Exploring two strategies for teaching procedures

Antonija Mitrovic, Moffat Mathews and Jay Holland

Intelligent Computer Tutoring Group, University of Canterbury, Christchurch, New Zealand
{tanja.mitrovic, moffat.mathews, jay.holland}@canterbury.ac.nz

Abstract. Due to high cost and complexity of Intelligent Tutoring Systems (ITS), current systems typically implement a single teaching strategy, and comparative evaluations of alternative strategies are rare. We explore two competing strategies for teaching database normalization. Each data normalization problem consists of a number of tasks, some of which are optional. The first strategy enforces the procedural nature of the data normalization by providing an interface that requires the student to complete the current task (i.e. a part of the problem) before attempting the next one. The alternative strategy provides more freedom to the student, allowing him/her to select the task to work on. We performed an evaluation study which showed that the former, more restrictive strategy results in better problem-solving skills.

Keywords: teaching strategies, procedural tasks, evaluation

1 Introduction

Ideally, ITSs should support multiple teaching strategies and adapt them for each student. Current ITSs typically implement a single teaching strategy, due to high development costs. Different teaching strategies might require a lot of development work; for example, the system's interface might need to be changed to support a different style of interaction. There are also difficulties in the evaluation of ITSs. For those reasons, evaluating competing teaching strategies for the same domain is rare.

Many factors influence ITS design, such as the limited capacity of working memory [1], the cognitive load [2] and the nature of the task. Instructional tasks can be arranged on a spectrum from strictly procedural (sequential), in which the student needs to learn a well-defined algorithm, to non-procedural, in which students are free to start from any part of the problem or apply actions in any order [3]. The solution search space for sequential tasks is much smaller than that of non-sequential tasks [4], as the student only has to concentrate on the solution space for a part of the task rather than the whole task. An example non-procedural task is software design: the student does not have to start at a particular point and there is no sequence they must follow; the solution search space is much higher as they keep track of what they have done, the consequences of what they have done, and what is left to do.

So, how should one teach procedures to novices? In this paper, we explore whether there is a difference between forcing students to adhere to the sequence of actions or leaving them to answer problems steps in any order they wish. Our hypothesis is that

adfa, p. 1, 2011.

students taught via the non-sequential method would be less efficient and solve fewer tasks, while students in the sequential method group would tackle more problems, and also more complex ones, have a higher rate of success, and be more efficient.

We discuss data normalization in the following section. Section 3 then presents two versions of NORMIT, implementing the sequential and a less-restrictive strategy. We present the study and its results in Section 4, and end with conclusions.

2 Data normalization

Data normalization is the technique of refining an existing relational database schema in order to ensure that all relations are of high quality [5]. Normalization is usually taught via a series of lectures that introduce the relevant concepts followed by paper-based exercises. Students find data normalization very difficult [6, 7], as it is very theoretical and requires a good understanding of the relational data model, various types of keys (primary, candidate, foreign keys and superkeys), Functional Dependencies (FD), normal forms and normalization algorithms.

Data normalization is a procedural technique: the student goes through a number of tasks to analyze the quality of a database. Each problem consists of a relation schema and a set of FDs (which does not have to be complete). For example, the student might be given a relation R(A, B, C, D, E) (typically the semantics of the attributes id not given) and the set of FDs: {A → B, AB → C, D → AC, D → E}.

The normalization procedure as implemented in NORMIT consists of eleven tasks described below. Please note that we refer to elements of the procedure as *tasks* rather than *steps*, as each of them contains a number of actions the student has to perform, including in some cases relatively complex algorithms. Therefore we refer to them as tasks to make it clear that the tasks are relatively complex compared to what is generally assumed by a step in the ITS research. The first eight tasks are necessary to determine the highest normal form the relation is in. If the relation is not in Boyce-Codd Normal Form (BCNF), the student needs to apply the relational synthesis algorithm to derive an improved database schema via tasks 9-11.

1. Identify the candidate keys for the given table. There may be one or more keys in a table; e.g. the only key in the above problem is D.
2. Find the closure of a given set of attributes. In the above example, to make sure that D is the key of relation R, we could determine that its closure consists of all attributes of relation R.
3. Identify prime attributes. Prime attributes are those attributes that belong to any candidate keys. In the above problem, D is the only prime attribute.
4. Simplify FDs by applying the decomposition rule, if necessary. In this task, a FD with more than one attribute on the right-hand side (RHS) is replaced with as many FDs as there are attributes on RHS. In the above problem, D → AC would be replaced with two FDs: D → A and D → C.
5. Determine the normal forms for the given relation.
6. If the student specified that the relation is not in 2NF, he/she needs to identify FDs that violate that form (i.e. partial FDs).

