing and visualization packages, except for bibliometrix package [11]. Bibliometrix is an R package specifically designed to support bibliometric analyses. In particular, it can import bibliographic data both directly and indirectly from SCOPUS, Clarivate Analytics’ WoS, PubMed, Digital Science Dimensions and Cochrane databases, and perform a wide range of bibliometric analysis methods, including citation, co-citation, bibliographic coupling, scientific collaboration, co-word, and co-authorship analysis.

From these analyses, some visualizations of the data have been produced using R and a few libraries, particularly, ggplot, VOSViewer, and Bibliometrix. Additional visualizations were produced in Biblioshiny, a shiny app providing a web-interface for bibliometrix. Finally, we have curated a bibliometric summary and compiled a brief report of the findings, which we interpreted to answer the research questions presented in Subsection 1. Figure 1 presents the steps performed in this research.

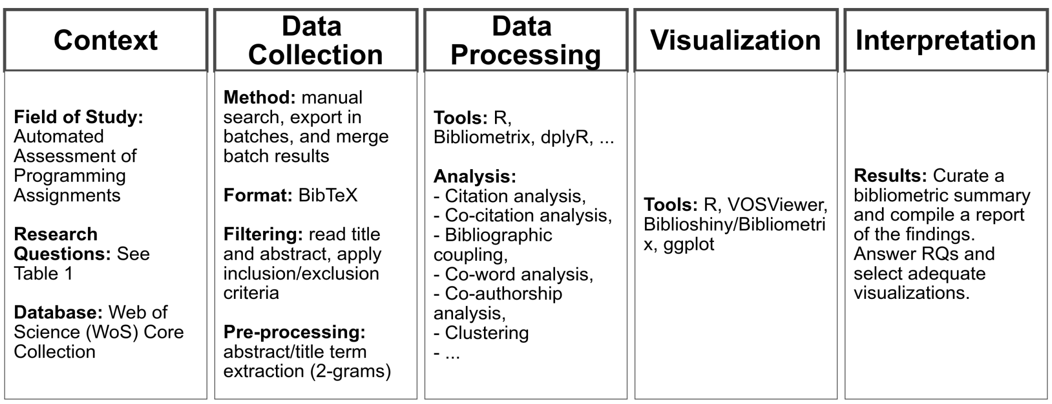

Figure 1. Schema of the steps performed in this research

3. Results

The results of the analysis are detailed in this section, where each subsection answers one of the groups of research questions presented in Section 1. Subsection 3.1 provides a summary of the data used in the analysis, including answers to **RQ1**. Subsection 3.2 encompasses the results related to the authors' analysis (i.e., **RQ2**). Subsection 3.3 demonstrates the results regarding the analysis of citations (i.e., **RQ3**). Subsection 3.4 presents answers to **RQ4**, which pertains to topics and keywords. Subsection 3.5 presents answers to **RQ5**, which contains questions about the feedback on automated assessment of programming assignments.

3.1. Data Summary

Literature on automated assessment of programming assignments has had a continually growing interest during the analyzed period (2010-2022), as depicted in Figure 2 through a visualization of the number of publications per year with a linear trend and the associated confidence interval. The trend line reflects a growth rate of approximately 7.1% in the annual scientific production during the timespan 2010-2022. Nevertheless, a slight decrease in the number of publications between 2019 and 2020 is noticeable, an exceptional situation that can be associated with the COVID-19 pandemic crisis. The years with the highest number (99) of publications, i.e., the peak years, were 2018 and 2021. This responds to **RQ1-1**.

Each of the collected documents was cited by an average of 7.254 other publications, with an average rate of 1.045 per year. Thus, in response to **RQ1-2**, it takes an average of 11.483 months to receive the first citation. The year with the most citations per publication on average was 2014, with a mean of 14.31 citations per publication. However, the year in which publications were most cited per citable year (i.e., following years captured in the analysis) was 2017, having an average of 1.81 citations per publication and citable year. Figure 3 shows the average citations of a document per citable year, for each year of publication. For example, a publication of 2010 (i.e., with 13 citable years) has 0.624 citations on average per year (the lowest of this dataset if we exclude the last year, whose information may not be complete).

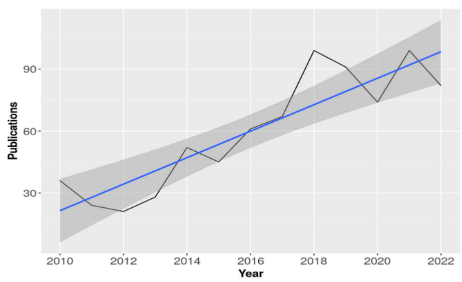

Figure 2. Number of publications per year with the linear trend (blue line) and its confidence interval

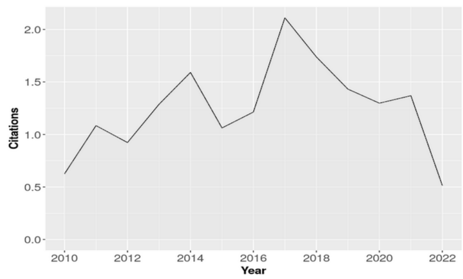

Figure 3. Average publication citations per document in citable years

The set of 779 selected publications consists of 7 review papers, 222 journal articles, 545 proceedings papers, and 5 classified as other editorial material. These come from 455 different sources, including journals, and books, among others. The top-20 publication sources (**RQ1-3**), presented in the tree-map of Figure 4, account for almost one-fifth of the total publications. The Proceedings of the 51st ACM Technical Symposium on Computer Science Education is the source with the highest number of articles collected (15), followed by ACM Special Interest Group on Programming Languages (SIGPLAN) Notices with 12 publications, ACM Transactions on Software Engineering and Methodology with 11 publications, and Information and Software Technology with 10. Computers & Education, Empirical Software Engineering, Journal of Systems and Software, Science of Computer

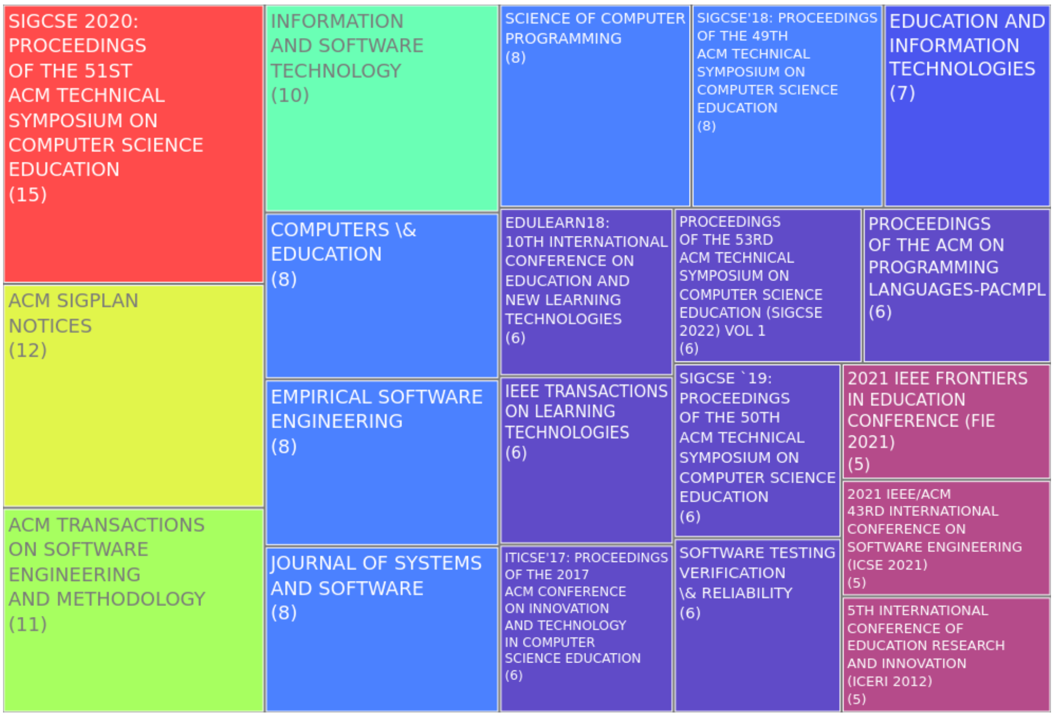

Figure 4. Top-20 sources of publications in a tree-map

Programming, and the Proceedings of the 49th ACM Technical Symposium on Computer Science Education, with 8 publications each, complete the top-5 sources.

