Computer lab work on theory

Emma Enström
School of Computer Science and Communication, KTH
Lindstedtsvägen 3
SE-100 44, Stockholm, Sweden
emmaen@csc.kth.se

Viggo Kann
School of Computer Science and Communication, KTH
Lindstedtsvägen 3
SE-100 44, Stockholm, Sweden
viggo@nada.kth.se

ABSTRACT

This paper describes an attempt to introduce computer lab exercises on NP-completeness proofs in a class already containing computer lab exercises on algorithms and data structures. In the article we are interested in the answer of the following question: Can the students feel that their understanding of theoretical computer science is improved by performing a computer lab exercise on the subject?

The class is mandatory for students in a computer science program, and is taken by about 130 students each year. Theory of NP-completeness proofs with reductions has previous years been examined on an individual assignment with written solutions handed in and later explained orally by the student to a teacher. The new assignment is performed as a computer lab exercise where students are working in small groups of two. This exercise is placed before the individual assignment, and is examined first by running automated test cases and later by an oral presentation in lab to a teacher.

An improvement can be observed of the students' average results since the new assignment was introduced. This is not enough to prove the benefit of using the new assignment. However, the responses to questionnaires at course evaluations show that almost all students think that the assignment really gave them better understanding of polynomial reductions in NP completeness proofs. The students' result on the new assignment corresponds closely to their results on the following individual assignment. Seemingly, the new assignment predicts accurately who is going to pass the following assignment.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science education

General Terms

Experimentation, Theory

Keywords

reductions, learning theoretical computer science, computer lab work

1. INTRODUCTION

Computer lab assignments in computer science education are a common and successful way to get students to train their ability to solve programming problems, to program, and to make them learn different constructions in programming languages and concepts of programming. They are nowadays used in almost every computer science course except in plain theory courses. We think that there are elements of theoretical computer science that may be trained in computer lab assignments, which is traditionally not done, except for algorithms and automata simulations [8].

Algorithms, Data structures and Complexity is a compulsory course in the second year of the Computer Science and Engineering Master of Science program. The course is taken by about 130 students each year. Two thirds of the course is theoretical and one third consists of lab assignments. The theory of NP-completeness proofs and reductions has previously been examined mainly in an individual home assignment, where the student hands in written solutions and later explains them to a teacher. Since this assignment was considered difficult by many students we invented a new computer lab assignment, on a basic difficulty level, which students work with in small groups of two. This assignment is placed before the individual assignment, so that the students should be better prepared to solve the harder problem. In this paper we describe this new assignment and evaluate it using student feedback, and we also try to measure how the student performances have changed. The computer lab assignment is aided by an automated "judge", a system originally constructed for running test suites on submissions to programming contests, the same judge that is used for other lab assignments in this course.

Among the other exams for this class are two individual home assignments, one on algorithms and the one on NP completeness proofs mentioned above, a final written mandatory exam on the basics of theory, and an optional oral exam for higher grades. The class was a class that the students needed to work continuously with from start, already before the new assignment was added. Perhaps it is so that they worked continuously with other things than the theory of NP previously.

Parallel to the introduction of the new assignment, the grading system was adapted to the European Credit Transfer System ECTS with rewritten criteria and the structure of the existing written exams slightly altered. Because of this it is problematic to compare results from the periods before and after the introduction.

2. PREVIOUS STUDIES

Reductions as a "habit of mind" is a concept dealt with by education research [3](p 133) . In the context of proving NP-hardness the originally "hard" subject of reductions is complicated by using reductions backwards or as a thought experiment. You do not solve a new problem by using a previously known solution of another problem, you assume that you can solve the new problem and show that then you would also be able to solve a known NP-hard problem. It is possible that all proofs by contradiction are considered unnatural by the students, but here the principle only resembles proof by contradiction. (Assuming you can do this, you would also be able to do that). Reductions could be considered to be a *threshold concept* [7], proving hard to learn by many students, but once mastered a new level of understanding of complexity is reached.

Higgins and Bligh implemented an assignment in a "diagram based" domain with automatic assessment [5], including producing certain components for the diagrams for students to use. They argue that one of the benefits is that the students can practise an unlimited number of times before committing on a final version of their diagram, increasing their performance considerably. They also argue that formative assessment is considered expensive or resource-intense and automatic assessment is considered the opposite, making it ideal to perform formative assessment by automatic assessment methods and tools. The less positive findings were that some parameters are completely left aside of the process, for instance the readability of the diagram produced, good overall feedback on the students' performance and meaningful naming of the diagram components. Also there could be too much focus on shortcomings as the feedback is associated with failing test cases. The authors account for several criteria of good formative feedback [5].

