Computational thinking as a tool in primary and secondary mathematical problem solving: a literature review

KIM ANDRÉ STAVENÆS REFVIK AND ANNETTE HESSEN BJERKE

In this systematic literature review, we investigate the connections between computational thinking and problem solving in the context of primary and secondary mathematics education. We do this by exploring how and at which steps in the mathematics problem-solving process seven peer reviewed studies report on the inclusion of computational thinking concepts, practices and perspectives. Overall, the studies show that it is possible and at times beneficial to include computational thinking in mathematics problem solving. However, more research is needed to see whether simply including computational thinking and its programming tools enhances students' problem-solving skills in mathematics.

In recent years, several countries have added computational thinking (CT) to school curricula. Whilst countries like Denmark, Hungary, Italy, Portugal and Turkey have introduced CT as a separate informatics subject, Sweden, Finland and France have incorporated CT into existing subjects, including mathematics (Bocconi et al., 2016; Gueudet et al., 2017; Heintz et al., 2017; Hemmi et al., 2017). Mathematics has been found to provide a meaningful context with a relevant set of problems for applying CT (Hambrusch et al., 2009). With Norway's 2020 curriculum revision, it joined countries introducing CT into mathematics curricula across grade levels (Directorate of Education, 2020). In the new curriculum, CT is described as "a process for developing strategies to decompose and solve a problem systematically, with or without the use of digital tools" (Directorate of Education, 2020), indicating a close relationship between problem solving and CT.

In the context of education and schooling, CT was first introduced by Papert (1980). Whilst his notion of CT did not include a definition, he

Kim André Stavenæs Refvik, *Volda University College*
Annette Hessen Bjerke, *Oslo Metropolitan University*

Refvik, K. A. S. & Bjerke, A. H. (2022). Computational thinking as a tool in primary and secondary mathematical problem solving: a literature review. *Nordic Studies in Mathematics Education*, 27 (3), 5–27.

related CT to the construction of ideas that are explicative, accessible, and powerful (Papert, 1996). He argued that the process of learning is transformed when programming computers (Papert, 1980). Hence, he introduced the programming language Logo as a learning environment for students to explore geometry and learn programming, but because of difficulties in learning programming languages, Papert's idea of a *CT for all* was to some extent ahead of its time (Resnick et al., 2009). Papert saw that implementing this type of thinking had challenges because the "visions of how to integrate *computational thinking* into everyday life were insufficiently developed" (Papert, 1980, p. 182). This assertion was later supported by empirical studies of Logo programming; teachers were found to provide more assistance than instruction, with few children improving their thinking skills (Kurland & Pea, 1985; Kurland et al., 1986). As a result, Logo disappeared from the school context within a decade (Noss & Hoyles, 1996).

In 2006, Wing's approach gave CT a renewed focus of attention. She modified Papert's ideas and defined CT as "the thought processes involved in formulating a problem and expressing its solution in a way that a computer – human or machine – can effectively carry out" (Wing, 2006, p. 33). Thus, CT is not restricted to computation or designing computer programmes; it represents a type of analytical thinking that connects to mathematical thinking through various ways of solving problems (Wing, 2008). We shall now see how, in the wake of Wing (2006), a variety of definitions and operationalisations were set forth.

CT facets

There has for long been little or no agreement about precisely what CT encompasses (Brennan & Resnick, 2012; Grover & Pea, 2013; Shute et al., 2017; Zhang & Nouri, 2019). To capture its complexity, some well-received operationalisations of CT have influenced and guided ongoing research, such as Weintrop et al.'s (2016) taxonomy and Brennan and Resnick's (2012) CT framework. Due to how Brennan and Resnick's (2012) CT framework (see table 1) makes clear distinctions between computational concepts (the concepts students engage with as they program), computational practices (the practices students develop as they engage with the concepts) and computational perspectives (the perspective students form about the world around them and about themselves), we continue in this review to use this way of speaking about different CT dimensions. We do this because, together, these dimensions appear to capture many CT facets in the growing body of research involving CT.

Table 1. *Operationalisation of Brennan and Resnick's (2012) CT framework*

CT Concepts	CT Practices	CT Perspectives
Sequences	Being incremental and iterative	Expressing
Loops	Testing and debugging	Connecting
Parallelism	Reusing and remixing	Questioning
Events	Abstracting and modularising	
Conditionals		
Operators		
Data		

Despite the lack of consensus noted above, CT in education has recently received considerable attention in policy initiatives (Bocconi et al., 2018; Hsu et al., 2018) in being a necessary 21st-century skill and competence in problem solving (Voogt et al., 2015). Recognising that problem solving has long played a key role in mathematics education and is a top priority in curriculum development (Liljedahl & Cai, 2021), it is easy to argue for including CT in mathematics education: in a world of high demands, the ability to *efficiently* interpret new information, in which problem-solving skills are a prerequisite, is viewed as becoming more important than an individual's specific knowledge (Green & Gilhooly, 2005).

Hence, problem solving (in the context of mathematics education) has been used as an argument to include CT in mathematics (Barr & Stephenson, 2011; Kallia et al., 2021; Pérez, 2018; Selby & Woollard, 2013; Sneider et al., 2014; Wing, 2008). This approach could give CT a central role in mathematics education: problem solving is found everywhere in mathematics curricula, textbooks and lesson plans (Cai et al., 2011; English & Gainsburg, 2015; Jäder et al., 2020; Lester et al., 2016; Lye & Koh, 2014). Indeed, as we now shall see, problem solving has over the years been granted the status of an essential competency in mathematics (Niss & Højgaard, 2002, 2019).

Mathematical problem solving

One of the most prominent ways of looking at the concept of problem solving dates to Polya's (1954) four-step model for solving a problem. In the context of mathematics education, Polya's (1954) contribution and Schoenfeld's (1985) six-step refinement (reading, analysis, exploration, planning, implementation, and verification) are essential in providing stepwise outlines for approaching problem-solving tasks. As problem solving involves taking on tasks that are significantly different from those

learned by heart, a major part of the challenge is in deciding which bits of one's toolkit of mathematical skills will help (Burkhardt & Bell, 2007, p. 395) and if and to what extent more recent technological advances and associated CT can help in the problem-solving process.