3.2. Authors

The set of publications selected for analysis contains documents from 2120 distinct authors, of which 56 authored at least one of the 62 single-authored documents that are part of this set. The average number of authors per document is 3.26, whereas excluding the 62 single-authored publications, there are 3.47 co-authors per document on average. From these co-authorships, we could identify that 20.28% are international co-authorships. Figure 5 shows a histogram of the number of authors per publication that responds to **RQ2-1**. The most common are publications with 2 to 4 authors (72.5% of all publications), being 3-author publications the ones in the majority.

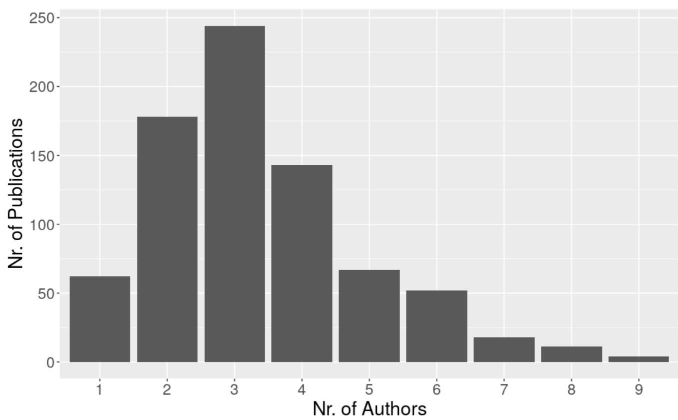

Figure 5. Distribution of publications by the number of authors

In respect to the "most prolific authors" (i.e., the authors who have made more publications) as asked in **RQ2-2**, Figure 6 shows the top-10, sorted in descending order from top to bottom, of authors who have made more contributions to the field and, for those, the number of publications and citations per year. From this perspective, the authors who are more active recently, such as Fraser G. and Edwards S. H., and those who were more active at the beginning of the timespan, such as Kim M., Queirós R., and Leal J. P., are easier to identify. Nevertheless, the most impactful works are that of Monperrus M., who studies mostly automated program repair techniques. The work of Fraser G., which concentrates on software testing techniques, and Kim D., which investigates techniques for automated generation of feedback, complete the podium regarding authors' impact. This can be confirmed by measuring the authors' h-index (6, 5, and 5, respectively).

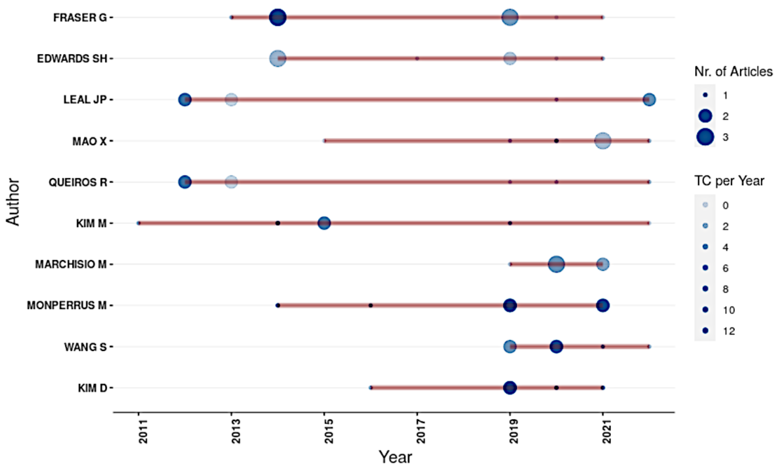

Figure 6. Productivity of authors over time (TC stands for Times Cited)

Having the most prolific authors, we construct a histogram of the number of authors per publication for each of them separately to answer **RQ2-3**. Figure 7 illustrates the result. Among the most prolific authors, the only authors who have worked alone are Queiros R. (1) and Monperrus M. (1), while all others have no single-authored publications. Nevertheless, Edwards S. H. and Marchisio M. publish mostly in small groups with one or two co-authors. Interestingly, Mao X., Wang S., and Kim D. have only worked in large groups of 4 or more authors.

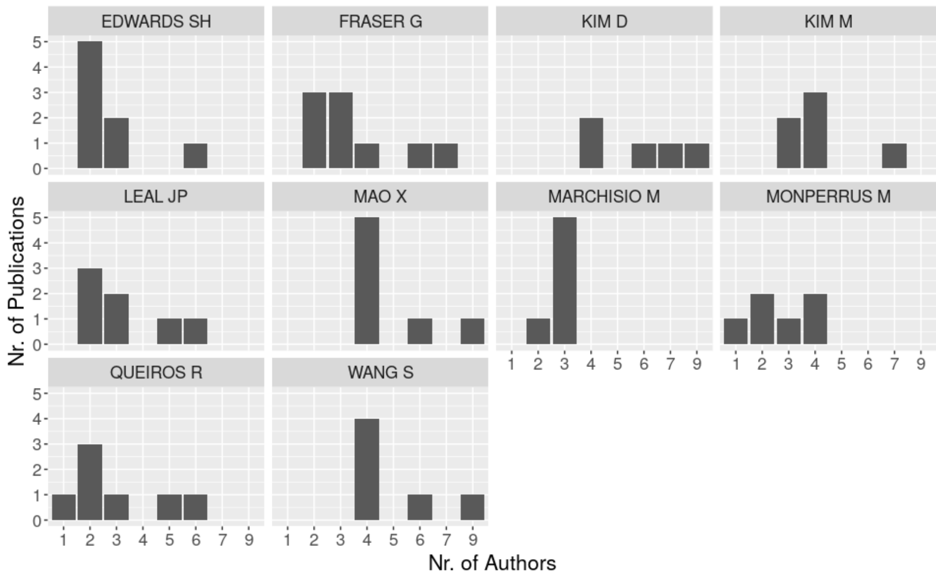

Figure 7. Number of authors per publication for the most productive authors

There are several researchers collaborating on projects that publish together very often. **RQ2-4** aims to identify them. To this end, we have created a graph of the authors' collaboration networks, presented in Figure 8. Bolder edges indicate a stronger partnership relation between authors, i.e., the authors participated together in the same publications more often. For instance, Leal J. P. and Queiros R. authored many of their publications together. The same for Tonisson E. and Sade M., who also collaborated less frequently with Lepp M. and Luik P.