These criteria are that good formative feedback should facilitate self-assessment, encourage dialogue between students and with teachers, clarifying what is to be considered "good performance", provide opportunities to improve, provide information to students about their progress, encourage positive beliefs, and provide information to the teacher about how the teaching was working.

In [9] Suleman describes a system, resembling the one used in this study, for automatic assessment. Among the conclusions are that some students set off to try to outwit the system rather than solving the problem, and actually sometimes believe that they have solved the problem in their assignment while the system is still making trouble. Other findings are similar to [5], the inability of the system to "correct" the solution aesthetically, for instance concerning variable names, and also the inability to deal with *almost* successful solutions.

Often presumed benefits from lab exercises in science, or rather arguments that are used to recommend lab exercises since they are presuming beneficial effects on learning, are described by Hult [6](p 48) as giving support for learning that is meaningful, socializing the student into the scientific society, providing skills, and motivating the students.

Hult's attitude towards these presumed benefits is critical. It cannot be taken for granted that the laboratory exercises really fits into the class at all. He argues that the students learn specific techniques and methods, without even having a fair chance of understanding them. A majority of the students can leave the lab without understanding what they have been doing. In addition to that, the exercises are designed to reinvent knowledge or illustrate theories instead of, as in research, to invent and discover new knowledge. He also means that post-lab activities should be used more often to show the connection between lab exercises and theory.

Hemmi [4] writes about two aspects of the concept of proof, explanation and conviction, and that they can occur separately. Students who meet proof only to explain theories are not as familiar with the aspect of convincing an audience of the truth in some claim, and therefore rather use more intuitive approaches to get convinced. This is done also among mathematicians, but they use the intuitive feeling of something being true as an excuse for constructing a proof.

3. THE NEW ASSIGNMENT

The design of the new exercise was motivated by the wish to give the students tools for evaluating their solutions and get some feedback before presenting their solutions to the teacher, in a part of the course that each year revealed a need for more practicing among many students: the subject of NP-completeness proofs. This should be done without causing extra work load on the teachers. Standard benefits of formative and automated assessment are valid in this situation. The need for an extra assignment was dealt with by translating an existing "algorithm" for proving NP-hardness to an exercise in transforming input from one problem to input for another. Given that problem A is known to be NP complete and problem B is a problem previously unknown, then for any instance x of A you can represent the reduction as

```
A(A's input x) =
  y ← some transformation of x
  return B(B's input y),
```

where B(y) has the same answer as A(x). Correctness of the reduction, and that the problem is actually in NP, is proven by the students in an oral presentation after they have worked in pairs with the exercise. We provide input specifications for the problems involved, and the students write programs that via standard in and standard out perform necessary operations on inputs of one problem to transform it to an instance of another problem. During their work, they are allowed unlimited number of submissions of their program to an automated assessment system, that runs a test suite and complains by email if the test cases fail. The system was designed to aid programming contests but is also used for checking correctness of other laboratory exercises of this particular course. It is relatively similar in function as the system web-based system architecture described by [1]. The system can either compare output of the students' program with output from a reference solution, or test the output of the students' program for certain criteria. There are mainly two things that the students can get help with from the system in this case: if they are trying to reduce the wrong problem, they will produce illegal data format and get complaints that reveal what kind of data the system expected, and if there are special cases that they have not thought about, the instance of problem B could have

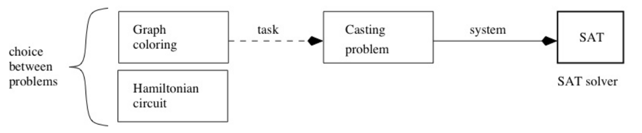

Figure 1: Overview of the task and the reductions

the wrong answer and then they know that they are not finished yet. To know, before presenting the solution to a teacher, that the solution meets with all criteria that the test case constructor could come up with, is supposedly releasing some pressure on the students as their thoughts can not be completely wrong.

The assignment considered the computational complexity of a "Casting problem". Given a set of actors, a set of roles with lists of possible actors for the role, and a set of scenes where different roles played together and hence was not allowed to be occupied by the same actor, the problem is to determine whether it is possible to fill all roles according to the restrictions. An additional restriction is that two of the actors are very valuable to the people responsible of casting and should always be assigned at least one role each. Since they hate each other, these divas are never allowed to play in the same scene. The task was to show the NP-hardness of the Casting problem by implementing a reduction of either Graph coloring or Hamiltonian circuit to it. The reduction is straightforward from Graph coloring, which the students are supposed to realize.

