JOURNAL OF LEARNING ANALYTICS

SOLAR SOCIETY for LEARNING ANALYTICS RESEARCH

Volume 9(2), 66–86. https://doi.org/10.18608/jla.2022.7363

Using HeuristicsMiner to Analyze Problem-Solving Processes: Exemplary Use Case of a Productive-Failure Study

Christian Hartmann[1], Nikol Rummel[2], and Maria Bannert[3]

Abstract

This paper presents a fine-grained process analysis of 22 students in a classroom-based learning setting. The students engaged (and failed) in problem-solving attempts prior to instruction (i.e., the Productive-Failure approach). We used the HeuristicsMiner algorithm to analyze the data of a quasi-experimental study. The applied algorithm allowed us to investigate temporally structured think-aloud data, to outline productive and unproductive problem-solving strategies. Our analyses and findings demonstrated that HeuristicsMiner enables researchers to effectively mine problem-solving processes and sequences, even for smaller sample sizes, which cannot be done with traditional coding-and-counting strategies. The limitations of the algorithm, as well as further implications for educational research and practice, are also discussed.

Notes for Practice

- The presented learning analytics tool (i.e., the HeuristicsMiner algorithm) is applicable for small samples (e.g., for analyses at the classroom level).
- The information mined by the HeuristicsMiner algorithm can be directly used for educational practice (e.g., to script sequences of effective learning activities).
- The analyses presented here identify productive and unproductive problem-solving strategies, which help students to prepare effectively for future learning (e.g., in later instruction).

Keywords

Productive-Failure approach, classroom setting, process mining, HeuristicsMiner, problem-solving strategies

Submitted: 20/09/2021—**Accepted:** 05/04/2022—**Published:** 09/08/2022

Corresponding author [1] Email: christian.hartmann@tum.de Address: Technical University of Munich, Department of Educational Sciences, Arcisstrasse 21, 80333, Munich, Germany. ORCID ID: https://orcid.org/0000-0003-3109-1104

[2] Email: nikol.rummel@rub.de Address: Ruhr University Bochum, Institute of Education Science, Universitätsstrasse 150, D-44801, Bochum, Germany. ORCID ID: https://orcid.org/0000-0002-3187-5534

[3] Email: maria.bannert@tum.de Address: Technical University of Munich, Department of Educational Sciences, Arcisstrasse 21, 80333, Munich, Germany. ORCID ID: https://orcid.org/0000-0001-7045-2764

1. Introduction

In the learning sciences, successes in problem solving and learning are often explained in terms of specific activities. For instance, in problem-based learning (see Hmelo-Silver, 2004, for an overview), it is assumed that learners benefit from actively discovering learning content within authentic problem situations. In research on collaborative learning (see Kreijns et al., 2003, for an overview), it is assumed that learning is supported by interactive activities, such as explanations. If students engage in self-regulated learning (see Panadero, 2017, for an overview), then it is hypothesized that metacognitive activities, such as monitoring their learning processes, are needed to learn successfully. During the past several decades, educational researchers have become increasingly interested in exploring essential learners' activities to better understand why certain pedagogical methods lead to better learning and problem-solving outcomes, as well as the fact that digital learning environments serve as extended data sources (e.g., as log files that can be extracted from student learning) and due to the increasing possibilities of innovative analysis methods (cf. Martin & Sherin, 2013).

To analyze problem-solving and learning activities, classroom-based educational research often relies on data from student think-aloud recordings; specifically, students will express what they think while engaging in problem solving (see Young,

2005, for an introduction and discussion of think-aloud methods). Think-aloud data are highly important for investigating problem solving and learning because they provide adequate access to student intentions, thoughts, and actions throughout their problem-solving and learning processes. As a continuation of frequent application, think-aloud data have recently been used to investigate elementary students' self-regulated learning activities (Heirweg et al., 2019), to explore how children interact with a social robot tutor (Ramachandran et al., 2018) or to examine how secondary school students read texts (Rogiers et al., 2020). A well-established approach to analyzing think-aloud data (also described in the previously mentioned studies) is coding-and-counting strategies, wherein researchers use audio recordings of student think-alouds to code and analyze the frequencies of shown thoughts and/or behaviour.

Despite the wide use of coding-and-counting strategies in the learning sciences, however, there is critical debate about whether they are an adequate method for analyzing think-aloud data (see Reimann, 2009). This debate is not reduced to the methodological limitations of coding-and-counting strategies; research is also becoming increasingly concerned with the temporal nature of theoretical approaches to explaining problem solving and learning (for the case of research on self-regulated learning, see Molenaar & Järvelä, 2014). In other words, problem solving and learning are understood as a series of closely intertwined activities. The extent to which certain processes (specifically, patterns consisting of many different sequential activities) determine the success of learning and problem-solving processes is gaining importance in research on learning science as they provide a more fine-grained explanation for successful or unsuccessful problem-solving and learning activities. However, applying coding-and-counting strategies (i.e., extracting only the frequencies of a set of learning activities from data like think-aloud data) does not sufficiently address the temporal nature of problem-solving and learning processes. Csanadi et al. (2018) argue that coding-and-counting strategies do not account for specific relations between coded thoughts and/or behaviour; specifically, they argue that the richness of think-aloud data is significantly reduced. For example, while a student who thinks aloud expresses his or her thoughts in temporally structured (and possibly related) sequences, this process is not represented by aggregated frequency counts. For some research questions, the temporal structure of think-aloud data does not seem to be essential. For instance, a study by Neuman and Schwarz (1998) showed that university students' problem-solving performance was improved by only three out of four types of self-explanations. However, if researchers are interested in how certain activities (e.g., self-explanations) occur and correspondingly depend on each other during the problem-solving process, the temporal structure of think-aloud data significantly gains importance.

In this paper, we analyzed process data from a Productive-Failure study as a use case. Productive-Failure describes a teaching approach (for a more detailed description, see section 2) in which learners first attempt to solve a problem on their own and fail in solving the problem correctly. It is assumed that failing in problem solving effectively prepares learners for subsequent instruction, wherein the correct problem solution is taught. We analyzed the process data from the Productive-Failure study to demonstrate that student problem-solving activities are involved in a temporally structured process. Additionally, we aimed to show how this process can be effectively analyzed using the HeuristicsMiner algorithm to take full advantage of the richness of think-aloud data (e.g., for building more fine-grained theories). Using the definition of the Society for Learning Analytics Research (SoLAR), we define learning analytics as "the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environment in which it occurs" (Siemens et al., 2011, p. 3). We applied a learning analytics tool, the HeuristicsMiner algorithm (Weijters et al., 2006), to analyze the temporal nature of think-aloud recordings. For example, the HeuristicsMiner algorithm has already been used to analyze self-regulated learning (Bannert et al., 2014), computer-supported collaboration (Sobocinski et al., 2017), and organizational processes (Bingham & Haleblian, 2012). In the analyses presented in this paper, we illustrate how HeuristicsMiner can be used to investigate the temporal nature of student problem-solving and failing processes with think-aloud data from a classroom-based quasi-experiment with secondary school students (for a description of the original study, see Hartmann et al., 2020). We demonstrate and discuss that an important advantage of HeuristicsMiner lies in the fact that the algorithm produces highly enriched data, which is also beneficial for small sample sizes (e.g., classroom environments). As the analyses reported in this paper will demonstrate, learning analytics has a high potential for small-scale and classroom-based research; specifically, it sufficiently addresses the temporality problem-solving and learning processes of students. We illustrate (in the context of the Productive-Failure approach as an exemplary use case) how learning analytics makes theory building and data exploration more effective and how learning analytics reveals possibilities for supporting student learning processes in educational practice.

2. Theoretical Background

In the previous section, we introduced why an analysis of the temporal nature of problem-solving and learning processes contributes to research on learning science. In this section, we further illustrate the advantages of analyzing problem-solving

ISSN 1929-7750 (online).

and learning processes over coding-and-counting strategies with the example of the Productive-Failure approach. Therefore, we first introduce the Productive-Failure approach, as well as essential research on the approach. Subsequently, we argue that the theory that underlies the Productive-Failure approach explains the effectiveness of problem solving prior to instruction by temporally dependent processes. The Productive-Failure approach is particularly interesting as an exemplary use case because (to the best of our knowledge) existing research on Productive-Failure has not yet sufficiently addressed these temporal characteristics of theorized processes; therefore, the process analysis presented in this paper also contributes to research on Productive-Failure. The following sections address this lack of research by assessing and analyzing the postulated problem-solving processes using learning analytics.

2.1. Problem Solving Prior to Instruction: The Productive-Failure Approach

Research on the Productive-Failure approach is subject to a far-reaching discourse within educational research (cf. Koedinger & Aleven, 2007), in which the question is asked as to whether students should receive explanations early in the learning process (i.e., via direct-instruction approaches) or if they should be engaged in their own problem-solving activities without immediate guidance (i.e., via problem-based learning). In the context of this discourse, Productive-Failure describes a learning approach in which students attempt to solve a problem before they are taught about the correct problem solution (see Kapur, 2012). If students attempt to solve a problem (which they cannot yet solve correctly), then they most likely generate multiple intuitive solution ideas. However, it is unlikely that they will arrive at the correct solution, or that they will completely solve the underlying problem through their (failed) attempts. The effectiveness of the Productive-Failure approach has been proven in numerous studies (for an overview, see Loibl et al., 2017): Studies have found that experiencing failures in problem solving prior to instruction supports students in conceptually understanding the correct problem solution more effectively than students who have first been taught about the correct solution and apply it to a problem (e.g., DeCaro & Rittle-Johnson, 2012; Kapur, 2012, 2014; Loibl & Rummel, 2014a, 2014b; for an overview, see Loibl et al., 2017).

From these studies, Loibl et al. (2017) and Kapur (2014) have derived the following three mechanisms to explain why engaging in (failed) problem-solving attempts effectively prepares students for the acquisition of conceptual knowledge from subsequent instruction:

1) the activation of prior knowledge
2) the awareness of knowledge gaps
3) the identification of deep structures in the target concept

Loibl et al. (2017) described these mechanisms as a consecutive process; specifically, mechanisms that build upon the other. Because students have no specific knowledge about the correct problem-solving strategy, they must activate their prior knowledge to generate intuitive solution ideas. Additionally, because students mostly generate different solution ideas (some studies have found this to be more effective, e.g., Kapur, 2014), it is assumed that, in addition to the activation of prior knowledge, students also develop different conceptualizations or representations of the problem and (at least some) of the deep features of the correct solution. Loibl and Rummel (2014a) further argued that, by activating previous knowledge and differentiating (e.g., comparing) between possible solutions, students recognize that their previous knowledge is insufficient to solve the problem (i.e., they develop an awareness of knowledge gaps). If the student solution attempts are then considered in the following instruction (cf. Loibl & Rummel, 2014a), it is assumed that students will better understand the deep features of correct problem solutions because they can lean on their prepared knowledge gaps, as well as their already activated and modified mental models. These explanation approaches highlight the temporal character of the problem-solving-and-failure processes; specifically, the aforementioned problem-solving activities most likely occur in certain sequences (in a temporal order) and most likely depend on each other.

