Journal of Mathematical Behavior 55 (2019) 100704

Contents lists available at ScienceDirect

Journal of Mathematical Behavior

journal homepage: www.elsevier.com/locate/jmathb

Localized considerations and patching: Accounting for persistent attributes of an algorithm on a contextualized graph theory task

John Griffith Moala[a,*], Caroline Yoon[b], Igor' Kontorovich[c]

[a] Massey University, New Zealand
[b] The University of Auckland, New Zealand
[c] The University of Auckland, New Zealand

ARTICLE INFO

Keywords:
Algorithm
Graph theory
Aptness
Patching
Localized considerations

ABSTRACT

Students often hold on to algorithms that are inappropriate for the problem at hand, despite being presented with evidence of their inappropriateness during testing. Past research has tended to focus on the rejection and replacement of an algorithm in its entirety, rather than small refinements to an otherwise intact algorithm. In this study, we take a fine-grained view, focusing on the iterative refinement and augmentation of an algorithm, rather than its wholesale replacement. We ask: how do students decide what to change and what to keep when revising their algorithm upon testing? We analyze the collaborative work of three students on a graph theory task which invited the students to develop an algorithm for a contextualized problem. Two terms, localized considerations and patching, are introduced to describe how the group revised and validated their algorithms while retaining one specified attribute of their initial algorithm throughout.

1. Introduction

Discrete mathematics is widely regarded as the mathematics of our time (Dossey, 1991). It forms the foundation for modern day topics such as computer science, cryptography, optimization, and graph theory, with widespread applications in big data, discrete dynamical systems in ecology, networks in industry and the social sciences (Committee on the Mathematical Sciences in 2025, Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, Council, 2025Committee on the Mathematical Sciences in et al., 2013Committee on the Mathematical Sciences in 2025, Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, Council, 2025).

At the heart of discrete mathematics lies the *algorithmatizing approach* (Maurer & Ralston, 1991) which entails creating an algorithm that can be used to find a solution to a given problem, as opposed to merely finding a solution. The algorithmatizing approach is typically iterative, beginning with the creation of a provisional algorithm, which is then tested on a problem, and the results are examined to inform revisions to the algorithm and further testing (Hart & Martin, 2018). The results of testing are critical indicators of the algorithm's aptness, and can help diagnose what needs to be fixed. Yet mathematics education research demonstrates that students often hold on to algorithms that are inappropriate for the problem at hand, despite being presented with evidence of their inappropriateness during testing, and opportunities to revise them (e.g., Booth, Barbieri, Eyer, & Paré-Blagoev, 2014; De Bock, Van Dooren, Janssens, & Verschaffel, 2002).

Our study investigates the persistence of students' inappropriate algorithms during multiple test-and-revise iterations in a single

* Corresponding author.
E-mail addresses: j.g.moala@massey.ac.nz (J.G. Moala), c.yoon@auckland.ac.nz (C. Yoon), i.kontorovich@auckland.ac.nz (I. Kontorovich).

https://doi.org/10.1016/j.jmathb.2019.04.003
Received 2 January 2018; Received in revised form 12 April 2019; Accepted 15 April 2019
Available online 28 April 2019

task. We ask: how do students decide what to change and what to keep when revising their algorithm upon testing? We conduct a fine-grained analysis of the collaborative work of three students on The Jandals Problem — a graph theory task which invited the students to develop an algorithm for a contextualized problem (Yoon, Moala, & Chin, 2016). The group's initial algorithm specified a priori attributes that the solution for the problem should have, and then searched through potential solutions for the one which possessed the specified attributes. Then, noticing that their initial algorithm was inappropriate, the group engaged in three iterations of testing and revising, producing three revised algorithms that were strikingly similar to the initial algorithm, but still inappropriate for the problem at hand.

We operationalize the persistence of the group's inappropriate algorithm through three test-and-revise iterations, as the persistence of a particular specified attribute of the students' initial algorithm. We introduce two terms, *localized considerations* and *patching*, to describe how the group revised and validated their algorithms while retaining one specified attribute of their initial algorithm throughout.

2. Background literature

Mathematics education literature contains examples of, and explanations for, students persisting with algorithms that are inappropriate for the problem. For instance, some students persist in applying algorithms based on linear reasoning in non-linear situations (De Bock et al., 2002). One of the word problems discussed in De Bock et al. (2002) invited students to create an algorithm for determining the increase in the volume of an object, if each of its dimensions is doubled. The authors inferred an improper use of linear reasoning from students' use of algorithms such as: *doubling the volume of the object*. The authors suggested that these students needed to reject their linear algorithms in favor of non-linear algorithms.

Booth et al. (2014) discussed the persistent errors in students' algebraic problem solving. For example, the *negative sign error* refers to instances in which, for a linear equation such as: $-12x = -6x - 4x - 4$, students begin by subtracting $-4x$ from both sides of the equation, and then subtracting $-6x$ from both sides, reducing the equation to $-2x = -4$. Accordingly, the authors suggested particular interventions that would prevent students from applying such algorithms.

One explanation for the persistence of students' inappropriate algorithms can be inferred from studies pertaining to the use of counterexamples as a means for facilitating revisions of students' incorrect conceptions. Zazkis and Chernoff's (2008) research on counterexamples shows that while all counterexamples are mathematically equivalent in their potential to reveal erroneous conceptions, not all counterexamples invoke cognitive conflict, and even if they do, cognitive resolution does not necessarily follow. Incidentally, studies have found that when students encounter counterexamples to their inappropriate algorithms, they do not always reject their incorrect algorithms (e.g., see Clarke & Veith, 2003; Ferrarello & Mammana, 2018; Lappan & Bouck, 1998). Extending Zazkis and Chernoff's (2008) argument to the context of algorithms, it can be hypothesized that some counterexamples are better than others in facilitating the rejection of students' inappropriate algorithms (cf. Klymchuk, 2012; Zaslavsky & Ron, 1998).

Another explanation for the persistence of faulty or inappropriate algorithms is that they are so entrenched in students' intuitive knowledge (Fischbein, 1987) that they are inaccessible for reflection, correct without a need for any further justification, and persistent in the face of conflicting evidence (see De Bock et al., 2002). A countering viewpoint rests on students' tendencies to consciously apply certain algorithms deliberately to situations wherein they are not applicable (e.g., applying algorithms for adding and subtracting natural numbers to fractions; or using linear algorithms for non-linear problems). This viewpoint differs from the intuitive-knowledge explanation in the sense that these students no longer unconsciously apply certain algorithms, but rather do so in an explicit and deliberate way, with the conviction that these algorithms work everywhere.

Underlying beliefs, habits, and the norms that support them may also be contributing factors to the persistence of students' inappropriate algorithms. Students may dismiss efficient and effective algorithms for ones given to them by the teacher, believing that the teacher always knows better. Furthermore, even when some students realize that an algorithm given to them by their teacher is producing an error, they reject the error because they think that the algorithm has to be correct since it came from the teacher (see De Bock et al., 2002; Verschaffel, Greer, & De Corte, 2000; Wyndhamn & Säljö, 1997).