In recent decades, with technological advances resulting in computational, modelling and programming tools important to solving mathematical problems (Carreira & Jacinto, 2019; Liljedahl & Cai, 2021), problem solving has been increasingly influenced by technology and digital competency (Geraniou & Jankvist, 2019). These advances have "given rise to a new set of competencies that need to be developed" in individual problem solvers (Liljedahl & Cai, 2021, p. 727) and their teachers (Nordby et al., 2022a).

Acknowledging the affordances and advances stemming from including digital tools in problem solving led Carreira and Jacinto (2019) to bring together two frameworks: the outline of the problem-solving process proposed by Schoenfeld (1985) and a digital literacy framework proposed by Martin and Grudziecki (2006). This resulted in the *Mathematical problem solving with technology* (MPST) framework that addresses the use of digital resources when solving a task or problem (Carreira & Jacinto, 2019).

The MPST framework

As the aim of our review is to better understand CT's role in mathematics problem solving, we present the MPST framework (Carreira & Jacinto, 2019) in table 2, while mapping it onto the five steps in Schoenfeld's (1985) problem-solving process, as Carreira and Jacinto (2019) did. We see the MPST framework as a way of talking about the role of not only digital resources in problem-solving processes, but also the thinking behind it in CT. While students engage in MPST, "learners have the opportunity to expand or enhance not only important problem-solving heuristics [...] but also to construct and incorporate ways of reasoning associated with the use of the tool" (Santos-Trigo, 2019, p. 86) through CT. The role of heuristics has always been central to mathematical problem solving. However, in line with the increasing influence of digital technology in mathematics education, we see in the literature that digital technologies and programming languages have the potential to expand or enhance students' problem-solving strategies.

In this literature review, we seek to better understand the role of CT in mathematics problem solving in the context of primary and secondary mathematics education and thus pose two research questions (RQs):

Table 2. *An adaptation of Carreira and Jacinto's (2019, p. 45) mapping of the work of Schoenfeld (1985) and Martin and Grudziecki (2006).*

Five-step problem-solving model (Schoenfeld, 1985)*		**Mathematical problem solving with technology (Carriera and Jacinto, 2019)****	
Reading	Reading and ingestion of the problem conditions.	Grasp	Appropriation of the situation and the conditions in the problem and early ideas on what it involves.
Analysis	In analysis, an attempt is made to understand a problem entirely, select an appropriate perspective, and reformulate or simplify the problem.	Notice	Initial attempt to comprehend what is at stake, namely the mathematics that may be relevant and the digital tools that may be necessary.
		Interpret	Placing affordances in the technological resources in pondering mathematical ways of approaching the solution.
Exploration	In exploration, one finds a variety of problem-solving heuristics. A search for relevant information can be incorporated into the analysis/planning/ implementation sequence.	Integrate	Combining technological and mathematical resources within an exploratory approach.
		Explore	Using technological and mathematical resources to explore conceptual models that may enable the solution.
Planning and implementation	Planning and implementation deal with questions like if the plan is well-structured and how the implementation is assessed and monitored.	Plan	Outlining an approach to achieve the solution based on the analysis of the conjectures explored.
		Create	Carrying out the outlined approach, recombining resources in new ways which will enable the solution and create new knowledge objects, units of information or other outputs which will contribute to solve-and-express the problem.
Verification	Verification deals with questions regarding if and how the problem's solution is verified, tested or assessed.	Verify	Engaging in activities to explain or justify the solution achieved based on the mathematical and technological resources.
		Disseminate	Present the solutions or outputs to relevant others and consider the success of the problem-solving process.

* The descriptions of the different steps are quoted from Schoenfeld (1985, pp. 297–298).
** The descriptions of the different steps are quoted from Carreira and Jacinto's (2019, p. 45) mapping of Martin and Grudziecki's (2006) process of digital technology problem solving and Schoenfeld's (1985) stages of mathematical problem solving.

1. How and at which stages in the problem-solving process does primary and secondary mathematics research report on the inclusion of CT?

2. To what extent, if any, do the included studies report on a) improved mathematics learning and b) improved mathematics problem-solving skills among students after the inclusion of CT?

This paper is organised as follows. We first present our methodological approach, including search strategies with inclusion and exclusion criteria, and an outline of the steps taken in the analysis. We next present our findings before concluding by discussing the implications of our

findings and offering suggested avenues for future primary and secondary mathematics education and associated research.

Methodology

A systematic review inspired by Shute et al.'s (2017) methodology was applied to provide a comprehensive understanding of how CT can be included in mathematics problem solving, and how mathematics problem solving can gain from the inclusion of CT. In this section, we give an account of our search strategy with two selection stages before we provide a description of data extraction and analyses.

Search strategy

An initial search for research related to CT was conducted to gain an overview of how CT has been addressed. This initial search enabled the identification of several key terms that could help capture the publications of interest for this literature review and served to help the researchers familiarise themselves with the field.

Wing (2006) marked a change from the *constructing* central to Papert's (1980) Mindstorms to *thinking* and a focus on thought processes. Thought processes are central to Wing's approach: CT is described as "a kind of analytical thinking. It shares with mathematical thinking in the general ways in which we might approach solving a problem" (Wing, 2008, p. 3717). After Wing, it became clear that CT was more complex than programming (Lye & Koh, 2014; Rodríguez-Martínez et al., 2019; Wing, 2008) and might even differ from programming skills (Ioannidou et al., 2011; Israel et al., 2015). CT emphasises thought processes, with problem-solving strategies deriving from computer sciences that could benefit educational settings. For this reason, this review begins in 2006, when Wing's (2006) influential view entered the field.

The first papers read were all by Wing (2006, 2008, 2011). Reading her papers and examining related readings increased our understanding of relevant keywords for CT (e.g. *computational literacy*, *computational modelling*, *algorithmic thinking* and *algorithms*). Similarly, reading relevant literature from the field of mathematics education helped to find relevant keywords for mathematics problem solving (e.g. *mathematical thinking*, *problem-solving strategies*, *problem-solving model* and *mathematical competence*). Moreover, because of the focus of our RQs, the search terms *elementary school*, *primary school*, *secondary school* and *high school* were included in searches to capture the target age span of this review.