2.2. Investigating the Productive-Failure Problem-Solving Process

Even if the effectiveness of the Productive-Failure approach is empirically supported (e.g., Kapur, 2014; Loibl & Rummel, 2014a, 2014b) and theoretically well founded (see Loibl et al., 2017), it remains unclear as to whether these mechanisms can actually explain the effectiveness of the approach. As we will argue in the following section, the described mechanisms most likely occur during a temporally sequenced problem-solving process not yet explored in detail and possibly not sufficiently addressed by coding-and-counting strategies. Consequently, there is little evidence that (meta-)cognitive, motivational, or emotional processes indeed occur throughout the problem-solving process and whether these processes may explain the effectiveness of the Productive-Failure approach. Based on central research on (mathematical) problem solving, in the following sections, we first describe what problem-solving activities can be expected if students attempt to solve a problem prior to being taught about the correct solution.

ISSN 1929-7750 (online).

In most studies on the Productive-Failure approach (e.g., Loibl & Rummel, 2014a, 2014b; Kapur, 2014), the utilized mathematical problem was embedded in a story about soccer. The students were shown a dataset with the number of goals scored by three soccer players over a ten-year period and were asked to identify the most reliable soccer player. In this scenario, the goal state would be a mathematical value (or a graphical solution) representing how reliable the three soccer players are. Adequate problem-solving strategies would be the formula for standard deviation. As the Productive-Failure approach predicts, students with no specific prior knowledge about standard deviation (or similar mathematical approaches) formulate inadequate problem-solving strategies to correctly solve the problem. Even if the goal state is clearly defined, the lack of prior knowledge makes it necessary for the students to first define the goal state for themselves (i.e., a mathematical representation of reliability). Even if some students may have a clear representation of the goal state at the beginning of the problem-solving process, due to a lack of prior knowledge, it is likely that most students first (and throughout the problem-solving process) must define the problem's goal state to identify suitable strategies (i.e., their solution attempts). A possible strategy would be to reduce the selection of possible problem-solving strategies by establishing concrete evaluation criteria for the goal state and by continuously evaluating their suitability (Schoenfeld, 1985, 2016). However, because of a lack of prior knowledge, students cannot be sure of the validity of their solution attempts and their defined goal states before being instructed. Therefore, it seems plausible that the problem-solving process is characterized by conclusions, comparisons between solutions, formulation of expectations, and the continuous monitoring, evaluation, and reorientation of student problem-solving attempts, as well as redefining the goal state. A study by Granberg (2016) provides further support for this assumption.

Granberg (2016) analyzed failed student problem-solving processes for mathematical problems. She described student problem-solving processes as being chronological and cyclical. Ideally, during the problem-solving process, students first read and analyze the problem definition to activate prior knowledge, after which they evaluate their prior knowledge and, if required, build new knowledge. This modified knowledge can then be used to plan and execute further problem-solving steps, and this cycle of problem-solving activities continues until the problem is solved. She further categorized student behaviours using 24 student conversation records, computer activities, and interviews. She also identified productive and unproductive problem-solving sequences. In productive problem-solving sequences, students cyclically went through the previously mentioned problem-solving steps. Problem-solving sequences were unproductive if the students often switched between activities; specifically, they were unproductive if the sequences did not build on each other and if the student superficially passed through the sequences. Most importantly, Granberg (2016) described student problem-solving-and-failing activities as being temporally sequenced processes.

Even though process analyses of a Productive-Failure condition by Brand et al. (2018) showed that the problem-solving-and-failing activities assumed by Granberg (2016) did occur during the problem-solving process measured, there is still little empirical work regarding the assessment and analysis of concrete processes during the problem-solving phase. In addition, Brand et al. (2018) analyzed absolute (and relative) frequencies of student problem-solving activities, but they did not analyze the temporal structure; specifically, they did not analyze how the activities occurred during problem solving and whether they depended on each other. By these models of problem solving, and according to the theory of the Productive-Failure approach (see Loibl et al., 2017), we assume that the temporal structure of the data and the dependencies between problem-solving activities are highly relevant in explaining the effectiveness underlying problem-solving-and-failing prior to instruction. This issue has also been reflected in the context of self-regulated learning (cf. Azevedo, 2014; Bannert et al., 2014; Reimann et al., 2014; Molenaar & Järvelä, 2014; Winne, 2014; Van Laer & Elen, 2018). Most recently, it has been shown that temporal sequences of multimodal data assessed in different learning contexts may help to better explain student learning performance (Järvelä & Bannert, 2021).

3. The Present Process Analysis

In the process analysis presented in this paper, we demonstrate how learning analytics (compared to coding-and-counting strategies) can be used to better address the temporal characteristics of problem-solving and learning processes. As an initial exemplary use case, we explored data from a Productive-Failure study. We analyzed effective (and ineffective) problem-solving processes of secondary school students in a classroom context using the HeuristicsMiner algorithm. This exploration is linked to existing theory and explanations of the effectiveness of the Productive-Failure approach and aims to further develop this theory by demonstrating how the HeuristicsMiner algorithm can be effectively applied to explore think-aloud data. Because the Productive-Failure problem-solving process has not yet been sufficiently investigated, in addition to the application of the HeuristicsMiner algorithm, the present study further contributes to research on the Productive-Failure approach using a fine-grained process exploration. We explored the problem-solving processes of 22 students who engaged in 45 minutes of problem solving prior to instruction. For our analyses, we used data from a quasi-experimental study by Hartmann et al. (2020), in

ISSN 1929-7750 (online).

which the authors investigated whether students need to generate their own solution attempts to be effectively prepared for a subsequent instruction or whether observing another student in generating problem-solving attempts would also be effective (cf. "Vicarious Failure"; Kapur, 2014; Hartmann et al., 2021). The process analysis presented in this paper is an important extension of the analyses by Hartmann et al. (2020) and has not yet been published; specifically, we used the video and audio recordings made during the quasi-experiment to gain detailed insights into student problem-solving processes. As our study is explorative (i.e., not explanative), we did not formulate any hypotheses about potential processes mined by HeuristicsMiner or the impact of mined problem-solving activities on student learning outcomes.

4. Methods

4.1. Sample

Hartmann et al. (2020) conducted this study in secondary schools in Germany, and they implemented four conditions. The condition considered for this paper (i.e., Productive-Failure) consisted of 24 students. Because two of the 24 students had enhanced mathematical abilities, prior knowledge, and extraordinary performance on the conceptual knowledge post-test, we removed them from our analysis as outliers. Our final sample, therefore, consisted of $N = 22$ students (11 female and 11 male; median age 16.26; $SD = 0.73$). These students came from the same school and participated during their free time after class (not during their regular lessons).

4.2. Procedure

In the quasi-experimental study by Hartmann et al. (2020), the students went through four phases. First, students completed a 20-minute pre-test before the 45-minute problem-solving phase conducted in a normal classroom after regular classes. As they worked on the problem, students were supervised by a teacher from the school and the two experimenters. Students were instructed to generate as many solution attempts for the mathematical problem as possible by using a blank paint sheet on a tablet PC. They were instructed that it was fine if they could not come up with the solution right away and that they should keep trying. While students worked on the mathematical problem-solving task, neither the teacher nor the experimenters instructed the students on the target concept nor provided any type of support on problem-solving strategies. While students worked on the problem, all handwritten solution attempts were screen recorded, including any notes made. In addition, students were asked to think aloud while problem solving; specifically, they were told to verbalize any thoughts while problem solving. The students were set far apart so that they could not hear each other and student voices were recorded with dictation devices. In addition to screen and audio recordings, we conducted no additional log data. Before starting the problem solving, the experimenters introduced the think-aloud method throughout the student problem-solving process and labelled paper cards reminded the students to express their thoughts. After the problem-solving phase, an experimenter taught the students for 45 minutes about the correct solution for the task. This instruction was similar to that used by Loibl and Rummel (2014a, 2014b), wherein students were taught about the correct solution for the problem-solving task by contrasting the correct solution to their own attempts. Immediately after instruction, students completed a conceptual knowledge post-test; specifically, the students worked on tasks that required them to explain the components of the correct problem solution.

4.3. Materials and Measures

4.3.1 Problem-Solving Task

Hartmann et al. (2020) used a mathematical problem-solving task adopted from Kapur (2014) and Loibl and Rummel (2014a, 2014b). The task utilized the concept of variance by using a soccer example; specifically, Hartmann et al. (2020) instructed students to identify the most consistent of three soccer players using their number of goals scored over ten years. Students who attempted to solve this problem generated solution attempts that differed in how many of the four components of the canonical solution were taught during instruction (for a more detailed description, see Loibl & Rummel, 2014a). A suitable solution strategy for this problem (e.g., standard deviation) would consider that 1) deviations must be summed up for all data points, 2) negative differences cancel each other out, 3) differences must be calculated from a fixed reference point, and 4) dividing by the number of data points would increase the comparability of results. Since students had not been previously instructed on the best solution strategy, they would mostly fail to solve the problem correctly, even if they could produce diverse solutions.

4.3.2 Mathematical Ability and Prior Knowledge (Pre-test)

Hartmann et al. (2020) assessed student mathematical ability by using the two of their most recent grades in mathematics. The prior knowledge levels of the students were assessed with a pre-test, which included six items that required students to interpret and draw graphs, to apply their knowledge of descriptive statistics, and to draw and interpret a boxplot. Within the pre-test, Hartmann et al. (2020) also checked if students already knew the correct problem solution. Note that the pre-test gauged

relevant knowledge and related concepts, rather than specific knowledge about the targeted concept.

4.3.3 Conceptual Knowledge (Post-test)

The post-test used by Hartmann et al. (2020) assessed the conceptual knowledge with four items. The items asked the students to explain graphical representations, to sort datasets regarding their consistency, and to identify and explain typically made errors (mean and range, three points). Hartmann et al. (2020) and Loibl and Rummel (2014a) provide more detailed information and statistics on the conceptual knowledge post-test (e.g., the internal consistency of items).

4.3.4 Quantity and Quality of Solution Attempts (Process Coding)

To assess the quantity of solutions generated by students, Hartmann et al. (2020) counted the number of different solution attempts and assessed their quality by assigning each solution a score ranging from zero (none of the correct solution components) to four (all four correct components). Hartmann et al. (2020) then used the solution with the highest number of included components to evaluate the overall quality of all student solution attempts.

