Worked Examples and their use in Computer Science

Ben Skudder

I. Introduction

We examine some of the literature around worked-examples. We show how worked examples have been used in fields like engineering, statistics and programming, and suggest it's implication and possible benefits for use in Computer Science education.

II. What are Worked Examples?

According to [6] "Worked-out examples typically consist of a problem formulation, solution steps, and the final answer itself". A problem is presented, accompanied with step-by-step instructions which lead to the solution. These are usually textual but may include pictures, diagrams or animations. Students are expected to study the worked example and from it learn how they might apply it to similar problems.

III. How are they supposed to work?

The studies usually use Cognitive Load Theory (CLT) as their theoretical background. According to CLT humans have a limited working memory, where only a few bits of information can be processed at one time. Humans also have a long-term memory with a much larger capacity. Long-term memory consists of a set of schemas, and with practice can automatically be recalled and applied with minimal impact on working memory. The aim of learning in this framework is to help students form appropriate schemas which can be used to solve familiar and novel problems.

Much of the more modern literature distinguishes between different types of cognitive loads the most common being extraneous (ECL) and germane (GCL). ECL is the cognitive load caused by activities which do not assist with the formation of schemas, and GCL is the cognitive load caused by those that do. Many studies aim to expand on previous uses of worked examples by decreasing ECL or increasing GCL.

IV. Ways of presenting worked examples

The ways of presenting worked examples include:

a) Examples only: where students just study provided worked examples. An example of it's use is in [11]. The provision of examples over giving problems to solve is supposed to reduce ECL and direct student's attention to the relationships between different problem steps, and students can construct relevant problem-solving schemas around it. Problem-solving with no guidance, however, requires a large cognitive load for novices, but all the effort goes to finding an answer rather than schema formation.

b) Example-problem blocks: A block of worked examples of various types are provided to study, then a set of related problems are given which students are expected to solve. [3] examines the use of example-problem blocks. The provision of problems to solve as well as examples is supposed to motivate students to apply and practice what they've learned from examples, which helps foster schema formation and automatic processing.

c) Example-problem pairs: One of the most common ways of presenting worked examples, where each example is paired with a problem similar to the example for students to complete. Students alternate between studying a worked example and solving a related problem. [11], [3], [10], [6] provide examples of their use. This is supposed to foster learning better than example-problem blocks, as students can better select and recall the most relevant example (i.e. the one just studied) to relate the problem to when they are given one directly after the other. Separating them will make it harder to recall the relevant example to relate to the current problem.

d) Faded worked examples: A complete worked example is presented, then another worked example with one step missing is presented, and students are expected to fill in the missing step. They are presented with a series of worked examples, with an extra step removed each time, until a student is presented with just a problem to solve. The most common orders for fading steps are known as forward fading - where steps are removed starting from the beginning, and backwards fading - where steps are removed from the end first. [10], [6], [9], [2] show examples of these techniques.

According to [10] and [4], as a student gains expertise from studying worked examples, the benefits of studying them over problem solving disappears, as partial schema formation means that the elements that were once a source of GCL become a source of ECL. At this point problem-solving elicits GCL rather than ECL. To ease this transition, faded worked examples begin with a fully worked example, but as they study it and gain expertise steps are removed to encourage a manageable amount of problem solving, fostering GCL. By the end of the fading sequence, students will have studied many of the worked example steps and will be able to problem solve on their own.

These basic forms of presenting worked examples are often augmented with other techniques, such as:

e) Subgoal labeling: A technique where groups of steps are given a label, to help organize the information into a meaningful structure. This is presented in [5]. According to [5] subgoal labels allow students to focus on groups of steps rather than individual steps, giving them fewer problem-solving steps to consider and, reducing cognitive load. The highlighted structure given by subgoals is also supposed to assist with schema formation, or provide ”mental model frameworks” to internally explain how problems are solved.

f) Self explanation prompts: Self-explanation is a process some learners undergo when provided with a worked examples. Students who try to explain to themselves the reasons for a step or set of steps in an example were found to learn more than those who don't [6], so self-explanation prompts are designed to elicit such self-explanations. Self-explanation prompts can be in the form of asking students to justify a step or choosing what principle a particular step is invoking. According to the more recent CLT such prompts when employed correctly would be a source of GCL.

V. How well do they work?

The benchmark for evaluating worked examples is usually some form of problem solving. Problem solving usually requires a student to solve a problem, and are told when they've successfully provided a solution. Usually a set of questions is given, and some of these questions are swapped for worked examples people in the problem solving condition solve all the questions, and people in the examples condition study several examples and solve some problems.

They are also often evaluated for their ability to promote near transfer and far transfer. Near transfer is the ability of students to solve questions which are isomorphic to the ones they saw in their training phase, whereas far transfer it the ability for students to solve novel problems which use many of the same skills from the training phase, but in a different sequence or with some of the learned techniques requiring minor modifications.

