CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Empowering Instructors with Collective Intelligence: Learning Problem-Solving Paths to Facilitate Feedback Generation

Thyago Tenório[1,2], Luiz Rodrigues[1], Seiji Isotani[1], Ig Ibert Bittencourt[2]

[1]Instituto de Ciências Matemáticas e de Computação (ICMC) – USP
Avenida Trabalhador São Carlense, 400 – 13566-590 – São Carlos – SP – Brasil

[2]Instituto de Computação (IC) – UFAL
Avenida Lourival Melo Mota, S/N – 57072-900 – Maceió – AL – Brasil

{tm.thyago,lalrodrigues,sisotani}@usp.br, ig.ibert@ic.ufal.br

Abstract. *The feedback on the Intelligent Tutoring Systems (ITS) providing the necessary support and guidance to students successfully complete a given task, improving their learning. However, building proper feedback demands time, whereas instructors are often overloaded, and several interactions with experts of the domain knowledge to know the several paths students, which is often infeasible and increases the costs and the complexity to develop ITSs at scale. To address this problem, we proposed a novel approach to build feedback using students' collective intelligence (CI), where our ITS might learn such paths incrementally building a knowledge graph with more detail than those predicted by instructors, optimizing the identification of problem-solving paths, and facilitating the iterative designing of meaningful and fine-grained feedback for the instructors by presenting domain model's updated, meaningful visualizations. To evaluate our approach we developed an ITS in a domain of numerical expressions that was used by 99 students. As a result, we observed that our approach helps to create a knowledge graph with a quality equivalent to that built by specialists in less time and considerably reducing the instructor's overload.*

1. Introduction

Feedback is prominent for learning. In an analysis of several meta analyses, [Wisniewski et al. 2020] found that feedback has an overall medium effect on student learning, especially on cognitive and motor skills outcomes, and that its type significantly moderates that effect. Specifically, [Wisniewski et al. 2020] indicate high-information feedback (i.e., corrective plus self-regulatory information) seems to be the most effective type. Students will benefit from feedback both when they successfully complete a learning task (e.g., confirming their correct approach) as well as when they face difficulties (e.g., explaining mistakes, guiding on how to avoid/correct them). Similarly, feedback is an essential part of advanced educational technologies, such as Intelligent Tutoring Systems (ITS), which will tutor students during problem-solving tasks [VanLehn 2006].

However, providing proper feedback is challenging for instructors. In ITS, the common practice is to design all feedback before deployment [VanLehn 2006]. First, this adds another task for the already overloaded instructors to spend time with. Second, to ensure feedback is available since the first use, this requires instructors to predict the

possible solutions/steps students might undergo while working on the learning activities. Nevertheless, feedback timing might also affect is effectiveness [Wisniewski et al. 2020], which could maximize the learning gap left by the lack of technological support. Thereby, either in the context of ITS or regular classrooms, the ideal scenario for providing feedback to students demand time and understanding of their solutions/steps.

Collective Intelligence (CI) is an approach that might help to address those challenges. CI refers to a group of individuals collectively doing things that are intelligent [Leimeister 2010]. Among its benefits, CI enables sharing ideas easily, quickly, and safely, which is often the case of discussion forums, blogs, and wikis [Baker and Green 2005]. Additionally, CI enables constructing knowledge bases, as the world's largest encyclopedia, Wikipedia [Malone et al. 2010], and Duolingo [Von Ahn 2013]. Despite CI holds great potential for the educational domain [Tenório et al. 2021, Meza et al. 2018], the researchs using CI to help instructors with feedback generation are limited [Tenório et al. 2020].

In a literature review, [Tenório et al. 2021] found the main purposes of CI on online educational technologies are improving collaboration on the educational process and creating educational content. Despite the attention to educational content creation, the authors found such studies mostly explore resource translation, video lectures and subtitles, and concept maps. In contrast, [Tenório et al. 2022] introduced a process to use students CI to author ITS's inner-loops. In an experimental study using an ITS equipped with CI, the authors found that their process generated a knowledge graph similar to the one instructors generated. However, the available research is limited to a computational process aimed for ITS, which cannot be easily used to augment instructor intelligence. Therefore, this paper sought to answer *how can we use Collective Intelligence to augment instructor intelligence and optimize the process of designing feedback?*

To answer that question, we introduce an algorithm that iteratively learns problem-solving patterns from students' solutions, as well as the (in)frequent paths students might undergo, and generates meaningful visualizations to augment instructor intelligence. Note that despite our approach depends on technology to process student data, the visualizations do not. Then, we demonstrate this approach in action with a case study for a numerical problem and discuss how it can be further used in practice to augment instructor intelligence. Based on that, this study helps instructors, regardless of their access to technology, with a tool that generates meaningful visualizations of problem-solving patterns based on students CI. Thus, our contribution is twofold, empowering instructors by removing the effort required to identify such patterns, hence, optimizing the generation of feedback for each problem-solving step.