4.3.5 Problem-Solving Activities (Process Coding)

Think-aloud and screen recordings were used to gain more detailed information about students' problem-solving processes. We applied a coding manual that included seven (disjunctive) problem-solving activities derived from the Productive-Failure approach (e.g., Loibl et al., 2017; described in section 2.1), a model of mathematical problem solving by Schoenfeld (1985) (described in section 2.2), and an analysis of our learning environment (e.g., the problem-solving task). Schoenfeld (1985) describes mathematical problem solving as a process in which problem solvers first plan their approach, explore possible solutions and conditions for success, then implement, and finally evaluate their attempted solutions. Based on this theoretical model adapted to our learning setting, we argue that the seven problem-solving activities included in the coding manual are essential parts of the student problem-solving-and-failing process. Table 1 shows the coding manual, specifically, the seven codes, including a description and examples for each from student think-aloud protocols.

The seven problem-solving activities of the 22 students were first coded by one rater who watched and listened to their video and audio recordings. Once the rater was able to assign a student's statement to one of the seven categories, the timestamps (i.e., the start and end time of an activity in mm:ss) for this problem-solving activity were registered, as well as the identified activity assigned. The coder was instructed to assign only one category to a single timestamp so that multiple coding was avoided and — while coding — had access to all the think-aloud protocols such as the raw audio data. After the first round of coding, the timestamps identified by the first rater were recoded to assess interrater reliability. The second rater recoded seven of the 22 student recordings consisting of 242 timestamps. A total of 214 of the 242 timestamps (88.43%) were rated by the two raters with identical problem-solving activities. Accordingly, the raters disagreed on 28 timestamps (11.57%). In addition, a Cohen's κ (unweighted) test was performed to determine if there was sufficient agreement between the two raters. There was substantial agreement, with $\kappa = .86$ (95% CI, .81 to .91), $p < .001$. Both coders reported no difficulty applying specific categories of the coding manual to the data. In addition, both coders noted that the audio data had good audio quality and participants expressed their ideas clearly.

4.4. Process Analysis Using HeuristicsMiner

We analyzed student problem-solving processes by applying the HeuristicsMiner algorithm (cf. Sonnenberg & Bannert, 2015, 2018), which is based on petri nets and hidden Markov models. The HeuristicsMiner algorithm extracts the order of coded activities (in the present study, student problem-solving activities are shown in Table 1); specifically, in what sequences (also referred to as "dependencies") the seven problem-solving activities occurred (in relation to another activity) during the problem-solving process. For instance, it is reasonable to assume that students first execute a solution attempt [3] and then continue with drawing a substantiated conclusion [5] based on the data produced by their calculations. However, it is also possible that students finish the execution of their solutions by drawing unsubstantiated conclusions [4] or by alternately executing and drawing conclusions.

In particular, the algorithm extracts dependency values (ranging from –1 and 1), thus indicating a causal relation between two activities (e.g., for the assumption that activity B follows [⇒] activity A). A dependency score of 1 indicates that activity B always follows activity A (as expected), and a score of –1 indicates that activity A always follows B (contrary to the assumption).

ISSN 1929-7750 (online).

Table 1. Coding Manual Used to Assess Student Problem-Solving Activities

Code	Description	Examples
[1] Description of the solution	The student names and/or describes a possible solution attempt (e.g., mean value or range) before executing the solution attempt.	*One solution approach would be to calculate the mean!* *To calculate the mean, I first must add up all values and then…!*
[2] Analysis of the goal state	The student uses the problem-solving task to analyze (and understand) the goal state of the problem (i.e., the concept of reliability). The student may (repeatedly) read the task materials and/or define general criteria for reliable data. Based on the student definition of the goal state, he or she may also express expectations, prognoses, or hypotheses concerning the results of a not yet executed solution attempt.	*Now I must read the task description again to see if I have thought about everything!* *I think consistency of data means that…!*
[3] Execution of the solution	The student executes a solution attempt; specifically, they write down their calculations. The student may also describe what they are doing during the solution attempt; however, this only refers to what they are doing in the moment of execution (not a general description of the solution attempt).	*I now sum up all values and divide the sum by the number of years!* [participant writes down numbers]
[4] Unsubstantiated conclusion	The student draws a conclusion from the results of the solution approach, however, without substantiated reasoning. For instance, the student may select the most reliable player (without arguing why the player is most reliable) or simply describe the results of the solution attempt. The student may compare solution attempts concerning their results, but only descriptively, without further argumentation. For instance, the student may only stress that the same player was selected as the most reliable in the two solutions (without arguing what this means for their conclusion).	*Here you can see that the number of goals scored by the three soccer players fluctuates!* [no reason added] *Player A is the most consistent player here!* [no reason added]
[5] Substantiated conclusion	The student formulates a substantiated conclusion by analyzing the results of the solution attempt using data generated by the solution attempt to decide whether a player is more or less reliable. The student might support their conclusion by highlighting central similarities or differences between solution approaches to argue why the results support their conclusion.	*The values of Player A fluctuate less, so he is the most consistent!* *In this solution and my previous solution, the goals scored by Player A fluctuate the most and this shows that he is less reliable, compared to the others!*
[6] Unsubstantiated evaluation	The student evaluates the solution approach, but without explicit reasoning. The student may express irritation during (or after) implementing the solution approach, without giving reasons. However, it remains unclear what exactly the student is evaluating and the evaluation remains unsubstantiated. For instance, the student indicates that a certain solution approach does (or does not) work, but does not elaborate on this more precisely.	*I do not know if this is right!* [no reason added] *Something is weird with this solution!* [no reason added]
[7] Substantiated evaluation	The student expresses irritation with the generated solution. The student evaluates the quality of the solution attempt. The student reflects on whether a solution does (or does not) work and can argue explicitly (at least in part) why.	*Calculating the range does not work, because it includes only the extreme values, but ignores the distribution of all remaining values!*

The further the dependency score approaches zero, the less both activities depend on each other. The dependency score can be interpreted as the probability of how frequently activities occur in a sequence. In addition to direct sequences (e.g., A⇒B), HeuristicsMiner also takes into account long-distance relations (e.g., A⇒B⇒C⇒A⇒B⇒C⇒A⇒B⇒C), as well as length-one loops (e.g., A⇒C⇒B or A⇒C⇒C⇒B or A⇒C⇒C⇒C⇒B) and length-two loops (e.g., A⇒C⇒D⇒B or A⇒C⇒D⇒C⇒D⇒B or A⇒C⇒D⇒C⇒D⇒C⇒D⇒B).

In addition, the algorithm considers AND-relations (sequences that occur together) and OR-relations (sequences that exclude each other) to extract the dependency scores. An important criterion for the selection of the heuristic miner was how the algorithm handles "noise" (all activities that an algorithm extracts from the data that do not correspond to the actual

ISSN 1929-7750 (online).

behaviour shown). When analyzing problem-solving activities, noise can occur at various stages of data analysis, such as in the coding process itself or in the way the algorithm uses data to build a model. In selecting the algorithm for our use case, the key question was how the algorithm handles activities that are not frequently coded or occurred frequently but unsystematically. For example, learners may rarely perform important activities when solving problems, so they could be overlooked by the algorithm. The problem with very frequently performed behaviour is that relationships to other activities are more likely and possible relations therefore might be overestimated by the algorithm. Both of these together can lead to noise that can critically bias the extracted model and must be addressed by the algorithm. Compared to alternative algorithms such as Alpha Miner, HeuristicsMiner (a developed version of the Alpha algorithm) considers less frequent and frequently occurring activities in a robust way. For example, while the Alpha algorithm only extracts transitions between occurring activities as a workflow net (regardless of how frequently activities occur), HeuristicsMiner considers both frequencies as an extraction criterion and recurring, repetitive patterns (e.g., loops). This allows for the identification of systematic patterns related to less frequent activities as well as higher-level patterns of frequently recurring activities.

5. Results

5.1. Students' Individual Prerequisites, Performance, and Problem-Solving Activities

The 22 students indicated an average of $M = 3.91$ ($SD = 0.73$) for mathematical ability (lowest grade 1, highest grade 6) and scored $M = 4.32$ ($SD = 1.19$) on the prior knowledge pre-test (0 to 10 points). The students invented $M = 4.41$ ($SD = 1.26$) solution attempts; the best solution attempt included, on average, $M = 1.77$ ($SD = 0.97$) components of the correct solution, as taught during subsequent instruction (four components). The students reached $M = 3.00$ ($SD = 1.22$) points on the conceptual knowledge post-test, which was given directly after instruction. As shown in Table 2, there were no significant correlations between these variables. Accordingly, neither the students' individual prerequisites (their mathematical abilities and relevant prior knowledge) nor the assessed measures of their problem-solving performance (solution quantity and quality) were associated with conceptual knowledge acquisition from the instruction after problem solving.

Table 2. Pearson Correlations of Students' Individual Prerequisites and Performance (including 90% confidence intervals)

$N = 22$	Mathematical ability	Prior knowledge	Number of solutions	Quality of solutions
Prior knowledge	-.31 [-.60,.06]			
Number of solutions	.33 [-.04,.61]	.18 [-.19,.51]		
Quality of solutions	.17 [-.20,.50]	-.10 [-.44,.27]	.04 [-.33,.40]	
Conceptual knowledge	.25 [-.12,.56]	.29 [-.08,.59]	-.06 [-.41,.31]	.06 [-.31,.41]

To understand how student problem-solving activities effectively prepared them to gain conceptual knowledge from later instruction, we further explored student problem-solving processes; specifically, we explored the seven coded activities based on student think-aloud recordings (see Table 2). These recordings were 40 minutes and 37 seconds on average (we provided 45 minutes for problem solving). We coded M = 27.45 activities of the 22 students, which took (on average) 32 minutes and 9 seconds of the entire problem-solving process. The remaining 8 minutes and 28 seconds could not be allocated to the coded activities because students were silent or engaged in off-task behaviour. Taken together, our coding procedure covered most of the student think-aloud recordings.

Table 3 shows the relative and absolute frequencies separated by the seven coded problem-solving activities. The students predominantly described [1] and executed [3] their problem-solving attempts. Taken together, both activities comprise 54.80% of all coded activities. Less frequently, students drew unsubstantiated [4] (16.06%) or substantiated conclusions [5] (10.43%), and they unsubstantially evaluated their solution attempts [6] (10.76%). The least common activities were substantiated evaluations [7] (3.81%) and analysis of the goal state [1] (4.14%).

Table 4 displays the correlations between the seven activities and the measures reported in section 5.1 (mathematical ability, prior knowledge, number of solutions, quality of solutions, and conceptual knowledge). As shown in Table 4, students who invented a higher number of solutions also engaged more in describing and executing them. In addition, higher-quality solutions were associated with more frequent descriptions [1] throughout the problem-solving process. Apart from the description [1] and execution [3] of the problem-solving attempts, the remaining activities occurred relatively rarely. In particular, students less frequently defined the goal state of the problem [2] and indicated a lower number of substantiated evaluations [3] of their solution attempts. Interestingly, the more frequently the students indicated unsubstantiated evaluations

[4] (e.g., doubts about their solutions without adequate reasoning), the lower their conceptual knowledge acquisition from the instruction after problem solving.

