Teaching algorithms in upper secondary education: a study of teachers’ pedagogical content knowledge

Jacqueline Nijenhuis-Voogt\textsuperscript{a}, Durdane Bayram-Jacobs\textsuperscript{b}, Paulien C. Meijer\textsuperscript{c} and Erik Barendsen\textsuperscript{a,d}

\textsuperscript{a}Institute for Science Education, Radboud University, Nijmegen, The Netherlands; \textsuperscript{b}Eindhoven School of Education, Eindhoven University of Technology, Eindhoven, The Netherlands; \textsuperscript{c}Radboud Teachers Academy, Radboud University, Nijmegen, The Netherlands; \textsuperscript{d}Department of Computer Science, Open University, Heerlen, The Netherlands

ABSTRACT

Background and Context: Computing education is expanding, while the teaching of algorithms is less well studied.

Objective: The aim of this study was to examine teachers’ pedagogical content knowledge (PCK) for teaching algorithms.

Method: We conducted semi-structured interviews with seven computer science (CS) teachers in upper secondary education (students aged 15–18). The data were analyzed qualitatively.

Findings: We found two patterns of variation in teachers’ PCK. First, we detected variation in the teachers’ goals related to their view of algorithms: they either focused on “thinking” about the algorithm as an object, or focused on “thinking and making”, where the algorithm is also regarded as a program. Second, we found variation in teachers’ knowledge about responding to differences between students, which may be generic or topic-specific. Furthermore, our findings reveal that teachers consider class discussions to play a significant role as an instructional method for provoking reflection.

Implications: Our findings regarding PCK may be beneficial for the development of teacher education and professionalization activities for CS teachers.

Introduction

An algorithm, which is a fundamental concept in computer science (CS), can be described as a set of computational steps needed to transform an input into an output (Cormen et al., 2009). Using a programming language, an algorithm can be implemented in code to be executed by a computer. Programming languages “come and go” and are prone to developments and trends, but algorithms are foundational (Schwill, 1994).

Gal-Ezer et al. (1995) highlighted the lasting value of algorithms in the fast-changing world of computer technology and introduced a curriculum for CS in secondary education centered on algorithms. Several studies have underlined the importance of teaching algorithms in secondary education (e.g. Dagienė & Jevisikova, 2012; Yadav et al., 2018), both because algorithms are fundamental concepts (Hromkovič & Lacher, 2017) and...
because algorithm development and understanding improves students’ problem solving and communication competences (Lessner, 2013).

In various countries, algorithms are incorporated into the curriculum because they are seen as an essential topic in CS education (e.g. Computing at School Working Group, 2012; CSTA, 2017). Likewise, in the new CS curriculum in the Netherlands, launched in 2019, the topic of algorithms is much more prominent than it was previously (Barendsen et al., 2016). It is presumed that the focus in CS education should not be on students mastering a specific programming language but on the learning of fundamental concepts (such as algorithms) and problem-solving skills (Solomon, 2007).

Consequently, it is interesting to examine how to teach this topic, exploring teachers’ practices and their underlying reasoning regarding teaching algorithms. To this end, Sentance and Csizmadia (2017) investigated teacher perspectives on computing education in their search for effective pedagogies for teaching particular aspects of computer science, such as algorithms. The results of their study highlighted several successful teaching approaches, but also described the challenges that teachers face, including the challenges of “students willingness or ability to problem solve” (Sentance & Csizmadia, 2017, p. 479). Moreover, Yadav, Gretter et al. (2016) examined CS teachers’ experiences, reporting that it can be challenging for teachers to address each student’s needs because of the uniqueness of each individual’s problem-solving approach.

Hence, teachers face the challenge of teaching algorithms to students with diverse abilities and approaches. This highlights the need for sufficient education and adequate professional development to support both in-service and pre-service teachers responding to this challenge. Computing education is expanding, requiring the training of many new teachers (Leyzberg & Moretti, 2017; Yadav & Berges, 2019). To enrich teacher education, it is common and effective to examine the knowledge of experienced teachers and the broad range of variation therein (Verloop et al., 2001).

We are therefore interested in the practices and knowledge of experienced teachers when teaching algorithms, specifically geared towards addressing different student abilities and approaches to learning to solve problems. Teachers need a specific type of knowledge to teach algorithms effectively, which is different from content knowledge or general pedagogy. Shulman (1987) described this knowledge as pedagogical content knowledge (PCK), which “represents the blending of content and pedagogy into an understanding of how particular topics, problems, or issues are organized, represented, and adapted to the diverse interests and abilities of learners, and presented for instruction” (Shulman, 1987, p. 8).

PCK concerns fundamental teacher knowledge, and many researchers have examined the knowledge of, among others, language teachers (e.g. Liu, 2013; Meijer et al., 1999) and science teachers (e.g. Alonzo & Kim, 2016; Bayram-Jacobs et al., 2019; Henze et al., 2008). For PCK in CS education, Hubbard (2018) reviewed the existing research literature and signified a need for further research into the aspects of PCK unique to the discipline of computing. Prior PCK research into CS education examined topics such as programming (Barendsen et al., 2014; Saeli et al., 2011), the design of digital artifacts (Rahimi et al., 2016), and modeling (Grgrurina et al., 2017); however, there is little scientific understanding of teachers’ PCK for teaching fundamental concepts such as algorithms.

Because this topic is underexposed in educational research, we conducted a small-scale, in-depth study among experienced teachers, with the purpose to identify possible
types of variation and potential patterns within teachers’ PCK. Such results can be used for preparing subsequent PCK research on a broader and larger population, as well as for informing the design of professional development activities.

We interviewed seven CS teachers with experience in teaching algorithms. This paper begins with a conceptual framework on teaching algorithms and PCK. We will then go on to formulate the research questions and describe the method used. This is followed by a description of our findings and a discussion of the results. Finally, the conclusions are presented with recommendations for implications.

**Conceptual framework**

**Teaching algorithms**

Much research in recent years has highlighted the importance of teaching algorithms; for example, in their review of curriculum developments, Webb et al. (2017) found that there is consensus that algorithms should be incorporated as a key concept in CS curricula. Before examining the knowledge of teachers (Fenstermacher, 1994) regarding teaching algorithms, it is worth investigating the knowledge for teachers resulting from previous research, in particular the suggested challenges and ongoing debates regarding teaching algorithms.

The concept of algorithms was examined by e.g. Haberman et al. (2005), who stated that the solution of an algorithmic problem consists of both a well-designed algorithm and a description of the problem analysis and the characteristics of the solution, such as correctness and efficiency. These two aspects, construction and analysis, are included in the definition of “algorithmic thinking” given by Futschek (2006), who characterized algorithmic thinking as the abilities connected to constructing and understanding algorithms. These abilities include analyzing given problems, specifying a problem precisely, finding the basic actions that are adequate to the given problem, constructing a correct algorithm, thinking about all possible special and normal cases of a problem, and improving the efficiency of an algorithm (Futschek, 2006, p. 160). Algorithmic thinking is recognized as one of the elements of computational thinking (CT). Aho (2011) defined CT as “thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms”. Selby and Woollard (2013) identified several abilities related to CT, including abstraction, generalization, decomposition, and algorithmic thinking.

In a study of students’ understanding, Perrenet et al. (2005) defined four abstraction levels for the algorithmic concept. Level 1 is the execution level, where an algorithm is viewed as “a specific run on a specific machine”; the execution time depends on the machine. Level 2 is the program level, where an algorithm is seen as a process, described by a specific executable program written in specific programming language; the execution time is determined by the input. Level 3 is the object level, where an algorithm is viewed as an object that is not linked to a specific programming language; the execution time is considered in terms of “magnitude of order as function of the input”. Level 4 is the problem level, where an algorithm is seen as a black box; the problem leads to thinking about what type of algorithm would solve the given problem. At this level, “problems can be categorized to suitable algorithms; a problem has an intrinsic complexity” (Perrenet et al., 2005, p. 65).
This hierarchy of abstraction levels to consider an algorithm works on two levels, as Statter and Armoni (2020) described in their study on teaching abstraction in CS to 7th grade students (ages 13–14). On the one hand, this hierarchy can be used to assess students’ understanding of the concept of an algorithm. On the other hand, it “represents the practice of classical CS abstraction” (Statter & Armoni, 2020, pp. 8:7). Levels 1, 2, and 3 are different modes of the representation of an algorithm, where level 1 is the most specific and level 3 the least specific. Level 4 describes the problem that will be solved by the algorithm. Armoni (2013) used this hierarchy as a guide for the teaching of abstraction in CS and described a framework for teaching abstraction in the context of algorithmic problem solving. This framework includes several guidelines, e.g. to make a clear distinction between the program level and the object level and to start the problem-solving process on a high level, focusing on the verbal representation of the algorithm without including any programming language constructs.