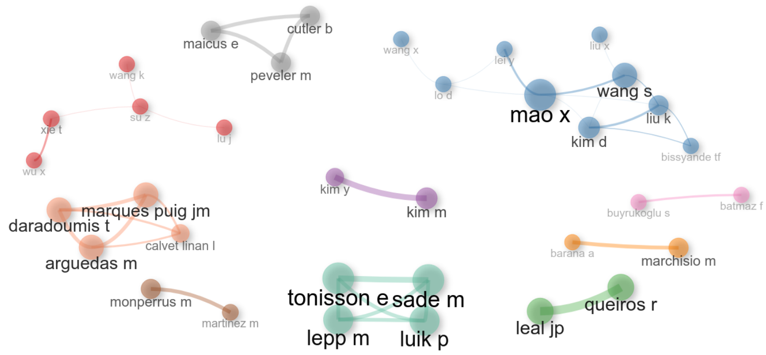

Figure 8. Authors' collaboration networks

Regarding **RQ2-5** which aims to find out the main affiliations from the authors, there are 636 distinct identified affiliations within the collected publications. Note that a publication can count to more than one affiliation if it involves either authors with multiple affiliations or documents with multiple authors resulting from a collaboration between different institutions. The top-20 most prolific affiliations, alone, account for more than 39% of the identified affiliations. Carnegie Mellon University is the institution with the most publications (25), followed by North Carolina State University (20), and the University of Porto (19). The Nanjing University and the University of Tartu, both appearing with 18 publications, complete the top-5. The top-20 most prolific affiliations are presented in Figure 9, which alone account for more than 37.5% of the identified affiliations.

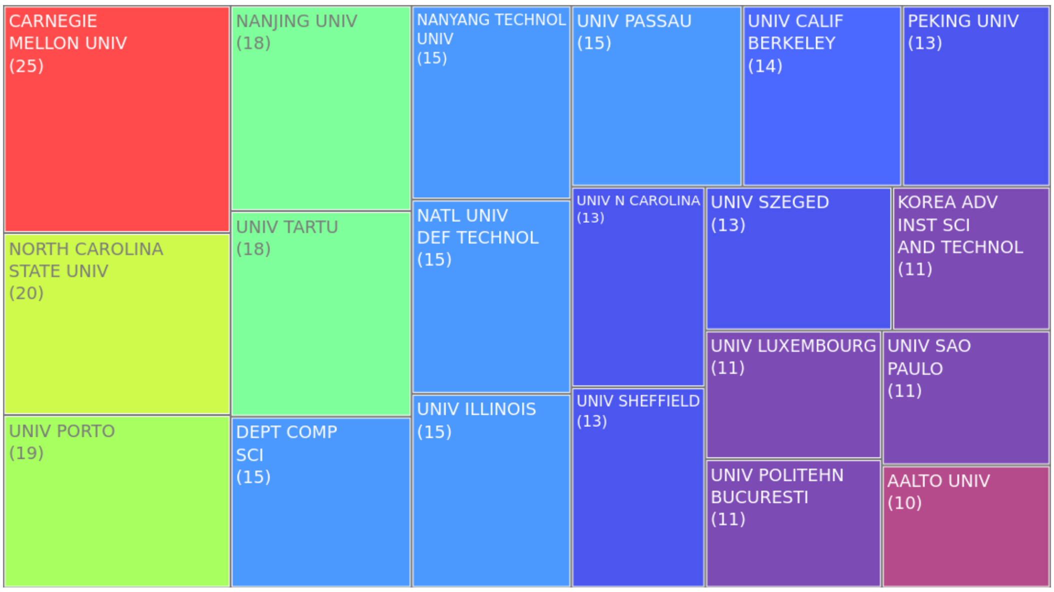

Figure 9. Top-20 authors' affiliations by productivity in a tree-map

3.3. Citations

The most influential publications (**RQ3-1**) are those having more citations, but should also take into consideration the year of publication and the area of the publication citing it (i.e., a citation from a publication in the same area has a different weight from one in another area). To handle the former, we used the Normalized Citation Score (NCS) of a document, which is calculated by dividing the actual number of cited publications by the expected citation rate for publications of the same year. To address the latter, we have made a twofold answer.

On the one hand, the local NCS (i.e., citations within the collected data) determines the most influential publications within the area. The top-5 publications under such conditions are: "Context-Aware Patch Generation for Better Automated Program Repair" by Wen et al. [12]; "Marking student programs using graph similarity" by Naudé et al. [13]; "A distributed system for learning programming on-line" by Verdú et al. [14]; "TBar: revisiting template-based automated program repair" by Liu et al. [15]; and "A system to grade computer programming skills using machine learning" by Srikant S. [16].

On the other hand, looking at all the citations provides a global perspective on the most influential publications. The top-5 publications in this regard are: "PerfFuzz: automatically generating pathological inputs" by Lemieux et al. [17]; "Automated Feedback Generation for Introductory Programming from Assignments" by Singh et al. [18]; "Context-Aware Patch Generation for Better Automated Program Repair" by Wen et al. [12]; "Automated Assessment in Computer Science Education: A State-of-the-Art Review" by Paiva et al. [5]; and "Precise Condition Synthesis for Program Repair" by Xiong et al. [19].

When two publications are cited together by other documents (co-citations) frequently, it indicates they are likely to address the same topic, i.e., they are semantically related. **RQ3-2** aims to identify the most relevant of those co-citations. The answer is provided in the historiographic map proposed by E. Garfield [20] (see Figure 10), which presents a chronological network map of the most relevant co-citations from a bibliographic collection. This map identifies six separate groups corresponding to different topics, namely: **Group I (Light Blue)** includes advancements on automated program repair techniques [12,15,21]; **Group II (Pink)** contains two studies analyzing difficulties faced by novice programmers in automated assessment tools [22,23]; **Group III (Green)** encompasses works on automated feedback for CS projects [24,25]; **Group IV (Yellow)** includes publications on automated program repair techniques and tools [26,27]; **Group V (Red)** captures works exploring the automated assessment of the computational thinking skills of novice programmers [16,28, 29]; **Group VI (Blue)** captures a group of works aiming to improve feedback on automated assessment [13,30,31].

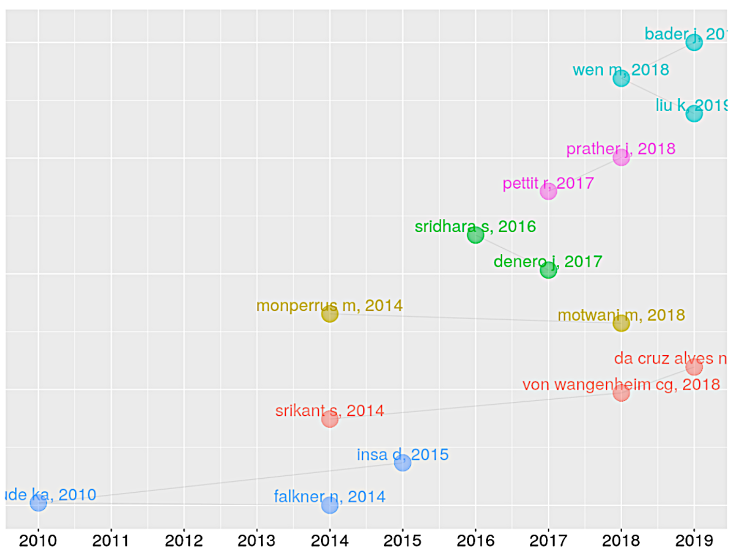

Figure 10. Historiographical representation of a network map of most relevant co-citations