Table 3. Relative and Absolute Frequencies of Problem-Solving Activities (*N*=22)

Activity		Relative frequency	Absolute frequency
[1]	Description of the solution	22.35%	135
[2]	Analysis of the goal state	4.14%	25
[3]	Execution of the solution	32.45%	196
[4]	Unsubstantiated conclusion	16.06%	97
[5]	Substantiated conclusion	10.43%	63
[6]	Unsubstantiated evaluation	10.76%	65
[7]	Substantiated evaluation	3.81%	23
	∑ absolute		604

Table 4. Pearson Correlations of Student Problem-Solving Activities and Performance (including 90% confidence intervals)

N = 22		Mathematical Ability	Prior knowledge	Number of solutions	Quality of solutions	Conceptual knowledge
[1]	Description of the solution	.27 [-.10,.57]	.03 [-.33,.39]	**.58 [.27,.78]**	-.03 [-.39,.33]	.06 [-.31,.41]
[2]	Analysis of the goal state	-.05 [-0.40,.32]	-.23 [-.55,.14]	-.12 [-.46,.25]	.13 [-.25,.47]	.02 [-.34,.38]
[3]	Execution of the solution	**.37 [.01,.64]**	-.12 [-.46,.26]	**.38 [.03,.65]**	**.43 [.09,.69]**	-.22 [-.54,.15]
[4]	Unsubstantiated conclusion	.28 [-.09,.58]	-.01 [-.37,.36]	.28 [-.10,.58]	.22 [-.16,.53]	**-.41 [-.06, -.67]**
[5]	Substantiated conclusion	.26 [-.11,.57]	.01 [-.35,.37]	.11 [-.26,.45]	.24 [-.14,.55]	-.06 [.31, -.41]
[6]	Unsubstantiated evaluation	-.31 [-.60,.06]	-.09 [-.44,.27]	.02 [-.34,.38]	-.01 [-.37,.36]	-.07 [-.42,.30]
[7]	Substantiated evaluation	-.09 [-.44,.28]	-.04 [-.39,.33]	-.23 [-.54,.15]	.32 [-.04,.61]	.27 [-.10,.57]

Note: Bold marked 90% CI scores do not include the zero; that is, these correlations marginally reached significance.

Even if the frequencies and correlations provide initial insights into student problem-solving processes, the specific connection between (sequences) patterns of problem-solving activities, as well as their correspondence to student conceptual knowledge acquisition, remains blurred. In other words, counting the frequencies of student problem-solving activities alone is not a sufficient description of the underlying problem-solving process. For instance, as indicated by the negative correlation shown in Table 4, students who gained more conceptual knowledge from the later instruction seem to show fewer unsubstantiated evaluations [6] in comparison to better learners. However, this activity is most likely involved in the temporally structured problem-solving process; specifically, the activities may occur in certain sequences and may depend on each other (e.g., they only occur in a certain order), and connections to student learning outcomes may only become noticeable if we address this temporal characteristic in our process analysis. Therefore, we analyzed student problem-solving processes by applying HeuristicsMiner (cf. Weijters et al., 2006), an algorithm that indicates different patterns of relations (dependencies) between the coded activities within the problem-solving process.

5.2. Identifying Low- and High-Performing Students

Because the HeuristicsMiner algorithm produces a holistic spaghetti model of dependencies between problem-solving activities during the process, the model outcome cannot be directly linked to student conceptual knowledge acquisition (e.g., using multiple regressions). An efficient strategy to relate the process model produced by the algorithm to student learning outcomes is to contrast the performance of different groups of participants (cf. Bannert et al., 2014). Accordingly, we divided the 22 students into two distinct groups based on their median score (*Md* = 3.25) on the conceptual knowledge post-test, which resulted in a high-performing (*n* = 11; range between 3.5 and 5 points) and low-performing group (*n* = 11; range between 0.5 and 3 points). We argue that these two groups provide the most effective contrast in patterns of problem-solving processes to

student conceptual knowledge acquisition from the later instruction because further exploratory analysis revealed that these two groups only differed significantly on their conceptual post-test performance, not on the remaining variables (e.g., mathematical ability or prior knowledge).

As shown in Table 5, Bayesian independent sample T-tests revealed that our data for student mathematical ability, prior knowledge, number of solution attempts, and quality of the invented solution are approximately two times better explained by the null hypothesis (i.e., that there are no differences between students on these measures; BF_{01}), compared to the alternative hypothesis (i.e., that there are differences between students on these measures; BF_{10}). However, the Bayesian T-tests only provided weak to moderate evidence towards the null hypotheses, even though it was better supported by our data than the alternative hypothesis. The rather weak evidence may be due to statistical power; specifically, the small sample size. Because we divided the groups based on student performance on the conceptual knowledge post-test, the Bayesian T-Test for assessing conceptual knowledge strongly supported the assumption that our data on conceptual knowledge are better explained by the alternative hypothesis (BF_{10}), compared to the null hypothesis (BF_{01}).

Table 5. Differences in Individual Prerequisites and Performance Between High- (n = 11) and Low-Performing Students (n = 11)

Bayesian Independent Samples T-Test								95% Confidence Interval	
		Mean (SD)	BF_{01}	BF_{10}	Error %	Cohen's d	Mean difference (SE)	Lower	Upper
Mathematical ability (Grades 1 to 6)	High (1)	4.00 (0.71)	2.31	0.43	0.005	-0.24	-0.18 (0.32)	-0.85	0.48
	Low (0)	3.82 (0.78)							
Prior knowledge (0 to 10 points)	High (1)	4.41 (1.07)	2.49	0.40	0.005	-0.15	-0.18 (0.52)	-1.26	0.90
	Low (0)	4.23 (1.35)							
Number of solutions (Absolute)	High (1)	4.09 (1.38)	1.57	0.64	0.009	0.51	0.64 (0.53)	-0.47	1.75
	Low (0)	4.73 (1.10)							
Quality of solutions (Category 0 to 4)	High (1)	1.73 (1.01)	2.56	0.39	0.005	0.09	0.09 (0.42)	-0.79	0.98
	Low (0)	1.82 (0.98)							
Conceptual knowledge (0 to 7 points)	High (1)	3.95 (0.47)	< 0.01	2107	2.296e -7	-2.58	-1.90 (0.32)	-2.57	-1.25
	Low (0)	2.05 (0.93)							

Table 6 shows the relative and absolute frequencies separated by high- and low-performing students. Overall, the two groups did not significantly differ concerning the problem-solving activities they performed. Concerning the best differences, the low-performing students executed 16 more problem-solving attempts [3] and drew 17 more unsubstantiated conclusions [4] than the high-performing students. However, these differences did not correspond to the relative frequency, but exclusively to the absolute frequencies.

Table 6. Relative and Absolute Frequencies of Students' Problem-Solving Activities Separated by Low (n=11) and High Performers (n=11)

	Activity	**Low** (n = 11) Relative frequency (absolute)	**High** (n = 11) Relative frequency (absolute)
[1]	Description of the solution	21.60 (70)	23.21 (65)
[2]	Analysis of the goal state	3.70 (12)	6.64 (13)
[3]	Execution of the solution	32.72 (106)	32.21 (90)
[4]	Unsubstantiated conclusion	17.59 (57)	14.29 (40)
[5]	Substantiated conclusion	11.11 (36)	9.64 (27)
[6]	Unsubstantiated evaluation	10.80 (35)	10.71 (30)
[7]	Substantiated evaluation	2.47 (8)	5.36 (15)
	∑ absolute	324	280

We verified the differences between the conditions on the absolute frequencies of coded activities with seven Bayesian independent sample T-tests. As shown in Table 7, the data on all seven problem-solving activities were better explained by the null hypothesis (i.e., no differences between low and high performers in the absolute frequency of shown problem-solving activities; BF_{01}), compared to the alternative hypothesis (i.e., differences between the two groups; BF_{10}). However, even if the

Bayesian factors only provided weak to moderate evidence for the null hypothesis, no difference between the groups explains our data better than to expect differences between low and high performers.

Table 7. Differences in Mean Absolute Frequency of Student Problem-Solving Activities Between High (n=11) and Low-Performers (n= 11)

Bayesian Independent Samples T-Test									95% Confidence Interval	
			Mean (SD)	BF_{01}	BF_{10}	Error %	Cohen's d	Mean difference (SE)	Lower	Upper
[1]	Description of the solution	High (1) Low (0)	5.91 (1.97) 6.36 (1.86)	2.33	0.43	0.005	0.24	0.45 (0.82)	-1.25	2.16
[2]	Analysis of the goal state	High (1) Low (0)	1.18 (1.17) 1.09 (0.94)	2.56	0.39	0.005	-0.09	-0.09 (0.45)	-1.04	0.85
[3]	Execution of the solution	High (1) Low (0)	8.18 (1.72) 9.64 (3.08)	1.35	0.74	0.007	0.58	1.45 (1.06)	-0.76	3.67
[4]	Unsubstantiated conclusion	High (1) Low (0)	3.64 (2.34) 5.18 (2.56)	1.21	0.83	0.005	0.63	1.55 (1.05)	-0.64	3.73
[5]	Substantiated conclusion	High (1) Low (0)	2.46 (2.07) 3.27 (2.24)	1.96	0.51	0.005	0.38	0.82 (0.92)	-1.10	2.74
[6]	Unsubstantiated evaluation	High (1) Low (0)	2.73 (2.05) 3.18 (2.68)	2.42	0.41	0.005	0.19	0.45 (1.02)	-1.67	2.58
[7]	Substantiated evaluation	High (1) Low (0)	1.36 (1.63) 0.73 (1.10)	1.73	0.58	0.008	-0.46	-0.64 (0.59)	-1.87	0.60

Taken together, the low- and high-performing students were similar, concerning the absolute frequencies of their problem-solving activities, their prerequisites (i.e., mathematical ability), and the outcomes of the problem-solving process (i.e., solution quantity and quality). However, the analyses considered thus far do not provide sufficient information about student problem-solving processes; specifically, in what sequences the activities occur during the problem-solving process. We assume that if some of these sequences occur exclusively in either the low- or high-performing group, then they could potentially explain the differences in student knowledge acquisition from instruction after problem-solving; this was analyzed with HeuristicsMiner.