Questions have been raised about whether teaching algorithms should include teaching students to implement a developed algorithm in code. In light of the described hierarchy (Perrenet et al., 2005), these questions refer to whether teaching algorithms only concerns the object level or also includes the program level. Futschek (2006) stated that students learn algorithmic thinking by solving problems, and signaled that for novices, specifying the solution in a programming language might shift the attention away from the problem-solving process. He therefore suggested that the algorithm should be treated at the object level and described in pseudo-code, meaning students do not have to deal with the syntax of a programming language. On the other hand, Grover et al. (2015) described a course aimed at teaching algorithmic thinking skills that included programming. In their course, pseudo-code was used as a step before coding; hence, algorithms were taught at the program level.

Taken together, these studies relate to the debate regarding the use of computer tools and the need for the creation of digital artifacts for learning CT. The Computer Science Unplugged project provides ways to develop CT without using computers (Bell et al., 2009); other approaches to teach algorithms without computers have been suggested, such as inventing and playing algorithms (Futschek & Moschitz, 2010), teaching graph algorithms by games and puzzles (Gibson, 2012), as well as the use of metaphors and analogies to help students form correct mental models (Forišek & Steinová, 2012). However, in a review of the state of the field of CT, Grover and Pea (2013) stated that introducing computing concepts without the use of a computer may keep “learners from the crucial computational experiences involved in CT’s common practice” (p. 40).

Along the same line, Denning (2017) determined “Traditional CT” versus “New CT”, describing as one of the important differences that “in Traditional CT programming ability produces CT, and in New CT learning certain concepts produces programming ability”. Traditional CT can be recognized in the definition of “algorithmization”, one of the so-called “fundamental ideas” of CS: “the entire process of designing, implementing and running an algorithm” (Schwill, 1997, p. 288), therefore focusing on both the described object level and the program level. New CT can be exemplified by the CS Unplugged project, in which algorithms do not need to be linked to a specific programming language, referring to the object level. However, it is interesting to note that the CS Unplugged material is being extended with “plugging it in” exercises. In a review of the use of CS Unplugged, Bell and Vahrenhold (2018) state that “the ‘programming vs.
Unplugged’ debate should not be an ‘either-or’ debate” (p. 507), instead advocating a combination that creates valuable learning. It is interesting to examine teachers’ knowledge in this respect.

Teaching students to construct and understand algorithms is regarded as difficult in secondary education. There are several reasons for this; for example, teaching students to analyze algorithms is viewed as an abstract topic (Dagienė & Jevisikova, 2012; Gal-Ezer & Zur, 2002). In addition, teachers, having varied backgrounds, might lack content knowledge (Cutts et al., 2017; Yadav, Gretter et al., 2016). Moreover, in their guide to teaching CS, Hazzan et al. (2014) characterized problem-solving techniques as creative processes and described the "discovery process" from the understanding of the requirements to the problem solution (p. 77). Teaching such processes is difficult, as reported by Sentance and Csizmadia (2017). In their study, teachers expressed “the challenge of developing, promoting and sustaining problem-solving strategies and techniques amongst the students they teach” (p. 481).

Concerning these challenges, several instructional strategies have been recommended. Hazzan et al. (2014), for example, suggested scaffolding as a strategy for teachers to support their students in the problem-solving process. Scaffolding had a central place in the study of Hasni and Lodhi (2011), who offered a guideline for designing lab exercises. They claim that detailed steps in the exercises may support students through the process of problem solving. The process of scaffolding was also reported as an instructional strategy by Lye and Koh (2014) in their review of teaching and learning CT. In addition, these authors suggested that students should be engaged in reflection, although this “is seldom found in the K–12 studies reviewed” (p. 59).

Assessment of algorithmic and computational thinking is found to be challenging as well (Basso et al., 2018; Grover, 2017). In addition, assessment is often combined with a specific programming environment (e.g. Wangenheim et al., 2018; Werner et al., 2012), suggesting a strong emphasis on programming skills.

Another challenge for teaching algorithms, as already mentioned, is the diversity of students. Not all students have prior programming experience, and some students progress faster than others (Sentance & Csizmadia, 2017). It may also be a challenge to make the teaching interesting for all students. Black et al. (2013) examined teachers’ perspectives on how to make computing interesting, and their results might also be relevant for learning algorithms. These authors stated that "students would be attracted by a feeling of recreation – in-the-moment enjoyment” (Black et al., 2013, p. 256). It will be interesting to examine what teachers consider to be enjoyable for students in the teaching of algorithms, what variations they observe related to this aspect, and how teachers respond to that.

Teaching algorithmic thinking or (the wider concept of) CT is not only aimed at learning to develop, use, and evaluate algorithms, but also concerns more generic skills. In a study on the role of the CS education community in the development of CT, Barr and Stephenson (2011) reported several of these skills, including being confident in dealing with complexity and being persistent when working with difficult problems (p. 51). Likewise, in a study of the conceptions and misconceptions of CT among Italian primary school teachers, Corradini et al. (2017) described “transversal skills” that are related to computational thinking. These skills include “create”, “communicate and collaborate”, and “be persistent when dealing with complex problems” (Corradini et al., 2017, p. 138). Barr
and Stephenson (2011) referred to these skills as dispositions and pre-dispositions, suggesting that they are prerequisites for learning CT; however, these skills may also be regarded as the intended outcome of teaching algorithms. Likewise, Sentance and Csizmadia (2017) found that teachers reported the problem that students lack resilience (suggesting it is a prerequisite), but the teachers also recommended strategies to build up resilience to failure in students, hence it becomes a result. Similarly, in a study of the PCK of Lithuanian teachers with respect to programming, Barendsen et al. (2014) found that newly qualified teachers regarded logical thinking to be a fundamental requirement for learning to program, while it was considered a result by more experienced teachers. It will be interesting to examine the knowledge of teachers in the Netherlands with regard to this aspect.

**Pedagogical content knowledge (PCK)**

The concept of PCK was introduced by Shulman (1986), who described it as a “special amalgam of content and pedagogy” (Shulman, 1987, p. 8), pointing out that PCK is domain-specific. PCK is seen as the knowledge teachers use to transform their understanding of the subject or topic they are teaching in order to teach their students more effectively (Henze & Van Driel, 2015). Many scholars have highlighted the relevance of the work of Shulman and suggested different conceptualizations of PCK (e.g. Gess-Newsome, 1999; Hashweh, 2005; Magnusson et al., 1999; Park & Oliver, 2008). In this study, we used four components of the widely adopted PCK model of Magnusson et al. (1999): *knowledge of goals and objectives for teaching a specific topic in the curriculum, knowledge of students’ understanding of this topic, knowledge of instructional strategies concerning this topic,* and *knowledge of ways to assess students’ understanding of this topic.* These components are shaped by the teachers’ orientations to teaching science.

The strength of a teacher’s PCK is determined by both the richness of the PCK components and the internal coherence of the teacher’s knowledge (Barendsen & Henze, 2019; Bayram-Jacobs et al., 2019). The coherence is reflected in the interconnection between the PCK components, apparent in the alignment of the various PCK components and in the explicit reasoning regarding more than one PCK component. To provide a better insight into PCK development, Park and Chen (2012) recommended the investigation of the interaction of the different PCK components. Hashweh (2005) also emphasized these interconnections when he defined PCK as a set of pedagogical constructions and asserted that these constructions result from the “interaction between different teacher knowledge and beliefs categories” (p. 279). The interaction of these components and the resulting coherence in teachers’ knowledge is required to transfer this knowledge into classroom practice. Barendsen and Henze (2019) described the significance of the coherence of these components, stating that these elements “need to be interconnected to enable effective scaffolding of students’ learning” (p. 1143).

Research into teacher knowledge may contribute to defining the content of that knowledge in terms of PCK components and the coherence of the teacher’s knowledge in terms of the interconnections of these components. By studying the content of teachers’ PCK, we aim to identify whether there is a “shared body of knowledge” (Meijer et al., 1999) that can be described by *patterns* in teachers’ PCK.
A variety of methods have been suggested to study teachers’ PCK, such as classroom observation (Barendsen & Henze, 2019), action research (Brandes & Armoni, 2019), or teaching vignettes (Yadav, Berges et al., 2016). A commonly used tool is the Content Representation (CoRe) format developed by Loughran et al. (2004), in which different “prompts” may be used as interview tools with groups of science teachers. This tool is suitable for eliciting teachers’ declarative PCK because it studies what teachers report to know, as opposed to dynamic PCK, the knowledge that is shown in action (Alonzo & Kim, 2016).