Test cases for the automated testing are instances of graph coloring where the system knows the answer. Since the casting problem instances, that the programs of the students produced, cannot be predicted, we have reduced the casting problem to SAT and an existing SAT solver handles the problem instances, see Fig. 1.

There is one technical complication in the setup, namely that the students might very well produce instances of the casting problem that reduced to SAT cannot be solved by the SAT solver in reasonable time. This mainly happens when the students perform unnecessary or dysfunctional operations in their program. Then the system will abort and report that it is not capable of checking this submission. The students then have the choice to try changing their program to make it pass all tests in the stipulated time, or , convinced that they are right, still present their solution to a teacher.

The main didactical complication is that the students can perceive the assignment as a marked trail, where they by trial and error are supposed to get the system to accept their solution. The risk of this is meant to be reduced by the fact that there are two problems to choose from, but in reality a student can go for the same choice as everyone else. There is also a wish from the students' side to get more details of the test cases, which in this assignment would support the trial-and-error strategy that would place the correctness proof construction *less* in focus. Therefore, the error messages from the system are deliberately sparse. At the same time, it would be convenient to give more detailed hints at some point during the process.

The assignment was presented as an experiment in 2007 and some students volunteered to try it. After that, another group of students were required to solve the same problem as compensation for previous failures at or lack of results for the individual assignment on the subject. All students' results, questions and assignment and course evaluation answers with comments were collected and analyzed qualitatively, and some improvements were made to the assignment. After this, it became a mandatory part of the class and has been used as such for two years. The students' grades on different exams before and after the assignment becoming mandatory have been compared, and course evaluations have been analyzed.

4. RESULTS

The results were observed more attentively in 2007, and the internet evaluation was more extensive that year. Main differences between the different years is that a larger group of students claimed that they learned from the assignment in 2008 and 2009. The way the students behaved and the type of questions they often asked were similar. Some students were trying to write a program to solve the casting problem. This resulted in error messages on the data format, which at least could help the students to understand that something was wrong. The students as a group could be considered to have a practical orientation towards the assignment. Some questions were emailed to teachers where students asked about the data formatting, both as a result of trying to reduce the wrong problem and for other reasons. The data formatting issues seemed to be considered the key to success.

4.1 Interaction with the system

When they submitted solutions to the system, the students often continued doing this repeatedly even after their solution was accepted. Their interaction with the system running the test suite was iterative. The design clearly promotes competition. They competed for the shortest running times as this was the only quality criteria returned when a solution was accepted.

4.2 Connection with reality

Comments from a few students about "not introducing any scenes that did not exist in the first place" as a reason for not dealing with the reduction in the simplest possible way showed that they expected a little too much from the reduction. They actually wanted a direct connection, perhaps 1-1, between the problems. When you have an arbitrary instance of graph coloring, you need not to be concerned with whether there were any scenes in that instance. The scenes are part of the instances of casting. The students were not comparing structures, they were seeking for the "right" way to represent graph coloring as casting.

Others tried to get rid of unnecessary information that was allowed in both graph coloring and casting, for instance double edges, or attempted to first find out which roles where all in the same scenes as one another. That would correspond to solving another NP complete problem, clique, which is not allowed when performing a reduction that has to have polynomial running time.

Some comments from students during exams revealed that they interpreted the method as "showing equivalence between problems" not on a structure level but on a concrete level – all instances of the new problem would need to get (at least or exactly) one instance of the known problem "mapped" on it in order for this idea to work. Trying to get convinced that this will happen includes something similar to trying to reduce in both directions between problems.

This is typically more complicated than designated by the assignment constructor, and the student that attempts this will indeed feel bothered by the problem.

4.3 Evaluation of the assignment

A moderate try to evaluate the results of year 2008 and 2009 shows that the average grade for the second individual assignment, on the topic of NP completeness proofs, had increased slightly compared to the grade on the first individual assignment (on algorithms), suggesting that the assignment on NP completeness proofs was these years not perceived as more difficult than the rest of the course, as it was in 2007. Results are shown in Table 1.