5.3. Mining Problem-Solving Activities of High- and Low-Performing Students

We used the software ProM (version 5.2) to analyze student problem-solving processes with the HeuristicsMiner algorithm (Weijters et al., 2006). The software allows users to define thresholds; specifically, whether the algorithm ignores or less frequently considers strongly related sequences to indicate dependencies in problem-solving activities. We adopted the following default threshold values: relative-to-best-threshold =.05; positive observations = 10; dependency threshold =.90; length-one-loops threshold =.90; length-two-loops threshold =.90; long-distance threshold =.90; dependency divisor = 1; AND threshold =.10. Additionally, all-activities-connected heuristics were used, and long-distance dependency heuristics were disabled. Based on these settings, the algorithm generates dependency scores, thus indicating the relation between two problem-solving activities. For instance, if students first described [1] and then executed [2] a solution attempt, then the algorithm provides a dependency score and the absolute frequency of this relation, which underlie the calculation of the dependency. Whereas a dependency score between approximately .75 and .99 indicates a rather strong dependency between two activities (i.e., they most often occur dependently), values close to .50 represent a weaker, but still considered, relation. Dependencies that do not meet the defined thresholds are ignored by the algorithm; accordingly, these dependencies are not part of the process model. We calculated two independent process models: one for the low-performing group and one for the high-performing group. Table 8 displays all dependency scores and frequencies as outlined by the algorithm (separated by the low- and high-performing groups). Figures 1 and 2 illustrate the values of Table 8 in the form of a spaghetti model. Whereas the bold links in both figures represent dependencies, which only occur in one of the two groups, gray shaded links represent dependencies that occurred in both.

In the following section, we further describe the results of the process analyses (see Table 8, Figure 1, and Figure 2) based on two categories. First, some processes occurred in both the low- and high-performing groups. Even if these processes may be less interesting for understanding the differences in student performance on the conceptual knowledge post-test, they may indicate basic problem-solving activities. Second, exclusive dependencies did exist; specifically, dependencies that occurred either in the low- or high-performing groups. These dependencies seem to be highly interesting for understanding differences in student conceptual knowledge acquisition. As the outcomes of HeuristicsMiner can be further interpreted in different ways, this is just one example of how the rather complex outcomes of the algorithm can be categorized.

Table 8. Dependency Scores and Frequencies of Student Problem-Solving Activities Indicated by HeuristicsMiner for Low and High Performers

Dependencies between activities	**Starting activity**		**Following activity**	**Low** (n = 11) Dependency (Frequency)	**High** (n = 11) Dependency (Frequency)
[1⇒3]	[1] Description of the solution	⇒	[3] Execution of the solution	.73 (57)	.70 (48)
[1⇒6]	[1] Description of the solution	⇒	[6] Unsubstantiated evaluation	.90 (16)	.88 (15)
[1⇒7]	[1] Description of the solution	⇒	[7] Substantiated evaluation	.50 (3)	-
[2⇒3]	[2] Analysis of the goal state	⇒	[3] Execution of the solution	.50 (8)	.75 (12)
[2⇒4]	[2] Analysis of the goal state	⇒	[4] Unsubstantiated conclusion	.50 (4)	-
[2⇒5]	[2] Analysis of the goal state	⇒	[5] Substantiated conclusion	.67 (2)	-
[3⇒1]	[3] Execution of the solution	⇒	[1] Description of the solution	.91 (35)	-
[3⇒2]	[3] Execution of the solution	⇒	[2] Analysis of the goal state	.50 (9)	.75 (6)
[3⇒3]	[3] Execution of the solution	⇒	[3] Execution of the solution	.99 (22)	-
[3⇒4]	[3] Execution of the solution	⇒	[4] Unsubstantiated conclusion	.97 (34)	.96 (38)
[3⇒5]	[3] Execution of the solution	⇒	[5] Substantiated conclusion	.53 (31)	.50 (19)
[3⇒6]	[3] Execution of the solution	⇒	[6] Unsubstantiated evaluation	.90 (3)	.89 (11)
[4⇒1]	[4] Unsubstantiated conclusion	⇒	[1] Description of the solution	.96 (25)	.86 (20)
[4⇒2]	[4] Unsubstantiated conclusion	⇒	[2] Analysis of the goal state	.50 (4)	-
[4⇒3]	[4] Unsubstantiated conclusion	⇒	[3] Execution of the solution	.97 (15)	.96 (16)
[4⇒4]	[4] Unsubstantiated conclusion	⇒	[4] Unsubstantiated conclusion	.96 (8)	.94 (1)
[4⇒6]	[4] Unsubstantiated conclusion	⇒	[6] Unsubstantiated evaluation	.75 (16)	-
[4⇒7]	[4] Unsubstantiated conclusion	⇒	[7] Substantiated evaluation	-	.75 (3)
[5⇒1]	[5] Substantiated conclusion	⇒	[1] Description of the solution	.94 (18)	.92 (11)
[5⇒2]	[5] Substantiated conclusion	⇒	[2] Analysis of the goal state	.67 (2)	-
[5⇒3]	[5] Substantiated conclusion	⇒	[3] Execution of the solution	.92 (11)	.91 (8)
[5⇒4]	[5] Substantiated conclusion	⇒	[4] Unsubstantiated conclusion	.67 (11)	-
[5⇒5]	[5] Substantiated conclusion	⇒	[5] Substantiated conclusion	.93 (1)	.92 (1)
[5⇒6]	[5] Substantiated conclusion	⇒	[6] Unsubstantiated evaluation	.67 (8)	-
[5⇒7]	[5] Substantiated conclusion	⇒	[7] Substantiated evaluation	-	.67 (3)
[6⇒1]	[6] Unsubstantiated evaluation	⇒	[1] Description of the solution	.90 (9)	.88 (13)
[6⇒3]	[6] Unsubstantiated evaluation	⇒	[3] Execution of the solution	.90 (9)	.89 (12)
[6⇒4]	[6] Unsubstantiated evaluation	⇒	[4] Unsubstantiated conclusion	.75 (14)	.67 (7)
[6⇒5]	[6] Unsubstantiated evaluation	⇒	[5] Substantiated conclusion	.67 (7)	-
[6⇒6]	[6] Unsubstantiated evaluation	⇒	[6] Unsubstantiated evaluation	.96 (5)	-
[7⇒5]	[7] Substantiated evaluation	⇒	[5] Substantiated conclusion	-	.67 (6)
[7⇒6]	[7] Substantiated evaluation	⇒	[6] Unsubstantiated evaluation	.50 (3)	.75 (4)

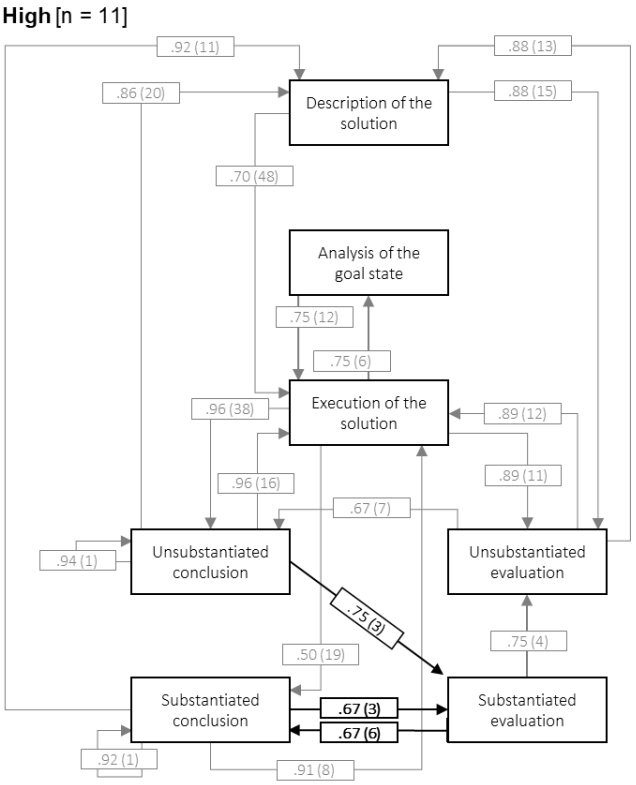

Figure 1. Heuristic nets of the high-performing students' problem-solving process

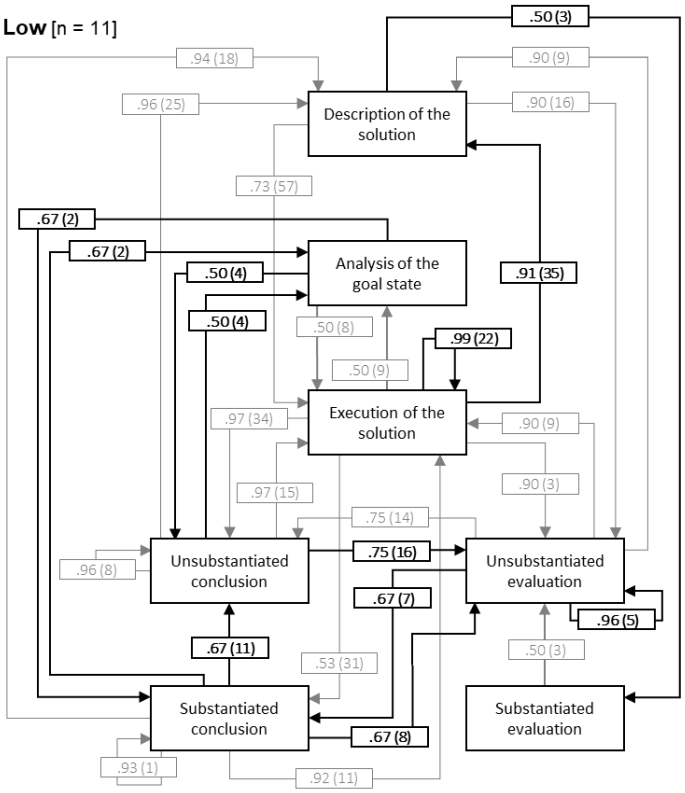

Figure 2. Heuristic nets of the low-performing students' problem-solving process

ISSN 1929-7750 (online).

5.4. Identifying Student Problem-Solving Strategies

Based on these two categories of dependencies, we outlined eight strategies (A to H) to prepare the complex process model for discussion (see Table 9). We generated these strategies by initially interpreting the data and determining that strategies are dependent; specifically, the strategies may (cyclically) tether if they shared the same problem-solving activities. We exclusively assigned the dependencies outlined by the algorithm to only one strategy. We decided to define the problem-solving activity in which students described their solution attempts [1] as a starting point because this activity appeared to be the first shown activity for most students. In the following section, we describe extracts from the heuristics net (see Figure 1, Figure 2, and Table 8). Strategies A to D included nonexclusive strategies; specifically, those dependencies between activities that occurred in both the high- and low-performing groups. However, we highlighted whether the dependency scores and/or frequencies of the outlined dependencies noticeably differed between the two groups. Strategies E to H assumed exclusive strategies; specifically, those strategies that only occurred in one group.