Teachers’ PCK for a certain topic is closely connected to their practice because teachers have experience in teaching this subject and know, for example, the best practices in terms of instructional strategies. As we are interested in the reasons behind teacher practices and the content of their knowledge for teaching algorithms, we investigated teachers’ PCK for this topic.

**Aim of the study**

This study aims to gain insight into the PCK of CS teachers for teaching algorithms. As elaborated in the previous sections, computing education is expanding and the teaching of algorithms is nontrivial, particularly with regard to the challenges related to teaching this subject, such as responding to student differences. The education of new CS teachers and professional development activities for in-service teachers are therefore required. The knowledge of experienced teachers may contribute to improving teacher training; therefore, this study seeks to examine teachers’ PCK and their responses to the above challenges. In light of our operationalization of the concept of PCK and its quality indicators, and our wish to discover indicators of common knowledge, we have translated this aim into the following research questions:

1. What is the content of the teachers’ knowledge for the different PCK components for teaching algorithms?
2. How are the components of the teachers’ PCK interconnected?
3. What patterns can be found in the teachers’ PCK for teaching algorithms?

**Methods**

Our study capturing teachers’ PCK was carried out in a naturalistic, qualitative setting (Cohen et al., 2011). Since the purpose of our analysis is descriptive, a phenomenological rather than a grounded theory approach was chosen. We conducted an in-depth, small-scale interview study in which teachers were asked about their knowledge regarding the elements of PCK (Goals, Instructional Strategies, Students’ Understanding, and Assessment) on teaching algorithms and algorithmic thinking. The interview data were analyzed thematically to identify characteristics and patterns within the teachers’ PCK.

**Participants and context**

We contacted CS teachers in the Netherlands, where CS is offered as an elective subject and is taught in grades 10–12, i.e. for students aged 15–18. Teachers were recruited from regional networks of CS teachers. At the time of the data collection, teaching algorithms
Table 1. Teaching experience of teachers participating in the study.

<table>
<thead>
<tr>
<th>Teacher</th>
<th>Teaching experience (years)</th>
<th>Teacher education</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16–20</td>
<td>CODI</td>
</tr>
<tr>
<td>2</td>
<td>6–10</td>
<td>CS Education (BSc)</td>
</tr>
<tr>
<td>3</td>
<td>6–10</td>
<td>CS Education (student)</td>
</tr>
<tr>
<td>4</td>
<td>6–10</td>
<td>CS Education (MSc)</td>
</tr>
<tr>
<td>5</td>
<td>6–10</td>
<td>CS Education (MSc)</td>
</tr>
<tr>
<td>6</td>
<td>16–20</td>
<td>CODI</td>
</tr>
<tr>
<td>7</td>
<td>1–5</td>
<td>CS Education (MSc)</td>
</tr>
</tbody>
</table>

was not compulsory in the curriculum; therefore, we deliberately selected teachers with experience in teaching algorithms. In 2019, a new curriculum for CS in upper secondary education was implemented and some teachers already prepared themselves for the curriculum reform, e.g. because they were involved in the development of teaching materials. In the new curriculum, algorithms are mentioned as an explicit learning goal, both in the skills domain and in the knowledge domain (Barendsen et al., 2016). Students should learn problem-solving skills and find solutions in terms of algorithms and data. In addition, students should be able to recognize and apply standard algorithms and investigate the correctness and efficiency of digital artifacts. The new curriculum consists of a core curriculum and several elective themes, one of them being “Algorithmics, computability and logic”.

The participating teachers all focused on teaching algorithms in their lessons, although the then-current curriculum did not include the concept of algorithms specifically. All teachers reported to teach standard algorithms (e.g. for searching or sorting). Some teachers combined the teaching and learning of the analysis of algorithms (e.g. with respect to efficiency) with a programming course, others had developed their own course for teaching standard algorithms and taught their students to analyze these algorithms.

One female and six male teachers participated in the study. Table 1 presents the professional experience of the teachers participating in the study. Their teaching experience varied from 1–5 years to 16–20 years, with most having 6–10 years of experience. Teacher education varies substantially among Dutch CS teachers (Grgurina et al., 2018). Among our participants, two teachers followed an in-service teacher training program for two years, known as the CODI program. From 1998–2005, teachers with a teaching degree in any other subject could participate in the CODI program to become a certified CS teacher. One of the participating teachers is currently studying to become a licensed CS teacher. The other four teachers have Bachelors or Masters degrees in CS education.

We had planned seven interviews with the intention to invite more interviewees if still new ideas would come up (Mason, 2010). However, we concluded that we had reached this so-called data saturation after these seven interviews.

**Data collection**

We used semi-structured interviews to capture the teachers’ declarative PCK. Some of our interview questions were taken from the CoRe questionnaire (Loughran et al., 2004). Other questions were added with the aim of eliciting the teachers’ knowledge of how
to respond to student differences. Because we focused mostly on teachers’ practices during lessons, the question regarding assessment was directed at formative assessments. The interview questions were used to start the conversation, and we formulated follow-up questions when necessary.

The research started with a pilot study to test and refine the interview questions. During the pilot interviews we noticed that, when asked about their instructional strategies, the teachers tended to focus on their own behavior in the classroom, elaborating on what they did, what they prepared, and so on, while neglecting to mention what their students do in class. We therefore adjusted the interview questions, starting with a question about the students’ activities (Henze & Van Driel, 2015). Interview questions are listed in Table 2.

The interviews started with an introduction to explain that our research is focused on eliciting teachers’ knowledge regarding teaching algorithms and algorithmic thinking. We introduced a definition of algorithmic thinking based on the definition of Futschek (2006), and we handed the teachers a card presenting this definition (see Figure 1, translated from Dutch).

We asked whether the teachers recognized this definition to ensure we had a shared understanding of the topic we were discussing. All teachers concurred with this definition, confirming that they recognized the various aspects of the definition. We then asked teachers to recall a lesson on algorithms and algorithmic thinking and continued with the interview questions.

The interviews to capture teachers’ declarative PCK lasted between 22 and 46 minutes, this does not include introductory talk, which makes it comparable to other PCK studies

Table 2. Interview questions used in this study.

<table>
<thead>
<tr>
<th>Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. What do you see your students doing during lessons on algorithms and algorithmic thinking?</td>
</tr>
<tr>
<td>2. What do you do during these lessons?</td>
</tr>
<tr>
<td>3. What do you intend for your students to learn?</td>
</tr>
<tr>
<td>4. Why is it important for students to learn this?</td>
</tr>
<tr>
<td>5. What elements do your students enjoy or find interesting about this topic?</td>
</tr>
<tr>
<td>6. What elements do your students think of as easy about this topic? How do you recognize this?</td>
</tr>
<tr>
<td>7. What elements do your students think of as difficult about this topic? How do you recognize this?</td>
</tr>
<tr>
<td>8. Do you notice during lessons whether students fully comprehend what is taught? How can you tell?</td>
</tr>
<tr>
<td>9. Do differences in the prior knowledge of students play a role when teaching this topic? How do you deal with that?</td>
</tr>
<tr>
<td>10. For this topic, how do you handle the different interests of the students?</td>
</tr>
<tr>
<td>11. Are there other differences between students that you observe and take into account in your lessons? How do you do that?</td>
</tr>
</tbody>
</table>

Algorithmic thinking has to do with abilities that are connected to constructing and understanding algorithms. The following abilities play a role in this:

- analyzing given problems and specifying them precisely
- constructing correct algorithms
- analyzing algorithms, e.g., with respect to efficiency.

Figure 1. Definition of algorithmic thinking.
(e.g. Henze et al., 2008). Some teachers were able to quickly explicate and verbalize their knowledge and to come up with relevant examples directly. For other teachers, more inquiring was needed to elicit their knowledge as they needed more time to consider their answers. Furthermore, some teachers were in the middle of teaching a class about algorithms and could draw on “fresh knowledge”, while others needed more time to remember what happened during these lessons. The interviews were audio-recorded and transcribed verbatim.