Year	Assignment 1		Assignment 2		$\frac{\text{failed2}}{\text{failed1}}$
	avg. grade	failed	avg. grade	failed	
2007	3.3	9%	3.2	13%	1.4
2008	1.5	22%	1.8	21%	0.95
2009	2.2	17%	2.3	14%	0.82

Table 1: Average grades on the individual assignments. Absolute values of grades are not comparable between years.

Since there could be other causes for the difference in performance, these results need to be accompanied by other indications. In the evaluations 2008 and 2009 as many as 78% and 69%, respectively, answered that the lab assignment gave them better understanding of NP reductions, and only 6% and 3%, respectively, did not know NP reductions beforehand and did not get better understanding.

There is also a correlation between the students' results on the computer lab assignment and the following individual assignment on the same subject. If nothing else, the results on the lab assignment seem to predict whether or not the student will pass the next assignment.

5. DISCUSSION

How do the descriptions by Hult [6] translate to computer science, and theoretical computer science in particular? By meaningful we can refer to learning of general skills and habits of mind. The term can be explained by opposing it to learning details about problems or algorithms, or scaffolding for learning the specific details about how to solve the casting problem instead of the concerns to deal with while constructing a NP completeness proof. The socialization process relevant for theoretical computer science can also be thought of as assimilating habits of mind. It could be gradual adaptation of the way of expecting, dealing with and producing formal proofs along the practical work, or thinking of reductions as a general method for solving problems as well as for proving. The skills relevant in computer lab exercises in general are programming and problem solving skills, but in theoretical computer science also proving skills are relevant. The motivational part of the new exercise seems to be the automated testing system. Interaction with the system, that is, iterative submissions of code to it, is frequent even after the system accepting the solution.

The examination demands that the student understands what has been showed by the testing system and where to give complementary information to fulfill the proof. The process, however, is not likely to stress all parts of the proof equally. Few students think it is relevant to formally state why their solution has a particular design, even if they give vague arguments that shows what they have been considering during the design process.

What students already know of when starting with the assignment is firstly the system. This could give them a push in the right direction. Further, they have been participating in classes where NP-completeness is discussed and have, if they are ambitious, practised on several problems provided by the teachers.

5.1 Framing of the assignment

Framing of the assignment is essential for its ability to fulfill its purposes. During the first year, the framing was not the desired one. The framing of the assignment once it became mandatory fits to the description of pre- and post-lab activities described by [6]. With questions on theory hinting important features to take into account, the assignment being placed where its learning outcome supposedly is high and it becoming a mandatory part of the examination, the assignments fits into the class and can also be used at a time where it can support learning.

5.2 The criteria of good formative feedback

The assignment described in this article matches some of the criteria accounted for by [5]. It encourages to dialogue since the students are working in pairs and it provides opportunities to improve since there is no limit of submissions to the assessment system. The information to the students is not considered high-qualitative by the students, but it is not clear whether they are comparing the information in the feedback to intended purposes or to expectations of feedback far beyond what we want to provide. The teacher gets information on the students succeeding or not, but no specific information about how to improve teaching. Positive beliefs are probably only encouraged by the students that are successful in their task, but on the other hand this is meant to be, and is, a vast majority. However the failure in submitting a program that is accepted might enforce negative beliefs, especially if "everyone else" apparently manages it well. The assignment is not flexible in what is requested by the students to get certain feedback, and this is one point on which improvements would be desirable but difficult.

5.3 Clashing of purposes

That students were eager to have more information about the data formatting maybe depended on the way the system gave its error messages – it mainly complained about data formatting. Complaints occurred when the students tried to do something less correct and the system was unable to interpret that. Students were also concerned about the feedback. Since the error messages contained information on what the system expected to read, this ought to result in the student finding out about his or her mistake. Of course, the system will not be able to give feedback on the original idea of the students', it can only tell whether the data supplied by the students' programs is a valid instance of the casting problem. If so, it solves the instance produced by the program and compares the solution to the known solution of the problem instance of graph coloring. When these match, the program is accepted. This does not mean that anything goes at the presentation – if there are severe errors the students will not pass the presentation even if the system accepted their program. However, this was not the issue that the students were complaining about. They wanted to know the test case specifications, they wanted information like "you failed on this particular test case". The level of detail that we would like to supply is between the current level (wrong format or direction of reduction/fail/pass) and test case specifications.