Table 9. Nonexclusive and Exclusive Problem-Solving Strategies

Nonexclusive Strategies Applied in Both Groups (A to D)			
A	[1] Description of the solution	⇒	[3] Execution of the solution
	[3] Execution of the solution	⇒	[4] Unsubstantiated conclusion
	[4] Unsubstantiated conclusion	⇒	[3] Execution of the solution
	[4] Unsubstantiated conclusion	⇒	[4] Unsubstantiated conclusion
	[4] Unsubstantiated conclusion	⇒	[1] Description of the solution
B	[3] Execution of the solution	⇒	[5] Substantiated conclusion
	[5] Substantiated conclusion	⇒	[3] Execution of the solution
	[5] Substantiated conclusion	⇒	[5] Substantiated conclusion
	[5] Substantiated conclusion	⇒	[1] Description of the solution
C	[2] Analysis of the goal state	⇒	[3] Execution of the solution
	[3] Execution of the solution	⇒	[2] Analysis of the goal state
D	[1] Description of the solution	⇒	[6] Unsubstantiated evaluation
	[6] Unsubstantiated evaluation	⇒	[1] Description of the solution
	[6] Unsubstantiated evaluation	⇒	[3] Execution of the solution
	[3] Execution of the solution	⇒	[6] Unsubstantiated evaluation
	[6] Unsubstantiated evaluation	⇒	[4] Unsubstantiated conclusion
	[7] Substantiated evaluation	⇒	[6] Unsubstantiated evaluation
Exclusive Strategies Applied Only in One of the Groups (E to H)			
E	[4] Unsubstantiated conclusion	⇒	[7] Substantiated evaluation
	[4] Unsubstantiated conclusion	⇒	[6] Unsubstantiated evaluation
	[6] Unsubstantiated evaluation	⇒	[6] Unsubstantiated evaluation
F	[5] Substantiated conclusion	⇒	[7] Substantiated evaluation
	[7] Substantiated evaluation	⇒	[5] Substantiated conclusion
	[1] Description of the solution	⇒	[7] Substantiated evaluation
	[6] Unsubstantiated evaluation	⇒	[5] Substantiated conclusion
	[5] Substantiated conclusion	⇒	[4] Unsubstantiated conclusion
	[5] Substantiated conclusion	⇒	[6] Unsubstantiated evaluation
G	[3] Execution of the solution	⇒	[1] Description of the solution
	[3] Execution of the solution	⇒	[3] Execution of the solution
H	[4] Unsubstantiated conclusion	⇒	[2] Analysis of the goal state
	[2] Analysis of the goal state	⇒	[4] Unsubstantiated conclusion
	[5] Substantiated conclusion	⇒	[2] Analysis of the goal state
	[2] Analysis of the goal state	⇒	[5] Substantiated conclusion

5.4.1 Nonexclusive Strategies Applied in Both Groups (A to D)

Strategy A. This strategy describes the most frequently applied sequences, which most likely led students from one problem-solving attempt to another. After describing their problem-solving attempts, students in both groups often continued executing them [1⇒3]. After executing their problem-solving attempts, students often drew unsubstantiated conclusions, only reporting the results of their solution attempts without explicitly reasoning as to why the attempt supported their conclusion [3⇒4]. In some cases, students returned from unsubstantiated conclusions to executing their solution attempts [4⇒3], most likely because they derived their conclusions from the initial results. Interestingly, in the low-performing group, students returned from unsubstantiated conclusions to unsubstantiated conclusions [4⇒4] eight times (which happened only one time in the high-performing group). In other words, compared to the high-performing group, low-performing students more

ISSN 1929-7750 (online).

frequently repeated conclusions without reasoning through the problem-solving process. After unsubstantiated conclusions, students in both groups often continued in describing another solution attempt [4⇒1], even if this situation occurred five counts fewer in the high-performing group; in some cases, high-performing students continued with different activities.

Strategy B. This strategy describes how students in the two groups moved towards substantiated conclusions. All students most frequently drew substantiated conclusions after executing their solution attempts [3⇒5]. Although this dependency frequently occurred (31 counts for the low-performing groups and 19 counts for the high-performing group), a dependency score close to .50 indicated that both activities were rather loosely coupled. After drawing substantiated conclusions — in one case within both groups — the students returned to executing their solution attempts [5⇒3] and/or repeatedly drew substantiated conclusions [5⇒5]. As with unsubstantiated conclusions (cf. [4⇒4], strategy A), students seemed to repeatedly draw substantiated conclusions throughout the problem-solving process. Students of both groups often went from substantiated conclusions to the description of another solution attempt [5⇒1], even if they did this less frequently when compared to unsubstantiated conclusions (cf. [4⇒1], Strategy A).

Strategy C. This strategy describes how the analysis of the problem's goal state occurred throughout the problem-solving process in both groups. Both mined process models revealed that students analyzed the goal state of the problem before [2⇒3] and after [3⇒2] executing their solution attempts. Accordingly, in the low- and high-performing groups, the analysis of the goal state did not seem to take place at the beginning of the problem-solving process (e.g., related to the description of the solution attempt [1]), but occurred throughout the execution of the solution attempts. However, although these sequences occurred frequently in both groups, the score in the high-performing group was noticeably higher (.75) than in the low-performing group (.50). This indicates that high-performing students included the analysis of the goal state and followed reliably throughout the execution of their solution attempts.

Strategy D. This strategy outlines how students in both groups proceed with unsubstantiated evaluations; specifically, they evaluated their solution attempts without adding elaborated arguments. Unsubstantiated evaluations often took place after students described their problem-solving attempts [1⇒6]; specifically, before students most likely executed a solution attempt. However, unsubstantiated evaluations also appeared in the opposite direction, i.e., descriptions followed the evaluations [6⇒1]. Accordingly, both groups superficially evaluated their solution attempts before or after describing another solution attempt (which we interpreted as the starting point for the attempt). In addition, students in both groups (more frequently in the high-performing group) indicated unsubstantiated evaluations before [6⇒3] and after [3⇒6] they executed their problem-solving attempts. In both groups, unsubstantiated evaluations also preceded unsubstantiated conclusions [6⇒4]; however, this occurred seven times more in the high-performing group. In other words, if students evaluated their solutions without deeper argumentation, then they also subsequently drew unsubstantiated conclusions. Low-performing students exhibited this sequence 14 times; specifically, seven times more than the high-performing group (high: .67; low: .75). Furthermore, in some cases (four times in the high-performing group, .75; three times in the low-performing group, .50), students first substantially evaluated their solutions before moving to substantiated evaluations [7⇒6].

The strategies reported thus far (A to D) included problem-solving sequences, which, despite some differences in frequency and dependency scores, occurred in both groups. However, the spaghetti model, as illustrated in Figures 1 and 2, already outlines noticeable differences between the groups concerning problem-solving processes, which either low- or high-performing students exclusively applied.

5.4.2 Exclusive Strategies Applied Only in One of the Groups (E to H)

First, the dependencies between problem-solving activities generated for the high-performing group included a smaller number of exclusive dependencies. Specifically, whereas the high-performing group indicated only three exclusive sequences, the low-performing group exhibited 12 exclusive relations. This indicates that, compared to the low-performing students, the high-performing students may have more efficient problem solving; specifically, they showed less exclusive, but likely more straightforward and elaborated sequences. In other words, whereas low-performing students rather arbitrarily applied many exclusive strategies, high-performing students showed fewer exclusive sequences, but those sequences seem to be effective (e.g., substantiated conclusions and evaluations). We further defined four strategies (E to H) to describe these exclusive sequences in more detail.

Strategy E. This strategy describes differences between the groups, concerning how students related unsubstantiated conclusions to (unsubstantiated and substantiated) evaluations. In contrast to low-performing students, in three cases, high-performing students substantially evaluated their solutions after they unsubstantially drew conclusions from their solution attempts [4⇒7]. In other words, if high-performing students did not add arguments to their conclusions, then they argued why the solution attempt would sufficiently (or insufficiently) solve the problem. In contrast, in the low-performing group, unsubstantiated conclusions preceded unsubstantiated evaluations [4⇒6] 16 times. When low-performing students did not

ISSN 1929-7750 (online).

argue their conclusions, they evaluated their solutions, but without using reasoned arguments, as the high-performing students did. In addition, low-performing students repeated unsubstantiated evaluations five times [6⇒6].

Strategy F. This strategy describes how the two groups linked substantiated conclusions and substantiated evaluations in different ways. Only high-performing students drew three counts more substantiated conclusions and subsequently (and substantially) evaluated their solution attempts [5⇒7]. In addition, only high-performing students showed this dependency by six times in reverse order [7⇒5]. Furthermore, the low-performing students did show dependencies, including substantiated evaluations; however, this differed from the high-performing students three times after describing a solution attempt [1⇒7], although the dependency score was weak. Only low-performing students drew substantiated conclusions seven times after unsubstantiated evaluations [6⇒5], 11 times before unsubstantiated conclusions [5⇒4], and eight times before unsubstantiated evaluations [5⇒6]. Taken together (in contrast to the high-performing students), low-performing students did not proceed dependently with substantiated conclusions and evaluations.

Strategy G. This strategy describes two dependencies, which only and frequently occurred in the low-performing group. Low-performing students described a solution attempt 33 times after executing a solution [3⇒1], and they engaged in repeatedly executing the solution 22 times [3⇒3]. These dependencies indicate that low-performing students often went from executing a solution attempt to executing and/or describing another attempt without proceeding with alternative (and probably more effective) activities; for instance, conclusions or evaluations.

Strategy H. This strategy describes how students in the low-performing group exclusively related (substantiated and unsubstantiated) conclusions to the analysis of the goal state of the problem. Over four instances (although with a low dependency score), low-performing students drew unsubstantiated conclusions either before [4⇒2] or after [2⇒4] analyzing the goal state of the problem. The same pattern was found for substantiated conclusions, in which low-performing students performed two times before [5⇒2] and two times after [2⇒5] analyzing the goal state.

In summary, the exclusive strategies highlight noticeable differences between the low- and high-performing students. The most interesting pattern likely consists of the fact that high-performing students were shown a straight problem-solving process (i.e., less exclusive strategies) compared to the low-performing group. In addition, high-performing students exclusively combined substantiated elaborations and evaluations, which indicates that high-performing students may elaborate more deeply on their problem-solving attempts, thus likely explaining their better performance on the conceptual knowledge post-test.

6. Discussion

In the analysis presented in this paper, we explored an effective problem-solving process in a classroom context via learning analytics. We applied the HeuristicsMiner algorithm to explore the problem-solving process of 22 students who engaged in 45 minutes of problem solving prior to instruction. Our study further aimed to illustrate (using data from a Productive-Failure study) how learning analytics (e.g., process mining) can be used to understand problem-solving and learning processes in more detail.