**Data analysis**

Data were analyzed to capture teachers’ PCK by studying the content of the PCK components and checking the interconnections of these components, as previously reported by Alonzo and Kim (2016), Barendsen and Henze (2019), and Bayram-Jacobs et al. (2019). The data analysis started with repeated readings of the transcripts to develop an understanding of the data as a whole. In the data analysis process, a combination of a deductive and an inductive coding strategy was used to develop a code list that can serve as a basis for analysis. The units of analysis were coherent text segments where a teacher formulated an idea about teaching. The data were coded using the Atlas.ti qualitative data analysis software.

**Data analysis for content of PCK components**

First, to capture the content of each teacher’s PCK, the corresponding interview transcripts were analyzed for the four PCK components from the PCK model described by Magnusson et al. (1999): Goals (GO), Instructional Strategies (IS), Students’ Understanding (SU), and Assessment (AS). We developed an initial code list based on the model of Magnusson et al. (1999), while also using in vivo coding (Miles et al., 2014) to identify codes that describe the content of the teachers’ PCK within each component on a detailed level. For example, from the model, the main code “IS–class activities” was derived, i.e. knowledge of activities that can be used to help a class of students comprehend specific concepts. Because we noticed important differences in what the teachers mentioned regarding these class activities, we added codes to capture this variation (e.g. “IS–class activities – where students cooperate”).

During this open coding process, we prepared a table in which we listed the various quotations from each teacher for each code to search for similarities. For example, by examining all quotations regarding “difficult to learn” concepts, we found that teachers not only referred to specific aspects of learning algorithms (e.g. abstract problems or asymptotic notation), but also referred to the learning process itself (students do not know where to start when solving problems). This approach helped us to analyze the data thoroughly and capture more details. At the end of this phase, we investigated the content of the teachers’ knowledge for the different PCK components for teaching algorithms (as an answer to research question 1).

**Data analysis for interconnection of PCK components**

As a next step, we investigated the interview transcripts for interconnection between the PCK components. We examined the consistency and alignment of different PCK components per teacher and used memo-writing (Charmaz, 2006) to generate seven memos,
one memo per teacher. The background information about a teacher (e.g. education, experience), and a summary of the content of each PCK component were included in each memo. In this way, we obtained interconnections per teacher and we subsequently revisited the data to verify whether our findings were accurate. We also examined text fragments that were coded with more than one PCK component, indicating an interconnection.

To elucidate the nature of the interconnection we used the codes in the “interconnection” category. The main codes for this category (e.g. Connection GO–SU) were created prior to coding (deductive coding), and the subcodes emerged during data analysis (inductive coding). For example, we looked for text segments where teachers reported how they used certain instructional strategies to meet specific goals and applied the code “Connection GO–IS – specific instructional strategies aimed at achieving certain goals”. Both the the written memos and the text fragments coded with interconnection codes supported us in the interpretation process for understanding how the components of the teachers’ PCK are interconnected (as an answer to research question 2).

The final coding scheme is shown in Table 3. This scheme is also used to organize the results section where examples are given to clarify the codes.

Data analysis for patterns
The analysis thus far can be characterized as within-case. In the last step of the data analysis, we performed a cross-case analysis (Miles et al., 2014) with the goal of identifying shared knowledge (cf., Meijer et al., 1999). To this end, we looked for common patterns in the teachers’ PCK.

We used selective coding (Boeije, 2010) and looked for connections between the categories found in earlier phases to make sense of the data. As Boeije (2010, p. 114) explains, “selective coding is aimed at integrating the loose pieces of your earlier coding efforts and can be considered a logical step after the segmenting of the data”. In this process, we identified overarching categories (“core categories”, as described by Boeije (2010)) characterizing common and differentiating features that are reflected frequently in the data, towards possible typifications of teachers’ PCK. Thus, while examining the codes for Goals, Students’ Understanding, Instructional Strategies, and Assessment concurrently, by continually comparing specific incidents in the data (Taylor & Bogdan, 1984), we identified patterns, which in our case were the cross-case similarities, differences, and component interdependencies in the teachers’ PCK.

Trustworthiness
In this subsection, we discuss the aspect of trustworthiness, a term used to refer to the quality and rigor of qualitative research. Lincoln and Guba (1985) defined criteria to assess the trustworthiness in the context of naturalistic inquiry, comprising credibility, transferability, dependability and confirmability.

We assured credibility by our research design, consisting of interviews with teachers with the use of CoRe questions. This is a common research method to capture teachers’ PCK.

The transferability of the study is supported by the detailed descriptions of the data, in qualitative research referred to as “thick descriptions”. By giving detailed accounts of both
**Table 3. Coding scheme for PCK.**

<table>
<thead>
<tr>
<th>Categories</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goals</td>
<td>Goals for teaching algorithms</td>
</tr>
<tr>
<td></td>
<td>• learning to analyze algorithmic problems (e.g. decomposition)</td>
</tr>
<tr>
<td></td>
<td>• learning to solve algorithmic problems</td>
</tr>
<tr>
<td></td>
<td>• learning to code an algorithm</td>
</tr>
<tr>
<td></td>
<td>• learning to analyze quality aspects of algorithms</td>
</tr>
<tr>
<td></td>
<td>• being able to apply problem-solving skills outside CS</td>
</tr>
<tr>
<td></td>
<td>• learning transversal skills while solving algorithmic problems</td>
</tr>
<tr>
<td>Motivations for objectives</td>
<td>• preparing for a role in society</td>
</tr>
<tr>
<td></td>
<td>• working with computers as tools</td>
</tr>
<tr>
<td>Students' understanding</td>
<td>Variation in student approaches to learning because of:</td>
</tr>
<tr>
<td></td>
<td>• different abilities</td>
</tr>
<tr>
<td></td>
<td>• different interests</td>
</tr>
<tr>
<td></td>
<td>• other reasons (e.g. different levels, motivation, gender)</td>
</tr>
<tr>
<td>Prior knowledge and skills</td>
<td>• programming experience</td>
</tr>
<tr>
<td></td>
<td>• mathematical knowledge</td>
</tr>
<tr>
<td></td>
<td>• analytical skills</td>
</tr>
<tr>
<td>Activities that are difficult or easy to learn</td>
<td>• difficult: how to start solving problems</td>
</tr>
<tr>
<td></td>
<td>• difficult: abstract problems, generalizing</td>
</tr>
<tr>
<td></td>
<td>• difficult: analyzing and optimizing an algorithm</td>
</tr>
<tr>
<td></td>
<td>• difficult: describing the analysis of algorithms, asymptotic notation</td>
</tr>
<tr>
<td></td>
<td>• easy: concrete problems, puzzles</td>
</tr>
<tr>
<td>Engaged by</td>
<td>• engaged by satisfaction, gratification of finding a solution</td>
</tr>
<tr>
<td></td>
<td>• engaged by solving puzzles, &quot;unplugged&quot;</td>
</tr>
<tr>
<td>Disengaged by</td>
<td>• engaged by understanding &quot;how it works&quot;</td>
</tr>
<tr>
<td></td>
<td>• disengaged by specifying algorithm (e.g. flowchart)</td>
</tr>
<tr>
<td>Instructional strategies</td>
<td>Representations for algorithms</td>
</tr>
<tr>
<td></td>
<td>• describe algorithm in different languages (natural or programming)</td>
</tr>
<tr>
<td></td>
<td>• express algorithms in schematic diagrams (e.g. flowchart)</td>
</tr>
<tr>
<td></td>
<td>• use illustrations, examples (e.g. puzzles)</td>
</tr>
</tbody>
</table>

(Continued)
Table 3. (Continued).