The persistence of students' incorrect algorithms may also be due to students not checking whether their algorithms are valid. For instance, a student's algorithm for subtracting one positive integer from another might be: subtract the smaller digit from the corresponding larger digit. The algorithm works for $29 - 17$, but yields the same result of 12 for $27 - 19$. From the lack-of-validation perspective, this student has not, say, checked whether, for example, $19 + 12 = 27$; or whether $27 - 19$ and $29 - 17$ are equivalent (e.g., Booth et al., 2014; Brown & Quinn, 2006).

Overall, we notice that past research has tended to focus on the persistence and the desired rejection and replacement of an inappropriate algorithm in its entirety. We also notice that past research seems to equate the rejection of an algorithm with recognizing its inappropriateness. Consequently, past explanations for the persistence of students' inappropriate algorithms are given in terms of things which the students did not do (e.g., not validating their algorithm; not being able to transition from linear to non-linear algorithms). Such explanations shed little to no light on what students actually do—the mechanisms by which they revise their inappropriate algorithm upon testing, irrespective of whether the resultant algorithm is appropriate or inappropriate. In our study, we take a fine-grained view, focusing on the iterative refinement and augmentation of an algorithm, rather than its wholesale replacement. We ask: How do students decide what to keep in and what to remove from their inappropriate algorithms?

3. The Jandals Problem: A graph theory algorithmatizing task

In this paper, we focus on problems that require finding a solution among a set of potential solutions. The Jandals Problem (Yoon

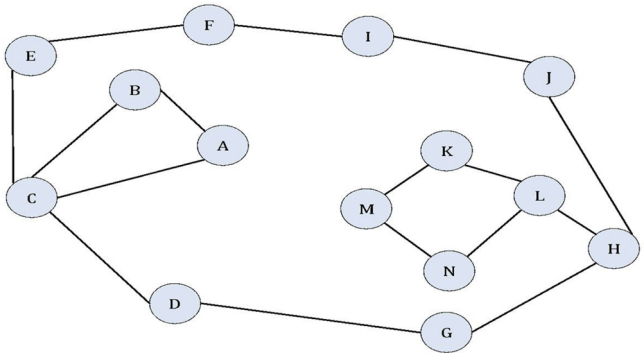


Fig. 1. Friendship Network 1.

et al., 2016), which we used in our study to motivate students to engage in creating and refining algorithms, is an example of such a problem. The Jandals Problem begins with some warm up questions that familiarize students with diagrammatic representations of networks (graphs) within the context of friendship associations, where a vertex represents a person, and an edge between two vertices represents a friendship between two people. After the warm-up, a scenario is posed: Xanthe, an exchange student in New Zealand learned that New Zealanders use the word jandals to refer to what she commonly calls flip-flops. Upon returning home, Xanthe wants to spread the word *jandals* throughout different networks of friends. The task instructions state:

> Create an algorithm (method) that Xanthe can use to figure out the first person whom she should share the word with first in each friendship network to ensure that the word gets passed on to everyone in the network as rapidly as possible. She assumes that a person will share the word with all of his/her friends on one day, and each of those friends will share it with their friends the next day. Ensure that your algorithm will work for any friendship network, not just the one given [Fig. 1] (Yoon et al., 2016, p. 8).

Students are initially presented with only the sample network shown in Fig. 1. Once students have created and agreed on an algorithm, they are presented with additional friendship networks one at a time (see Figs. 2 and 3) on which they are encouraged to test and revise their algorithms.

4. Conceptual framework

4.1. Algorithms

Algorithms can be thought of as a set of rules or instructions for solving a specific problem (Thomas, 2014). One type of algorithm for solving The Jandals Problem involves counting the number of days it takes for the word to spread throughout the network, starting from each person. The quickest starting person is then the person corresponding to the minimum number of days. Another type of algorithm could specify a priori the attributes which the quickest starting person should have, and then search the network for the person which possesses the specified attributes. The first type of algorithm involves comparisons among the candidates, whereas the latter checks each candidate independently of the others, searching for a candidate with specific attributes.

The algorithms that were refined by the group of students in our study are of the latter type. Revising such algorithms involve updating the list of specified attributes of the desired solution by deciding which attributes to keep, which to discard, and adding new attributes to the list. The removal of some attributes from the list of specified attributes suggests their inappropriateness from the students' perspective, while maintaining and addition of other attributes, indicates their inappropriateness. In this way, the revision of the list of specified attributes can be construed in terms of their contextual appropriateness.

4.2. Considerations of aptness

We anticipate that even within a single task, contextual appropriateness is multi-dimensional. That is, what might be appropriate for one aspect of the task, might be not (or less) appropriate for a different aspect of the same task. To address the multi-dimensionality of contextual appropriateness, we turn to the construct of *considerations of aptness to the task* (Kontorovich, Koichu,Leikin,& Berman, 2012).

In the context of tasks asking students to pose mathematical problems, Kontorovich (2016) discussed considerations of aptness to the task as a means of capturing "uncertainties and doubts of a poser" (p. 246) in the attempt to satisfy the explicit but often vague requests of the task. In the present study, we use considerations of aptness to the task broadly to capture students' attempts to address the task's requests for an algorithm and to evaluate the extent to which the requests are met.

The Jandals Problem above contains multiple requests such as: finding the "quickest person" (i.e., the one from whom the word reaches everyone else the fastest) in a network; creating an algorithm that identifies the "quickest person" in any network; and framing the algorithm as a letter. Thus, we anticipate different considerations of aptness to the task, depending on which request(s) the students are addressing. An algorithm might be apt for one aspect of the task (e.g., finding the best person in a given network), but

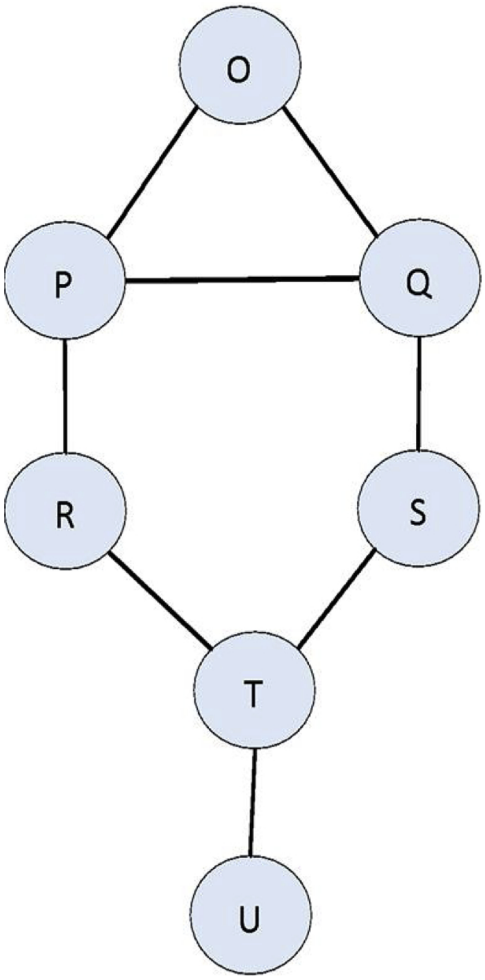


Fig. 2. Friendship Network 2.