First, our findings illustrate how learning analytics techniques support educational research. Learning analytics offers fine-grained descriptions of the problem-solving and learning processes of students, which would have remained hidden when using conventional analytical methods; for example, coding-and-counting absolute and relative frequencies. Our study supports the conclusion by Csanadi et al. (2018), who analyzed (computer-supported) collaborative problem-solving processes with an epistemic network analysis (cf. ENA, Shaffer et al., 2016), as well as with coding-and-counting-based strategies. Csanadi et al. (2018) concluded that the sole analysis of absolute and relative frequencies of problem-solving (and learning) activities ignores the temporal nature of verbal data; accordingly, detailed process analyses are inevitable to sufficiently address the temporal characteristics of problem-solving and learning processes. Bannert et al. (2014) drew a similar conclusion, concerning self-regulated learning processes. The findings reported in this paper support this conclusion for a Productive-Failure classroom setting (cf. Kapur, 2012). Our analysis of absolute and relative frequencies showed no significant difference between the low- and high-performing students; specifically, the seven problem-solving activities occurred similarly in both groups. Therefore, they did not seem to help explain student learning outcomes. In addition, the number and quality of student solutions (such as prior knowledge and mathematical ability) did not explain student performance on the conceptual knowledge post-test. As indicated by our findings, only unsubstantiated conclusions marginally tended to be negatively correlated with the conceptual knowledge acquisition of students. In other words, if students did not argue their conclusions, then they learned less. However, as low- and high-performing students similarly and frequently showed unsubstantiated conclusions, it remains unclear as to why not arguing conclusions may be less effective for learning. As has been argued, frequencies do not provide information about the temporal nature of the coded activities; specifically, dependencies between sequenced activities have been shown during problem-solving. In contrast, the HeuristicsMiner algorithm used in our study was able to determine that conclusions

and evaluations occurred very differently during the problem-solving-and-failing processes of the low- and high-performing students. The algorithm also determined that such detailed descriptions of problem-solving (and learning) processes have important advantages for educational research and practice.

Foremost, process mining is necessary for theory building in the context of educational science. For instance, most research explains the effectiveness of the Productive-Failure approach using a consecutive process. Specifically, during problem-solving, students first activate their prior knowledge to generate multiple solution ideas (Kapur, 2012) and fail, thus becoming aware of their knowledge gaps (Loibl & Rummel, 2014a). This problem-solving-and-failing process is assumed to prepare the students to recognize deep features of the correct solution, as is taught in the instruction after problem-solving (for an overview, see Loibl et al., 2017). In conclusion, the theory underlying the Productive-Failure approach strongly leans on process-oriented explanations. Therefore, methods need to address these temporal characteristics of theory, and the HeuristicsMiner algorithm used in the present study meets this requirement. As only one example (and likely the most interesting outcome of the mined process model), our analysis indicated that, if high-performing students were not able to reason through their conclusions, then they tended to reason as to why their solution attempt solved (or did not solve) the problem (i.e., substantiated evaluations). This outlined sequence of the high-performing students corresponds with the assumption that becoming aware of one's knowledge gaps (due to the problem-solving-and-failing process) explains the effectiveness of the Productive-Failure approach (Loibl et al., 2017; Loibl et al., 2014a). If high-performing students realized that they could only derive unsubstantiated conclusions from their solution attempts (thus reaching an impasse in their problem solving), then they tried to evaluate (with elaborated reasoning) whether their solution attempt was useful. As an outcome of this process, they may overcome their impasse and arrive at a better solution attempt or realize exactly why they cannot efficiently solve the problem. This can also be described as meta-cognitive processing; specifically, learners try to reflect on and explain their problem-solving behaviour (cf. Berardi-Coletta, 1995). If students already elaborate on their solution attempts prior to instruction (cf. awareness of knowledge gaps), they might connect the later presented knowledge on the canonical solution approach more effectively. In contrast, if low-performing students were not able to reason through their conclusions, then they analyzed the goal state of the problem but did not return to substantiated evaluations. Even if this also indicated metacognitive behaviour as students try to redefine the goal state of the problem, if the students do not elaborately evaluate their attempts, then they most likely cannot become aware of their knowledge gaps or identify better solution attempts and therefore might be less effectively prepared for the later instruction. In addition, these results may indicate a difference between low- and high-performing students because high performers have a better representation of the problem's goal state from the outset of problem-solving, compared to low-performing students since the high performers did not need to return to analyzing the problem's goal state after they struggled with drawing well-founded conclusions. Since this is only one example of the interesting sequences outlined in our analysis, we argue that the mined patterns have a high potential to specify and extend the theory used in the context of Productive-Failure and educational research in general.

A further advantage of the process-mining technique (as presented in this paper) is represented by the fact that it does not require large-scale sample sizes. Therefore, the HeuristicsMiner algorithm is suitable for small sample sizes and even single individuals. As illustrated in this paper, the algorithm can reveal a high amount of detailed temporal information, even with small samples (or one individual), which makes the algorithm applicable to small-scale classroom settings. In addition, the rules behind the algorithm (see section 4.5 for more details; Weijters et al., 2006) are simple and transparent, as the algorithm refers to the order of events (A⇒B) and relates these to alternative events (B⇒A).

However, it should be noted that the use of the HeuristicsMiner algorithm is also limited. One important shortcoming of the algorithm may be because it does not rely on inferential statistics (cf. Null-Hypothesis-Testing), which consider sample and population estimates. Therefore, the algorithm is applicable for describing and exploring data of a sample, rather than for explanatory studies, in which data from a sample are used to make general predictions for an entire population. An interesting avenue for further research is making the algorithm applicable for explanatory research and hypothesis testing. An initial promising direction is conformance checking; specifically, the opportunity to compare two models from different samples to verify similarities and deviations (see Van der Aalst et al., 2012, Sonnenberg & Bannert, 2018).

Nevertheless, future research must also provide clear guidelines to mindfully set the algorithm's thresholds. The algorithm requires defining thresholds for all types of dependencies; specifically, how clear dependencies must be and how frequently they must occur to be considered in the spaghetti model. The outcome of HeuristicsMiner is a heuristic net of dependency scores (including absolute frequencies) of all sequences explored by the algorithm (see Table 8, Figure 1, and Figure 2). Depending on the predefined parameters, the algorithm includes or excludes less frequently occurring dependencies between activities for the model; specifically, the complexity of the spaghetti model extracted by the algorithm strongly varies following the predefined thresholds. For the generalizability of the data presented here, this means that depending on the parameter setting, the model (and thus the interpretations derived from it) changes significantly. For our application case, it cannot yet

be sufficiently clarified according to which criteria these parameters are to be set. It may be argued that, for a pure exploration of data, researchers can theoretically decide when a mined model is too complex for theory building (i.e., if the thresholds were too liberal) or too trivial (i.e., if the thresholds were too strict). However, Weber et al. (2013) argued that, if the thresholds of the algorithm are set based on the visual complexity of a mined model, then users cannot be confident in the mining accuracy because they do not know what noise may affect the algorithm's outcome; in particular, the complex interactions between the parameters to be adjusted. To the best of our knowledge, there are no clear guidelines on how to set the algorithm's thresholds, which may be an important avenue for future research. At this point, concerning the use case presented here, a theoretical consideration of interpretability — a theory-driven parameterization oriented to the complexity of the spaghetti model — appears to be an applicable and possibly still valid option.

In addition, the relationships between mined activities may be interfered with by unsystematically performed activities or by the noise produced by unreliably applying the coding manual. Even if the HeuristicsMiner algorithm is assumed to be robust for noisy data (because less frequently occurring activities are also included by the mining procedure; see Sonnenberg & Bannert, 2015), and when considering the rather vague parameter setting previously mentioned, systematically occurring noise (e.g., less frequent and irrelevant dependencies) can critically bias the mined process model, based on how the thresholds were set. Furthermore, in addition to the threshold limitations, the reliability of the process model also strongly depends on the reliability of think-aloud coding. As was argued in this paper, educational research often uses think-aloud data to make the thinking processes of learners explicit. However, in the analysis presented in this paper, learners may not express any thoughts while problem solving, which critically affects the outcomes of the mined model. In addition, the mined model does not handle or correct noise produced because of incorrect data coding. Thus, the coding of think-alouds and, correspondingly, the mined model, strongly depend on the reliability of the coded think-aloud data even if interrater reliability was sustainable, based on the complexity of the applied coding manual and research question, this may not always be the case. To appropriately align the coding with a research question, decisions must be made; for instance, how fine-grained the processes to be coded are or which parts of think-aloud protocols should be considered (for an overview and guidance see Chi, 1997). The quality of the model obtained with HeuristicsMiner is highly dependent on the quality and type of coding. In our study, we coded more general problem-solving activities that can be used for initial exploration and theory building. However, these activities may be too fuzzy for more specific hypothesis testing. One advantage of HeuristicsMiner is that also very detailed processes of, for example, one or a few subjects can also be mapped, making the algorithm applicable to different coding.

Another limitation is associated with the rather indirect relation between dependency scores and student learning performance, because we had to group the students into low and high performers. Even if students in the high-performing groups performed better on the conceptual knowledge post-test than the low-performing group, the algorithm does not outline which activities within the groups predicted the individual learning outcomes of the students. In addition, the process models mined by HeuristicsMiner may be highly sensitive for the chosen grouping procedure; grouping students based on continuous variables cannot be handled as a clear cut-off. Therefore, it may be interesting for further research to investigate how information mined by the algorithm can be broken down to the individual level; specifically, future research should be conducted to indicate which individuals contributed (as well as how much was contributed) to the summarized dependency scores mined by the algorithm (i.e., individual variance in dependencies). The individual variances can then be integrated into inferential statistics of explanatory studies (e.g., as a part of mediation models). It may also be an interesting option not to divide cases by a median split, but to use more complex clustering methods (cf. Bogarín et al., 2014); however, this would correspondingly result in higher demands on the datasets to be analyzed (for instance, in terms of sample size).

Nevertheless, in addition to these limitations, the descriptive heuristics net mined by the algorithm can already contribute to explanatory studies, concerning implementation checks. In the context of educational research (which also applies to the context of productive failure), theoretical assumptions are often tested in experimental studies; specifically, at least two conditions are compared to identify (and internally validate) differences between groups. However, it often remains unexplored as to whether participants within the conditions behave in the way expected. For example, in the context of self-regulated learning, students in an experimental condition may be given metacognitive prompts to improve their self-regulated learning processes (e.g., Bannert et al., 2015). However, when compared to a control group (without metacognitive prompts), individuals may not change their self-regulated learning behaviour after they have received the prompts. The process mining presented in this paper would be an ideal method to explore whether participant behaviour matches theoretical assumptions. In some cases, this may also lead researchers to discover the reasons for unexpected and contradictory findings.

7. Conclusion

The analyses presented in this paper demonstrate the potential of learning analytics in the context of a classroom-based

ISSN 1929-7750 (online).

learning environment (i.e., the Productive-Failure approach). We initially explored productive and unproductive problem-solving strategies and highlighted how such an exploration would be highly advantageous for educational research and practice. Nevertheless, as learning analytics is a new and fast-growing field in educational science, we still need to establish clear standards and guidelines for how to use different tools, such as HeuristicsMiner, to improve research in this field.