This first search string was tested in the ERIC and Web of Science databases. A reading of abstracts and keywords in the resulting publications

helped us refine the search string to narrow the search. The final search was conducted in four databases: ERIC, Web of Science, Academic Search Ultimate and EBSCO's Education Source.

Inclusion and exclusion criteria. In selection stage one (see figure 1), the search resulted in 739 publications from the four databases. After duplicate removal ($n = 212$), the remaining 527 papers were subject to a screening of the title, abstract and introduction based on the criteria in table 3.

A total of 495 publications were excluded during this first screening, leaving 32 papers for the second screening, which consisted of full-text reading. The first round of full-text reading resulted in a significant

Table 3. *Inclusion and exclusion criteria*

Type of criterion	Inclusion	Exclusion
Type of publication	Peer-reviewed papers Peer-reviewed conference papers	Reports Dissertations Books/book chapters* Editorials
Publication period	2006 – March 2021	
Type of study	Empirical papers	Literature reviews Theoretical papers**
Language	English Nordic	
Educational level	Primary and secondary education	Pre-school/ kindergarten education Higher education Special education
Setting	School context Summer school***	Out of school setting
Object of study	Students (engaging in problem solving activities) Teachers (orchestrating problem solving activities)	
Key terms in the title or abstract or introduction	Mathematics/Math/Maths Problem solving Computational thinking Programming Algorithmic thinking Computer science Primary/elementary school/education Secondary/high school/education	STEM (science, technology, engineering and mathematics)
Conceptualisation	Computational thinking is defined/ operationalised/ conceptualised	Programming when not being situated in the context of computational thinking

* Books/book chapters were excluded because it is not always easy to determine whether they are peer reviewed.
** Literature reviews and theoretical papers are excluded because of our object of study.
*** Summer schools were included when the students' normal school context was clearly outlined.

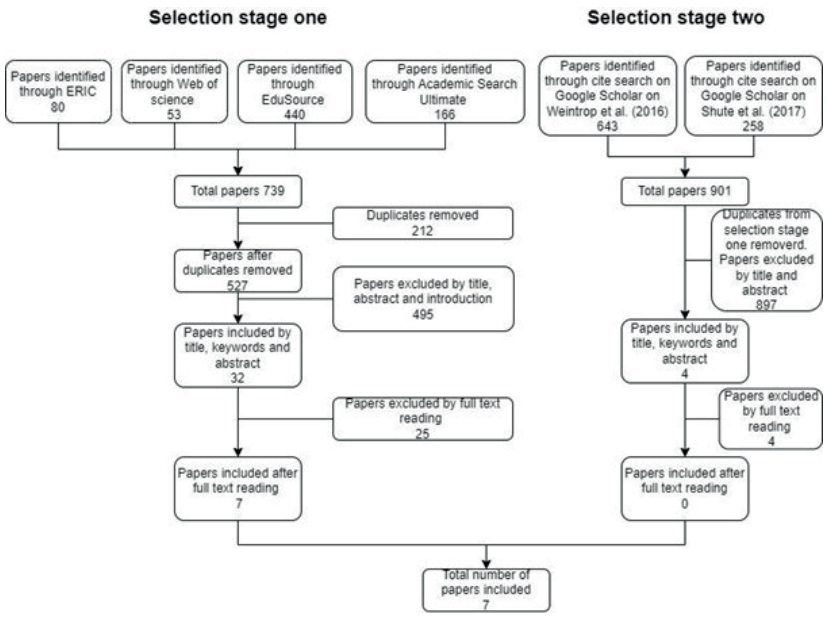

Figure 1. O*verview of the search process*

reduction from 32 to seven papers: this was due to how we used our RQs to exclude papers that were unclear on what they meant by CT; that is, they neither defined nor operationalised CT, focusing instead on programming.

As figure 1 shows, selection stage two involved identifying papers through a Google Scholar search (see Haddaway et al., 2015) that returned all papers citing Weintrop et al. (2016) or Shute et al. (2017). These two texts were chosen because of their abundant citations and their theoretical contributions to the field in the form of a CT taxonomy (Weintrop et al., 2016) and a model showing the links between CT and mathematical thinking (Shute et al., 2017). This stage did not add any publications, so we ultimately had seven papers for this systematic literature review.

Data extraction and analysis

Analysing the seven studies involved several steps. First, a spreadsheet was used to capture important features of each paper[1], such as information about the context, participants, aims, mathematical domain, theoretical frameworks, CT and problem-solving definitions and operationalisations, RQs, methods and results. Next, due to the authors' operationalisation of the review's RQs, columns were added and filled in after a second reading. These first two steps were conducted by both authors

separately leaving us with two spreadsheets consisting exclusively of quotes from the included papers. In the third step, the authors compared and merged their spreadsheets, adding comments and notes in green to record the authors' common understanding of selected quotes. In addition to the features listed above, the spreadsheet now included columns capturing the following aspects: description of design, how reported data are analysed, programming languages and how they are used, what they report on, what the students in a study are assigned to do or engage in, any learning among students reported, main message and the concepts and constructs in relation to how they see CT.

The combined results of these features, with a special focus on the report of any learning among students, enabled us to answer RQ2 on the extent, if any, to which studies report on improved mathematics learning and improved mathematics problem-solving skills among students after the inclusion of CT.

To answer RQ1, on how and at which stages in the problem-solving process primary and secondary mathematics research reports on CT inclusion, a fourth step in our analysis process was taken. The way in which Carreira and Jacinto (2019) described the different processes in solving mathematical problems with technology (see table 2) enabled us to search the included papers in detail to determine when during the problem-solving process CT was included. Once this was established, we used Brennan and Resnick's (2012) operationalisation of CT (table 1) to assess whether inclusion was in the form of CT concepts, practices or perspectives. Table 4 provides an alphabetically ordered overview of the publications included in this literature review, building on relevant features from the spreadsheet.