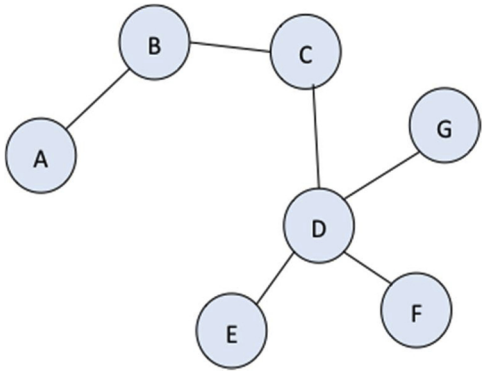


Fig. 3. Friendship Network 3.

not (or less) apt for another (e.g., explaining the algorithm to a client). Thus, the aptness of an algorithm for the task, even one which is “mathematically correct”, varies according to variations in purpose (i.e., for what and for whom?). Furthermore, although The Jandals Problem has multiple explicit requests, these requests are open to interpretation. For instance, questions that may be asked in response to the task instructions above are: What happens if there is more than one quickest person in a friendship network? What if you create an algorithm that finds a quickest person, but also finds non-quickest persons? What does "any friendship network" look like? The answers to these questions are not provided in the task instructions. As such, the answers to these questions will differ

Table 1
The group's four algorithms in order of appearance in their work.

Name	Description of the algorithm
Algorithm 1	Share the word with someone who tells three people.
Algorithm 2	Share the word with someone who tells three people, and one of those three people tells two other people.
Algorithm 3	Share the word with someone who tells three people, and each of those three people tells one other person.
Algorithm 4	Share the word with someone who tells three people, and two of those three people each tells one other person.

depending on how the requests stated in the task instructions are interpreted, and will influence what is considered to be an apt algorithm.

5. Method

5.1. Participants, setting, and data collection

At the time of data collection, the participants of our study, Chad, Gil and Lome (pseudonyms), were enrolled in a pre-degree course in algebra at a large New Zealand university. None of them had studied graph theory before and were recruited as part of a larger research project that explored students' engagement with discrete mathematics through contextualized tasks (Yoon, Chin, Moala, & Choy, 2018). Chad, Gil and Lome worked together on The Jandals Problem in a one-hour session, which took place outside of class time and was video-recorded. The group worked in the presence of a researcher, who answered clarification questions but avoided providing mathematical hints.

5.2. Data analysis

The overall analysis involved an *open interpretation* of the data (Clement, 2000), which is appropriate for constructing new (preliminary) hypotheses about students' learning processes, rather than developing the reliability of established constructs. Our analysis involved multiple stages. At the first stage, we identified the algorithms[1] that the group created and considered apt for the task (see Table 1 below). We assumed and operationalized an algorithm *as apt for the task* from the group's perspective if the group indicated that they were satisfied with presenting it as their response to the task.

All four algorithms we identified (Table 1) comprise a list of specified attributes that the person to be found in a given friendship network should have. For example, Algorithm 1 specifies that the person to be found should have the attribute of "tells three people". Note, "[a person who] tells three people" is equivalent to "[a person who] has three friends" and "[a node] that is adjacent to three nodes". We use "tells three people" in our analysis as this was how the students referred to the attribute. Implementing Algorithm 1 requires searching the friendship network for a person that "tells three people". Similarly, implementing Algorithm 2 requires searching the network for a person with two attributes: 1) "tells three people"; and 2) "one of those three people tells two other people".

At the second stage, we identified the specified attributes that appeared in all four algorithms, which turned out to be the "tells three people" attribute. The presence of the "tells three people" attribute in all four of the group's algorithms, meant that all four algorithms would produce incorrect solutions for networks in which the quickest person does not tell three people. We operationalize the group's transition from one incorrect algorithm to another in terms of the persistence of the tells three people attribute across successive iterations of the group's algorithms.

At the third stage, we examined how the group created each of the four algorithms, tracing the evolution in, and the justifications for, aptness of both the algorithms and the specified attributes. We distinguished between the aptness of an algorithm for the task, and the aptness of a specified attribute. The former was clarified above. For the latter, we assumed and operationalized a specified attribute as apt from the group's perspective if the attribute was included in an algorithm which the group considered apt for the task. Conversely, when the group declared an algorithm inapt for the task, we assumed that at least one specified attribute was inapt. In sum, we operationalized *apt attributes* as those that were kept in (or added to) the algorithm, and *inapt attributes* as those that were removed from an algorithm.

One researcher (first author) created the initial explanations for how the students revised their algorithms, and then two other researchers (co-authors) independently evaluated the viability of the initial explanations by the extent to which they captured the three different iterations in which the students created an algorithm.

6. Results and analysis

We present four chronological episodes from Chad, Gil and Lome's work on The Jandals Problem that describe how the group revised and validated their algorithms, while retaining the specified attribute of "tells three people" throughout. Each episode begins

[1] Throughout the session, the students and researcher switched between "algorithm" and "method", and we preserve both when describing and analyzing the group's work.

with our *account-of* (Mason, 2002) the group's work, in which we describe the emergence, testing and revision of algorithms, followed by our *account-for* (Mason, 2002), in which we offer explanations for the group's revisions.

During our analysis of the data, we noticed two common themes in our accounting for the group's revisions, which we have termed *localized considerations* and *patching*. Localized considerations refers to the practice of evaluating the aptness of something based on a limited subset of information. Localized considerations can be advantageous in some situations and disadvantageous in other situations. For instance, if one wants to find a "good" node in a graph to start running an algorithm, among a large number of potentially good starting nodes, then it might be efficient to concentrate on a small subset of nodes, and start at the best point within the small subset. However, localized considerations can be disadvantageous in situations that require determining and confirming that a node is "the best" across all alternatives. Patching refers to the practice of keeping some specified attributes, and removing others and/or adding new ones without altering the fundamental structure of the algorithm being revised. We develop *patching* and *localized considerations* more fully through the episodes below.

6.1. Episode 1: the emergence of the elimination method and uncertainty about its aptness as an algorithm

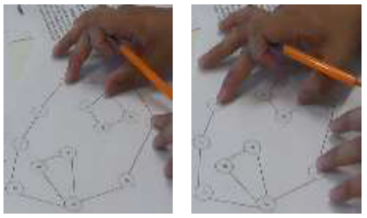
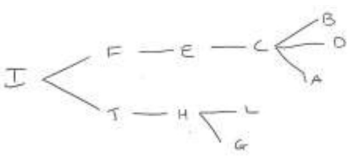
Account-of

After the group reads the task instructions, Gil looks at Fig. 1 and says:

1 *Gil*: We just need to find a person that would spread it quickly. Like…
2 *Chad*: *E*.
3 *Lome*: *E* would tell *F*. *F* would tell *I*.
4 *Chad*: *E* would tell *F* and *C*.
5 *Lome*: Yeah. But it [*pointing at E*] would take long to get to *H*. If you start from *D*, it will go straight from *G* to *H*.
6 *Gil*: Say you start at *D*, it would be, day one [*pointing at C and G*]. Day two, *C* will tell *E, B, A*, and *G* will tell *H*. Then day four…
7 *Lome*: I have an idea. Pick a person, figure out how many days it would take for the word to reach everyone. Like, start at *D*, find out how many days it would take, write that down, move to the next one…like an elimination method.
8 *Gil*: But how do we find a good one?
9 *Lome*: A good starting point?
10 *Chad*: If you go to *H* you can tell three people [*pointing at J, L, G*] in one day, then…
11 *Gil*: On second day, *K, N, I, D*…
12 *Chad*: Yeah…so, it's like, umm…find the one with the least days possible.
13 *Lome*: Should we do it by ourselves? Or work as a team? I'll do my own thing then I'll get back to you guys.