7. If the student specified that the relation is not in 3NF, he/she needs to identify FDs that violate that form (i.e. transitive FDs).
8. If the student specified that the relation is not in BCNF, he/she will be asked to identify FDs that violate that form.
9. For relations that are not in BCNF, reduce LHS of FDs. This task checks whether some of the attributes on the LHS can be dropped while still having a valid FD.
10. Find minimal cover (i.e. the minimal set of FDs).
11. Decompose the table by using the minimal cover.

3 Two versions of NORMIT

NORMIT [8, 9] teaches data normalization in a task-by-task manner, showing only one task at a time which the student needs to complete before moving on to the next task. The student can submit a solution at any time, which the system then analyses and presents feedback. At any point during the session, the student may change the problem, review the history of the session, examine the student model or ask for help on the current task. The system currently contains 50 problems and new problems can be added easily. NORMIT is a constraint-based tutor, and its knowledge base is represented as a set of 82 (problem-independent) constraints. Each constraint is relevant for a particular task of the procedure. Some constraints are purely syntactic, while others compare the student's solution to the ideal solution (generated by the problem solver). The short-term student model consists of a list of violated/satisfied constraints for the current attempt, while the long term model records the history of usage for each constraint. Please see [8] for information about NORMIT.

The original version of NORMIT enforces the procedural nature of the data normalization by forcing the student to complete the current task before being able to move on to the next task. An alternative strategy would allow the student to work on any task of the procedure in any order. To implement that strategy, we developed a less restrictive interface which shows all the tasks on a single page, thus allowing the student to approach the problem in different ways. In order to work on a particular task, the student clicks the *Edit* button which expands the page by adding specific elements for that task. The functionality provided by the modified interface is essentially the same as in the original tutor, but the interaction is slightly different. We also had to modify the system's knowledge base to support this new style of interaction. In the original version of NORMIT, constraints are task-specific: the very first test in each constraint specifies the task the constraint is relevant for. In the new version, the student is free to select the task, and therefore the constraints cannot be restricted to specific task. There are 75 constraints in the non-procedural version of NORMIT.

4 Evaluation Study

We performed an evaluation study with the students enrolled in an introductory database course at the University of Canterbury. Our hypothesis was that procedural version of the tutor would result in higher learning in terms of problem-solving skills and

conceptual knowledge. Prior to the experiment, the students had four lectures on normalization. The study was conducted at the scheduled lab times on October 5th or 6th, 2011(the students were divided into two streams). The session length was 100 minutes. The students in the control group used the original, procedural version of the system, while the experimental group used the new, non-procedural version. The participation was voluntary, and 33 students participated in the study. All students enrolled in the course were free to use the system after the study if they so wished.

The students were randomly assigned to one of the two conditions, and were given an online pre-test, with four multi-choice questions. The initial two question required students to identify the correct primary key and the highest normal form for a given table. For the remaining two questions students needed to identify the correct definition of a given concept. A similar test was used as the post-test at the end of the sessions. Both tests were short on purpose as the session was of limited length. The consequence of short pre/post tests, however, is the limited coverage of the domain.

Table 1. Statistics from the study (standard deviations given in parentheses)

Group	Experimental (14)	Control (14)	Significant?
Pre-test mean (SD)	1.9 (1.3)	2.3 (1.3)	no
Post-test mean (SD)	3.3 (1)	3.5 (0.8)	no
Gain	1.5 (1)	1.2 (1.5)	no
Normalized gain	0.7 (0.4)	0.6 (0.5)	no
Improvement pre-to-post	t=5, t<0.01	t=2.9, p<0.01	
Time	68 (27)	74 (15)	no
Attempted problems	4.7 (2.2)	8.3 (3.3)	p<0.05
Solved problems	3.7 (2.4)	6.6 (3.7)	p<0.05
Attempts	33 (24)	101 (56)	p<0.01
Known at start	35 (15)	37 (17)	no
Learnt constraints	6.9 (5.9)	5.4 (3.7)	no
Used constraints	53 (18)	63 (20)	p=0.09
Problem complexity	2.3 (1.3)	4.4 (2.5)	p<0.01

We excluded data about students who interacted with the system for less than 10 minutes and/or have made no attempts at problems, which resulted in 14 students in each group (Table 1). There was no difference between the two groups on the pre-test and post-test performance, as well as on the gains, normalized gains and interaction time. Both groups improved significantly during the session (determined by comparing their pre/post test results by a matched t-test).

We then analyzed the learning behavior by examining the student logs. The control group students attempted and solved significantly more problems and made significantly more attempts than their peers. The latter is easy to explain: the control condition had to go through each task in order to solve problems, while the experimental condition participants could only work on a subset of tasks.

Another measure of learning is the number of constraints that were learnt during the session. To see whether a constraint is known at the start, we require that the student has applied it correctly on at least 4 out of 5 initial attempts at that constraint. As

reported in Table 1, there was neither significant difference on the number of constraints known at start, nor on the number of learnt constraints.