Results

The seven papers were analysed to identify how CT and problem solving were related in the context of primary and secondary mathematics education research. We organise this section in accordance with our two RQs, the first being concerned with how and at which stages in the problem-solving process primary and secondary mathematics research report on the inclusion of CT, and the second asking to what extent, if any, the included studies report on improved mathematics learning and problem-solving skills among students after the inclusion of CT.

The inclusion of CT in problem-solving processes

To set forth how the seven papers report on the inclusion of CT in different mathematics problem-solving processes, we draw on the analysis

Table 4. *Presentation of included papers*

Study	Participants	Type of study/ duration	Data	Programming language	Aim
Calder (2018)	New Zealand N = 26, one class Ages 10–11	Case study/ 2 weeks	Daily students blogs, student and teacher interviews, classroom observation	Scratch (block-based programming language)	Gain insights into a teacher and her students' perceptions of Scratch, and the ways that mathematical thinking might develop through coding with Scratch.
Cui and Ng (2021)	Hong Kong N = 8 Ages 12–14*	Design based research/ 3 day camp	Video from four angles, screenshots of program codes	Block-based programming language with physical objects (Arduino)	Document the challenges that arise within the context of problem solving in CT-based environments.
Kaufmann and Stenseth (2021)	Norway N = 3 Age 13	Instrumental case study/ 1 week	Video capturing conversations and the screen	Processing (a text-based programming language)	Investigate how pupils in lower secondary apply programming when working with a mathematical problem.
Ng and Cui (2021)	Hong Kong N = 8 Ages 11-13*	Design based research/ 3 day camp	Video from four angles, screenshots of program codes	Block-based programming language with physical objects (Arduino)	Utilize mathematical problems for which CT concepts and computational problem-solving practices were necessary and useful for finding the solutions.
Pei et al. (2018)	USA N = 120 Age 15	Experimental/ 6 school days	Video of two classes, 14 pairs of one-to-one pre-/post-interviews	Lattice land (a computational tool)	Argue that CT practices and mathematical habits of mind are distinct, yet mutually supportive constructs that have a home in contemporary mathematics classrooms.
Psycharis and Kallia (2017)	Greece N = 66 Ages 17–18	Quasi-Experimental/ 3 hour course intervention	Pre/post test data	Pascal (text-based programming language)	Investigate the impact/effect of computer programming curriculum intervention on reasoning skills, problem solving skills and self-efficacy in mathematics.
Rodríguez-Martínez et al. (2019)	Spain N = 47 Age 11	Quasi-experimental	Pre/post test data	Scratch (block-based programming language)	Study the effect of programming activities on both the development of students' computational thinking and mathematical learning, and to evaluate the potential of programming activities to foster students' acquisition of mathematical ideas and CT.

*= Cui and Ng (2020) and Ng and Cui (2020) report from the same design based research projects, but give different age spans for the participants.

conducted in stage four of the process described above. The way in which Carreira and Jacinto (2019) describe the different processes in solving mathematical problems with technology (see table 2), and how the lens of Brennan and Resnick (2012) enabled us to see CT inclusion as involving CT concepts, practices, or perspectives (see table 1) helped us better understand how CT's inclusion is reported in mathematics problem-solving processes. Table 5 gives a schematic overview of our findings.

Five of the seven papers had a thorough description of the mathematics problems assigned to different groups of students. Because of their research design, those in the last two columns provided no detailed information about problem-solving processes during interventions and are for that reason less present in the next sections where we take a closer look at how the different papers report on use of CT in each of the steps given in column two in table 5.

At the early stage of *grasping* the problem at hand, which involves ingesting the problem (Schoenfeld, 1985), an appropriation of the problems was necessary to make it fit with the predetermined digital tool or programming environment at hand. At this stage, those five studies providing descriptions of the mathematics problem at hand report on how students include CT in terms of different CT practices. Most often, we see how the chosen programming language and mathematical concepts together form the problem-solving task assigned. For instance, in Cui and Ng (2021) and Ng and Cui (2021), who report on the same study, the

Table 5. *Inclusion of CT in different problem-solving processes*

Schoenfeld's (1985) Problem-solving model	Carreira and Jacinto's (2019) Mathematical problem solving with technology model	Calder (2018)	Cui and Ng (2021)	Kaufmann and Stenseth (2021)	Ng and Cui (2021)	Pei et al. (2018)	Psycharis and Kallia (2017)	Rodríguez-Martínez et al. (2019)
Reading	Grasp	Practise	Practise	Practise	Practise	Practise		
Analysis	Notice							
	Interpret		Practise		Practise	Practise		
Exploration	Integrate	Concepts	Concepts	Concepts	Concepts		Concepts	Concepts
	Explore					Practise		
Planning/ implementation	Plan and Create*	Practise	Concepts	Practise	Concepts	Practise		
Verification	Verify	Practise	Practise	Practise	Practise	Practise		
	Disseminate	Perspective						

* Because it was hard to separate Plan and Create when analysing other people's research, we have followed Schoenfeld (1985) who combines them.

students were asked to develop a program for a physical object (Arduino) to present a green light for prime numbers and a red light for composite numbers. Here, the students needed to appropriate the mathematical problem entirely and convert it so it could be solved in the given block-based programming language, using CT practices like translating and decomposing (Cui & Ng, 2021). In Kaufmann and Stenseth (2021), the students were given a runnable code in the Processing programming language to make a wheel rotate. The students had to understand both the program code and the mathematics involved to change the spinning of the wheel, which was their task. Kaufmann and Stenseth (2021) report on students' engaging in troubleshooting as a CT practice (Brennan & Resnick, 2012). Calder (2018) reports on students developing a game in Scratch to facilitate year 1 students' understanding of numbers, which happens through an "iteration" of action and reflection practice, whereas Pei et al. (2018) report on a study that used the digital environment Lattice Land, in which the students engaged in CT practices when breaking the polygons into a series of sub problems. Hence, we see an extensive use of different CT practice at this stage.