Then each student determines in his own way, how many days it would take for the word to spread throughout the network from different starting persons (see Table 2). The students collectively consider *C, I, L, J, H, G,* and *M* as starting persons. Both Chad and

Table 2
The students' elimination methods.

Student	Description	Illustration/Diagram	Starting Persons considered
Lome	The first letter in the first circle denotes the starting person. The other letters in the first circle denote the people the word reaches on day one. The letters in the second circle denotes the people that the word reaches on day two, and so forth. The total number of circles is the number of days that it takes for the word to spread throughout the entire network (e.g., six days from *C*).	 *C, H, L, M*	*C, H, L, M*
Chad	The starting person is marked with one finger (e.g., person *H*). Then, different fingers are used to mark the people the word reaches on day one—the friends of *H* (i.e., *G. L. J*); day two—the collective friends of *G, L, J* that have not been reached; and so forth until all persons in the network have been marked.	 *H, I, J, L*	*H, I, J, L*
Gil	The first letter (e.g., *I*) in the tree-like structure denotes the starting person. A line is drawn from *I* to each of *I*'s friends (*F*, *J*). Then for both *F* and *J*, a line is drawn from it to each of its friends (but not the friends that are already in the tree). The foregoing step is repeated for each of *F* and *J*'s friends, and so forth, until all persons are in the tree.	 *G, I*	*G, I*

Lome get four days starting at *H,* which made it the quickest person. (Gil incorrectly calculated that *G* yields six days, when in fact it also yields four days). Then Lome remarks:

14 *Lome*: So we've solved the problem! I got four days. You [*Chad*] got four days. See if you [*Gil*] can get four days…If you get four, I reckon we've solved it.

Lome then looks back at the task instructions, turns to the researcher and says:

15 *Lome*: What's an algorithm? This [*points to his diagrams as shown in* Table 2] is not an algorithm is it?
16 *Researcher*: An algorithm is like a method. So it's not your solution, it…
17 *Gil*: It's like the way you got it.
18 *Researcher*: Yeah, so that she can use it for any other network, because this is just one of many different friendship networks across the campus.
19 *Lome*: Can we say we just did elimination method?
20 *Researcher*: You've got to explain it as well as you can so that Xanthe can use it for a different one that she is given. [*All students nod*].

Account-for

The episode began with Gil uttering the need to find "a person who would spread the word quickly" (see [1]). Lome and Chad considered *E* as a starting person (see [2–3]), but Lome promptly realized that *D* was a better starting person (see [5]). Starting from *D*, Gil identified whom the word reaches on successive days (see [6]), which led Lome in [7] to generalize this approach for any person. This generalized recursive approach was coined as an "elimination method."

Spurred by Gil's question in [8], the group considered in [9–12] the aptness of the elimination method for finding 'a good starting person' in Fig. 1. In [12], Chad interpreted 'a good starting person' as 'the person corresponding to the least days possible', implying that the numbers obtained (via the elimination method) for each person would be compared at some point. While Table 2 shows that each student had distinct versions of the elimination method, the students endorsed each other's results, and considered the obtainment of the same number of days for a particular person in the network as an indication that their methods were correct. Thus, the group considered their three versions of the elimination method equally apt for them to find the quickest person in Fig. 1.

The practice of localized considerations is evident in how the students develop and evaluate the aptness of the elimination method. To recall, localized considerations refers to determining the aptness/inaptness of something with respect to a limited subset of information. In this episode, the group justified their choice of person *D* over *E* by considering how quickly the word reaches person *H*, rather than how quickly the word reaches everyone in the network. Table 2 also shows that each student applied his elimination method only on a subset of persons in the network. Yet, they were happy to conclude that person *H* was the quickest out of all persons in the network.

In [15], Lome sought clarification from the researcher regarding the term "algorithm". Lome's act can be viewed as an attempt to expand the initial considerations of aptness for the elimination method. Whereas the aptness of elimination method (to the task) was initially considered entirely with respect to finding the quickest person in Fig. 1, the group recognized [15–20] that the aptness of the elimination method needed to be determined with respect to other aspects. While Lome's first question in [15] suggests uncertainty about what counted as an algorithm, he immediately sought confirmation that the elimination method was not an algorithm. Gil and the researcher [16–18] described what an algorithm in general is — a method; the way you got it — and what an apt algorithm for the task ought to do (e.g., one that Xanthe can use to find the quickest person in any network). The episode ended with uncertainty over the aptness of the elimination method[2] for the task, as evidenced by Lome's question in [19].

6.2. Episode 2: the patching of algorithm 1

Account-of

After the discussion in [15–20], Lome turns to Gil and says:

21 *Lome*: See if you can get that four one [a person from whom it takes four days].
22 *Gil*: Which one did you [looking at Lome)] start with?
23 *Lome*: H. You start with H.
24 *Chad*: So it's just one method?
25 *Lome*: No it means if we're given this one [pointing at Fig. 1] she might be given a different one, but she needs to figure out how to do her one from the algorithm we give her.
26 *Gil*: Yeah. So it'll be like, the algorithm would be like, the starting person should tell three people because [*points to* Fig. 1] if you told *H*, *H* would tell *L*, *G*, and *J*. And then, it spreads.
27 *Lome*: Yeah. Cool [*nodding with Chad*].

[2] The elimination method is a basic version of the most efficient algorithm for the task in The Jandals Problem (see Jungnickel, 2008).

The researcher points to the task instructions and says:

28 *Researcher*: Can I get you to read what the method needs to do?

[*The students look at the task instructions, and Chad reads it aloud*]

29 *Lome*: So [*looks at* Fig. 1] we said she should share it with someone who tells three people. But then mind you, if she starts at L, L tells three people but it doesn't work as fast.
30 *Gil*: Yeah, that's true.
31 *Lome*: So maybe [points to H] the starting person tells three people [points to L, *J*, and G] but one of those three people [points to L] tells two other people.
32 *Gil*: Yeah [nodding]
33 *Lome*: Because this person [*points to C*] tells four people, but none of those people tells two other people. Yeah, that's an algorithm!

The researcher asks, "Are you happy?" The students all reply, "Yes."

Account-for

At the beginning [21–23], Lome sought confirmation that *H* was the quickest person in Fig. 1. Then, after Chad and Lome further clarified what an apt algorithm for the task should do, Gil proposed a candidate – "share the word with someone who tells three people"[3] (Algorithm 1) which the group approved (see [24–26]). The elimination method and Algorithm 1 exemplify respectively the two types of algorithms discussed in Section 4.1. Whereas the elimination method constitutes a set of instructions applied recursively to the persons in the network, resulting in numbers that are compared, Algorithm 1 searches the network for a person that has specified attributes. Nevertheless, Algorithm 1 was generated by noticing particular attributes of the person (*H*) that the elimination method yielded. Algorithm 1 was considered apt for the task because the quickest person (*H*) in Fig. 1 was easily identified by the specified "[starting person] tells three people" attribute. And while the elimination method was, from the group's perspective, apt for finding the quickest person in Fig. 1 it was not considered something that could be offered to Xanthe (the client).