2. Learning Problem-Solving Paths

To provide feedback at each step for the student, the instructor/ITS must have the integrated knowledge of the possible resolutions and, consequently, the possible states, steps, and actions in a kind of knowledge network. So, we defined a knowledge graph ($\mathbb{KG} = (V, A)$), with V being the set of all possible states ($\mathbb{E}$) and A being the set of all possible steps ($\mathbb{S}$). An example of $\mathbb{KG}$ represented visually can be seen in Figure 1.

In the $\mathbb{KG}$, each node represents a possible state of the problem, highlighting the initial states with a dashed edge line and the final states with a double-edged line. A state

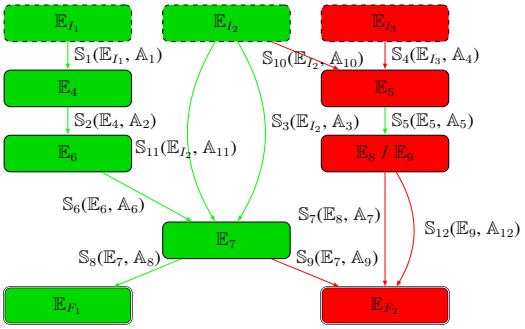

Figure 1. Example of Visual Representation of a Knowledge Graph $\mathbb{KG}$

can contain more than one input and output step. In turn, each edge (represented by a directed arrow) represents a possible step to be performed by students using an action $\mathbb{A}$. A resolution $\mathbb{R}$ is a path in the graph. Other information using colors are also included, as the correctness of the states (i.e., green nodes are correct states and red nodes are incorrect states) and the validity of the steps (i.e., green edges are valid steps and red edges are invalid steps). In the graph, a correct resolution only contains green vertices and edges, while an incorrect resolution contains one or more red vertices or edges.

To build each KG for the ITS problems, we had developed an algorithm that iteratively learns problem-solving patterns from students' solutions. Briefly, the algorithm receive as input a resolution $\mathbb{R}$ and the evaluations of feedbacks received from the students and performs the following steps:

- Retrieve the current knowledge graph of problem P. If there is no registered graph, it is necessary to initialize a new empty knowledge graph or the instructor could be initialize a new graph with at last one correct resolution.
- Evaluate and update values referring to elements (states or steps) in the knowledge graph linked to feedback evaluations.
- For each step $\mathbb{S}$ contained in the resolution $\mathbb{R}$:
  - I. Retrieve the source state of step $\mathbb{S}$ in the graph. If this state is not represented in the graph, it must be created. If $\mathbb{S}$ is the *first step* of R, this state must be marked as initial state in the graph.
  - II. Retrieve the target state of step $\mathbb{S}$ in the graph. If this state is not represented in the graph, it must be created. If $\mathbb{S}$ is the *last step* of R, this state must be marked as final state in the graph.
  - III. Retrieve the step in the graph whose source state, target state and action are equal to step S. If it does not exist in the graph, it must be created.
- For each retrieved/created state/step in the before instruction:
  - I. Update the values referring to correctness and validity in the knowledge graph of the elements (states and steps) involved in the resolution $\mathbb{R}$.

The last instruction aims to update the correctness values of the states and steps of the $\mathbb{KG}$. To estimate these values, we define mechanisms **based on the idea that correct/incorrect steps and states will be present in correct/incorrect resolutions, respectively**, and after we validate with the students. So, considering C_R as the set of the resolutions used to build the $\mathbb{KG}$, C_{RC} the subset of the correct resolutions and C_{RI} the

subset of the incorrect resolutions, the PE function (Equation 1), which indicates whether an element (state or step) is present or not in a resolution and the PEC function (Equation 2) that counts how many times a given element is present in a set of resolutions, we defined the correctness function C($\mathbb{E}$ or $\mathbb{S}$) as indicated in Equation 3. In summary, the equation counts how many times an element appeared in the correct resolutions and subtracts how many times it appeared in incorrect resolutions and finally divides by the total number of appearances in the set of resolutions. The range of the function is $[-1, 1]$.