The iterative interaction combined with the requests for more information about the test suite can be interpreted as attempts to use the system for test-driven development. This might be a useful strategy for professionals, but in that case they also have to construct their own test cases. Here, two aims of the assignment clash, namely the aim to give feedback through the process and the aim to give the students an opportunity to practise reductions in the context of NP before the individual assignment. Proving a reduction is correct can hardly be handled by an unknown automated test suite, it lies in the design of test cases. Our findings here are consistent with those of Suleman [9] – students can be working on outwitting the system, with the assignment only as side-scene if they solely wish to get acceptance from the system.

At least two different approaches are possible to avoid the clash between different aspects of an assignment. One origins from programming contests, where students interaction pattern differs very much from when used in class. In a programming contest, submitting a solution that is not accepted renders penalty. This feature is not used in class because it would make it impossible to aim for feedback during the process, and the findings of Higgins and Bligh [5] support this choice as they have seen students' results improve when allowed to practise in interaction with an assessment system an unlimited amount of iterations. The other alternative, that could keep the students from requesting more test info, is to simply ask them another question: to provide a test suite that can be compared to the present one by the system. Instead of performing the reduction, the task would be designing test cases that could be used for determining whether a solution is likely to be OK. Perhaps this better resembles the task to perform a reduction with its correctness proof. The black box would not contain test cases, rather different programs designed to (not) perform the task of reducing in various levels of success. As a second step, the students could do the present assignment, reducing with support from a system.

5.4 The expectations of "reality"

The tendency to treat the reduction as a connection between problems, or instances of problems, already existing can be compared to the conclusions of [2, 3], that students tended to reduce abstraction. Some students that tried to solve the casting problem instead of reducing graph coloring, perhaps considered it "cheating" not to solve it, as the students of Armoni et al. If you are merely stating that instances of one problem can be transformed to a subset of the instances of the new problem, this does not seem enough to some students.

5.5 Further evaluation

It is well-known that it is troublesome to obtain certain proof of this assignment being successful. If we consider the new assignment useful and important, it would not be ethical to not give it to all students, so there can be no control group. One other idea could be producing a different type of assignment, that we also believe is beneficial. Then the students could be divided into groups presenting different assignments and their results could be compared. Unfortunately, if neither of the assignments would be good, this would not be visible in that comparison. It is possible to argue that the mere addition of an assignment makes the students study harder, and therefore increases their knowledge. In that case, we can assume that any assignment added to the class would have provided an opportunity to learn more for the students. This makes it somewhat more interesting to compare assignments without being able to measure the benefit.

6. CONCLUSIONS

We have shown that the addition of a computer lab exercise on theory is likely to improve the possibilities of learning theory in class. However, it is obviously a difficult task to evaluate teaching methods, especially if they are supposedly beneficial for the students and already running.

7. REFERENCES

[1] M. Amelung, P. Forbrig, and D. Rösner. Towards generic and flexible web services for e-assessment. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 219–224, New York, NY, USA, 2008. ACM.

[2] M. Armoni. Reductive thinking in a quantitative perspective: the case of the algorithm course. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 53–57, New York, NY, USA, 2008. ACM.

[3] M. Armoni, J. Gal-Ezer, and O. Hazzan. Reductive thinking in undergraduate CS courses. In *ITICSE '06: Proc. 11th ann. SIGCSE conf. on Innovation and technology in Comp. Sci. education*, pages 133–137, New York, NY, USA, 2006. ACM.

[4] K. Hemmi. *Approaching Proof in an Community of Mathematical Practice.* PhD thesis, Stockholm University, Dep. of Mathematics, Stockholm, 2006.

[5] C. A. Higgins and B. Bligh. Formative computer based assessment in diagram based domains. In *ITICSE '06: Proc, 11th ann. SIGCSE conf. on Innovation and technology in Comp. Sci. education*, pages 98–102, New York, NY, USA, 2006. ACM.

[6] H. Hult. Laborationen – myt och verklighet. CUP:s rapportserie 6, Linköpings universitet, 2000. Critical review of different arguments for including laboratory exercises in science classes.

[7] J. Meyer and R. Land. *Overcoming Barriers to Student Understanding: Threshold concepts and troublesome knowledge.* Routledge, 2006.

[8] S. Rodger and T. Finley. *JFLAP – An Interactive Formal Languages and Automata Package.* Jones and Bartlett, 2006.

[9] H. Suleman. Automatic marking with Sakai. In *SAICSIT '08: Proc. 2008 ann. research conf. of the South African Inst. of Comp. Sci. and Information Technologists on IT research in developing countries*, pages 229–236, New York, NY, USA, 2008. ACM.