Lome in [29] disqualified Algorithm 1 from being apt for the task because it identified not only the quickest person (*H*) but also a non-quickest person (*L*). We infer, from Lome's actions, two goals that an apt algorithm for the task must accomplish:

1) must identify the quickest person; that is, the quickest person must have the specified attribute(s);
2) must not identify a non-quickest person; that is, a non-quickest person must not have the specified attribute(s).

In light of these two goals, we can say that focusing only on the first goal (localized considerations) led to Algorithm 1 being deemed an apt algorithm for the task. Then, the emergence of the second goal, expanding of the initial considerations, destabilized the aptness of Algorithm 1. The inaptness of Algorithm 1 for the task led Lome to propose another algorithm (Algorithm 2) [31–33]. Diagram 1 summarizes the group's transition from Algorithm 1 to Algorithm 2.

We note that Algorithm 2 resembles the initial one closely: both comprise a list of attributes that specify the person to be found, and both lists include the attribute of "tells three people". Algorithm 2 can be construed as Algorithm 1 plus "one of the three people tells two other people", and the group's transition from Algorithm 1 to Algorithm 2 [29–33] can be interpreted in terms of patching. To recall, patching involves mending an inapt algorithm by keeping some of the specified attributes, and removing others and/or adding new ones while keeping the underlying structure of the algorithm intact. As employed in the current episode, patching works by keeping the specified attribute(s) that satisfy the first goal (must identify the quickest person), removing all others, and then adding an attribute so that the resultant algorithm satisfies both goals (must identify the quickest person and only the quickest person). Note, an attribute "satisfies the first goal" if the quickest person under consideration possesses the attribute. And, an attribute "satisfies the second goal" if the non-quickest person(s) under consideration does not possess the attribute.

We demonstrate the practice of patching on the transition from Algorithm 1 to Algorithm 2. The "[someone who] tells three people" specified attribute of Algorithm 1 satisfies the first goal (mentioned above), and because it is the only specified attribute in

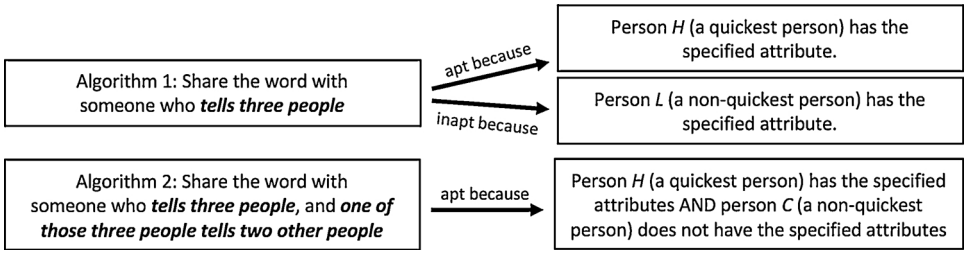


Diagram 1. The transition from Algorithm 1 to Algorithm 2.

[3] This is not exactly what was said but they seem to treat it as such.

Algorithm 1 no attribute is (or can be) removed. Then, the specified attribute of "one of the three people tells two other people" is added to Algorithm 1, and the resultant algorithm is confirmed to satisfy both goals. We see localized considerations going hand-in-hand with patching when the group confirmed the aptness Algorithm 2 for the task. Specifically, the group determined the aptness of Algorithm 2 by focusing only on persons *H* and *C*, without systematically scanning the other persons in the network.

6.3. Episode 3: Algorithm 2 is patched and considerations of aptness are expanded but remain localized

Account-of

After Lome writes down their second algorithm, the researcher hands the group a new friendship network (Fig. 2), and asks them to test their algorithm on it. The students, in turn, embark on finding the quickest person in the new network with their elimination methods from Episode 1. The group find that *P*, *Q*, *R*, *S*, and *T* are the quickest starting persons which all take three days, while *O* and *U* both take four days (note that it takes just two days from *R* and *S*, making them the only quickest persons). Then Lome asks Gil and Chad:

34 *Lome*: OK, so why wouldn't she tell like…U, but tell Q instead? What's the method? Obviously Q is quicker but why would she tell Q and not U?
35 *Gil*: Because, the algorithm…we said she has to tell someone who tells three people, so Q tells three people, S, P, and O. Then, one of them must tell two other people [He pauses, and stares at Fig. 2].
36 *Lome*: So is [looking at researcher] an algorithm like an equation, kind of?
37 *Researcher*: It's just a method, a set of instructions, do this first, and then do that next.
38 *Lome*: Oh yeah.
39 *Gil*: I think [pointing at Fig. 2] the method should be to just share the word with someone who tells three people.
40 *Lome*: No, because L [in Figure 11] tells three people, but L isn't the quickest.
41 *Gil*: Yeah, no, it should be…start with someone who tells three people, and one of them tells two other people.
42 *Lome*: But L [in Fig. 1] tells three people, and one of those three people [points to H] tells two people. So it [Algorithm 2] doesn't work. Yeah, see it actually doesn't work here either [pointing at Fig. 2] because T tells three people [R, U, and S], but none of them tells two other people.
43 *Chad*: Wait, so what's the algorithm?
44 *Lome*: It's supposed to be a set of instructions…that she can use on any network.
45 *Gil*: Yeah it doesn't matter what network it is…We need a general pattern.
46 *Lome*: I think it's an equation…because say she has hundreds of these [networks] she doesn't want to do elimination method for every one. What if there's a network with a thousand people? She'll be there for ages counting… So is that our next way, figure out an equation?
47 *Gil*: I'm thinking for this one [Fig. 2], if you start on T who tells three people, it works because they [pointing at R and S] don't tell the same person. And that's why L [in Fig. 1] doesn't work because the two people they tell [pointing at K and N] only tell one person [M].
48 *Lome*: Yeah! Can you say that again?
49 *Gil*: The starting person tells three people, and each of those three people tells a different person.
50 *Lome*: Wait…does that work on [pointing at H in Fig. 1]? Ah, it does. Okay, yeah that's it.

Account-for

At the beginning of the episode, the group used the elimination method, rather than Algorithm 2, as proposed by the researcher, to identify the quickest person(s) in Fig. 2. Yet, the group considered the elimination method inapt for the task. Lome's remarks in [34] suggest that an apt algorithm for the task should justify why one person was chosen over another – something which the elimination method did not do.

After Lome sought clarification from the researcher regarding what an algorithm is (see [36–38]), Gil re-proposed the second algorithm from Episode 2. However, Lome in [42] immediately declared Algorithm 2′s inaptness for the task, because it did not accomplish the two goals which an apt algorithm should accomplish, previously noted in Episode 2: the algorithm must identify the quickest person(s), and the algorithm must not identify a non-quickest person.