Declaration of Conflicting Interest

The authors declared no potential conflicts of interest concerning the research, authorship, and/or publication of this article.

Funding

The authors declared no financial support for the research, authorship, and/or publication of this article.

References

Azevedo, R. (2014). Issues in dealing with sequential and temporal characteristics of self- and socially-regulated learning. *Metacognition and Learning, 9*(2), 217–228. https://doi.org/10.1007/s11409-014-9123-1

Bannert, M., Reimann, P., & Sonnenberg, C. (2014). Process mining techniques for analysing patterns and strategies in students' self-regulated learning. *Metacognition and Learning, 9*(2), 161–185. https://doi.org/10.1007/s11409-013-9107-6

Bannert, M., Sonnenberg, C., Mengelkamp, C., & Pieger, E. (2015). Short- and long-term effects of students' self-directed metacognitive prompts on navigation behavior and learning performance. *Computers in Human Behavior, 52*, 293–306. https://doi.org/10.1016/j.chb.2015.05.038

Berardi-Coletta, B., Buyer, L. S., Dominowski, R. L., & Rellinger, E. R. (1995). Metacognition and problem solving: A process-oriented approach. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 21*(1), 205–223. https://doi.org/10.1037/0278-7393.21.1.205

Bingham, C. B., & Haleblian, J. (2012). How firms learn heuristics: Uncovering missing components of organizational learning. *Strategic Entrepreneurship Journal, 6*(2), 152–177. https://doi.org/10.1002/sej.1132

Bogarín, A., Romero, C., Cerezo, R., & Sánchez-Santillán, M. (2014, March). Clustering for improving educational process mining. *Proceedings of the 4th International Conference on Learning Analytics and Knowledge* (LAK '14), 24–28 March 2014, Indianapolis, IN, USA (pp. 11–15). ACM Press. https://doi.org/10.1145/2567574.2567604

Brand, C., Hartmann, C., & Rummel, N. (2018). Exploring relevant problem-solving processes in learning from productive failure. International Society of the Learning Sciences. https://repository.isls.org//handle/1/576

Chi, M. T. (1997). Quantifying qualitative analyses of verbal data: A practical guide. *The Journal of the Learning Sciences, 6*(3), 271–315. https://doi.org/10.1207/s15327809jls0603_1

Csanadi, A., Eagan, B., Kollar, I., Shaffer, D. W., & Fischer, F. (2018). When coding-and-counting is not enough: Using epistemic network analysis (ENA) to analyze verbal data in CSCL research. *International Journal of Computer-Supported Collaborative Learning, 13*(4), 419–438. https://doi.org/10.1007/s11412-018-9292-z

DeCaro, M. S., & Rittle-Johnson, B. (2012). Exploring mathematics problems prepares children to learn from instruction. *Journal of Experimental Child Psychology, 113*(4), 552–568. https://doi.org/10.1016/j.jecp.2012.06.009

Granberg, C. (2016). Discovering and addressing errors during mathematics problem-solving: A productive struggle? *The Journal of Mathematical Behavior, 42*, 33–48. https://doi.org/10.1016/j.jmathb.2016.02.002

Hartmann, C., van Gog, T., & Rummel, N. (2020). Do examples of failure effectively prepare students for learning from subsequent instruction? *Applied Cognitive Psychology, 34*, 879–889. https://doi.org/10.1002/acp.3651

Hartmann, C., van Gog, T., & Rummel, N. (2021). Preparatory effects of problem solving versus studying examples prior to instruction. *Instructional Science, 49*(1), 1—21. https://doi.org/10.1007/s11251-020-09528-z

Heirweg, S., De Smul, M., Devos, G., & Van Keer, H. (2019). Profiling upper primary school students' self-regulated learning through self-report questionnaires and think-aloud protocol analysis. *Learning and Individual Differences, 70*, 155–168. https://doi.org/10.1016/j.lindif.2019.02.001

Hmelo-Silver, C. E. (2004). Problem-based learning: What and how do students learn? *Educational Psychology Review, 16*(3), 235–266. https://doi.org/10.1023/B:EDPR.0000034022.16470.f3

Järvelä, S., & Bannert, M. (2021). Temporal and adaptive processes of regulated learning: What can multimodal data tell? *Learning and Instruction, 72*. https://doi.org/10.1016/j.learninstruc.2019.101268

Kapur, M. (2012). Productive failure in learning the concept of variance. *Instructional Science, 40*(4), 651–672. https://doi.org/10.1007/s11251-012-9209-6

Kapur, M. (2014). Productive failure in learning math. *Cognitive Science, 38*, 1008–1022. https://doi.org/10.1111/cogs.12107

Koedinger, K. R., & Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review, 19*(3), 239–264. https://doi.org/10.1007/s10648-007-9049-0

Kreijns, K., Kirschner, P. A., & Jochems, W. (2003). Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: A review of the research. *Computers in Human Behavior, 19*(3), 335–353. https://doi.org/10.1016/S0747-5632(02)00057-2

Loibl, K., & Rummel, N. (2014a). Knowing what you don't know makes failure productive. *Learning and Instruction, 42*(3), 305–326. http://dx.doi.org/10.1016/j.learninstruc.2014.08.004

Loibl, K., & Rummel, N. (2014b). The impact of guidance during problem solving prior to instruction on students' inventions and learning outcomes. *Instructional Science, 42*(3), 305–326. http://dx.doi.org/10.1007/s11251-013-9282-5

Loibl, K., Roll, I., & Rummel, N. (2017). Towards a theory of when and how problem solving followed by instruction supports learning. *Educational Psychology Review, 29*(4), 693–715. http://dx.doi.org/10.1007/s10648-016-9379-x

Martin, T., & Sherin, B. (2013). Learning analytics and computational techniques for detecting and evaluating patterns in learning: An introduction to the special issue. *Journal of the Learning Sciences, 22*(4), 511–520. https://doi.org/10.1080/10508406.2013.840466

Molenaar, I., & Järvelä, S. (2014). Sequential and temporal characteristics of self and socially regulated learning. *Metacognition and Learning, 9*(2), 75–85. https://doi.org/10.1007/s11409-014-9114-2

Neuman, Y., & Schwarz, B. (1998). Is self-explanation while solving problems helpful? The case of analogical problem-solving. *British Journal of Educational Psychology, 68*(1), 15–24. https://doi.org/10.1111/j.2044-8279.1998.tb01271.x

Panadero, E. (2017). A review of self-regulated learning: Six models and four directions for research. *Frontiers in Psychology, 8*, 422. https://doi.org/10.3389/fpsyg.2017.00422

Ramachandran, A., Huang, C. M., Gartland, E., & Scassellati, B. (2018, February). Thinking aloud with a tutoring robot to enhance learning. *Proceedings of the ACM/IEEE International Conference on Human–Robot Interaction* (HRI '18), 5–8 March 2018, Chicago, IL, USA (pp. 59–68). ACM Press. https://doi.org/10.1145/3171221.3171250

Reimann, P. (2009). Time is precious: Variable- and event-centred approaches to process analysis in CSCL research. *International Journal of Computer-Supported Collaborative Learning, 4*(3), 239–257. https://doi.org/10.1007/s11412-009-9070-z

Reimann, P., Markauskaite, L., & Bannert, M. (2014). e-Research and learning theory: What do sequence and process mining methods contribute? *British Journal of Educational Technology, 45*(3), 528–540. https://doi.org/10.1111/bjet.12146

Rogiers, A., Merchie, E., & Van Keer, H. (2020). What they say is what they do? Comparing task-specific self-reports, think-aloud protocols, and study traces for measuring secondary school students' text-learning strategies. *European Journal of Psychology of Education, 35*, 315–332. https://doi.org/10.1007/s10212-019-00429-5

Schoenfeld, A. H. (1985). Making sense of "out loud" problem-solving protocols. *The Journal of Mathematical Behavior, 4*(2), 171–191. https://psycnet.apa.org/record/1986-28488-001

Schoenfeld, A. H. (2016). Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics (Reprint). *Journal of Education, 196*(2), 1–38. https://doi.org/10.1177/002205741619600202

Shaffer, D. W., Collier, W., & Ruis, A. R. (2016). A tutorial on epistemic network analysis: Analyzing the structure of connections in cognitive, social, and interaction data. *Journal of Learning Analytics, 3*(3), 9–45. https://doi.org/10.18608/jla.2016.33.3

Siemens, G., Gašević, D., Haythornthwaite, C., Dawson, S., Buckingham Shum, S., Ferguson, R., Duval, E., et al. (2011). Open learning analytics: An integrated and modularized platform. Society for Learning Analytics Research. https://solaresearch.org/wp-content/uploads/2011/12/OpenLearningAnalytics.pdf

Sobocinski, M., Malmberg, J., & Järvelä, S. (2017). Exploring temporal sequences of regulatory phases and associated interactions in low- and high-challenge collaborative learning sessions. *Metacognition and Learning, 12*(2), 275–294. https://doi.org/10.1007/s11409-016-9167-5

Sonnenberg, C., & Bannert, M. (2015). Discovering the effects of metacognitive prompts on the sequential structure of SRL-processes using process mining techniques. *Journal of Learning Analytics, 2*(1), 72–100. https://doi.org/10.18608/jla.2015.21.5

Sonnenberg, C., & Bannert, M. (2018). Using process mining to examine the sustainability of instructional support: How stable are the effects of metacognitive prompting on self-regulatory behavior? *Computers in Human Behavior, 96*,

159–272. https://doi.org/10.1016/j.chb.2018.06.003

Van der Aalst, W., Adriansyah, A., & van Dongen, B. (2012). Replaying history on process models for conformance checking and performance analysis. *WIREs Data Mining and Knowledge Discovery, 2*(2), 182–192. https://doi.org/10.1002/widm.1045

Van Laer, S., & Elen, J. (2018). Towards a methodological framework for sequence analysis in the field of self-regulated learning. *Frontline Learning Research, 6*(3), 228–249. https://doi.org/10.14786/flr.v6i3.367

Weber, P., Bordbar, B., & Tino, P. (2013). A principled approach to mining from noisy logs using HeuristicsMiner. *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* (CIDM 2013), 16–19 April 2013, Singapore (pp. 119–126). IEEE Computer Society. https://doi.org/10.1109/CIDM.2013.6597226

Weijters, A. J. M. M., Aalst, van der, W. M. P., & Alves De Medeiros, A. K. (2006). Process mining with the HeuristicsMiner algorithm. (BETA publicatie: working papers; Vol. 166). Technische Universiteit Eindhoven.

Winne, P. H. (2014). Issues in researching self-regulated learning as patterns of events. *Metacognition and Learning, 9*(2), 229–237. https://doi.org/10.1007/s11409-014-9113-3

Young, K. A. (2005). Direct from the source: The value of "think-aloud" data in understanding learning. *Journal of Educational Enquiry, 6*(1), 19–33. http://hdl.handle.net/10453/6348