<table>
<thead>
<tr>
<th>Categories</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher activities to help students comprehend specific concepts</td>
<td>• coaching or scaffolding</td>
</tr>
<tr>
<td></td>
<td>• discussing solutions with the class</td>
</tr>
<tr>
<td></td>
<td>• lecturing or instructing a class</td>
</tr>
<tr>
<td>Class activities to help a class comprehend specific concepts</td>
<td>• where students are active in practical work</td>
</tr>
<tr>
<td></td>
<td>• where students think, reflect, evaluate</td>
</tr>
<tr>
<td></td>
<td>• where students cooperate</td>
</tr>
<tr>
<td>Assessment</td>
<td>Aspects of students’ learning that are important to assess</td>
</tr>
<tr>
<td></td>
<td>• developing an algorithm</td>
</tr>
<tr>
<td></td>
<td>• implementation of an algorithm</td>
</tr>
<tr>
<td>Specific instruments of assessment</td>
<td>• observing students during class</td>
</tr>
<tr>
<td></td>
<td>• asking questions to students</td>
</tr>
<tr>
<td></td>
<td>• giving students a test or a practical assignment</td>
</tr>
<tr>
<td>Interconnections</td>
<td>Connection GO–SU</td>
</tr>
<tr>
<td></td>
<td>• focus regarding goals is related to the appreciation of prior programming experience</td>
</tr>
<tr>
<td></td>
<td>• students like to find the solution of an algorithmic problem</td>
</tr>
<tr>
<td></td>
<td>• students are interested in relevant algorithmic problems</td>
</tr>
<tr>
<td>Connection SU–IS</td>
<td>• adapting instructional strategies to meet student differences</td>
</tr>
<tr>
<td></td>
<td>• scaffolding for topics that are difficult to learn</td>
</tr>
<tr>
<td>Connection GO–IS</td>
<td>• specific instructional strategies for reaching certain goals</td>
</tr>
<tr>
<td></td>
<td>• use of representations that are dependent on goals</td>
</tr>
<tr>
<td>Connection GO–AS</td>
<td>• specific types of assessment for tracking certain goals</td>
</tr>
</tbody>
</table>
the context and the results of the study, contextualized understanding may be achieved hence comparisons to other contexts might be made.

We have taken measures that contribute to the dependability of the study, especially with regard to the data analysis. The first and second authors met almost every week to discuss the coding process to make sure that the definitions were clear and that there was no overlap in the codes. The first author was the main coder. At different stages in the analysis process, the first and second author coded a part of a transcript separately and discussed all the disagreements until consensus was reached. The purpose of these discussions was to confirm interpretations, thus adding to the consistency of the analysis. Therefore, we chose to seek agreement rather than assess inter-rater reliability in a formal, quantitative sense (McDonald et al., 2019, p. 15). During meetings with all authors (once per six weeks), we deliberated whether we missed relevant data or whether the analysis put too much emphasis on certain information, to ensure a balanced data selection (Boeije, 2010). Furthermore, the third and fourth author often adopted a critical stance toward the analysis and played the devil’s advocate towards the findings to avoid biased interpretations. For example, we discussed how some elements of “Students’ Understanding” seemed to be generic (e.g. the variation in student approaches to learning) while others were more topic-specific (e.g. the activities that are difficult to learn, such as describing the analysis of algorithms), and whether these generic elements still should be included in an investigation concerning PCK (which, by definition, is topic specific). By re-examining the transcripts, we concluded that seemingly generic elements may refer to topic specific quotations. For example, a teacher reported how students differ and that students who score better come up with more complex and efficient solutions to algorithmic problems.

To ensure the confirmability of the study, all interviews have been transcribed verbatim. In addition, all authors were involved in the data analysis process. The various steps of the analysis were discussed in order to aim for “dialogical intersubjectivity”, described by Brinkmann and Kvale (2015) as “agreement through a rational discourse and reciprocal criticism between those interpreting a phenomenon” (p. 279).

Results

In this section, we first present the results of the analysis of the teachers’ PCK for teaching algorithms with respect to the four components of PCK, using the model proposed by Magnusson et al. (1999). In addition, the results of the analysis of the interconnections between these components will be described. We conclude this section with a description of the patterns we observed in the teachers’ PCK for teaching algorithms. The teachers’ quotes were translated from Dutch into English.

Teachers’ knowledge of teaching algorithms in terms of PCK components

Teachers’ knowledge of the goals of teaching algorithms

The analysis of teachers’ knowledge regarding goals for teaching algorithms resulted in several objectives (what students should learn) and in motivations for these objectives (why students need to learn this).

Goals for teaching algorithms. All teachers described learning to analyze algorithmic problems as an objective, as they aim for their students to learn decomposition and
expressed variation themes generic. items (dis)engaged From Teachers’ tives. Motivations aspects Other. These learn skills solving. Some where important If I want you to prepare children [for today’s software-led society], … then I think it is important that students learn to think algorithmically. (T3)

**Teachers’ knowledge of students’ understanding of teaching algorithms**

From the analysis of teachers’ knowledge about students’ understanding, four main themes emerged: “variation in student approaches to learning”, “prior knowledge and skills for learning”, “activities that are difficult or easy to learn”, and ‘elements students get (dis)engaged by’. A variety of items were expressed regarding these themes. Most of the items were topic-specific (i.e. specific for teaching algorithms), while other items (e.g. variation in student approaches to learning because of different interests) are more generic. This variation is apparent in the knowledge of all teachers, as all participants expressed both generic and topic-specific knowledge.
Variation in student approaches to learning. Participating teachers described the variation in the approaches to learning within a CS class and mentioned that these diverse approaches may be caused by the different abilities of students.

You see a difference in end levels emerging ... there are students who score better, they come with more complex and more efficient solutions. (T2)

According to the teachers, these various approaches may also stem from different interests or diverse levels or tracks of students. Additionally, teachers described how students differ in terms of motivation and gender.

Prior knowledge and skills. According to the teachers, prior knowledge and skills for learning algorithms is related to the programming experience, mathematical knowledge, and analytical skills of the students. Some teachers do not value students' prior programming experience as it may ensure that students become frustrated when they have to think first.

There are students who have programmed a lot, and I want them to stay away from a programming language and to think about the problem ... but then frustration emerges: 'why can I not start implementing right away?' (T3)

According to these teachers, prior programming experience does not help in thinking about solutions as students may be inclined to use a “trial-and-error” method. Regarding analytical skills, teacher T4 reported:

Some students have more developed analytical skills than others ... students who have those analytical skills do not have any problems, while others need to learn some sort of roadmap; they almost need an algorithm for how to write algorithms. (T4)

Activities that are difficult or easy to learn. Several activities were described as being either “difficult” or “easy” to learn. Puzzles as unplugged activities were mentioned as being “easy”, along with concrete problems connected to the students' own lives. For example, teacher T2 described discussing Dijkstra’s algorithm and relating it to planning a route with Google maps:

They think it is easy because you bring assignments that are relevant to their world. (T2)

The teachers observed that students have difficulty with how to start solving problems. Teachers also described that students have difficulty with abstract problems, generalizing, analyzing and optimizing algorithms, and the analysis of the running time of algorithms and asymptotic notation. According to teachers, that might be related to their poor understanding of the underlying mathematical knowledge.

That exponential (running time) is much bigger than quadratic does not mean much to them. (T5)

Elements students get (dis)engaged by. Several teachers mentioned that students enjoy solving algorithmic problems because it gives them cognitive satisfaction.

Most students feel a sense of satisfaction when they find a solution. (T1)

Teachers also mentioned solving puzzles is an element that students get engaged by, such as working on the Bebras challenge or searching for the shortest route on a map. Students
also get engaged when teachers show the relevancy of a new concept, and when students understand how it works and realize where they can apply it.

According to the teachers, students get disengaged by specifying algorithms and making schematic representations such as flowcharts:

In my view they do not take it seriously . . . for them it feels a bit like ‘dead weight’. (T6)

**Teachers’ knowledge of instructional strategies for teaching algorithms**

In analyzing teachers’ knowledge about instructional strategies, we focused on their knowledge of specific strategies that are useful for helping students to comprehend specific concepts related to algorithms. Three main categories emerged during the data analysis: representations, teacher activities, and class activities.

Representations. We found that the teachers use several representations for algorithms. Algorithms may be described in natural language, in pseudo-code, or in programming language. Most teachers start by asking their students to describe an algorithm in natural language:

I would introduce a general problem that they consider in their own language, without using jargon, and where they will say: that’s the way I would solve this problem. (T3)

Likewise, teachers use various schematic diagrams (such as flowcharts or Nassi-Shneiderman diagrams) to ensure that students think about the problem first. Some teachers ask their students to translate the pseudo-code or flowchart into code. Other representations that were used include illustrations and examples. Teachers mentioned using puzzles or unplugged activities such as playing cards when they teach sorting algorithms.

Teacher activities to help students understand specific concepts. Teachers mentioned several activities they carry out: lecturing or instructing, coaching or scaffolding, and discussing solutions with the class. Teachers mentioned discussing student solutions for algorithmic problems with the whole class. There are usually alternative solutions for an algorithmic problem and teachers mentioned to encourage students to evaluate the different solutions and reflect on them. Students also learn from each other when they see each other’s solutions.