$$\text{PE}(E, R) = \begin{cases} 1 & \text{se} \quad E \in R \\ 0 & \text{se} \quad E \notin R \end{cases} \qquad (1) \qquad PEC(E, C_R) = \sum_{i=1}^{|C_R|} PE(E, R_i) \qquad (2)$$

$$C(\mathbb{E}) \; or \; C(\mathbb{S}) = \frac{PEC(E, C_{RC}) - PEC(E, C_{RI})}{PEC(E, C_R)} \qquad (3)$$

Finally, we calculate the correctness of a resolution as the sum of the average correctness of the states E with the average of the validity of the steps S contained in R, divided by 2 to normalize the results, keeping the interval in $C(R) = [-1, 1]$, as presented in Equation 4 and considering C_E the set of states and C_S the set of steps of $\mathbb{R}$.

$$C(\mathbb{R}) = \frac{1}{2 * |C_E|} * \sum_{i=1}^{|C_E|} C(\mathbb{E}_i) + \frac{1}{2 * |C_S|} * \sum_{j=1}^{|C_S|} C(\mathbb{S}_j) \qquad (4)$$

A state/step/resolution is previously considered correct if C($\mathbb{E}/\mathbb{S}/\mathbb{R}$) $\geq$ 0.8 or incorrect if C($\mathbb{E}/\mathbb{S}/\mathbb{R}$) $\leq$ -0.8. After, we ask to students to confirm the results. The system analyzes each state/step of the student's solution and compares it with ones present in the graph, selecting that ones are presented in both or present on alternative paths that are in some way linked to the student's solution.

3. Learning A Numerical Problem's Paths

This section uses a numerical problem to exemplify our algorithm in action. We first provide a brief description of the data collection process, then we demonstrate the visualizations our algorithm generates based on each student solution.

3.1. Data Collection

To accomplish this process, we used a low-fidelity prototype of an ITS that featured the following numeric problem: $2 + [7 - 2 * 3 + (2^8 - 12)] - 4$, designed by a math instructor. The goal was to use a problem with reduced difficulty so that more users would be able to work on it. Concerning the prototype, it presented the problem description, five possible answers for the user to chosen from, and a area where the user could solve the problem step-by-step. Specifically, the latter started with a single blank line that the user could write the next step towards completely solving the process. Nevertheless, the prototype also allowed the user to add new lines so that the user could take as many steps as they wanted to answer the problem. Once the user confirmed their answer, the prototype saved the steps sequence, which would then be used by the algorithm.

To collect data, we used Amazon Mechanical Turk[1], a crowdsourcing tool widely used by social science researchers for data collection [Paolacci et al. 2010]. Accordingly, we chose to use it following recommendations on its benefits to increase sample size and external reliability [Landers and Behrend 2015]. We made our study available for five days, and enforced no restriction regarding participants' profiles to prevent selection bias. Once the user started their participation, the system would provide brief instructions concerning numerical expressions as well as the task to be done and ask the participant to self-report their math knowledge in the topic using a one to 10 scale (the higher the number, the higher the self-reported knowledge). Subsequently, the user would start solving the problem, which had a 10-minute limit.

As a result, we gathered 99 answers. On average, users took 237 seconds to complete the problem, used seven steps, and self-reported a 7.9 knowledge. Of all answers, 18 (18%) were mathematically invalid due to, for instance, resolutions without being step by step or texts outside the scope of the problem. Of the 81 valid answers, 56 (69%) and 25 (31%) were correct and incorrect, respectively. Interestingly, we received 12 different answers among the 25 incorrect ones. Next, we demonstrate our learning algorithm in action based on these data.

3.2. Iterative Learning

Figure 2 demonstrates the visualizations generated after the algorithm processed the first and second solutions. In Figure 2(left), the solution steps - represented as rectangles - demonstrate the steps the user went through to achieve the correct solution, which is indicated by their green colors. Nevertheless, all steps feature a orange shade to indicate they have been recently created. Figure 2(right), for instance, adds a second solving path and highlights it with the orange shade, whereas the already existent steps become all green. These colors similarly apply to arrows, which indicate transitions between steps. Moreover, each arrow feature a numeric value ranging from -1 to +1, which indicates the algorithm's confidence in that transition's correctness[2]. Following that pattern, Figure 3 demonstrates the visualization generated after processing wrong and invalid solutions with 20 solutions (25%) and the Figure 4 shows the graph with 40 solutions (50%).