The next step in Carreira and Jacinto's (2019) outline of Schoenfeld's (1985) problem-solving process is *notice*. At this stage, Carreira and Jacinto (2019) propose the need for an initial attempt to comprehend the mathematics the students find relevant and the digital tools that may be needed. As revealed in table 5, this step did not apply in any of the papers since the digital tools or environment were chosen in advance (by teachers or researchers) to fit the problem at hand.

Following how *interpreting* is described by Carreira and Jacinto (2019), we found that, when students accounted for the affordances in the digital tool or environment they were engaged with, they decomposed or reframed the problems. For instance, in Pei et al.'s (2018) study, the students decompose a polygon problem to be able to solve it within the possibilities embedded in Lattice Land, which is exactly what Carreira and Jacinto (2019) mean by the term interpreting (see table 2). Similarly, in Cui and Ng (2021) and Ng and Cui (2021), the students had to prepare mathematics problems to solve them in the block-based programming environment; in some cases, the students applied the same set of codes to new problems, which included the CT practices remixing and reusing (Ng & Cui, 2021). In this way, we see examples of CT practices at this stage as well.

Schoenfeld's (1985) exploration is divided into *integrating*, that is "combining technological and mathematical resources within an exploratory approach" and *exploring*, which means "using technological and mathematical resources to explore conceptual models that may enable the

solution" (Carreira & Jacinto, 2019, p. 45). We see that most studies report on integrating rather than exploring, even those in the last two columns in table 5: In both Psycharis and Kallia (2017) and Rodríguez-Martínez et al. (2019), students were taught how to explore and solve mathematical problems within a programming language or environment. Psycharis and Kallia (2017) report that, during the intervention, the students in the experimental group were taught how to develop a source code. The students' solutions were generalised so they could solve similar problems. This was even more apparent in Rodríguez-Martínez et al. (2019), where the students had a programming intervention in which they were acquainted with several CT concepts necessary to engage with Scratch (e.g. sequence, iteration or loop, event handling, conditionals). In papers of a more qualitative nature, like Calder (2018), the activity in Scratch helped students understand the link between the numerical value of an angle and the size of the corresponding visualised angle or turn. In Cui and Ng's (2021) and Ng and Cui's (2021) studies, the students explored the assigned problems while using different computational concepts, such as sequence, repetition or loops, events, and conditionals (Ng & Cui, 2021). While Kaufmann and Stenseth's (2021) contribution differs from the others in the way the students were given a piece of programming code to change, the students turned to the CT concept variable while exploring the code, which turned out to be especially important since this concept has different meanings in CT and in mathematics. Further, in Pei et al. (2018), we clearly see that the students' use of the dynamic and interactive characteristics of Lattice Land enabled them to explore the relationship between the shape of a triangle and the resulting area by moving points to change the triangle's shape while keeping its area fixed. In crafting a mathematical formula, the students used the CT practice of creating an abstraction. The other included papers did not report on any dynamic use of programming languages, but those who did, mainly report on the use of CT concepts at this stage, and to some extent, CT practices. As we shall see, this was also the case in the next step.

While Carreira and Jacinto (2019) describe the steps of *planning* and *creating* separately (see table 2), conducting a meta-analysis of other peoples' research made it difficult to distinguish between the two, and we have thus merged them. In most cases, we found planning and creating to be highly important – if not central – to using CT in mathematical problem-solving processes. In several papers, this was the explicit aim: the students were to develop a solution to a mathematical problem in a given programming language or digital environment (Calder, 2018; Cui & Ng, 2021; Ng & Cui, 2021; Pei et al., 2018). This made them engage in several CT concepts and practices. For example, the students in Calder (2018)

combined Scratch with geometric concepts while developing and modifying a program on numbers through iterations of action and reflection. In Pei et al. (2018), the students combined Lattice Land with geometry to develop new strategies and create an abstraction for solving polygon problems, while Ng and Cui (2021) used a block-based programming language with a physical tool with numbers in mathematics, reporting on how students learned, for example, how to use CT concepts like variables, events and conditions. Since the students in Kaufmann and Stenseth (2021) had a pre-produced programming code, they were engaged in troubleshooting as an iterative process, which was a slightly different approach to planning and creating.

In the *verifying* step, students across the included studies appeared to draw on different CT practices (such as debugging and troubleshooting) when justifying their solutions. For example, in Kaufmann and Stenseth (2021), Cui and Ng (2021) and Ng and Cui (2021), the students had to check, analyse, and modify their solutions. In Pei et al. (2018), verification occurred when students switched between formulating the relationship between variables and testing unique polygons. Overall, at this step, we found that students engaged in CT practices when explaining or justifying their solution to mathematical problems.

The final step of *disseminating* is about how the problem solvers presented their solutions to relevant others, a step less reported on in the included studies. However, Calder (2018) reported on students engaging in an "iteration" of action and reflection, which led students to modifying their solutions after having received feedback from the intended users of the block-based game they produced. Here, for the first time, we see how different CT perspectives come into play when the students presented and explained their program to their teachers and their year 1 peers.

Having seen how the papers in this review report on the inclusion of CT concepts and practices, and to a lesser extent CT perspectives in mathematical problem-solving processes, we now turn to our second research question.

Improved mathematics learning and problem-solving skills

While each included study reported on students' engaging in problem-solving tasks, the nature of the different research designs employed was found to be decisive for how they reported on a) improved mathematics learning and b) improved mathematics problem-solving skills in students after CT's inclusion. The crucial distinction is between those with an experimental design and associated hypothesis testing (Psycharis & Kallia, 2017; Rodríguez-Martínez et al., 2019) and those taking a more

qualitative perspective in the form of a case study (Calder, 2018; Kaufmann & Stenseth, 2021; Pei et al., 2018) or design-based research approach (Cui & Ng, 2021; Ng & Cui, 2021). We start by looking at the former.

Psycharis and Kallia's (2017) and Rodríguez-Martínez et al.'s (2019) pre- and post-test designs enabled them to propose different hypotheses on how their experimental groups (using Pascal and Scratch, respectively; see table 4) and control groups (not using any programming tools) deve-loped differently. When testing the effect of a computer programming curriculum intervention on problem-solving skills in mathematics, Psycharis and Kallia (2017) found a statistically significant improvement in the performance of students in the experimental group. However, they noticed that students of both groups were involved in only some steps of the problem-solving process and that those steps were insufficient to reveal significant differences in the students' problem-solving skills between groups. The authors thus rejected the hypothesis that computer programming would automatically improve students' general problem-solving ability.