Class activities to help the group comprehend specific concepts. The teachers reported using various class activities to facilitate the learning of algorithms. Teachers described activities where students are active in practical work; activities where students think, reflect, and evaluate; and activities where students cooperate with each other. For example, a teacher reported instructing his students to compare different sorting algorithms. Furthermore, teachers report encouraging their students to cooperate when solving algorithmic problems because they believe that it enhances student problem-solving skills.

By discussing it with others, you will find a better solution than when you think about a solution all by yourself. (T1)

**Teachers’ knowledge of assessment for teaching algorithms**

When analyzing the results of teachers’ knowledge of assessments, we found that they considered certain aspects of student learning important to assess, and that teachers utilized specific instruments of assessment.
Aspects of student learning that are important to assess. Teachers described assessing the development of algorithms (creative thinking about a solution), and some teachers reported assessing the implementation of algorithms in code. Another teacher asked questions regarding the complexity of algorithms:

Questions such as ‘what is the running time of the given code, is it $n^2$ or $2n$?’ are doable, but questions such as ‘think of an algorithm with a running time of $\lg(n)$’ are more difficult, and they are unable to come up with an example. (T4)

Specific instruments of assessment. Participating teachers mentioned that they assess students’ understanding by observing their students during class:

You can observe that from their facial expressions … when you give them the solution, some students are very expressive — they jump up and down. (T2)

Another way of assessing is asking questions during lessons:

Students have to draw a flowchart during class … and I ask them to explain what they did. (T7)

The teachers also mentioned tests or practical assignments as means of assessing their students’ knowledge.

**Interconnection of PCK components**

In addition to analyzing teachers’ knowledge for each PCK component, we investigated the interconnection of the different PCK components. The analysis revealed interconnections between Goals and Students’ Understanding (GO–SU), Instructional Strategies and Students’ Understanding (IS–SU), Goals and Instructional Strategies (GO–IS), and Goals and Assessment (GO–AS). These interconnections are illustrated in Figure 2 and described in this section.

![Figure 2. Interconnection between PCK components (Adapted from Bayram-Jacobs et al., 2019).](image-url)
Interconnection between goals and students’ understanding
The interconnection between GO and SU became apparent when we observed that the teachers’ focus regarding goals is related to their appreciation of students’ prior programming experience. Teachers who focus on “learning to analyze” and “learning to solve” are cautious about students’ prior programming experience, as these students might pay more attention to coding a solution than to finding an efficient algorithm.

When they tell me they have learned to program in a certain language I think, ‘OK, that’s nice’; however, it is questionable whether these students got all the concepts right and whether they have the right conceptual approach. (T3)

Teachers who, in addition to the aforementioned goals, focus on “learning to code an algorithm” value students’ prior programming experience as it might help them to think in a more ordered way.

When you have real coding experience, then you have learned structured thinking. That way, you can model your algorithm better, I think. (T2)

Furthermore, the GO–SU interconnection is evident in the teachers’ descriptions of the elements that students get engaged by during the teaching and learning of algorithms. For example, students like to find the solution of an algorithmic problem. As mentioned before, “learning to solve algorithmic problems” is one of the goals of teaching algorithms. Several teachers mentioned that students take pleasure in finding a solution for such a problem. Likewise, the GO–SU interconnection may be noticed when teachers reported about students who are interested in relevant algorithmic problems. Teachers described the motivations for teaching algorithms, such as “preparing for a role in society” and “working with computers as tools”. The teachers also reported that students get engaged by understanding “how it works”.

So you explain to students that you click on a button and the list gets sorted. It makes them realize: but how does it work? (T2)

In addition, a teacher described how he therefore employs the apps and systems used by students as illustrations to improve their algorithmic thinking.

Interconnection between instructional strategies and students’ understanding
Our analysis of the interviews revealed that the teachers base their use of instructional strategies on their knowledge of students’ understanding. Adapting instructional strategies to meet student differences. The teachers described how students differ in, for example, their abilities, motivation, and interests, and how these differences influence the activities they carry out or the activities they ask their students to carry out. This knowledge is often generic; it is not only applicable for teaching algorithms. For example, a teacher described how important it is to find the right level for the student activities, and how that might vary for each student:

It has to be challenging … it should excite, so the problem that you give them needs to be slightly more difficult than what they can handle. Then they will be challenged … and that differs for each student. (T1)
Another teacher explained how the various abilities of students play a role in creating groups for activities in which students cooperate. Sca olding for topics that are di cult to learn. Participating teachers described how they support their students when they encounter difficulties in solving algorithmic problems. A teacher appeared to use his previous experience to support his current students:

Especially when discussing solutions, I will put different emphases on the subject matter . . . In past years I have learned where students get stuck so I can anticipate difficulties slightly more easily. (T5)

**Interconnection between goals and instructional strategies**

As described under “Teachers’ Knowledge of the Goals of Teaching Algorithms” in the previous section, learning to analyze and learning to solve algorithmic problems are reported as goals when teaching algorithms. The teachers mentioned the use of several specific instructional strategies for reaching these goals.

You have to instruct your students and give them the tools that they can effectively solve something, either on their own or in a small group. (T3)

Teachers described how they encourage their students to think first, in order to be able to solve an algorithmic problem. A teacher specified how he asked his students to draw a roadmap and discuss these roadmaps using the “think – pair – share” methodology. Another teacher emphasized that more attention should be paid to teaching students to analyze algorithmic problems:

It is about the analysis of a problem and what we need to come up with a solution . . . What we miss in secondary education is to first think about what is needed, what is the goal, how we get there, and what steps can be distinguished. (T4)

Apparently, the goal of students being able to analyze and solve algorithmic problems requires specific instructional strategies where students learn to think about algorithmic problems and learn problem-solving methods.

The interconnection between GO and IS is also apparent in teachers’ use of representations for algorithms that are dependent on the goals they aim for. Teachers who emphasized that students should think first (focusing on goals such as learning to analyze and learning to solve) make use of natural language, pseudo-code, or flowcharts to represent algorithms. Teachers who in addition focus on “learning to code an algorithm” use the same representations, but they encourage their students to implement the algorithm in code too.

**Interconnection between goals and assessment**

The interconnection between GO and AS became apparent in teachers’ statements regarding specific types of assessment for tracking certain goals. For example, “learning to solve algorithmic problems” may be assessed by asking students to draw a flowchart. Teacher T7 described this:

Transforming the algorithm in a flowchart was done quite well . . . I observed them while drawing their flowchart on paper. There were not many students in the class; therefore, I was able to ask them individually to clarify what they did. (T7)
Another teacher mentioned that he would have liked to use tools to assess students’ thinking processes, because, for him, the focus is on “learning to solve an algorithmic problem”. This teacher advocated the assessment of conceptual knowledge:

We assess the implementation too much and I think we assess conceptual knowledge insufficiently. How do you assess whether students really understand a concept? (T3)

**Patterns in teachers’ PCK for teaching algorithms**

To answer our third research question about patterns in teachers’ PCK for teaching algorithms, we analyzed the content of teachers’ PCK. This analysis resulted in the identification of two patterns. The first pattern concerns the variation we observed in teachers’ focus regarding their goals associated with their knowledge of students’ understanding and instructional strategies. The second pattern is related to the teachers’ knowledge of responding to student differences when teaching algorithms.

**Different focus regarding goals for teaching algorithms**

Teachers appeared to be focused on either “thinking” or “thinking and making” as their goals when teaching algorithms because they differed in their belief of whether implementing the algorithm in code (“making”) has any additional value after designing an algorithm. This variation in goals is related to the teachers’ appreciation of students’ prior programming experience and their use of specific representations for algorithms. We observed that teachers who focus on “thinking” are cautious about students’ prior programming experience as they are concerned that the attention paid to the syntax of a programming language might distract students from thinking about a solution for an algorithmic problem. These teachers use natural language, pseudo-code, and flowcharts as representations for algorithms.