The states/steps colors change according to correctness values calculated for each student's solution added (see the yellow states/steps in Figures 3 and 4). In addition, there is another aspect that involves the size of the node and the thickness of the edge, which allows us to quickly visualize which are most frequent states and steps of the students. We highlighted in blue the most frequent correct path used by students, as well as added orange arrows pointing to the most frequent errors, which may involve errors of mathematical operations (1 and 3), or even misconceptions (2 and 4).

3.3. Graph Analysis

The final generated graph contains 117 states (18 initial, 86 support and 13 final), with 37 correct, 74 incorrect and 6 unknown and 160 steps. First, we asked to a instructor evaluate the graph and compared the instructor and algorithm results generating the confusion matrix presented in Table 1. The algorithm success rate was 87.18% (102 of 117) for the states and 97.50% (156 of 160) for the steps.

[1]https://www.mturk.com/

[2]Note that all transitions of Figure 2 have +1 confidence because both solutions were correct.

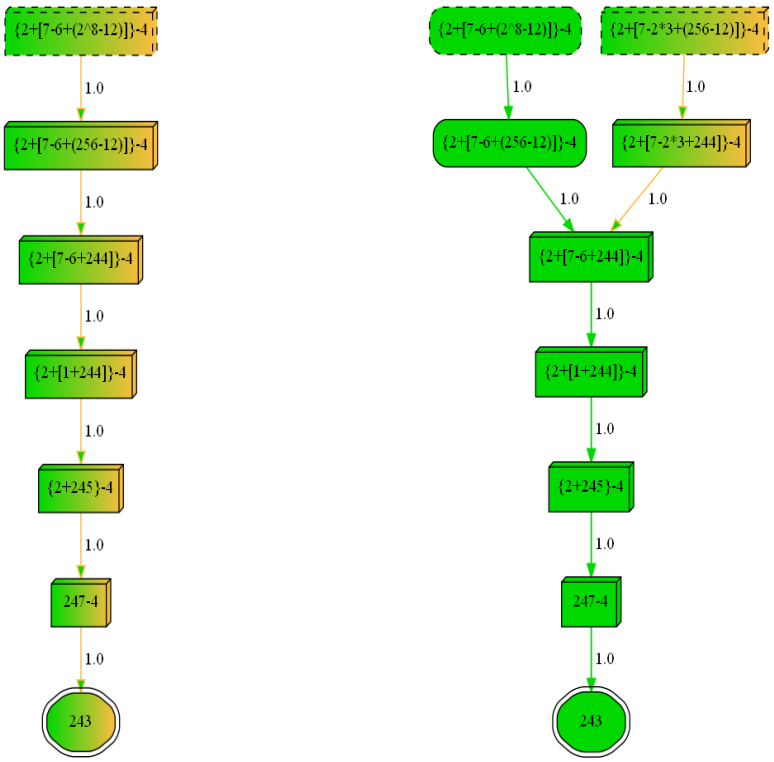

Figure 2. Visualizations generated after the algorithm processed one (left) and two (right) solutions

		Expert - States			Expert - Steps	
	State	**Correct**	**Incorrect**	**Step**	**Valid**	**Invalid**
P_1	**Correct**	29	8	**Valid**	74	3
	Incorrect	1	73	**Invalid**	0	82
	Inconclusive	6	0	**Inconclusive**	1	0

Table 1. Confusion Matrix with Model and Expert Data

However, we noted that the knowledge graph for a given problem tends to grow, becoming increasingly complex. So, we were concerned with analyzing growth/stability metrics of the graph to verify at what moment the changes will become minimal, since the number of added information on the graph tends to decrease for each new added solution. This problem is equivalent to the cold start problem in recommender systems, since it needed to verify what the number of solutions are necessary for the construction of the minimum recommended graph. So, we analyzed the growth of the number of created (total/correct/incorrect) nodes and edges, presented in Figures 5A and 5B, respectively.