3.4. Topics and Keywords

Information about current issues, trends, and methods in the field can be derived from keywords. Keywords are mandatorily provided by authors for each publication but are also assigned automatically by the indexing database, or even extracted as n-grams from the title or abstract. Therefore, for the group of research questions **RQ4** these are the properties that are the subject of analysis. For the first question of the group (**RQ4-1**), the answer is provided in Figure 11, through a thematic map based on the analysis of co-word networks and clustering using the authors' keywords. This approach is similar to the proposal of Cobo et al. [32]. It identifies four types of topics (themes) based on density (i.e., degree of development) and centrality (i.e., degree of relevance), namely: emerging or declining (low centrality and low density), niche (low centrality and high density), motor (high centrality and high density), and basic (high centrality and low density) topics. In emerging or declining topics, a cluster with "Android" is worth noticing as it is an indicator of the increasing interest in automatic assessment of mobile development assignments. Niche themes include some interesting topics such as Graph Similarity, which is a technique used in automated assessment for comparing source code semantically (e.g., compare to a known solution and derive feedback), and Automatic Question Generation as well as other topics related to tools of the domain itself (e.g., virtual programming lab, systems, and framework). Motor themes include topics such as Fault Localization, Debugging, Program Analysis, and Learning Analytics. Finally, among the basic themes, the clusters of Static Analysis – analyzing source code rather than its runtime behavior –, Automated Program Repair – a technique used to automatically correct programs, which is being applied to generate feedback –, and Test Generation are the most notable.

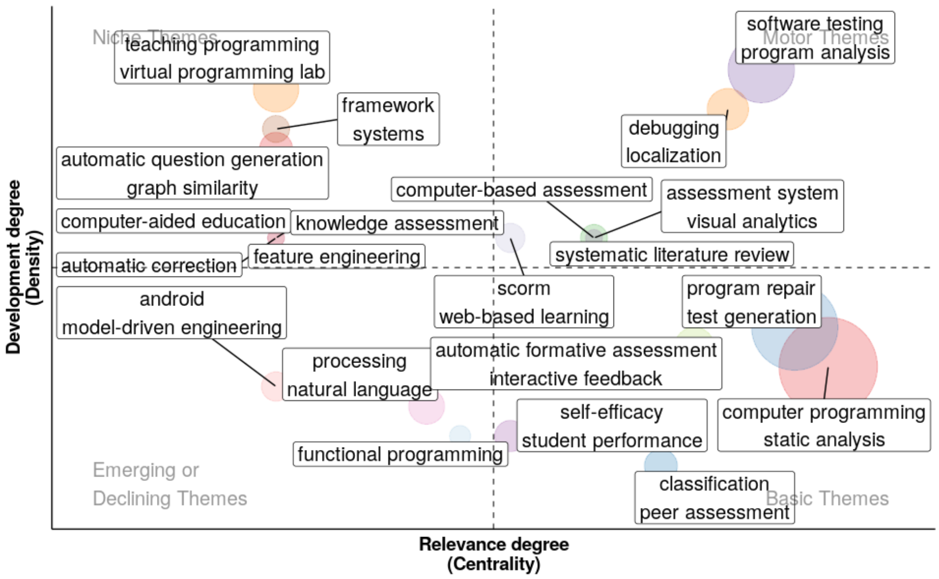

Figure 11. Thematic map based on authors' keywords

To answer **RQ4-2** and after, the analysis focuses on 2-grams extracted from the abstract. Figure 12 divides the decade into three equal-length sections (2010-2014, 2015-2018, and 2019-2022) and shows the thematic evolution between the three sections, based on analysis of the co-word network and the clustering of the authors' keywords [32]. From the first slice, it is noticeable the high interest in Test Generation and the evident importance of Machine Learning in the area already. The second slice includes a wide range of topics with a strong relation to attempts to improve feedback, such as Static Analysis, Automated Program Repair (e.g., apr techniques and program repair), and Symbolic Execution. Finally, in the third slice, the emphasis on topics such as Static Analysis, Automated Program Repair, and Test Generation is maintained.

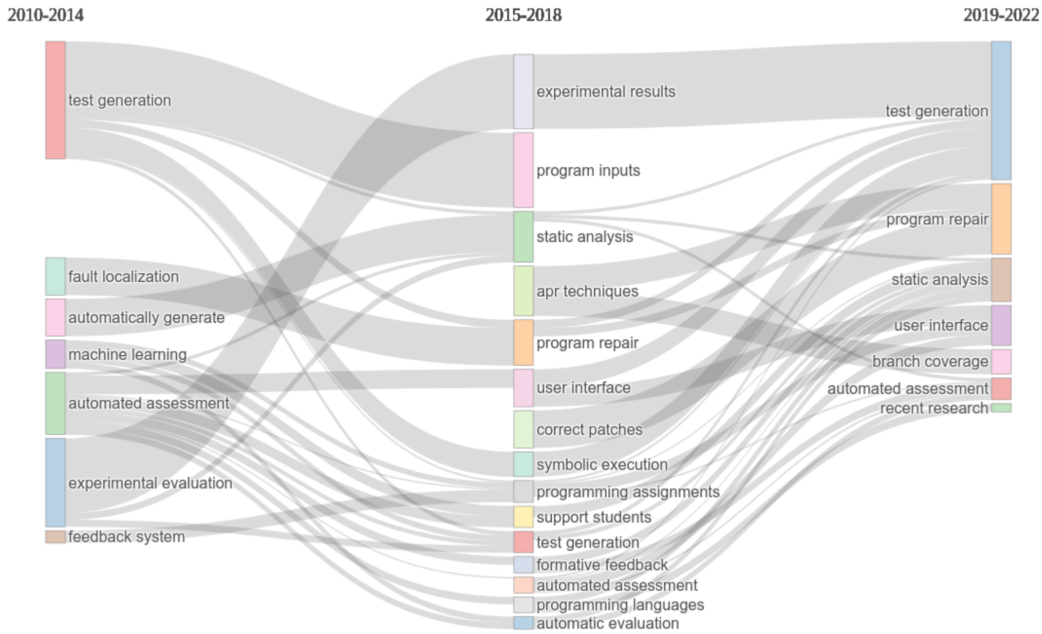

Figure 12. Thematic evolution based on authors' keywords

As for **RQ4-3**, Figure 13 presents a conceptual structure map created using Multiple Correspondence Analysis (MCA) – a data analysis method to measure the association between two or more qualitative variables – and Clustering of a bipartite network of the extracted terms. Using this approach, 2-grams are divided into four clusters, which can be described as follows: **Group I (Blue)** includes terms related to feedback and learning analytics; **Group II (Green)** seems related to static analysis; **Group III (Red)** contains 2-grams related to fault localization and test generation (e.g., fault localization and test generation); and **Group IV (Purple)** captures terms related to automated program repair (e.g., repair apr and program repair).