Additionally, we see an expansion of the group's considerations of aptness for Algorithm 2. In Episode 2, Algorithm 2's aptness was determined specifically with respect to persons *H* and *C* in Fig. 1. In the current episode, considerations were expanded to include *L* in Fig. 1 and *Q* in Fig. 2. Further expansion of the group's considerations of aptness occurred when the group clarified once again what an apt algorithm for the task should look like (see [43–46]). The expanded considerations were followed by Lome declaring and elaborating on the inaptness of the elimination method for the task. Specifically, Lome imagined a network with "a thousand people", for which the elimination method would take "ages [to implement]".

Then, in [47–50] Gil proposed a third algorithm (Algorithm 3). The transition from the inapt Algorithm 2 to Algorithm 3 is summarized in Diagram 2 . Again, we see that the third algorithm (like the first two algorithms) possesses the specified attribute of "[starting person] tells three people".

The persistence of the "tells three people" attribute is particularly fascinating in the current episode because the group had earlier found five quickest persons in Fig. 2, two of whom (*R* and *S*) do not "tell three people". Again, we can describe the transition from Algorithm 2 to 3, and the persistence of the "tells three people" attribute in terms of patching and localized considerations. Although

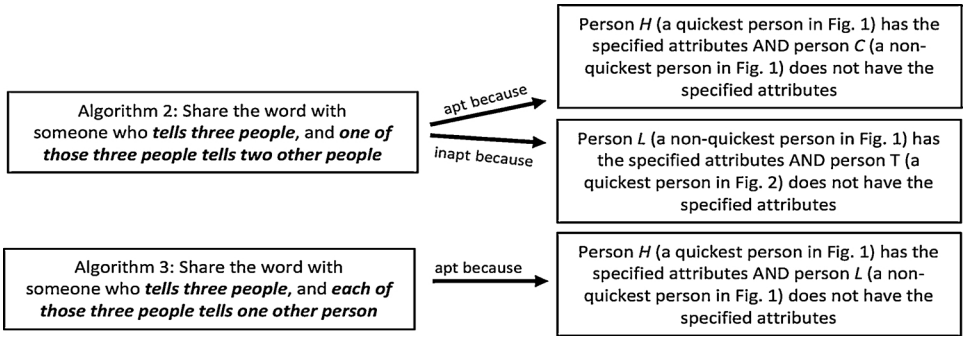


Diagram 2. The transition from Algorithm 2 to Algorithm 3.

the group identified five quickest persons in Fig. 2, the group localized their considerations to person *T* who so happened to have the "tells three people" attribute. Then, via patching, the group kept the specified attribute of Algorithm 2 [47–50] (i.e., "[starting person] tells three people") that identified the quickest person(s) being considered (i.e., *T*), and removed all other specified attributes (i.e., one of the three people tells two other people). Then, the attribute "each of the three people tells a different person" is added, yielding Algorithm 3, whose aptness for the task was confirmed only on persons *H* and *L* in Fig. 1.

6.4. Episode 4: the inaptness of the "tells three people" attribute is revealed

Account-of

After Gil writes down Algorithm 3 he turns to Lome and Chad and says:

51 *Gil*: Do you guys agree with that? [*Lome and Chad nod*] So if you start at *Q, Q* tells three people, then… [*he stares at* Fig. 2].
52 *Lome*: Then out of those three people *P* and *S* tell a different person.
53 *Chad*: So we can say that the starting person tells three people, and two of the three people, tell a different person each [*Gil and Lome nod*].
54 *Lome*: So in this network [Fig. 2] you tell…the one with the most friends? Wait, but if we start on *C* [*in* Fig. 1], *C* tells four people, which is the most but *C's* not the fastest one.
55 *Gil:* Yeah because the starting one tells three people.
56 *Lome*: Why is it three? What if you're given one that doesn't have a three?
57 *Gil*: Then I don't know. We already figured it out, now you just made it more complicated.
58 *Lome*: Let's just keep figuring it out.
59 *Gil*: So the way I worded it [Algorithm 3]…here [*in* Fig. 2] person *Q* tells three people [pointing at *O*, *P*, and *S*] and each of the three must tell a different person, but *O* cannot tell a different person.
60 *Lome*: Yeah but *P* and *S* can tell different ones. It's all good, because what you're saying is for these networks [*points to* Fig. 1 and 2] that we're given, the starting person tells three people and….
61 *Chad*: And at least two of them tell a different person each.
62 *Gil*: Yeah, out of these networks that we're given, but not other ones, because we don't know how they work yet…Still don't know the algorithm. We've figured it out for these ones, but not for other ones…We need something that will work on any network, and not so specific, something that will work not only on these two, but all hundreds of them.

The researcher gives the group another friendship network (Fig. 3). All three students apply the elimination methods on Fig. 3. Lome then says:

63 *Lome*: *C* takes two days. I think that's the quickest.
64 Chad: Yeah.
65 *Lome*: Our algorithm [*Algorithm 4*] doesn't work here, see *C* doesn't tell three. So, what about like, she share it with someone who tells two people, like *C*, and…Hmmm.
66 *Gil*: We need a pattern, a universal pattern, one that'll be like "yep, tell this person" [*both Lome and Chad nod*]. You know? Something like…easy.

Gil then places Fig. 2 in front of him, and says:

67 *Gil*: The good ones are *P*, *Q*, and *T*.
68 *Lome*: But why? Why is *U* not a good one? Obviously, we can see it but that's not good enough. I think it's an equation to solve it. Maybe if we can figure out the pattern, we can figure out the equation. So what's the pattern here?

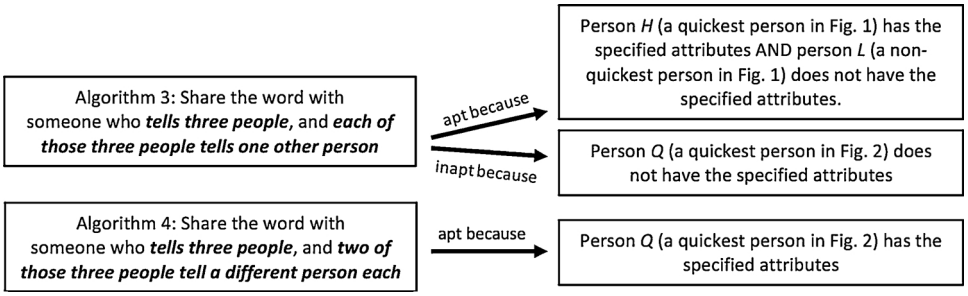


Diagram 3. The transition from Algorithm 3 to Algorithm 4.

Account-for

At the beginning, Gil declared the inaptness of Algorithm 3 (see [51]) because it would not identify person *Q*, a quickest person in Fig. 2. Lome and Chad then produced Algorithm 4, which still possessed the "tells three people" attribute. Once again, localized considerations and patching are evident in the creation of the fourth algorithm, and account for the persistence of the "tells three people" attribute (Diagram 3).