We observe in Figure 5A that the number of created nodes grows rapidly in the first 45 resolutions, decreasing the growth rate, becoming stable with 65 resolutions. On the other hand, Figure 5B shows that the number of created edges grows rapidly in the first 52 resolutions, but maintains the same behavior of the nodes. These findings are confirmed when we analyze the individual number of nodes and edges created individually for each solution, as shown in Figures 5C and 5D, respectively. We also analyzed the averages of nodes and edges created for each solution, as shown in Figures 5E and 5F. In both figures,

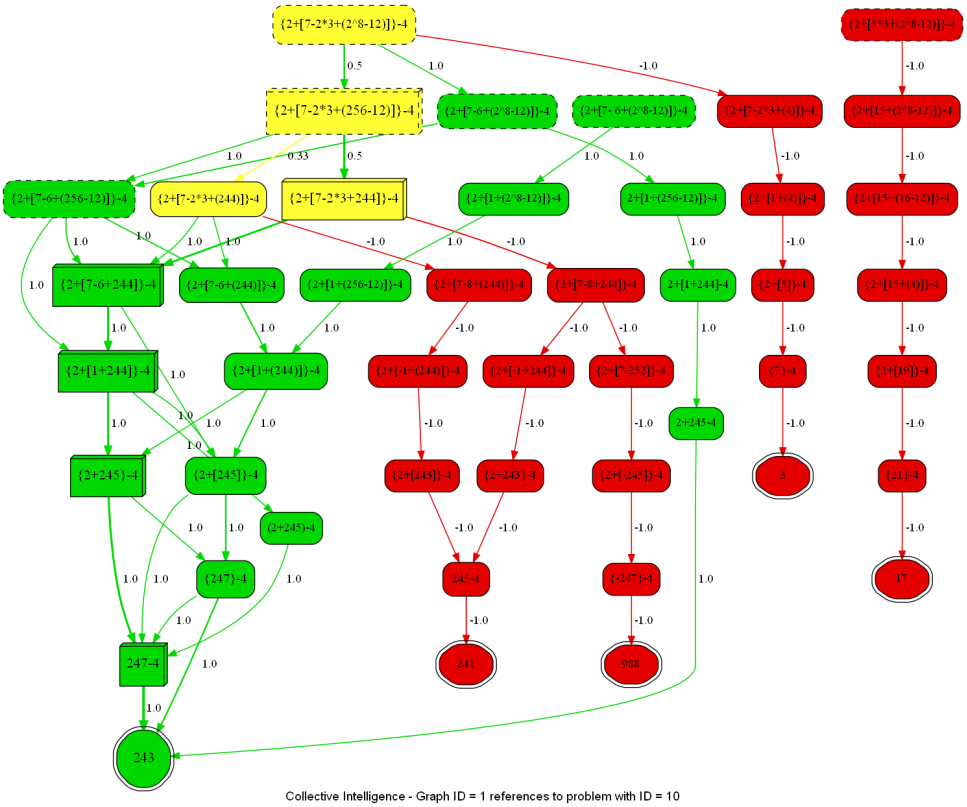

Figure 3. Visualization generated after the algorithm processed twenty solutions, including incorrect and invalid ones.

the creation averages start with a high value (7 nodes and 6 edges per resolution) and has a strong decay trend, ending with an average lower than one node and two edges created per resolution. In this sense, we conclude that at least 40 resolutions are necessary, with a recommended number of 65 resolutions to guarantee the stability of the graph.

4. Discussion - Augmenting Instructor Intelligence

This section discusses some situations our solution can be used to augment instructor intelligence in real learning settings. First, consider a math instructor at some underprivileged school with no internet or technology available. Neither this instructor nor their students can benefit from any of the several ITS available for this subject. Nevertheless, if interaction data from students' solutions are available, one can use our algorithm to create a graphical visualization of, among other information, students' common mistakes. Then, such visualization can be printed and used by the instructor to analyze such mistakes, come up with meaningful feedback for each one, and optimize this instructor's practice during lessons. Similarly, one follow this process to design confirmatory feedback for correct steps or warn students when they enter a incorrect step. Ultimately, such usage would enhance students' learning experiences by facilitating the instructor's process of finding problem-solving issues and preparing to promptly deal with them.