We explore some of the evidence behind the aforementioned categories of worked examples, some of which are taken from the domain of programming.

A. The expertise reversal effect

[4] demonstrate instances where providing worked examples can hinder learning for people who have some expertise in the domain compared to those who are novices. For novices worked examples directs their attention to important features of the problem and help in forming relevant problem-solving schemas. This is a better use of their cognitive resources than problem solving, which requires extensive search of the problems space which imposes a heavy cognitive load, none of which is related to schema formation (in the CLT framework, it's a heavy source of ECL but not GCL). However someone with some expertise already has partial or full schemas in long-term memory for them worked examples are redundant, and having to analyze them is a source of ECL rather than GCL. They cite examples of trades apprentices, students working with databases and other experiments where people with more experience fail to gain any benefit from worked examples, unlike novices. In these studies, as novices' expertise increases, they learn more from problem solving rather than studying examples.

B. Examples only

[11] compare worked examples on their own, example problem pairs, problem example pairs and problem solving on its own for teaching high school students to diagnose a faulty electrical circuit.

In studies examined before, they found example-only study improved learning and transfer over traditional problem-solving techniques, and the same was found in [11].

They found example-only and example-problem pairs to work more effectively than the other conditions. Students reported lower mental effort and scored better results upon testing than those in the other conditions. No difference was found between example-only and example-problem pairs. This contradicts what CLT might predict, because presumably having to solve a problem as well as examining a worked example would motivate a student to engage more with the learning process i.e. providing a source of GCL. They suggest that the tasks were so few and the training phase so short (30 minutes)

that with a longer training phase with more tasks there might be an observable difference.

C. Example-problem blocks

[3] compared using example-problem blocks, example problem pairs, alternating similar problem-solving task, and blocks of problem-solving tasks.

The tasks were 6 pairs of LISP programming questions, to solve after having gained some familiarity with LISP before the experiment proper started. Each pair tested the same skills, with one being the source problem and the other target. The idea was that the source provided a chance to initially learn to solve the problem, and the target allowed them to practice the techniques learned from the source. For the example-problem pairs and block conditions source problems were swapped for a worked example. In the block conditions, sources were separated from targets whereas in the pair conditions targets immediately followed sources.

Example-problem blocks were the worst preforming group in posttests. They spent as much time studying source examples as the example-pair group, and spent more time on the target problems. They suggested that difficulty in remembering the examples once they met the equivalent problem would hinder later problem solving, and that if students are unable to recall the appropriate example, the benefit of studying them over problem solving disappears. Indeed, both the problem-solving groups performed better than example-problem blocks group, suggesting the extra practice and fine-tuning afforded to the problem-solving block group outweighed the benefits of having worked examples. The example-problem pairs were the best performing group on posttests.

D. Example-problem pairs

As mentioned, [11] compared worked examples on their own, example problem pairs, problem example pairs and problem solving, and found example-only an example-problem pairs to work more effectively than the other conditions, and example-only and example pairs performed similarly, for reasons already suggested.

As mentioned, [3] compared example-problem pairs to example-problem blocks and alternating or blocked problem solving. The best performing group in posttests were the example-problem condition, who spent as much time studying examples as the blocked examples group.

[10] explores research comparing backwards and forwards fading with example-problem pairs. Previous research found the fading conditions in three experiments (one backwards for statistics problems, the other forward for physics problems, and another with both forward and backward fading) performed more accurately on posttests with regards to near transfer problems, though not far transfer problems. Students also produced fewer errors during learning. This suggests fading may offer better learning outcomes in a shorter amount of time for near-transfer tasks than example-problem pairs.

[6] explores the use of backwards fading with and without self-explanation prompts, compared to example-problem pairs with and without prompts for solving statistics problems. Under the equivalent prompt or no prompt conditions, backwards fading resulted in higher posttest results than example-problem pairs on both near and far transfer problems. An explanation of the effects of self explanation prompts is given later.

E. Faded worked examples

Asides from the research above from [10], which compared both backwards and forwards fading favorably to example-problem pairs, [10] conducted their own experiment to explore whether the sequence of fading affected near and far transfer more than the types of steps removed. In their two experiments no difference was found in learning outcomes or errors during learning. Previous research had also suggested that backwards fading worked would produce better results than forward fading on near transfer items. However, their own experiments did not confirm this. Their results also suggested that students learn most about those steps which are faded. The implication is that the learning activities elicited by removing steps focuses students on those area. For this reason, they suggest the earlier results must be attributed to the learning material they used and the type of the steps removed. The backward procedure removed those steps that may be 'prerequisites' or otherwise helped students learn principles which were helpful for earlier steps. Doing it the other way means they would not learn the important principles first, which would hinder subsequent learning.