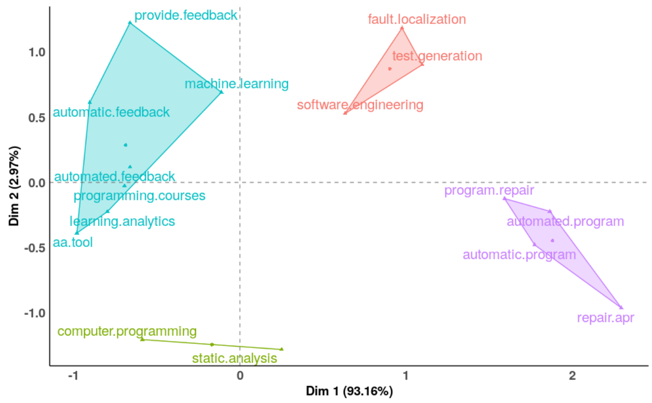

Figure 13. Conceptual structure map of abstract 2-grams obtained through MCA

With respect to **RQ4-4**, Figure 14 shows the ten most frequent abstract 2-grams by year, in which colors vary from blue (low occurrence in publications) to red (high occurrence), i.e., using a color temperature scale. The increasing interest in Static Analysis, Automated Program Repair, and Automated Test Generation is readily apparent. Although less visible, Machine Learning and Learning Analytics have also increased slightly over the years. This indicates a large growing interest in improving automated feedback generation, as most topics gaining popularity are related to source code analysis (Static Analysis and Machine Learning – in the current context) and fixing (Automated Program Repair, Automated Test Generation – including counter-example –, Fault Localization, and Machine Learning) techniques. Moreover, feedback for teachers, through Learning Analytics, seems to be now a topic of interest within the area of Automated Assessment. Note that, for this visualization, the set of generated 2-grams has been preprocessed to remove common terms (e.g., science, introductory, programming, paper, work, result, etc) and match synonyms (e.g., apr tool, repair tool, and program repair count for the same 2-gram). Each publication counts at maximum once for any 2-gram.

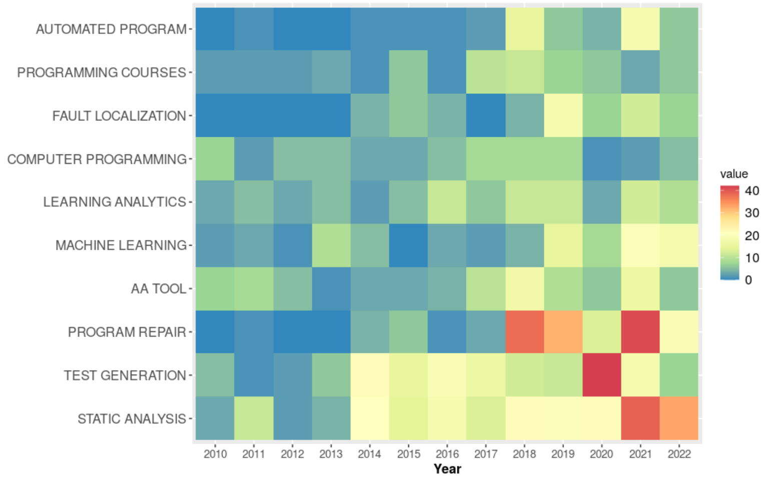

Figure 14. Top-10 most frequent abstract 2-grams by year

3.5. Feedback

The fifth group of research questions **RQ5** targets the subset of publications related to feedback. To this end, we filtered the dataset to include only publications whose keywords, title, or abstract contain the term “feedback”. The result is a set of 340 publications from 980 distinct authors, with an annual scientific production growth rate of 12.92%.

On this set of publications, we aim to re-analyze the topics to discover what exactly is being discussed about automatic feedback generation for programming assignments (**RQ5-1**). Figure 15 shows the most impactful/central clusters of 2-grams extracted from the abstract of these publications. The **Purple** cluster includes Automated Program Repair and Fault Localization. Cluster **Red** and **Blue** refer mostly to common terms in the area of automated assessment of programming assignments, such as “programming assignments”, “automated assessment”, or “assessment tool”. Nevertheless, the **Blue** cluster also includes terms related to the use of data for feedback purposes, namely “data-driven feedback” and “heatmaps”. Finally, the **Green** cluster includes some interesting branches such as “test generation” and “symbolic execution” (e.g., for generating counter-example test cases), and interactive feedback.

RQ5-2 asks what are the research lines within the feedback topic. To answer this question, we have rebuilt the historiographic map proposed by E. Garfield [20] for the new data. Figure 16 presents the result. The following relevant co-citations networks: **Green** captures feedback on exercises to stimulate the computational thinking skills of novice programmers [28,29]; **Purple** involves works evaluating the effects of feedback [33–36]; **Light Blue** encompasses works about automated program repairing [12,15,21,37]; **Pink** contains works comparing human and machine-generated feedback [38,39]; **Light Green** involves works providing feedback on student-developed test cases [40,41]; **Orange** has another series of works providing feedback on the accuracy of student-developed test cases [42–44]; **Red** includes efforts on exploring patterns of source code for feedback purposes [45,46]; **Blue** includes publications evaluating automated program repair techniques [26,27,47].

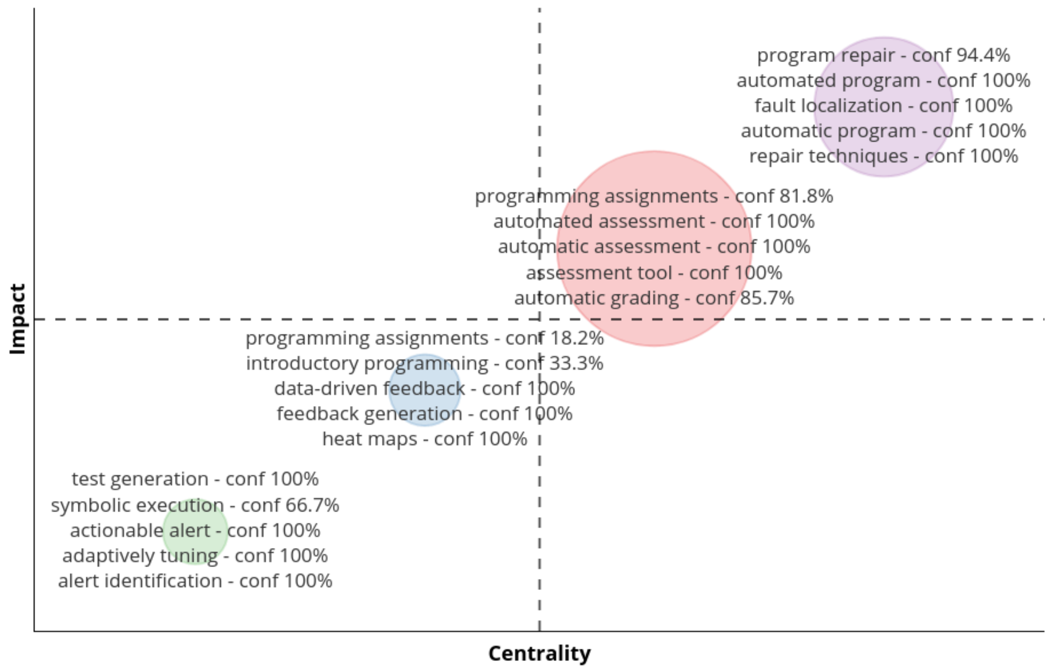

Figure 15. Frequent terms under feedback topic

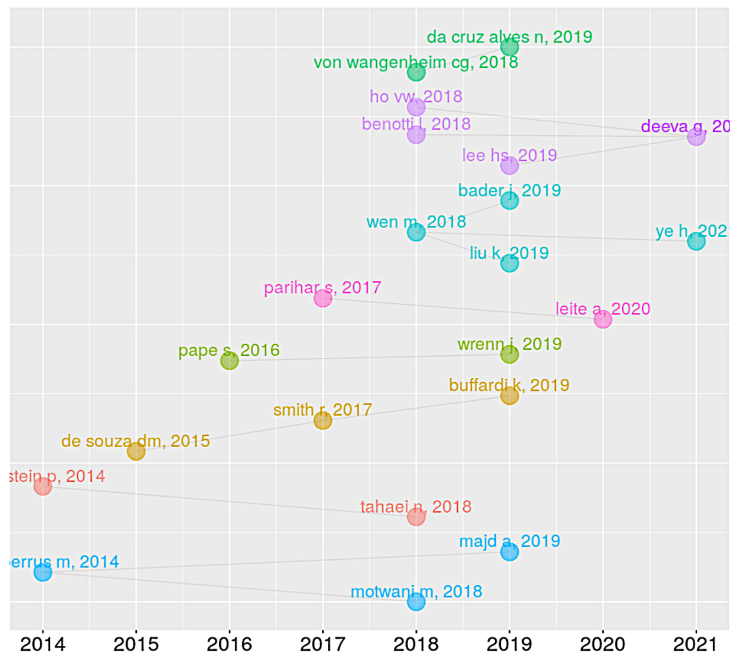

Figure 16. Historiographical representation of a network map of most relevant co-citations under feedback topic