Second, consider a math instructor with access to a ITS with our algorithm integrated. The ITS will generate all visualizations for the graph, updating them whenever a new solution is added and making them available to the instructor (including the entire

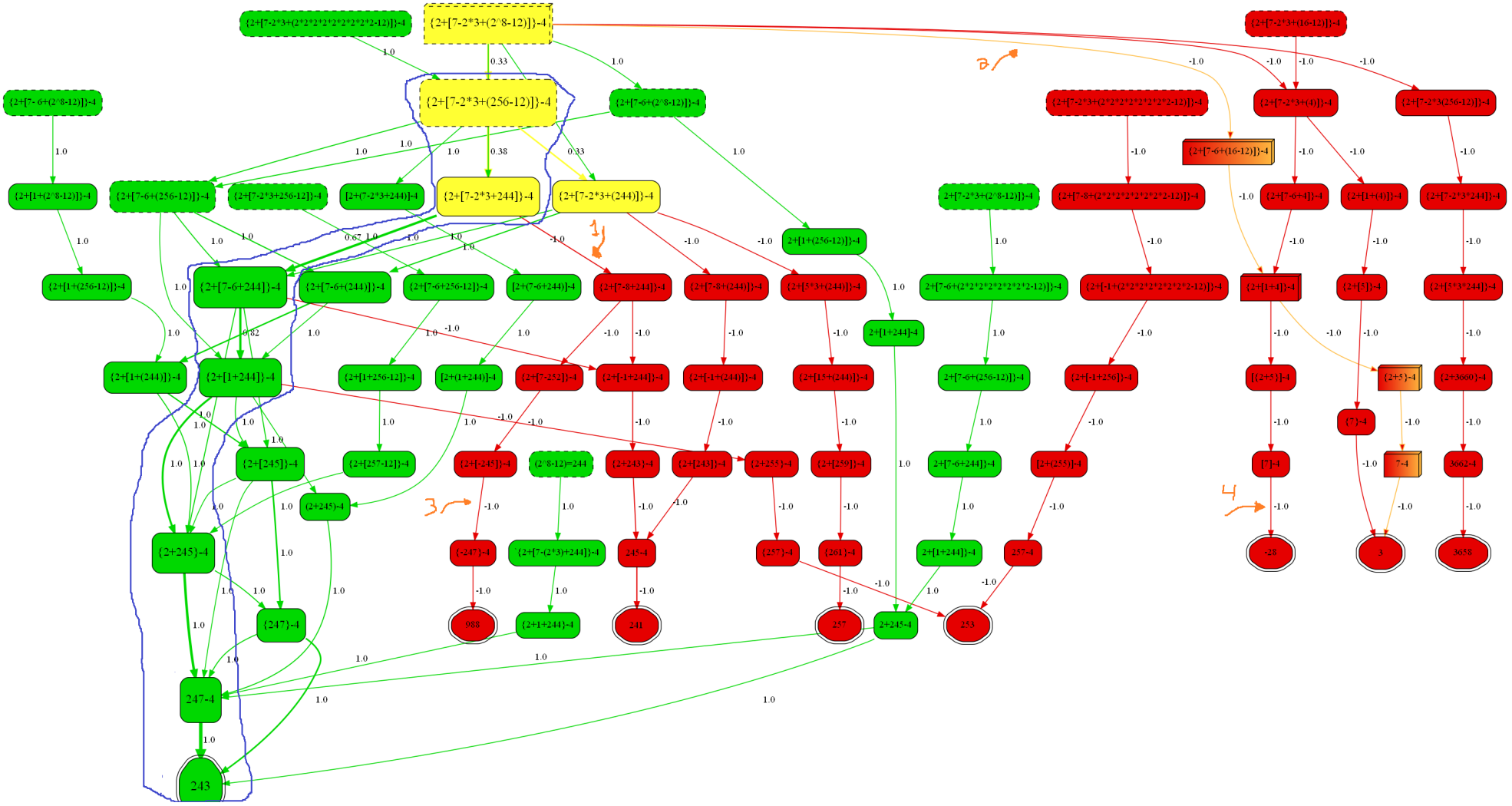

Figure 4. Building Knowledge Graph Using the Proposed Process With forty (50%) of the Solutions