For this pattern, the numbers were evenly balanced, with three (focus on “thinking”) and four (focus on “thinking and making”) teachers on each side. Teacher T3 may be regarded as an example of a teacher who focuses on “thinking” without focusing on “making”:

You can think very well about algorithms without being able to code. (T3)

This teacher reported being unhappy with the current teaching materials:

Because it is all about syntax, to me it seems that you offer them a programming language and hope that, while completing the assignments, students will get the idea of algorithmic thinking. (T3)

He reported focusing on problem solving:

Students learn that for every problem they could see around them (it does not matter from which discipline) they have sufficient tools to know how to solve the problem, automated or not. (T3)

On the other hand, teachers who focus on “thinking and making” have a more positive notion of students’ prior programming experience, pointing to the structured way of thinking that students developed when learning to program. Teacher T5 may be viewed
as an example of a teacher who focuses on “thinking and making”. Asked about what he intends his students to learn, this teacher answered:

I want them to be able to apply some algorithmic concepts such as backtracking and recursion … I think problem-solving skills are very important. If such a problem is offered to you, how do you look at it, how do you get from a problem to a solution? … I want them to be able to recognize problems, so they can see the key to a solution based on the problem description. I want them to recognize good and not so good solutions, then you are talking about complexity. (T5)

Later this teacher added:

That they recognize that there is a recipe for solving a problem. When they see that, and especially when they see their program works … they enjoy the reward when they have made it themselves. (T5)

Regarding prior programming experience, this teacher reported:

It helps if they have done something with Scratch or Robomind academy or similar, because they are aware that a computer only carries out what you tell it to do, not what you think … that you have to specify whatever needs to happen. When students have used Scratch, for example, they are aware of constructions such as ‘sequence’, ‘iteration’, and ‘selection’ so they figure it out faster. (T5)

This pattern is illustrated in Figure 3.

**Knowledge of responding to student differences**
We recognized a pattern in teachers’ knowledge of responding to student differences when teaching algorithms. As mentioned before, teachers’ knowledge regarding students’ understanding appeared to be generic or topic-specific, and all teachers expressed

---

![Figure 3](image-url)

**Figure 3.** Two foci in teachers’ PCK for teaching algorithms.
both types of knowledge. However, we identified a pattern in the teachers’ knowledge of responding to student differences (the interconnection between SU and IS): different teachers tended to express either more generic knowledge (three teachers) or more topic-specific knowledge (four teachers).

The teachers who exhibited more topic-specific knowledge of responding to student differences described how to support students in learning algorithms. They described focusing on scaffolding students’ problem-solving skills. When students encountered problems or got stuck, teachers reported providing hints to enable students to move on. A teacher described that the depth of his explanation of a concept, for example, Dijkstra’s algorithm, was dependent on the level of the students in a class. Furthermore, the teachers described using various representations of algorithms, specifically to ensure that all students would be able to understand the steps of an algorithm. Teacher T4 responded to students’ different interests by setting open assignments:

To give them open assignments … they should come up with their own algorithm and then write a roadmap and draw a flowchart for that algorithm … It should be an algorithm concerning something they are interested in, e.g. their hobby. (T4)

The teacher added an extra step in the problem-solving process to respond to students’ different skills:

I intentionally added an extra step, because smarter students or students with more affinity might be able to write pseudo-code right away. For others I added this step … ‘first write your solution in “normal Dutch” sentences, then rewrite your solution in pseudo-code’. (T4)

Other teachers expressed mostly generic knowledge regarding responding to students’ differences. For example, another teacher mentioned that he preferred to use open assignments. He related it to students’ differences and mentioned to use problems that are “doable” for all students, but he also emphasized:

Assignments need to be challenging, they should be just a little too difficult … but that differs for each student, sometimes between groups of students but most of the time for each student, because you really have all sorts of students in a class. (T1)

**Discussion**

Prior work has examined the PCK of CS teachers (Barendsen et al., 2014; Saeli et al., 2011); however, very little was found in the literature regarding teachers’ PCK for teaching algorithms. The present study was designed to examine CS teachers’ PCK for teaching algorithms, which we hoped might elucidate the specific knowledge needed to teach algorithms.

The results of our study indicate a pattern in the goals for teaching algorithms. The teachers focus on either “thinking” or “thinking and making” and, correspondingly, have a different appreciation of students’ prior programming experience and differing ideas regarding the use of specific representations for algorithms. These findings must be interpreted with caution because of the size of our sample; however, it is interesting to consider this pattern because teachers’ goals for teaching algorithms may be related to the abstraction levels for the algorithmic concept, as defined by Perrenet et al. (2005).
Teachers who focus on “thinking” seem to restrict their teaching to the object level, where an algorithm is viewed as an object and not linked to a specific programming language. Teaching students to analyze algorithms in terms of, for example, its efficiency is an example of teaching algorithms at the object level. On the other hand, teachers who focus on “thinking and making” address both the object level and the program level, as they require their students not only to reason about the algorithm as an object, but in addition to write the algorithm in a specific programming language.

All teachers aim for their students to learn to analyze algorithmic problems and to learn to analyze quality aspects of algorithms. However, teachers who focus only on “thinking” explain their motivations by pointing to the strong focus on syntax that occurs the moment students start writing code. Therefore, these teachers focus only on the level of algorithm as verbal description and not on the level of algorithm as program. For teachers, the clear distinction between these levels as recommended by Armoni (2013) might clarify what they are aiming for and what instructional strategies should be involved. Furthermore, verbalizing explicitly why they aim for a specific goal might contribute to Armoni’s (2013) call to “be persistent and precise”.

Our results may also contribute to the debate regarding CT. Denning (2017) distinguished between “traditional CT” and “new CT”, describing one of the characteristics of “new CT” as follows: “it is completely optional whether an algorithm will ever be translated into a program” (p. 37). This perspective seems consistent with the knowledge of the teachers who focus on “thinking”, because being able to write code is not required to think about algorithms. Therefore, it is possible that the focus of goals on either “thinking” or “thinking and making” for teaching algorithms suggests that the distinction of CT as described by Denning (2017) might be more generally applicable for teaching CS.

Regarding teachers’ knowledge about how to respond to student differences, we found a focus on either generic or topic-specific (i.e. specific for teaching algorithms) knowledge. We examined the teachers’ educational background and the number of years of teaching experience, and found that all of the teachers who focus more on topic-specific knowledge about responding to student differences had a background in CS, and may therefore have had more content knowledge. As PCK is represented as the blending of content and pedagogy (Shulman, 1987), more well-developed content knowledge may translate into more topic-specific knowledge about how to respond to student differences when teaching algorithms. Although the small sample size impedes the generalization of these findings, our results may point to the variety of knowledge needed for teaching algorithms, with both generic and topic-specific knowledge contributing to teaching this topic.

Another finding regarding teachers’ responses to student differences concerns the use of specific instructional strategies. Teachers reported that students need to be challenged; however, they also mentioned that the difficulty of consecutive algorithmic problems should not be increased too quickly. The teachers described how they needed to scaffold and invent extra steps or activities to help students develop their algorithmic thinking. In line with Vygotsky’s (1978) “zone of proximal development”, there is a thin line between “challenging” and “too hard to try”. As this line may be different for each student, teachers have to know their students well in order to give them the support they need. Although the teachers reported that analytical skills seem to be more helpful than programming experience for students when learning algorithms, they described adapting their
instructional strategy, particularly for students with programming experience who might become frustrated when they are required to write pseudo-code or draw flowcharts before starting to code. Teachers may need to support these students to analyze algorithmic problems first, in order to solve them.

The teachers described how they adapt their instructional strategies to meet the various student approaches to learning, which implies a consideration of their different interests. The teachers mentioned that students like it when teachers demonstrate the relevance of new concepts or use application situations, or contexts, from the students’ own lives. An element that students do not seem to enjoy is drawing schematic diagrams, although almost all the teachers reported using a form of schematic representation for algorithms. This suggests that teachers believe that, in order to support their algorithmic thinking, students should learn to represent their algorithm in a schematic diagram. A possible explanation for this is that reification (converting an abstract concept into a thing, in this case a diagram) might support student learning. The reification of an algorithm into code could have the same effect, although it might put too much focus on the syntax and distract from the problem-solving process. Futschek (2006) pointed to this risk and suggested that the algorithm should be described in pseudo-code. In the present study, teachers reported that students did not show a similar dislike for pseudo-code as for flowcharts or other diagrams; therefore, it would be interesting to further examine why teachers use these schematic diagrams.

Furthermore, our data analysis suggested several characteristics associated with the four components of PCK for teaching algorithms, Goals, Instructional Strategies, Students’ Understanding, and Assessment. One interesting finding is related to the teachers’ reports of the instructional strategies they use; the teachers mentioned several activities used to help students comprehend specific concepts, including “teacher discusses solutions with the class” and “students think, reflect, and evaluate”. The teachers emphasized that whole-class discussions about alternative solutions for algorithmic problems encouraged students to think and reflect on the different solutions. Indeed, teaching students to compare various algorithms and reflect on their differences supports their comprehension of the logical steps of algorithms. This finding is consistent with the conclusions of Lye and Koh (2014), who examined the intervention approaches used to foster CT. Lye and Koh (2014) found that reflection is one of the strategies that may advance computational practices, such as problem-solving activities. However, most of the studies reviewed by Lye and Koh (2014) that used this approach were studies of higher education. In the K–12 studies reviewed, reflection was seldom reported. The findings of our study suggest that it is worthwhile to use reflection in upper secondary education, as reviewing and reflecting on solutions for algorithmic problems might foster algorithmic thinking.