4. Discussion

Automated assessment of programming assignments is a research area with several years of investigation, but still, an increasing research interest as demonstrated by the significant and growing number of publications in the analyzed years. The only exception coincides with (and can be justified by) the COVID-19 pandemic situation, which occurred between the start of 2020 and the start of 2022. Most of these publications appear in journals and conference proceedings, with shares of nearly 30% and 70%, respectively. The number of citations has maintained a nearly constant rate over the years (see Figure 3).

4.1. Authors

Authors of publications in this area are typically active Computer Science educators. They explore new ways of facilitating their tasks of creating and assessing programming assignments and, at the same time, provide students with a richer and more personalized practicing experience to soften the difficulties of learning how to program. The research teams are mostly small, with 2 to 4 authors, and publish several times together. For instance, Queirós R. and Leal J. P. have authored 6 of their 7 publications in the area together (one of each is separate). Nevertheless, works with more than 4 authors are common in publications introducing techniques from static source code analysis [12,48,49].

4.2. Citations

The top-5 most influential citations, both local and global, have been identified, having one common publication. The total 9 most influential citations include a systematic literature review [5], an assessment tool [14], a technique for the assessment using graph similarity [13], a technique for the assessment using machine learning on graph representations of source code [16], three techniques for automated program repair [12,15,19], a worst-case test generation approach [17], and a technique to generate feedback given a reference solution and an error model [16]. This highlights the great interest among researchers in generating better feedback, as 7 out of 9 most influential citations have that end goal.

4.3. Topics

The systematic literature review on automated assessment by Paiva et al. [5] is the most closely related to the one and recent review. This review identified a new era of automated assessment in Computer Science, the era of containerization, among other interesting findings. In particular, the growing interest in static analysis techniques to assess not only the correct functionality of a program but also the code quality and presence of plagiarism. Furthermore, it notices the efforts towards better feedback primarily by introducing techniques from other research areas, such as automated program repair, fault localization, symbolic execution, and machine learning. Regarding automated assessment tools, more than half of the mentioned tools are open source. Finally, the increasing interest in incorporating Learning Analytics into automated assessment tools to help teachers understand student difficulties is also mentioned. A technical report by Porfirio et al. [50] presents a systematic literature mapping of the research literature on automatic source code evaluation until 2019, which also had similar findings. In particular, it (1) shows the increasing number of publications; (2) notices a few attempts to extract knowledge and visualize information about students from data produced during the automated assessment of source code (i.e., first attempts on Learning Analytics); and (3) demonstrates that functional correctness is the aspect receiving most attention.

The responses given in subsection 3.4 to research questions of group **RQ4** confirm most of the findings of previous works, namely the recent focus on static analysis approaches and the introduction of techniques from other research areas, such as automated program repair, fault localization, and machine learning. Traditional automated assessment based on running the program against a set of test cases is still the dominating strategy. Moreover, the high frequency of some keywords related to Learning Analytics corroborates the interest in integrating outcomes from this research area into automated assessment tools. Nevertheless, this research could not capture enough information to confirm the trend of containerization of automated assessment. As the conducted analysis had minimal human interference, if "docker" (or a related term) was neither a frequent keyword nor part of a frequent abstract 2-gram, then it was not identified. In contrast, in the aforementioned review [5], a number of publications were manually annotated with a predetermined set of tags after reading.

4.4. Feedback

Automatic feedback generation is the most explored topic in the area, representing more than 43% of the collected publications and with an annual scientific production growth rate of almost 6 percentage points higher than the area itself. Works under this topic range from the introduction of static analysis, machine learning, and source code analysis techniques to experiments assessing the quality of the tools/techniques against a dataset, experiments comparing the use of a tool/technique against either manual or before treatment feedback, and learning analytics approaches.

5. Conclusion

This paper presents a bibliometric study of the literature on automatic assessment in Computer Science from 2010 up to the end of 2022, based on the WoS Core Collection. The collected data shows the ever-increasing research interest in the area, particularly, in integrating and developing techniques to improve automatically generated feedback. Static analysis, machine learning, and source code analysis techniques used in other research areas opened room for improvement of current solutions, so it is important to continue pursuing this area in the coming years.

The analysis performed allowed us to answer all the research questions posed at the beginning of this study and presented in section 1. Results regarding topics are identical to those reported in a recently published systematic literature review on automated assessment in computer science [5] and, thus, validate each other. Novel results include, for instance, the identification of the most active/productive researchers in the field, their groups and affiliations, the collaborations between them, the most impactful publications, the evolution of important research lines, and the sources with most publications in the area. Moreover, the analysis of the subset of research related to automatic feedback generation allowed to identify the different branches being explored in this topic as well as the research lines.

Admittedly, this study has some limitations. In particular, the WoS Core Collection does not include publications from all sources. Second, the names of some authors and affiliations appear in different forms over the decade, which may introduce some bias into the analysis. In this case, the work of a database such as the WoS Core Collection to standardize affiliations and authors is important.

In the upcoming years we expect research in this area to continue growing. We foresee that the development of static analysis techniques to assess different aspects and types of programming assignments and the integration of source code analysis and machine learning techniques to improve automatically generated feedback will drive research in the area. Furthermore, we recommend another bibliometric study of this type (at least) in the next decade as, in such an active area, it is important to understand where research is heading to and for new researchers in the field to know the paths of research and authors to follow.

Author Contributions: Conceptualization, J.C.P., A.F., and J.P.L.; Data curation, J.C.P.; Formal analysis, J.C.P.; Funding acquisition, J.C.P.; Investigation, J.C.P.; Methodology, J.C.P., A.F., and J.P.L.; Project administration, J.C.P., A.F., and J.P.L.; Resources, J.C.P.; Software, J.C.P.; Supervision, A.F. and J.P.L.; Validation, J.C.P., A.F., and J.P.L.; Visualization, J.C.P., A.F., and J.P.L.; Writing—original draft, J.C.P.; Writing—review and editing, J.C.P., A.F., and J.P.L.. All authors read and agreed to the published version of the manuscript.

Funding: J.C.P.'s work is funded by the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology), Portugal – for the Ph.D. Grant 2020.04430.BD.