Lome then observed in [54] that person *Q* had the most friends in Fig. 2. This observation seemed to make Lome wonder whether an apt algorithm for the task should find the person with the most friends. However, Lome realized, by way of *C* in Fig. 1, that finding a person with the most friends contradicted the goal of not finding a non-quickest person. Then in [55] Gil responded to Lome, asserting that an apt algorithm for the task must have the "tells three people" attribute. Lome immediately [56] countered, expanding their considerations of aptness for an apt algorithm by imagining a friendship network in which no person tells three people. Subsequently, the group acknowledged that "tells three people" was not a necessary attribute of the quickest person in any network, and confirmed the aptness of Algorithm 4 with respect to Figs. 1 and 2. Thus, Lome's imaginary network destabilized the persistence of the "tells three people" attribute. Interestingly, both Figs. 1 and 2, like Lome's network, had quickest persons who do not tell three people, yet, the attribute persisted in both networks. One critical difference is that in Lome's network no person tells three people.

When they were given Fig. 3, the group further acknowledged Algorithm 4′s inaptness for the task, because it did not identify the quickest person *C*. Patching is evident when Lome determined that the "tells three people" attribute needed to be removed, and other attributes were to be added. In the latter part of [65] we see Lome attempting to create a new algorithm — i.e., "share the word with someone who tells two people […]", which in fact accomplished the goal of finding the quickest person *C* in Fig. 3. The session ended however before the group was able to produce a fifth algorithm.

7. Summary and discussion

Past research has tended to focus on the persistence and the desired rejection and replacement of an inappropriate algorithm in its entirety. Consequently, explanations for the persistence of students' inappropriate algorithms are given in terms of things which the students did not do (e.g., not validating their algorithm; not able to transition from linear to non-linear algorithms). Such explanations shed little to no light on the mechanisms by which students revise their inappropriate algorithm upon testing, irrespective of whether the resultant algorithm is appropriate or inappropriate. In the present study we took a more fine-grained view, focusing on the iterative refinement and augmentation of an algorithm, rather than its wholesale replacement.

Specifically, we investigated the persistence of inappropriate aspects of students' algorithms during multiple test-and-revise iterations in a single task. We asked: how do students decide what to change and what to keep when revising their algorithm upon testing? We conducted a fine-grained analysis of the collaborative work of three students on The Jandals Problem — a graph theory task which invited the students to develop an algorithm for solving a contextualized problem. The group's initial algorithm specified a priori attributes that the solution for the problem should have, and then searched through potential solutions for the one which possessed the specified attributes. Then, noticing that their initial algorithm possessed inappropriate attributes, the group engaged in three iterations of testing and revising, producing three revised algorithms that were strikingly similar to the initial algorithm, but still possessed inappropriate attributes. We introduced two terms, *localized considerations* and *patching*, to describe how the students revised and validated their algorithms while retaining one specified attribute of their initial algorithm throughout.

Analyzing the three iterations whereby the group transitioned from one algorithm to another, we first noticed that the group explicitly evaluated the aptness of an algorithm with respect to two goals: 1) the algorithm must identify the quickest person; and 2) it must not identify a non-quickest person. In light of these two goals, the group deemed an algorithm apt for the task if it accomplished both goals, and inapt otherwise. In each testing and revising iteration, we described the group's revision of their inapt algorithm as *patching*, which involves mending an inapt algorithm by keeping some of the specified attributes, removing others and/or adding new ones, while maintaining the underlying structure of the algorithm. More specifically, the group preserved the specified attribute(s) of the current algorithm that satisfied the first goal, removed all other attributes, and then added other attributes so that the resultant algorithm satisfied both goals. To recall, a specified attribute of an algorithm satisfied the first goal if the quickest person under consideration had the specified attribute. *Localized considerations*, which refers to determining the aptness/inaptness of something with respect to a limited subset of information, was evident when the students confirmed the aptness of their revised algorithm. For

example, the group would only test that the algorithm worked correctly on one network, rather than on both networks they were given; or they would test that the algorithm worked on the two given networks, but did not test the algorithm on any other network as requested by the task.

After having created four iterations of their algorithm, the group recognized in the final episode that their fourth algorithm was inapt because it did not identify the quickest person in Lome's imaginary network (which was concretized by Fig. 3). The group's response can be interpreted as an attempt to patch their algorithm, and in doing so, they determined that the "tells three people" attribute did not satisfy the first goal mentioned above (i.e., to identify the quickest person), eventuating in the removal of the attribute. In this way, we could say that it was through a particular iteration of patching that the "tells three people" attribute was finally removed.

We acknowledge that the two terms, localized considerations and patching, introduced in this paper, are tied to the methodology of our study, particularly the task that we used. Localized considerations might have been evident because of the multi-faceted nature of The Jandals Problem. That is, the task contained multiple requests such as: to find the quickest person in a network; to create an algorithm for a particular client (Xanthe); to create an algorithm that could identify the quickest person(s) in multiple networks that were structurally different (e.g., sometimes a network had one quickest person and other times had more than one quickest person); to frame the algorithm as a letter. The multi-faceted nature of the task allowed us to discuss how the aptness of something (e.g., an algorithm) was considered with respect to some facet of the task, but not other facets.

Moreover, both patching and localized considerations may have been influenced by the fact that the task motivated the development of an algorithm. Algorithms are often viewed and presented as linear steps that need to be taken, recipe-like, to achieve a certain outcome—indeed, our perspective in the present study has been to view an algorithm as a set of rules or instructions. Even though algorithms have an underlying structure, it is often their segments (the steps) that are emphasized, rather than their holistic nature. The emphasis on the algorithm as a linear list of segments may partly explain the students' tendency to revise their inapt algorithms using localized considerations and patching: both of which focus the students on addressing segments within their lists of attributes, even though it could have been more helpful to revise the underlying structure of their algorithm instead. The language of "debugging" in computer science might reinforce the emphasis on revising algorithms by focusing on segments, with error messages encouraging programmers to locate and fix specific lines of code. While such patching and localized considerations are important, our study shows how their overuse can result in the persistence of inapt algorithms that need to be revised at a deeper structural level. We propose that instruction on algorithms could be enhanced by giving students opportunities to compare and appreciate algorithms that differ on a holistic structural level, like the elimination method compared to the algorithms endorsed by the students in our study.

Localized considerations and patching might also be useful contributions to the discussion of helping students reject their persistent incorrect algorithm (e.g., see Booth et al., 2014; Clarke & Veith, 2003; Lappan & Bouck, 1998). As previously mentioned, one particularly useful strategy for getting students to reject an incorrect algorithm is the use of counterexamples. It can be inferred from past studies that some counterexamples may be more effective than others in facilitating the rejection of students' incorrect algorithms (e.g., see Zazkis & Chernoff, 2008). In the present study, the persistence of an incorrect algorithm was operationalized as the persistence of a specified attribute of the initial algorithm; and thus, we wonder how such an attribute might cease to appear in the students' algorithms.

All three networks (Figs. 1–3) that the students encountered in their work can be thought of as counterexamples to the group's algorithms, specifically because there is at least one quickest person in each of the three networks who does not have the "tells three people" attribute. However, it was only upon encountering Fig. 3 that the students rejected the "tells three people" attribute. From the perspective of the counterexamples literature, the network in Fig. 3 might be viewed as a more effective (convincing) counterexample than Fig. 1 and Fig. 2 because no person in Fig. 3 had the "tells three people" attribute; and as such the students realized that the "tells three people" attribute undermined the ability of their algorithm to find the quickest person in any network.