Rodríguez-Martínez et al. (2019) studied how Scratch can provide an opportunity to study mathematical concepts, such as the least common multiple (LCM) and greatest common divisor (GCD) in grade 6 students. The main objective was to analyse the potential of programming activities using Scratch for both the learning of mathematical ideas and the acquisition of CT. Their two-phased intervention revealed how the first phase, which involved specific instruction in computational concepts, led to a significant increase in students' CT levels. In the second phase, they found a relevant improvement in students' proficiency in solving word problems related to LCM and GCF in the experimental group (using Scratch); however, the improvement was not significant in comparison with the control group, who used a more traditional approach.

Turning to those studies using a case study approach, we found a more thorough outline of the activities the students were assigned. In Calder (2018), the students were challenged to design and build a mathematics game suitable for facilitating the number understanding of younger students; in Kaufmann and Stenseth (2021) the students were to upload and run a code and then revise it to meet some predetermined properties; and in Pei et al. (2018), the problem was to explore the shapes and sizes of lattice triangles to discover (or rediscover) some unexpected results when calculating the areas of increasingly more complex and less regular polygons. While none of these studies used any CT operationalisations to analyse or discuss their data, their focus on the problem-solving process and the close way in which they synthesise CT with mathematical thinking (Calder, 2018), mathematical habits of mind (Pei et al., 2018) and

mathematics as such (Kaufmann & Stenseth, 2021) enabled the various authors to report some tentative findings.

In Calder (2018), when developing mathematical games in Scratch, the students drew on geometric concepts like angle size and measurements. While Calder (2018) indicated that the students' understanding of coordinates and their relative position on a Cartesian plane seemed to be enhanced through this activity, there was less certainty about the extent to which new mathematical learning occurred during this process. Pei et al. (2018) investigated how a computational tool (Lattice Land) explores geometrical lattice ideas. When looking at one interview in detail (which they later compared with other interviews that supported their findings), they found that the student had gained strategies, after engaging in Lattice Land, that enabled her to calculate the area of a complex polygon, using a new boxing approach (which essentially meant detecting triangles in a complex polygon which she boxed in a rectangle whose area she was able to calculate). Kaufmann and Stenseth (2021) showed how students engaged in a programming task involving a circle made possible the integration of programming into mathematics education.

The last two papers – Cui and Ng (2021) and Ng and Cui (2021) – reported on the same design-based research study in which the students faced three problems: to construct a thermometer and a room capacity detector, to create a machine that calculates the bank deposit and balance for a designated scenario and to make a prime and composite number detector. While Cui and Ng (2021) reported on challenges students encountered while preparing tasks, programming, preparing computational abstractions and troubleshooting and debugging, Ng and Cui (2021) focused more on how students engaged in mathematical thought processes. Neither study reported any gained learning outcomes among students, but Ng and Cui (2021) did indicate that block-based programming encouraged students to deal with arithmetic sequences and geometric sequences and series without using any algebra. Based on this, they suggested that the introduction of algebraic thinking and representations in the primary grades can be achieved through CT concepts like variables and iterations.

While acknowledging that none of the included case studies and design-based research projects set out to measure changes in mathematics learning and problem-solving skills in individuals, we find that the studies do report on how including CT made the students more engaged and motivated to use mathematical ideas in their problem-solving process (Calder, 2018; Kaufmann & Stenseth, 2021) and how they "showed greater willingness to explore and share interesting findings" and discovered new methods for approaching unknown problems (Pei et al., 2018, p. 87). In

addition, the studies reported on improvements in students' mathematical thinking (Cui & Ng, 2021), creativity (Pei et al., 2018), argumentation (Kaufmann & Stenseth, 2021) and communication and collaboration skills (Calder, 2018).

Discussion and concluding remarks

Based on our close reading and careful analysis of seven studies, we assert that, together, the studies show that it is possible to include the CT concepts, practices and perspectives from Brennan and Resnick (2012) in different mathematics problem-solving processes (RQ1) and that students experienced some gains following this inclusion (RQ2). This supports Wing's (2006) initial suggestion – that CT and mathematical thinking share important features in how we might approach problems – and strengthens arguments for implementing CT through problem solving in mathematics education.

One conspicuous commonality across the included papers was assigning students a mathematical problem *and* a predetermined programming language or digital environment *in* which (Cui & Ng, 2021; Kaufmann & Stenseth, 2021; Ng & Cui, 2021) or *with* which to solve that problem (Calder, 2018; Pei et al., 2018; Psycharis & Kallia, 2017; Rodríguez-Martínez et al., 2019). For instance, in Kaufmann and Stenseth (2021), the students were told to work in the text-based development tool Processing to fine-tune a code, while the students in Calder (2018) were given the problem of creating a game for younger students using the block-based programming tool Scratch. This means, as table 5 reveals, that none of the included papers reports on students being challenged to find the relevant programming tool. This is not to suggest a shortcoming in the research literature – it is rather to raise an important issue for future mathematics teaching with CT, in line with the rare occurrence of the full integration of CT in mathematics education found in Nordby et al.'s (2022a) review. As an illustration of what we mean, consider how students, on their own initiative, use calculators, spreadsheets like Excel and dynamic software like GeoGebra as tools in their mathematics problem solving because they know that these devices can help them solve problems and spare them substantial amounts of work.

The introduction of CT into mathematics is still recent (Barr & Stephenson, 2011; Grover & Pea, 2013; Kallia et al., 2021), and teachers have not been trained in the programmes and tools available for bringing CT into mathematics classrooms (Nordby et al., 2022b); thus, the comparison with the foothold that calculators, spreadsheets and dynamic software have established in mathematics education may be unfair. Nevertheless,

we suggest that the comparison could be fruitful: as technology uses new approaches in mathematical problem solving (Carreira & Jacinto, 2019; Santos-Trigo, 2019), students will need to become "aware of which digital tools to apply within the different mathematical situations and context" and be "able to use digital technology reflectively in problem-solving and when learning mathematics" as part of their digital competency (Geraniou & Jankvist, 2019, p. 43). This is a goal worth pursuing in the future.