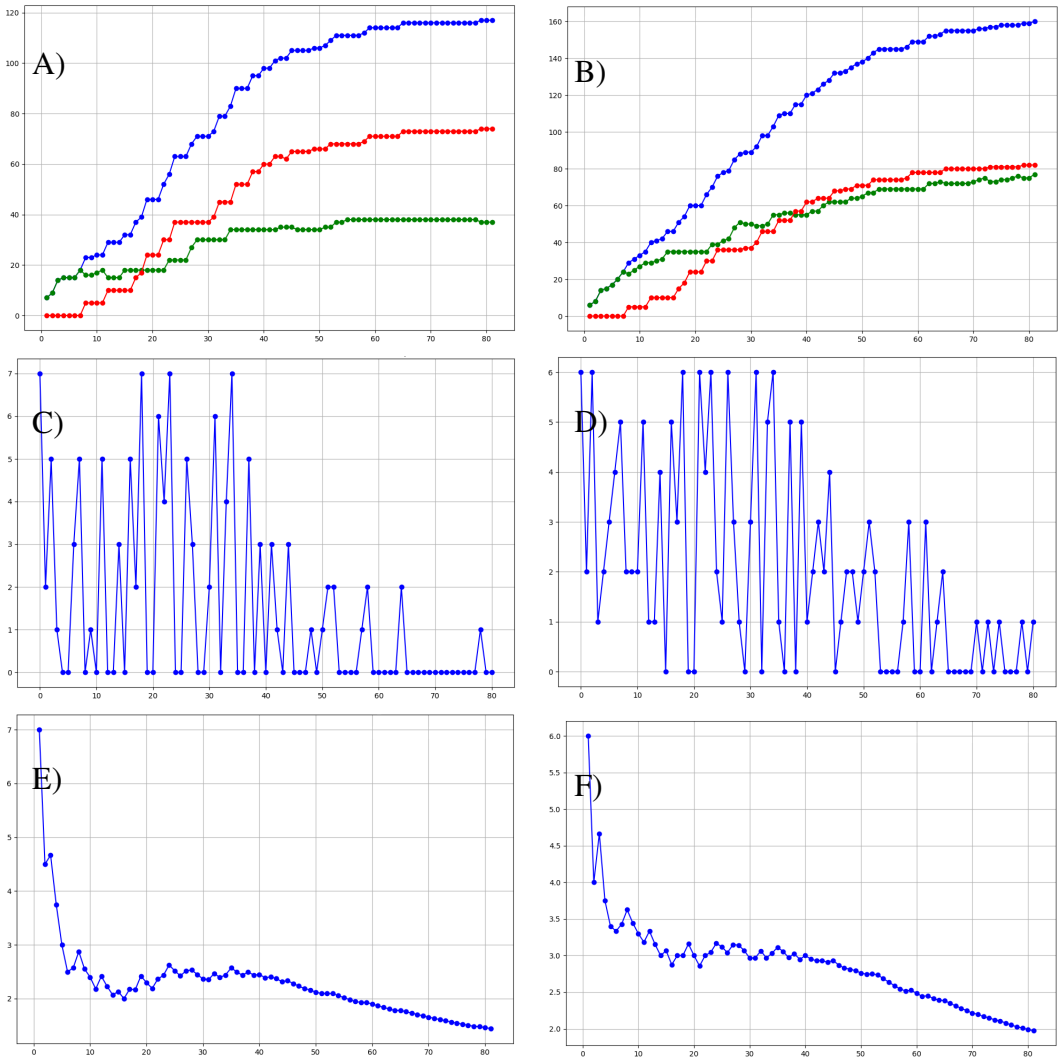

Figure 5. Metrics for Graph Analysis for Each Solution: A) Total Number of Created Nodes, B) Total Number of Created Edges, C)Indiv. Number of Created Nodes, D)Indiv. Number of Created Edges, E)Average of Created Nodes, F)Average of Created Edges

history) after a moderating automatically performed by the system to identify and eliminate inappropriate information. Thus, the instructor can focus on analyzing the patterns contained in the graph and perform a manual moderation with the aim of refining the quality of the information contained in the system to maximize the benefits of the process, which considerably decreases the instructor's workload. Furthermore, instead of creating abstract feedbacks for the problem itself, the instructor will be able to identify points in the graph where feedbacks are most needed, which will allow the construction of feedbacks that are more specific to the real needs of the students. In addition, the instructor can identify the main mistakes of the students (including miss conceptions) and to adjust the contents provided in the classroom in order to minimize the learning problems.

According the instructor's workload, working with step-by-step resolutions and more specific feedback requires additional work from the instructor, which is a weak point that can harm the results obtained with the application of step-by-step resolutions.

Some studies have estimated that 200–300 hours of development will be spent per hour of instruction, which can be reduced by half using tools like CTAT[Aleven et al. 2006]. In this sense, we evaluated the time spent to build the knowledge graph using the students solutions: 23,425 seconds, or approximately 6.5 hours. Even though there is no way to directly compare these numbers, we can observe that the time spent building the graph using CI is less than the time spent when experts are used. Furthermore, we show that the generated graph using CI had a high success rate (87.18% for nodes and 97.50% for the steps) and presents stability behavior from a determined number of solutions.