Moreover, the reflection is not carried out by students individually, but as a class activity in which students collaborate. The teachers also mentioned that they encourage their students to cooperate when solving algorithmic problems because “through cooperation, you will find a better solution”. This is in line with the findings of Sentance and Csizmadia (2017), who reported the successful strategies used by teachers in computing education. “Collaborative working” was one of the strategies they found and teachers in their study also commented on the “positive motivational impact” of collaborative working (Sentance & Csizmadia, 2017, p. 486). Yet, our findings also point to challenges related to student collaboration, as teachers face the task of composing well-balanced groups.
Students have different analytical skills, posing a potential challenge for teachers to organize collaboration so that all students are active and learn from each other.

The analysis of teachers’ knowledge about students’ understanding revealed that a student’s analytical skills are viewed as a helpful prerequisite for learning algorithms. The teachers mentioned that students with stronger analytical skills did not seem to encounter problems, contrary to students with less developed analytical skills for whom more support is required to help them understand how to solve algorithmic problems. According to the teachers, students enjoy the cognitive satisfaction of solving algorithmic problems or puzzles. This might be related to the “in-the-moment enjoyment” mentioned by Black et al. (2013) to make computing more interesting to students.

It is also interesting to note that “solving puzzles” was mentioned as “easy to learn” for students, and also as a topic that students enjoy. These results are in line with those of previous studies, e.g. the study of Futschek and Moschitz (2010), who recommended an approach for teaching algorithms that is both enjoyable and contributes to fast learning progress. Likewise, students mentioned “concrete problems” as being “easy to learn” and described enjoying “understanding how it works”. These results provide further support for the use of relevant and recognizable contexts for teaching algorithms (Nijenhuis-Voogt et al., 2021).

The findings regarding goals for teaching algorithms showed that teachers, when teaching algorithms, do not only aim for developing students’ algorithmic thinking. During the interviews, the teachers mentioned several goals, and most of them corresponded well with abilities connected to algorithmic thinking (Futschek, 2006), which we discussed with the teachers at the beginning of the interviews. However, the teachers also mentioned enhancing transversal skills, e.g. learning perseverance, as learning goals for teaching algorithms. Corradini et al. (2017) also mentioned transversal skills as an aspect of CT, and listed several skills such as creating, communicating, and being persistent when dealing with complex problems. As discussed in the conceptual framework section, transversal skills may be seen as a prerequisite to teaching algorithms (Barr & Stephenson, 2011), a result of teaching algorithms, or both (Sentance & Csizmadia, 2017). It is interesting to note that our findings suggest that teachers, when teaching algorithms, aim to develop students’ transversal skills, such as learning perseverance and learning to apply genuine effort in your work, instead of viewing these skills as prerequisite.

Implications, limitations, and future work

The present study suggests a pattern in teachers’ goals for teaching algorithms related to a hierarchy of abstraction levels for the algorithmic concept. At the same time, this pattern seems to refer to a well-known difference of opinion concerning the scope of CT. These findings may help us to understand the different views on goals for teaching algorithms, which may contribute to more well-reasoned discussions on these views in, for example, teacher education. In addition, our results may provoke reflection during professional development activities because they may help in-service teachers to reconsider the alignment of their goals and instructional strategies.

The findings of this study provide insight into teachers’ PCK for teaching algorithms in upper secondary education. The results reveal important recommendations for improving
teacher education and supporting teacher professionalization regarding teaching algorithms specifically and CS in general. This is especially relevant in the Netherlands, where teaching algorithms is a more prominent topic in the revised curriculum; however, the recommendations of this study might also contribute to teacher education and professionalization in other countries. Although this study focuses on teaching algorithms in upper secondary education, the findings may well have a bearing on teaching algorithms in higher education as well.

The small sample of participating teachers, which is a feature of qualitative studies, might be seen as a limitation of the study; however, it provided an opportunity to gather surprisingly rich data, as teaching algorithms was not specifically mentioned in the then-current curriculum. By delving deeper into the teachers’ knowledge, we were able to find remarkable patterns that may serve as a framework for future studies with a much larger sample. Furthermore, our sample may not be representative of the whole CS teacher population as we intentionally selected teachers with experience in teaching algorithms before this topic was included in the curriculum. Within the sample, there was a wide variety of obtained teacher education. This may have diversified the results as different teacher training may result in more variety in knowledge. Our data is not conclusive about a relation in teachers’ education and their reported PCK. Further research could usefully explore the effect of teachers’ education on their PCK.

During the data analysis process, we started with a deductive analysis with the use of the PCK model of Magnusson et al. (1999). This often used method of coding to capture teachers’ PCK may have influenced the inductive analysis that was carried out within the framework of the deductive analysis.

Our study serves as a good basis on which to design a quantitative or mixed study on a larger scale.

**Conclusion**

In this study, we elicited and described teachers’ PCK for teaching algorithms in order to contribute to the education of new CS teachers and professional development activities for in-service teachers. We have been able to identify a number of themes emerging from our data, including the finding that teachers’ focus on goals when teaching algorithms may be related to the abstraction level of their view on algorithms. In addition, our research pointed to the “generic” or “topic-specific” nature of knowledge regarding the responses to student differences, and we suggest that teachers need both types of knowledge. This illustrates that PCK is a blending of content and pedagogy (Shulman, 1987); when facing the challenge of meeting the needs of all students while teaching algorithms, both content knowledge and knowledge of pedagogy appear to be especially significant.

Our results point to the importance of using reflection as an instructional strategy, as well as the significance of high-quality teaching materials with attention paid to increasing the difficulty of successive problems. Teachers described that lessons should be relevant, as students like to see where they can apply their knowledge. This may be especially important for an abstract topic, such as (the efficiency and correctness of) algorithms.

We found a wide variety in the use of PCK for teaching algorithms. This diversity might enable teachers to apply their knowledge to match the level of their students and
respond to student differences. We hope that teacher education may benefit from these insights because teachers may be supported with a wide range of knowledge that can be applied whenever it is beneficial.

Acknowledgments

We would like to thank all teachers who participated in this study. This research received funding from the Dutch Ministry of Education, Culture and Science under the Dudoc program.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the Dutch Ministry of Education, Culture and Science under the Dudoc programme.

Notes on contributors

Jacqueline Nijenhuis-Voogt is a PhD candidate at the Institute for Science Education at Radboud University (Nijmegen, the Netherlands) and a computer science teacher in secondary education at GSG Guido (Amersfoort, the Netherlands). Her research focuses on computer science education, specifically on teaching algorithms and algorithmic thinking and on the use of a context-based approach for CS education.

Durdane Bayram-Jacobs is an assistant professor at the Eindhoven School of Education (ESoE), Eindhoven University of Technology (TU/e). Her research interests are: Socio-scientific Issues (SSI), science education for citizenship, Pedagogical Content Knowledge (PCK) of science teachers, formative evaluation of SSI lessons, and innovation in education.

Paulien C. Meijer is a professor of Teacher Learning and Development at the Radboud Teachers Academy at Radboud University (Nijmegen, the Netherlands). Her research focuses on teacher education, the development of teachers’ professional identity, workplace learning and teaching for creative learning.

Erik Barendsen is a professor of Science Education at Radboud University (Nijmegen, the Netherlands) and a professor of Computing Education at Open University of the Netherlands. His scientific interests include design-based and context-based teaching and learning in computer science and STEM subjects, computational thinking and its integration into the school curriculum, digital literacy, and teachers’ practical knowledge, in particular Pedagogical Content Knowledge (PCK).

ORCID

Jacqueline Nijenhuis-Voogt http://orcid.org/0000-0001-5156-0022
Durdane Bayram-Jacobs http://orcid.org/0000-0001-9502-1928
References


Hromkovič, J., & Lacher, R. (2017). The computer science way of thinking in human history and consequences for the design of computer science curricula. In V. Dagiene & A. Hellas (Eds.), *Informatics in schools: Focus on learning programming* (pp. 3–11). Springer International Publishing.