We end this paper by returning to table 5. There we see that the experimental studies (Psycharis & Kallia, 2017; Rodríguez-Martínez et al., 2019) report less on how CT was included in their studies. At the same time, we know from our RQ2 that it is precisely these studies that report measures of students' outcomes. The combined results from RQ1 and RQ2 lead us to propose more research using mixed methods as a way both to capture the details of how CT is included in problem-solving processes and to measure the outcomes of that inclusion. Adding to this, we would also like to draw attention to the challenges students encountered in, for instance, Cui and Ng's (2021) study, in preparing tasks (they encountered difficulties in reframing the problem), programming (students' misconception of CT concepts), creating computational abstractions (they had problems dealing with multiple variables and distinguishing them from how variables are used in mathematics) and CT practices such as troubleshooting and debugging. These challenges need to be addressed in future research initiatives, in the training of teachers and in planning the future implementation of CT in mathematics classrooms, including outside the problem-solving context.

References

Barr, V. & Stephenson, C. (2011). Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2 (1), 48–54. doi: 10.1145/1929887.1929905

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. & Engelhardt, K. (2016). *Developing computational thinking in compulsory education*. European Commission. https://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Nordic@BETT2018 Steering Group. doi: 10.17471/54007

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at AERA2012, Vancouver, Canada. https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Burkhardt, H. & Bell, A. (2007). Problem solving in the United Kingdom. ZDM, 39 (5), 395–403. doi: 10.1007/s11858-007-0041-4

Cai, J., Wang, N., Moyer, J. C., Wang, C. & Nie, B. (2011). Longitudinal investigation of the curricular effect: an analysis of student learning outcomes from the LieCal Project in the United States. *International Journal of Educational Research*, 50 (2), 117–136. doi: 10.1016/j.ijer.2011.06.006

Calder, N. (2018). Using Scratch to facilitate mathematical thinking. *Waikato Journal of Education*, 23 (2), 43–58. doi: 10.15663/wje.v23i2.654

Carreira, S. & Jacinto, H. (2019). A model of mathematical problem solving with technology: the case of Marco solving-and-expressing two geometry problems. In P. Liljedahl & M. Santos-Trigo (Eds.), *Mathematical problem solving: current themes, trends, and research* (pp. 41–62). Springer Nature.

Cui, Z. & Ng, O.-L. (2021). The interplay between mathematical and computational thinking in primary school students' mathematical problem-solving within a programming environment. *Journal of Educational Computing Research*, 59 (5), 988–1012. doi: 10.1177/0735633120979930

Directorate of Education. (2020). *Nye læreplaner – grunnskolen*. https://www.udir.no/laring-og-trivsel/lareplanverket/Nye-lareplaner-i-grunnskolen-og-gjennomgaende-fag-vgo/

English, L. D. & Gainsburg, J. (2015). Problem solving in a 21st-century mathematics curriculum. In L. D. English & D. Kirshner (Eds.), *Handbook of international research in mathematics education* (3rd ed., pp. 313–335). Routledge.

Geraniou, E. & Jankvist, U. T. (2019). Towards a definition of "mathematical digital competency". *Educational Studies in Mathematics*, 102 (1), 29–45. doi: 10.1007/s10649-019-09893-8

Green, W. H. & Gilhooly, K. (2005). Problem solving. In N. Braisby & A. Gellatly (Eds.), *Cognitive psychology* (2nd ed., pp. 301–333). Oxford University Press.

Grover, S. & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher*, 42 (1), 38–43. doi: 10.3102/0013189X12463051

Gueudet, G., Bueno-Ravel, L., Modeste, S. & Trouche, L. (2017). Curriculum in France: a national frame in transition. In D. R. Thompson, M. A. Huntley & C. Suurtamm (Eds.), *International perspectives on mathematics curriculum* (pp. 41–70). International Age. https://hal.archives-ouvertes.fr/hal-01599059

Haddaway, N. R., Collins, A. M., Coughlin, D. & Kirk, S. (2015). The role of Google scholar in evidence reviews and its applicability to grey literature searching. *PLoS ONE*, 10 (9). doi: 10.1371/journal.pone.0138237

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M. & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, 41 (1), 183–187. doi: 10.1145/1539024.1508931

Heintz, F., Mannila, L.-Å., Nordén, L.-Å., Parnes, P. & Regnell, B. (2017). Introducing programming and digital competence in Swedish K–9 education. In V. Dagiené & A. Hellas (Eds.), *Informatics in schools: focus on learning programming* (pp. 117–128). Springer. doi: 10.1007/978-3-319-71483-7_10

Hemmi, K., Krzywacki, H. & Partanen, A.-M. (2017). Mathematics curriculum: the case of Finland. In D. R. Thompson, M. A. Huntley & C. Suurtamm (Eds.), *International perspectives on mathematics curriculum* (pp. 71–102). Information Age.

Hsu, T.-C., Chang, S.-C. & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: suggestions based on a review of the literature. *Computers & Education*, 126, 296–310. doi: 10.1016/j.compedu.2018.07.004

Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H. & Basawapatna, A. (2011). *Computational thinking patterns*. Paper presented at AERA 2011, New Orleans, United States. https://files.eric.ed.gov/fulltext/ED520742.pdf

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M. & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82, 263–279. doi: 10.1016/j.compedu.2014.11.022

Jäder, J., Lithner, J. & Sidenvall, J. (2020). Mathematical problem solving in textbooks from twelve countries. *International Journal of Mathematical Education in Science and Technology*, 51 (7), 1120–1136. doi: 10.1080/0020739X.2019.1656826

Kallia, M., Borkulo, S. P. van, Drijvers, P., Barendsen, E. & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education*, 23 (2), 159–187. doi: 10.1080/14794802.2020.1852104

Kaufmann, O. T. & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, 52 (7), 1029–1048. doi: 10.1080/0020739X.2020.1736349