The control group participants attempted and solved approximately twice as many problems as their peers. At the first look, it seems contradictory that they acquired the same number of constraints and achieved similar results at the post-test as the experimental group. We therefore looked deeper into the logs. We identified all constraints relevant for attempts and report them in the *Used constraints* row of Table 1. There was a marginal difference in favour of the control group. Therefore, the control group students used more constraints to solve problems that the experimental group. A deeper look at the problems solved provides another interesting observation. The average complexity of problems solved by the experimental group is just over 2, while the control group solved problems of significantly higher complexities (the last row of Table 1). Given that there was no difference in background knowledge of the two groups, we can conclude that the significant difference in the problem-solving accomplishments comes from the difference in the interfaces. The procedural version of the tutor provided more guidance which in turn enabled the students to solve more problems, and also more complex problems, in the same amount of time.

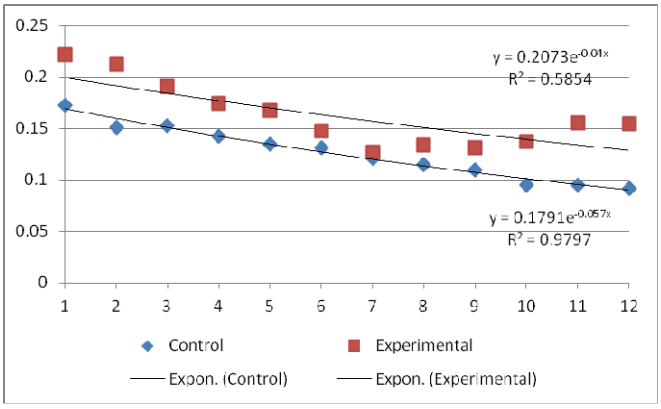

Fig. 1. Learning curves for the two groups

Figure 1 shows the learning curves for the two conditions (i.e. the proportion of violated constraints following the n^{th} occasion when a constraint was relevant, averaged across all students and all constraints). The R^2 fit to the exponential curve is good for the control, but is quite poor for the experimental group. The learning rate of the control group is also slightly higher. A closer inspection of the constraints learnt shows that the control group learnt more complex constraints, which is the consequence of higher average complexity of problems they attempted and solved.

5 Conclusions

There are many possible approaches to teach the same instructional domain. Due to high complexity of ITSs and high development costs, ITS developers usually imple-

ment only one teaching strategy. In this paper, we present two teaching strategies for data normalization, which differ in the amount of control students have in selecting which part of the problem to work on. The first strategy requires the student to follow the procedure closely, working on one task at a time and completing it before attempting subsequent tasks, while the other gives full control to the student. Our hypothesis was that the former strategy would result in better learning.

Our study shows that both strategies resulted in significant improvement from pre- to post-test. There was no significant difference between the two groups on the post-test; however, the post-test was short and its questions are of different nature compared to the problems in the ITS. We also looked at how many new knowledge elements (i.e. constraints) students learnt during the study. Although there was no significant difference in the amount of newly acquired knowledge, there was difference in the kinds of constraints learnt. The procedural version resulted in significantly higher number of problems attempted and solved in comparison to the non-procedural strategy. The average complexity of problems solved is also significantly higher in the case of procedural strategy. Therefore, closer adherence to the procedural nature of data normalization did result in higher problem-solving success.

Our study was of short duration and small in terms of the participants. We plan to to perform a bigger study in 2012 with NORMIT and also to conduct similar studies in other instructional domains.

References

1. Miller, G.A. The Magical Number Seven, Plus or Minus Two. The Psyhological Review, 63, 81-97, 1956.
2. Sweller, J. Cognitive Load Theory, Learning Difficulty, and Instructional Design, Learning and Instruction, 4, 295-312, 1994.
3. Mitrovic, A., Weerasinghe, A. Revisiting the Ill-Definedness and Consequences for ITSs. Dimitrova, V., Mizoguchi, R., du Boulay, B., Graesser, A (eds) Proc 14th Int Conf AIED pp. 375-382, 2009.
4. McCallum, A. K. Learning to Use Selective Attention and Short-Term Memory in Sequential Tasks. Proc. 4th Int. Conf. Simulation of Adaptive Behavior, 315-324, 1996.
5. Elmasri, R., Navathe, S. B. *Fundamentals of database systems*. Addison-Wesley, 2010.
6. Kung, H.-J., Tung, H-L., An alternative approach to teaching database normalization: A simple algorithm and an interactive e-Learning tool, *Journal of Information Systems Education*, 17(30), 315-325, 2006.
7. Phillip, G. C. Teaching database modeling and design: areas of confusion and helpful hints, *Journal of Information Technology Education*, 6, 481-497, 2007.
8. Mitrovic, A. The Effect of Explaining on Learning: a Case Study with a Data Normalization Tutor. In: C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (eds) Proc. Artificial Intelligence in Education, IOS Press, pp. 499-506, 2005.
9. Mitrovic, A. Fifteen years of Constraint-Based Tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22 (in print) http://dx.doi.org/10.1007/s11257-011-9105-9