Informed Consent Statement: Not applicable.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical re-strictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Samples of the compounds ... are available from the authors.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	Linear dichroism

References

1. Robins, A.; Rountree, J.; Rountree, N. Learning and Teaching Programming: A Review and Discussion. *Computer Science Education* **2003**, *13*, 137–172, [https://doi.org/10.1076/csed.13.2.137.14200]. https://doi.org/10.1076/csed.13.2.137.14200.
2. Mouza, C.; Codding, D.; Pollock, L. Investigating the impact of research-based professional development on teacher learning and classroom practice: Findings from computer science education. *Computers & Education* **2022**, *186*, 104530. https://doi.org/https://doi.org/10.1016/j.compedu.2022.104530.
3. Saikkonen, R.; Malmi, L.; Korhonen, A. Fully Automatic Assessment of Programming Exercises. *SIGCSE Bull.* **2001**, *33*, 133–136. https://doi.org/10.1145/507758.377666.
4. Ala-Mutka, K.M. A Survey of Automated Assessment Approaches for Programming Assignments. *Computer Science Education* **2005**, *15*, 83–102. https://doi.org/10.1080/08993400500150747.
5. Paiva, J.C.; Leal, J.P.; Figueira, A. Automated Assessment in Computer Science Education: A State-of-the-Art Review. *ACM Trans. Comput. Educ.* **2022**, *22*. https://doi.org/10.1145/3513140.
6. Ihantola, P.; Ahoniemi, T.; Karavirta, V.; Seppälä, O. Review of recent systems for automatic assessment of programming assignments. In Proceedings of the Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10; ACM Press: Berlin, Germany, 2010; pp. 86–93. https://doi.org/10.1145/1930464.1930480.
7. Souza, D.M.; Felizardo, K.R.; Barbosa, E.F. A Systematic Literature Review of Assessment Tools for Programming Assignments. In Proceedings of the 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET); IEEE: Dallas, TX, USA, 2016; pp. 147–156. https://doi.org/10.1109/CSEET.2016.48.
8. Paiva, J.C.; Figueira, Á.; Leal, J.P. Automated Assessment in Computer Science: A Bibliometric Analysis of the Literature. In Proceedings of the Advances in Web-Based Learning – ICWL 2022; Anonymous., Ed.; Springer International Publishing: Cham, 2022; pp. 000–000.
9. Andrés, A. *Measuring Academic Research*; Chandos Publishing (Oxford): Witney, England, 2009.
10. Clarivate. Web of Science Core Collection. https://www.webofscience.com/wos/woscc/summary/f82ac75a-44c0-4873-a40d-c59e1e79ef4e-79ee7f28/relevance/1, 2022. Accessed on 19th Mar 2023.
11. Aria, M.; Cuccurullo, C. bibliometrix: An R-tool for comprehensive science mapping analysis. *Journal of Informetrics* **2017**, *11*, 959–975. https://doi.org/10.1016/j.joi.2017.08.007.
12. Wen, M.; Chen, J.; Wu, R.; Hao, D.; Cheung, S.C. Context-Aware Patch Generation for Better Automated Program Repair. In Proceedings of the Proceedings of the 40th International Conference on Software Engineering; Association for Computing Machinery: New York, NY, USA, 2018; ICSE '18, p. 1–11. https://doi.org/10.1145/3180155.3180233.
13. Naudé, K.A.; Greyling, J.H.; Vogts, D. Marking student programs using graph similarity. *Computers & Education* **2010**, *54*, 545–561. https://doi.org/10.1016/j.compedu.2009.09.005.
14. Verdú, E.; Regueras, L.M.; Verdú, M.J.; Leal, J.P.; de Castro, J.P.; Queirós, R. A distributed system for learning programming on-line. *Computers & Education* **2012**, *58*, 1–10. https://doi.org/10.1016/j.compedu.2011.08.015.
15. Liu, K.; Koyuncu, A.; Kim, D.; Bissyandé, T.F. TBar: Revisiting Template-Based Automated Program Repair. In Proceedings of the Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis; Association for Computing Machinery: New York, NY, USA, 2019; ISSTA 2019, p. 31–42. https://doi.org/10.1145/3293882.3330577.
16. Srikant, S.; Aggarwal, V. A System to Grade Computer Programming Skills Using Machine Learning. In Proceedings of the Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Association for Computing Machinery: New York, NY, USA, 2014; KDD '14, p. 1887–1896. https://doi.org/10.1145/2623330.2623377.
17. Lemieux, C.; Padhye, R.; Sen, K.; Song, D. PerfFuzz: Automatically Generating Pathological Inputs. In Proceedings of the Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis; Association for Computing Machinery: New York, NY, USA, 2018; ISSTA 2018, p. 254–265. https://doi.org/10.1145/3213846.3213874.
18. Singh, R.; Gulwani, S.; Solar-Lezama, A. Automated Feedback Generation for Introductory Programming Assignments. In Proceedings of the Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation; Association for Computing Machinery: New York, NY, USA, 2013; PLDI '13, p. 15–26. https://doi.org/10.1145/2491956.2462195.