As mentioned above, [6] found that backwards fading resulted in higher posttest results than example-problem pairs on both near and far transfer problems.

[9] compares forward fading and backward fading. Those who used forward fading were found to outperform those using backwards fading. They suggest this has to do with the ease of the material they were learning. Having studied the first example, they may have gained all the initial knowledge they needed. According to the expertise reversal effect, if a student already has some expertise in the area, further learning is better gained by problem-solving and techniques like worked examples may hinder or decrease subsequent performance. This is

because for an expert, studying a worked example is a source of ECL rather than GCL, and problem-solving promotes GCL for those with some expertise in the targeted domain. Because forward-fading gets students to start problem-solving as the first, rather than the last step, the early problem-solving may have benefited students as opposed to those who had to wait until the final step to problem-solve.

F. Self-explanation prompts

[6] cites research with mixed results on the effects of activities designed to illicit self explanations. It has been suggested that self-explanations are a source of GCL helping students to from schemas around the materials they're learning, rather than e.g. just memorizing a set of steps to a solution. Experiments where students were prompted by an online tool to fill in templates for self-explanations , or where students were encouraged to write their own self-explanations as comments, failed to increase learning gains consistently. Another study found self-explanation prompts during the problem solving phase rather than during example study received positive results on learning.

In their own study into solving statistics problems, [6] prompted students with a set of principles a given step in the worked examples may be drawing from. Students were expected to choose one of the principles, and this was expected to foster self-explanations. Students in the self-explanation groups performed better on posttests for near and far transfer problem than those not prompted in the equivalent fading or example-pair groups not prompter. No extra time was required to achieve this result. The results for self-explanation prompts with backwards fading were replicated for university and high school students.

G. Subgoal labeling

[5] studied the use of subgoal labeling in video demonstrations and instructional material for creating mobile applications. In the subgoal conditions, the steps in the demonstration video and instructional material were labeled with subgoals grouping several steps into a cohesive group.

In posttests participants in the subgoal group better identified subgoals necessary to complete a solution whether or not they complete it correctly or not. They also were more likely to correctly complete the subgoals necessary for the assessments. Overall the subgoal condition outperformed their counterpart on both assessments immediately after training and assessments one week later. They did so spending less time on the assessments, and were less likely to drag out blocks in the assessments.

H. Summary

The evidence suggests certain worked example techniques (primarily example-problem pairs and faded worked examples) are an improvement over standard problem solving techniques, in terms of learning time and performance on near transfer tests in novices.

Though not all research indicates it is successful in easy or low effort tasks, when the student is not a novice in the domain the worked example target, it does appear to indicate faded worked examples in particular improve performance and decrease learning time on near transfer tasks, and techniques such as self-explanation prompts may promote far transfer as well if applied appropriately.

Since this research are on well structured domains like statistics, physics and programming, and given the well structured nature of Computer Science, much of this research provides promising avenues for teaching Computer Science. Research in adopting e.g. faded worked examples with self explanation prompts in teaching Computer Science as in [6] may yield positive results. As [10] suggests learning may improve if the correct sequence of fading is chosen, research into what steps should be faded first for a given problem would also help us understand how faded worked examples could most effectively be employed.

VI. Examples in Computer Science

There is not a lot of research into worked examples in Computer Science. Early research into CLT drew upon work in teaching LISP (e.g. [1]), where it was observed that students would rely heavily on provided examples as opposed to instructional texts. Much of the worked example literature rely on the results of these studies, but nonetheless worked examples have not been well explored or used in Computer Science education, as [8] and [7] observed. We've explained some examples of work in this area in Computer Science ([3], [5]).

One promising paper providing a framework to study or implement faded worked examples in Compuetr science is "Suggestions for Graduated Exposure to Programming Concepts Using Fading Worked Examples" [2], which we examine.

A. Suggestions for Graduated Exposure to Programming Concepts Using Fading Worked Examples

[2] provide a set of suggestions for implementing faded worked examples for an introductory course in C++ programming.

They decompose the task of programming in to components whose cognitive load they claim can be adequately managed. Their decomposition is based on two parts: the abstract algorithmic dimensions and the associated concrete programming constructs. The algorithmic dimensions they identify are design, implementation and semantics (the meaning of supplied code). The semantic dimension is divided in three, into assertion (students should be able to state true statements about the code at various point of execution), execution (given an input, provide the output) and verification (be able to test the code). The programming constructs chosen were selection, iteration and subroutine calls. Each of these would be taught in pairs (design of a selection algorithm, implementation of an iterative algorithm etc.), with the learning of each pair supported by sets of faded worked examples.