Suggestions from the counterexamples literature for helping students reject their incorrect algorithm seem to concern examining the students' algorithm, then finding a counterexample (a friendship network in our case) that would make the student realize that the inclusion of the attribute in the algorithm undermines the appropriateness of the algorithm. However, we note that there is an implicit assumption in such suggestions: recognizing the inappropriate nature of the attribute will result in its removal. More generally, such suggestions equate recognizing the inappropriateness of an algorithm with its rejection and replacement with a better algorithm. We add to the explanations and suggestions from the counterexamples literature by proposing that while Fig. 3 may have exposed the problematic nature of the "tells three people" attribute and motivated the need to revise their algorithm, it was through the application of patching, as explained in Episode 4 that the rejection of the "tells three people" attribute was finalized.

In general, while a counterexample may expose a faulty attribute and motivate students to revise their current algorithm, there is still a chance that after students revise their incorrect algorithm, employing practices such as patching and localized considerations, they will end up with another algorithm that still possesses the attribute. We hypothesize that in some cases an attribute might only be removed when it is no longer apt with respect to the practices by which students revise their algorithm. We also hypothesize that choosing a counterexample that would help students reject their incorrect algorithm, may require not only considering the algorithm carefully, but understanding how the students revise and validate their algorithm(s). The foregoing hypotheses may be fruitful avenues for further study.

The growing prominence of discrete mathematics coupled with the significance of algorithms in discrete mathematics have led many to suggest that enhancing students' competence in the algorithmatizing approach, will benefit students' mathematics learning as a whole (see Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in et al., 2013Committee on the

Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013Committee on the Mathematical Sciences in 2025 et al., 2013; Hart & Sandefur, 2018). However, very little research has explored how student-invented algorithms emerge. The study reported in this paper demonstrates that exploring the ways in which students develop their algorithms can shed light on problematic issues such as the persistence of inappropriate aspects of students' algorithms, and can lead to new suggestions for how to support students develop their algorithms. We propose that attempts to enhance students algorithmatizing competencies would benefit from more research that explores the nuanced ways in which students create, revise, and validate their algorithms.

Acknowledgements

The data presented in this paper were collected as part of a two-year long research project funded by the Teaching and Learning Research Initiative, and administered by the New Zealand Council for Educational Research.

References

Booth, J. L., Barbieri, C., Eyer, F., & Paré-Blagoev, E. J. (2014). Persistent and pernicious errors in algebraic problem solving. *The Journal of Problem Solving, 7*(1), 3.
Brown, G., & Quinn, R. J. (2006). Algebra students' difficulty with fractions: An error analysis. *Australian Mathematics Teacher, 62*(4), 28–40.
Clarke, E., & Veith, H. (2003). Counterexamples revisited: Principles, algorithms, applications. In N. Dershowitz (Ed.). *Verification: Theory and practice* (pp. 208–224). Berlin: Springer.
Clement, J. (2000). Analysis of clinical interviews: Foundation and model viability. In A. E. Kelly, & R. Lesh (Eds.). *Handbook of research design in mathematics and science education* (pp. 547–589). Mahwah, NJ: Lawrence Erlbaum Associates.
Committee on the Mathematical Sciences in 2025, Board on Mathematical Sciences and Their Applications, & Division on Engineering and Physical Sciences (2013). *The mathematical sciences in 2025.* Washington, DC: The National Academies Press.
De Bock, D., Van Dooren, W., Janssens, D., & Verschaffel, L. (2002). Improper use of linear reasoning: An in-depth study of the nature and the irresistibility of secondary school students' errors. *Educational Studies in Mathematics, 50*(3), 311–334.
Dossey, J. A. (1991). Discrete mathematics: The math for our time. In M. J. Kenney, & C. R. Hirsch (Eds.). *Discrete mathematics across the curriculum, K-12, 1991 yearbook* (pp. 1–9). NCTM.
Fischbein, E. (1987). *Intuition in science and mathematics.* Dordrecht: D. Reidel.
Hart, E. W., & Martin, W. G. (2018). Discrete mathematics is essential mathematics in a 21st century school curriculum. In E. W. Hart, & J. Sandefur (Eds.). *Teaching and learning discrete mathematics worldwide: Curriculum and research* (pp. 3–19). Cham: Springer.
Hart, E. W., & Sandefur, J. (Eds.). (2018). *Teaching and learning discrete mathematics worldwide: Curriculum and research.* Cham: Springer.
Jungnickel, D. (2008). *Graphs, networks and algorithms.* Berlin: Springer.
Klymchuk, S. (2012). Using counter-examples in teaching & learning of calculus: Students' attitudes and performance. *Mathematics Teaching-Research Journal Online, 5*(4).
Kontorovich, I., Koichu, B., Leikin, R., & Berman, A. (2012). An exploratory framework for handling the complexity of mathematical problem posing in small groups. *The Journal of Mathematical Behavior, 31*(1), 149–161.
Kontorovich, I. (2016). Considerations of aptness in mathematical problem posing: Students, teachers and expert working on billiard task. *Far East Journal of Mathematical Education, 16*(3), 243–260.
Lappan, G., & Bouck, M. K. (1998). Developing algorithms for adding and subtracting fractions. In L. J. Morrow, & M. J. Kenny (Eds.). *The teaching and learning of algorithms in school mathematics* (pp. 183–197). NCTM.
Mason, J. (2002). *Researching your own practice: The discipline of noticing.* Psychology Press.
Maurer, S. B., & Ralston, A. (1991). Algorithms: You cannot do discrete mathematics without them. In M. J. Kenney, & C. R. Hirsch (Eds.). *Discrete mathematics across the curriculum, K-12. 1991 yearbook* (pp. 195–206). NCTM.
Thomas, M. O. J. (2014). Algorithms. In S. Lerman (Ed.). *Encyclopedia of mathematics education* (pp. 36–38). Netherlands: Springer.
Verschaffel, L., Greer, B., & De Corte, E. (2000). *Making sense of word problems.* The Netherlands: Lisse.
Wyndhamn, J., & Säljö, R. (1997). Word problems and mathematical reasoning: A study of children's mastery of reference and meaning in textual realities. *Learning and Instruction, 7*(4), 361–382.
Yoon, C., Moala, J. G., & Chin, S. L. (2016). *The Jandals Problem.* Unpublished manuscript.
Yoon, C., Chin, S. L., Moala, J. G., & Choy, B. H. (2018). Entering into dialogue about the mathematical value of contextual mathematising tasks. *Mathematics Education Research Journal, 30*(1), 21–37.
Zaslavsky, O., & Ron, G. (1998). Students' understanding of the role of counter-examples. In A. Olivier, & K. Newstead (Vol. Eds.), *Proceedings of the 22nd conference of the international group for the psychology of mathematics education: Vol. 4*, (pp. 225–232). Stellenbosch, South Africa: University of Stellenbosch.
Zazkis, R., & Chernoff, E. (2008). What makes a counterexample exemplary? *Educational Studies in Mathematics, 68*(3), 195–208.