19. Xiong, Y.; Wang, J.; Yan, R.; Zhang, J.; Han, S.; Huang, G.; Zhang, L. Precise Condition Synthesis for Program Repair. In Proceedings of the Proceedings of the 39th International Conference on Software Engineering. IEEE Press, 2017, ICSE '17, p. 416–426. https://doi.org/10.1109/ICSE.2017.45.
20. Garfield, E. Historiographic Mapping of Knowledge Domains Literature. *Journal of Information Science* **2004**, *30*, 119–145. https://doi.org/10.1177/0165551504042802.
21. Bader, J.; Scott, A.; Pradel, M.; Chandra, S. Getafix: Learning to Fix Bugs Automatically. *Proc. ACM Program. Lang.* **2019**, *3*. https://doi.org/10.1145/3360585.
22. Pettit, R.S.; Homer, J.; Gee, R. Do Enhanced Compiler Error Messages Help Students? Results Inconclusive. In Proceedings of the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2017; SIGCSE '17, p. 465–470. https://doi.org/10.1145/3017680.3017768.
23. Prather, J.; Pettit, R.; McMurry, K.; Peters, A.; Homer, J.; Cohen, M. Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools. In Proceedings of the Proceedings of the 2018 ACM Conference on International Computing Education Research; Association for Computing Machinery: New York, NY, USA, 2018; ICER '18, p. 41–50. https://doi.org/10.1145/3230977.3230981.
24. DeNero, J.; Sridhara, S.; Pérez-Quiñones, M.; Nayak, A.; Leong, B. Beyond Autograding: Advances in Student Feedback Platforms. In Proceedings of the Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2017; SIGCSE '17, p. 651–652. https://doi.org/10.1145/3017680.3017686.
25. Sridhara, S.; Hou, B.; Lu, J.; DeNero, J. Fuzz Testing Projects in Massive Courses. In Proceedings of the Proceedings of the Third (2016) ACM Conference on Learning @ Scale; Association for Computing Machinery: New York, NY, USA, 2016; L@S '16, p. 361–367. https://doi.org/10.1145/2876034.2876050.
26. Monperrus, M. A Critical Review of "Automatic Patch Generation Learned from Human-Written Patches": Essay on the Problem Statement and the Evaluation of Automatic Software Repair. In Proceedings of the Proceedings of the 36th International Conference on Software Engineering; Association for Computing Machinery: New York, NY, USA, 2014; ICSE 2014, p. 234–242. https://doi.org/10.1145/2568225.2568324.
27. Motwani, M.; Sankaranarayanan, S.; Just, R.; Brun, Y. Do Automated Program Repair Techniques Repair Hard and Important Bugs? *Empirical Softw. Engg.* **2018**, *23*, 2901–2947. https://doi.org/10.1007/s10664-017-9550-0.
28. von Wangenheim, C.G.; Hauck, J.C.R.; Demetrio, M.F.; Pelle, R.; da Cruz Alves, N.; Barbosa, H.; Azevedo, L.F. CodeMaster - Automatic Assessment and Grading of App Inventor and Snap! Programs. *Informatics in Education* **2018**, *17*, 117–150. https://doi.org/10.15388/infedu.2018.08.
29. da Cruz Alves, N.; Wangenheim, C.G.V.; Hauck, J.C.R. Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study. *Informatics in Education* **2019**, *18*, 17–39. https://doi.org/10.15388/infedu.2019.02.
30. Falkner, N.; Vivian, R.; Piper, D.; Falkner, K. Increasing the Effectiveness of Automated Assessment by Increasing Marking Granularity and Feedback Units. In Proceedings of the Proceedings of the 45th ACM Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2014; SIGCSE '14, p. 9–14. https://doi.org/10.1145/2538862.2538896.
31. Insa, D.; Silva, J. Semi-Automatic Assessment of Unrestrained Java Code: A Library, a DSL, and a Workbench to Assess Exams and Exercises. In Proceedings of the Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2015; ITiCSE '15, p. 39–44. https://doi.org/10.1145/2729094.2742615.
32. Cobo, M.; López-Herrera, A.; Herrera-Viedma, E.; Herrera, F. An approach for detecting, quantifying, and visualizing the evolution of a research field: A practical application to the Fuzzy Sets Theory field. *Journal of Informetrics* **2011**, *5*, 146–166. https://doi.org/10.1016/j.joi.2010.10.002.
33. Deeva, G.; Bogdanova, D.; Serral, E.; Snoeck, M.; De Weerdt, J. A review of automated feedback systems for learners: Classification framework, challenges and opportunities. *Computers & Education* **2021**, *162*, 104094. https://doi.org/https://doi.org/10.1016/j.compedu.2020.104094.
34. Benotti, L.; Aloi, F.; Bulgarelli, F.; Gomez, M.J. The Effect of a Web-Based Coding Tool with Automatic Feedback on Students' Performance and Perceptions. In Proceedings of the Proceedings of the 49th ACM Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2018; SIGCSE '18, p. 2–7. https://doi.org/10.1145/3159450.3159579.
35. Ho, V.W.; Harris, P.G.; Kumar, R.K.; Velan, G.M. Knowledge maps: a tool for online assessment with automated feedback. *Medical Education Online* **2018**, *23*, 1457394, [https://doi.org/10.1080/10872981.2018.1457394]. PMID: 29608133, https://doi.org/10.1080/10872981.2018.1457394.
36. Lee, H.S.; Pallant, A.; Pryputniewicz, S.; Lord, T.; Mulholland, M.; Liu, O.L. Automated text scoring and real-time adjustable feedback: Supporting revision of scientific arguments involving uncertainty. *Science Education* **2019**, *103*, 590–622, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/sce.21504]. https://doi.org/https://doi.org/10.1002/sce.21504.
37. Ye, H.; Martinez, M.; Monperrus, M. Automated patch assessment for program repair at scale. *Empirical Software Engineering* **2021**, *26*, 20. https://doi.org/10.1007/s10664-020-09920-w.

38. Parihar, S.; Dadachanji, Z.; Singh, P.K.; Das, R.; Karkare, A.; Bhattacharya, A. Automatic Grading and Feedback Using Program Repair for Introductory Programming Courses. In Proceedings of the Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2017; ITiCSE '17, p. 92–97. https://doi.org/10.1145/3059009.3059026.
39. Leite, A.; Blanco, S.A. Effects of Human vs. Automatic Feedback on Students' Understanding of AI Concepts and Programming Style. In Proceedings of the Proceedings of the 51st ACM Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2020; SIGCSE '20, p. 44–50. https://doi.org/10.1145/3328778.3366921.
40. Pape, S.; Flake, J.; Beckmann, A.; Jürjens, J. STAGE: A Software Tool for Automatic Grading of Testing Exercises: Case Study Paper. In Proceedings of the Proceedings of the 38th International Conference on Software Engineering Companion; Association for Computing Machinery: New York, NY, USA, 2016; ICSE '16, p. 491–500. https://doi.org/10.1145/2889160.2889203.
41. Wrenn, J.; Krishnamurthi, S. Executable Examples for Programming Problem Comprehension. In Proceedings of the Proceedings of the 2019 ACM Conference on International Computing Education Research; Association for Computing Machinery: New York, NY, USA, 2019; ICER '19, p. 131–139. https://doi.org/10.1145/3291279.3339416.
42. Souza, D.M.D.; Isotani, S.; Barbosa, E.F. Teaching novice programmers using ProgTest. *International Journal of Knowledge and Learning* **2015**, *10*, 60–77. https://doi.org/10.1504/IJKL.2015.071054.
43. Smith, R.; Tang, T.; Warren, J.; Rixner, S. An Automated System for Interactively Learning Software Testing. In Proceedings of the Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2017; ITiCSE '17, p. 98–103. https://doi.org/10.1145/3059009.3059022.
44. Buffardi, K.; Valdivia, P.; Rogers, D. Measuring Unit Test Accuracy. In Proceedings of the Proceedings of the 50th ACM Technical Symposium on Computer Science Education; Association for Computing Machinery: New York, NY, USA, 2019; SIGCSE '19, p. 578–584. https://doi.org/10.1145/3287324.3287351.
45. Blikstein, P.; Worsley, M.; Piech, C.; Sahami, M.; Cooper, S.; Koller, D. Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences* **2014**, *23*, 561–599. https://doi.org/10.1080/10508406.2014.954750.
46. Tahaei, N.; Noelle, D.C. Automated Plagiarism Detection for Computer Programming Exercises Based on Patterns of Resubmission. In Proceedings of the Proceedings of the 2018 ACM Conference on International Computing Education Research; Association for Computing Machinery: New York, NY, USA, 2018; ICER '18, p. 178–186. https://doi.org/10.1145/3230977.3231006.
47. Majd, A.; Vahidi-Asl, M.; Khalilian, A.; Baraani-Dastjerdi, A.; Zamani, B. Code4Bench: A multidimensional benchmark of Codeforces data for different program analysis techniques. *Journal of Computer Languages* **2019**, *53*, 38–52. https://doi.org/https://doi.org/10.1016/j.cola.2019.03.006.
48. Yi, J.; Tan, S.H.; Mechtaev, S.; Böhme, M.; Roychoudhury, A. A Correlation Study between Automated Program Repair and Test-Suite Metrics. *Empirical Softw. Engg.* **2018**, *23*, 2948–2979. https://doi.org/10.1007/s10664-017-9552-y.
49. Adler, F.; Fraser, G.; Grundinger, E.; Korber, N.; Labrenz, S.; Lerchenberger, J.; Lukasczyk, S.; Schweikl, S. Improving Readability of Scratch Programs with Search-based Refactoring. In Proceedings of the 2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM); IEEE Computer Society: Los Alamitos, CA, USA, 2021; pp. 120–130. https://doi.org/10.1109/SCAM52516.2021.00023.
50. Porfirio, A.; Pereira, R.; Maschio, E. Automatic Source Code Evaluation: a Systematic Mapping. Technical report, Federal University of Technology, Paraná, Brazil (UTFPR), 2021. https://doi.org/10.13140/RG.2.2.36112.33287.