The provide concrete fully worked examples for all of the design-construct and implementation-construct pairs, and provide an example of semantic-assert and semantic for selection algorithm. They're suggestions have yet to be tested, but it does provide a concrete example for how faded worked examples could be used in Computer Science.

However, they could do with a greater justification of the use of backwards fading. Though backwards fading is often found to work well on reducing learning time or improving near transfer compared to some alternatives, [10] suggests this is an artifact of the teaching materials people use rather than something inherent in the backwards sequencing. The sequencing if fading could be examined to see which steps may be prerequisites for understanding other steps - [10] suggests these kinds of steps should be removed first.

Also, they suggest using 'ASSERTS' during the semantic part of training. This is designed to get students to assert what is known about certain parts of code in the form of code comment. This is motivated by the same principals motivating the use of self-explanation prompts in [6]. However, their explanations of how they will teach students to provide such assertions on their is light though their example does have an explicit process for coming up with assertions for selection, will this be helpful on problems which do not follow he same format?

As mentioned earlier, the research on self-explanation prompts is not unanimous. [6] suggests the interface allowing students to write down self-explanations may have an effect on whether it will be effective, and the prompts they provide in their own experiment require students to make choices from a list, rather than generating them on their own. This requires a low amount of activity from students. The scaffolding provided means they won't have to come up with assertions from scratch like in some previous studies, but the suggested 'ASSERTS' require a little more than picking options from a list. They themselves state "Including ASSERTs can be challenging for a student. However, the payoff is promising" [6]. What makes it challenging, and will this be too much for a novice or struggling student? Will the challenge be too great a source of ECL, negating the self-explanation effects like in some earlier studies cited in [6]? A greater examination on this point is required, both theoretical and empirical.

However, all in all, this paper still provides a clear framework for using and testing faded worked examples in Computer Science. Such techniques could straightforwardly be exteded to other C derived languages like C, Java or C♯, or any kind of imperative or procedural language. Other constructs or dimension of programming could be considered too.

REFERENCES

[1] John R. Anderson, Robert Farrell, and Ron Sauers. Learning to program in lisp. *Cognitive Science*, 8(2):87 – 129, 1984.

[2] Simon Gray, Caroline St. Clair, Richard James, and Jerry Mead. Suggestions for graduated exposure to programming concepts using fading worked examples. In *Proceedings of the third international workshop on Computing education research*, ICER '07, pages 99–110, New York, NY, USA, 2007. ACM.

[3] J. Gregory, Gregory Trafton, and J. Reiser. The contributions of studying examples and solving problems to skill acquisition, 1993.

[4] PaulChandler PaulSweller John Kalyuga, SlavaAyres. The expertise reversal effect. *Educational Psychologist*, 38(1):23 – 31, 2003.

[5] Lauren E. Margulieux, Mark Guzdial, and Richard Catrambone. Subgoal-labeled instructional material improves performance and transfer in learning to develop mobile applications. In *Proceedings of the ninth annual international conference on International computing education research*, ICER '12, pages 71–78, New York, NY, USA, 2012. ACM.

[6] R. Mason and G. Cooper. Transitioning from studying examples to solving problems: Effects of self-explanation prompts and fading worked-out steps. *ournal of Educational Psychology*, 95(4):pp. 774–783, 2003.

[7] R. Mason and G. Cooper. Why the bottom 10implication for introductory programming courses. In M. de Raadt and A. Carbone, editors, *Australasian Computing Education Conference (ACE2012)*, volume 123 of *CRPIT*, pages 187–196, Melbourne, Australia, 2012. ACS.

[8] Jeroen J.G. Van Merrinboer and Fred G.W.C. Paas. Automation and schema acquisition in learning elementary computer programming: Implications for the design of practice. *Computers in Human Behavior*, 6(3):273 – 289, 1990.

[9] R. Moreno, M. Reisslein, and G.M. Delgoda. Toward a fundamental understanding of worked example instruction: Impact of means-ends practice, backward/forward fading, and adaptivity. *Frontiers in Education, Annual*, 0:5–10, 2006.

[10] Alexander Renkl, Robert K. Atkinson, and Cornelia S. Grosse. How fading worked solution steps works - a cognitive load perspective. *Instructional Science*, 32(1-2):59–82, 2004. Date revised - 2010-09-01; Language of summary - English; Pages - 59-82; ProQuest ID - 811266033; Last updated - 2011-11-07; Corporate institution author - Renkl, Alexander; Atkinson, Robert K; Grosse, Cornelia S; DOI - OB-ccf31a17-c4ea-457b-a734mfgefd108; 13660440; 0020-4277; 1573-1952.

[11] Tamara van Gog, Liesbeth Kester, and Fred Paas. Effects of worked examples, example-problem, and problem-example pairs on novices learning. *Contemporary Educational Psychology*, 36(3):212 – 218, 2011.