Kurland, D. M. & Pea, R. D. (1985). Children's mental models of recursive Logo programs. *Journal of Educational Computing Research*, 1 (2), 235–243. doi: 10.2190/JV9Y-5PD0-MX22-9J4Y

Kurland, D. M., Pea, R. D., Clement, C. & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, 2 (4), 429–458. doi: 10.2190/BKML-B1QV-KDN4-8ULH

Lester, F. K. & Cai, J. (2016). Can mathematical problem solving be taught? Preliminary answers from 30 years of research. In P. Felmer, E. Pehkonen & J. Kilpatrick (Eds.), *Posing and solving mathematical problems: advances and new perspectives* (pp. 117–135). Springer. doi: 10.1007/978-3-319-28023-3_8

Liljedahl, P. & Cai, J. (2021). Empirical research on problem solving and problem posing: a look at the state of the art. *ZDM*, 53 (4), 723–735. doi: 10.1007/s11858-021-01291-w

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. doi: 10.1016/j.chb.2014.09.012

Martin, A. & Grudziecki, J. (2006). DigEuLit: concepts and tools for digital literacy development. *Innovation in Teaching and Learning in Information and Computer Sciences*, 5 (4), 249–267. doi; 10.11120/ital.2006.05040249

Ng, O.-L. & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: towards computationally enhanced mathematics education. *ZDM*, 53(4), 847–860. doi: 10.1007/s11858-020-01200-7

Niss, M. & Højgaard, T. (2002). *Kompetencer og matematiklæring: ideer og inspiration til udvikling af matematikundervisning i Danmark*. Undervisningsministeriets forlag. http://static.uvm.dk/publikationer/2002/kom/hel.pdf

Niss, M. & Højgaard, T. (2019). Mathematical competencies revisited. *Educational Studies in Mathematics*, 102 (1), 9–28. doi: 10.1007/s10649-019-09903-9

Nordby, S. K., Bjerke, A. H. & Mifsud, L. (2022a). Computational thinking in the primary mathematics classroom: a systematic review. *Digital Experiences in Mathematics Education*, 8, 27–49. doi: 10.1007/s40751-022-00102-5

Nordby, S. K., Bjerke, A. H. & Mifsud, L. (2022b). Primary mathematics teachers' understanding of computational thinking. *KI - Künstliche Intelligenz*, 36, 35–46. doi: 10.1007/s13218-021-00750-6

Noss, R. & Hoyles, C. (1996). *Windows on mathematical meanings: learning cultures and computers* (vol. 17). Springer. doi: 10.1007/978-94-009-1696-8

Papert, S. (1980). *Mindstorms*. Harvester Press.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1 (1), 95–123. doi: 10.1007/BF00191473

Pei, C., Weintrop, D. & Wilensky, U. (2018). Cultivating computational thinking practices and mathematical habits of mind in Lattice land. *Mathematical Thinking and Learning*, 20 (1), 75–89. doi: 10.1080/10986065.2018.1403543

Pérez, A. (2018). A framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education*, 49 (4), 424–461. doi: 10.5951/jresematheduc.49.4.0424

Polya, G. (1954). *How to solve it: a new aspect of mathematical method*. Doubleday Anchor Books.

Psycharis, S. & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science*, 45 (5), 583–602. doi: 10.1007/s11251-017-9421-5

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E.et al. (2009). Scratch: programming for all. *Communications of the ACM*, 52 (11), 60–67. https://dl.acm.org/doi/10.1145/1592761.1592779

Rodríguez-Martínez, J. A., González-Calero, J. A. & Sáez-López, J. M. (2019). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, 28 (3), 316–327. doi: 10.1080/10494820.2019.1612448

Santos-Trigo, M. (2019). Mathematical problem solving and the use of digital technologies. In P. Liljedahl & M. Santos-Trigo (Eds.), *Mathematical problem solving: current themes, trends, and research* (pp. 63–89). Springer. doi: 10.1007/978-3-030-10472-6_4

Schoenfeld, A. H. (1985). *Mathematical problem solving.* Academic Press.

Selby, C. & Woollard, J. (2013). *Computational thinking: the developing definition*. University of Southampton. https://eprints.soton.ac.uk/356481/

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. doi: 10.1016/j.edurev.2017.09.003

Sneider, C., Stephenson, C., Schafer, B. & Flick, L. (2014). Exploring the science framework and NGSS: computational thinking in the science classroom. *Science Scope*, 38 (3), 10–15. doi: 10.2505/4/ss14_038_03_10

Voogt, J., Fisser, P., Good, J., Mishra, P. & Yadav, A. (2015). Computational thinking in compulsory education: towards an agenda for research and practice. *Education and Information Technologies*, 20 (4), 715–728. doi: 10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K. et al. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25 (1), 127–147. doi: 10.1007/s10956-015-9581-5

Wing, J. (2006). Computational thinking. *Communications of the ACM*, 49 (3), 33–35. doi: 10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London: a Mathematical, Physical and Engineering Sciences*, 366, 3717–3725. doi: 10.1098/rsta.2008.0118

Wing, J. (2011). *Research notebook: computational thinking – What and why?* Carnegie Mellon University. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Zhang, L. & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K–9. *Computers & Education*, 141, 1–25. doi: 10.1016/j.compedu.2019.103607

Note

1 Because we had only seven studies, the authors were especially mindful of each included study's list of references. All references were examined, but no new papers meeting our inclusion criteria were found.

Kim André Stavenæs Refvik

Kim André Stavenæs Refvik is PhD. fellow at Volda University College. His research interest focuses on digital tools and technology in mathematics education and, more specifically, how students understand and use programming and computational thinking in mathematics. He is also engaged in teacher education and in-service teacher education on programming and mathematics.

kim.andre.stavenaes.refvik@hivolda.no

Annette Hessen Bjerke

Annette Hessen Bjerke is associate professor of mathematics education at Oslo Metropolitan University, Norway. Her research interests focus on mathematics teacher education, with specific emphasis on pre-service teachers' theory-practice transitions, and teachers' developing self-efficacy and subject matter knowledge. She is engaged in the introduction of computational thinking in mathematics across grades.

annette.hessen@oslomet.no