The literature results demonstrate that CI has substantial potential for use with online educational technologies to enhance collaboration, interaction, and the learning processes, thereby affecting not only students but also teachers [Tenório et al. 2021]. The findings in this work corroborate the potential of using CI to build educational content, allowing to build important information from the ITS domain model with a quality equivalent to that built by specialists in less time and considerably reducing the instructor's overload. In addition, we can use the students' CI not only by analyzing their answers and confirming with them about the correctness, but we can also use techniques to automatically identify which are the points in the graph that need more attention and feedback and, subsequently, ask to the students to register such feedbacks, which will be used by the ITS to help other students. These feedbacks can be evaluated by the students themselves, allowing the system to be able to identify and recommend better feedbacks, through an incremental process. The instructor would no longer be a content/feedback creator, but he becomes a moderator of the content, having more time to attend to students' needs.

5. Final Remarks

Despite feedback is prominent for learning, designing it is a time-consuming task that demands understanding the specific cases wherein students will be assistance. Aiming to address these challenges, this paper explores students Collective Intelligence as an alternative to reveal learning paths and facilitate feedback generation. Hence, we first detail an algorithm that learns such paths from students step-by-step solutions and generates meaningful visualizations that inform instructors on (in)correct paths, as well as the most frequent ones. We present a case study based on a numerical math problem to demonstrate this algorithm in action, discuss how it augments instructors intelligence, and investigate how long it takes to converge into a representative representation of that math problem.

Based on that context, this paper has a number of implications. For practitioners, we provide an algorithm that optimizes their practices by revealing the possible solutions, difficulties, misconceptions, and common and uncommon paths students might use to solve a problem. Hence, we reduce the overload needed to reflect on those aspects, making room to invest time in creating meaningful feedback and working on other tasks. For researchers, we introduce an algorithm that generates meaningful, static feedback and investigate its convergence for a specific math problem. Thereby, we open room for future studies on the effectiveness of the generated visualizations as well as convergence tests for other problems. Furthermore, by focusing on static visualizations, our approach enables reaching even those of underprivileged contexts wherein technological resources are limited. Thus, we contribute towards making learning inclusive and equitable for all by allowing underprivileged instructors and students to benefit from knowledge generated by those with.

References

Aleven, V., McLaren, B. M., Sewall, J., and Koedinger, K. R. (2006). The cognitive tutor authoring tools (ctat): preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, pages 61–70. Springer.

Baker, S. and Green, H. (2005). Blogs will change your business. *Business week*, 3931(5):56–65.

Landers, R. N. and Behrend, T. S. (2015). An inconvenient truth: Arbitrary distinctions between organizational, mechanical turk, and other convenience samples. *Industrial and Organizational Psychology*, 8(2):142–164.

Leimeister, J. M. (2010). Collective intelligence. *Business & Information Systems Engineering*, 2(4):245–248.

Malone, T. W., Laubacher, R., and Dellarocas, C. (2010). The collective intelligence genome. *MIT Sloan management review*, 51(3):21.

Meza, J., Jimenez, A., Mendoza, K., and Vaca-Cárdenas, L. (2018). Collective intelligence education, enhancing the collaborative learning. In *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*, pages 24–30. IEEE.

Paolacci, G., Chandler, J., and Ipeirotis, P. G. (2010). Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419.

Tenório, T., Isotani, S., and Bittencourt, I. I. (2022). Authoring inner loops of intelligent tutoring systems using collective intelligence. In *International Conference on Artificial Intelligence in Education*, pages 400–404. Springer.

Tenório, T., Isotani, S., Bittencourt, I. I., and Lu, Y. (2021). The state-of-the-art on collective intelligence in online educational technologies. *IEEE Transactions on Learning Technologies*, 14(2):257–271.

Tenório, T., Isotani, S., and Bittencourt, I. I. (2020). Inteligência coletiva como ferramenta de apoio na construção de loops internos em sistemas tutores inteligentes. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pages 1233–1242, Porto Alegre, RS, Brasil. SBC.

VanLehn, K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265.

Von Ahn, L. (2013). Duolingo: learn a language for free while helping to translate the web. In *Proceedings of the 2013 international conference on Intelligent user interfaces*, pages 1–2.

Wisniewski, B., Zierer, K., and Hattie, J. (2020). The power of feedback revisited: A meta-analysis of educational feedback research. *Frontiers in Psychology*, 10:3087.