Evaluating adaptive problem selection

Antonija MITROVIC and Brent MARTIN

Intelligent Computer Tutoring Group
Computer Science Department, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{tanja,brent}@cosc.canterbury.ac.nz

Abstract: This paper presents an evaluation study that compares two different problem selection strategies for an Intelligent Tutoring System (ITS). The first strategy uses static problem complexities specified by the teacher to select problems that are appropriate for a student based on his/her current level of ability. The other strategy is more adaptive: individual problem difficulties are calculated for each student based on the student's specific knowledge, and the appropriate problem is then selected based on these dynamic difficulty measures. The study was performed in the context of the SQL-Tutor system. The results show that adaptive problem selection based on dynamically generated problem difficulties can have a positive effect on student learning performance.

1. Introduction

Adaptivity is central to many modern computer systems, especially Web-enabled ones. An adaptive system makes the user's task simpler or, in some cases, doable. Intelligent educational systems also adapt to each individual student's needs, learning abilities and preferences. However, there is an important distinction between educational systems and other applications whose goal is to support users in performing specific tasks. Intelligent educational systems must support the student in learning a task, and therefore should support all aspects of the task to be learned. In contrast, the goal of other types of adaptive applications is to help the user perform a task faster or more efficiently. The support needed in educational systems therefore differs significantly from that needed by other kinds of adaptive systems. One of the crucial differences is that support in educational systems should fade over time, to allow the learner to resume control over the process, become independent and acquire metacognitive skills.

One of the adaptive decisions that ITSs make is problem selection. An appropriate problem is one that is challenging for the student, but still not too hard: the student should be able to solve the problem with the system's support. Most ITSs select problems based on the state of the student model (i.e. based on the student's knowledge), thus providing adaptive problem selection.

In this paper, we compare two problem selection strategies, both of which use the student model to adaptively select a problem. The difference between these strategies is that one of them uses static problem complexities assigned by the

domain expert, while the other one dynamically computes the difficulty of the problem for the given student at a certain moment during learning. The motivation for these measures comes from Brusilovsky [2]. Within the ITEM/IP system, task sequencing was based on problem *complexity* (also referred to as structural complexity [3]) and problem *difficulty* (also referred to as conceptual complexity [3]). In this system, the problem complexity was a static measure of how complex the problem was, in terms of the number of statements needed in the solution. On the other hand, problem difficulty was dynamically computed from the student model, and represented the number of concepts the students does not know.

In the next section we discuss the system we used in the study, and the two different versions of it that implement the two problem-selection strategies. In Section 3 we present our hypotheses and the design of the experiment. Section 4 presents the results, while the conclusions are given in Section 5.

2. SQL-Tutor and its versions used in the study

The goal of this project is to investigate two different problem-selection strategies, and to determine whether the more dynamic approach better supports learning. We performed an experiment in the context of SQL-Tutor, an intelligent tutoring system that teaches the SQL database language to university-level students. For a detailed discussion of the system, see [5, 7]; here we present only some of its features. SQL-Tutor consists of an interface, a pedagogical module—which determines the timing and content of pedagogical actions—and a student modeller, which analyzes student answers. The system contains definitions of several databases and a set of problems and their ideal solutions. Each problem is assigned a static level of complexity by the domain expert based on the domain concepts that are necessary to solve the problem. SQL-Tutor contains no problem solver: to check the correctness of the student's solution, SQL-Tutor compares it to the correct solution, using domain knowledge represented in the form of more than 600 constraints. It uses Constraint-Based Modeling (CBM) [8, 9] for both domain and student models.

Student may select problems in SQL-Tutor in one of several ways: they may work their way through a series of problems for each database, ask the system to select a problem on the basis of his/her student model, select a problem from the list, or select a type of problem they want to work on such that the system selects a problem of that type on the basis of their student model. For this study we developed two versions of the system, differing from each other in the problem selection strategy. In order to select a problem, both strategies use the student model to determine which problems are appropriate for the current level of student's knowledge.

In both versions of SQL-Tutor used in this study, when the student asks for a new problem, they will be presented with a page showing their student model and asking them to select the type of problem they would like to work on. This encourages the student to reflect on their knowledge in order to identify the type of problem they have difficulties with. To support this reflection we open the student model to the users. The constraint base of SQL-Tutor is large, and therefore it is not possible to show the student's progress directly in terms of constraints. Instead, we collapse the student model into six parts, corresponding to the six clauses of an SQL query. A previous study [6] showed that such a visualization of the student

model has a positive effect on learning, especially for less able students, and helps students select appropriate problems.

Figure 1 presents a screenshot from SQL-Tutor, showing the page for problem selection. The student model is displayed to the student in terms of their progress over the six clauses. To measure progress on a clause, we compute the percentage of constraints relevant to that clause that the student has used so far. The student model tracks how the student has used each constraint, and computes an estimate of the student's understanding of the constraint based on the last *n* uses of that constraint. We use these estimates to compute how well the student knows all the constraints relevant for the clause. The correctly known constraints are shown in green (the first part of the bars in Figure 1), while the ones the student has

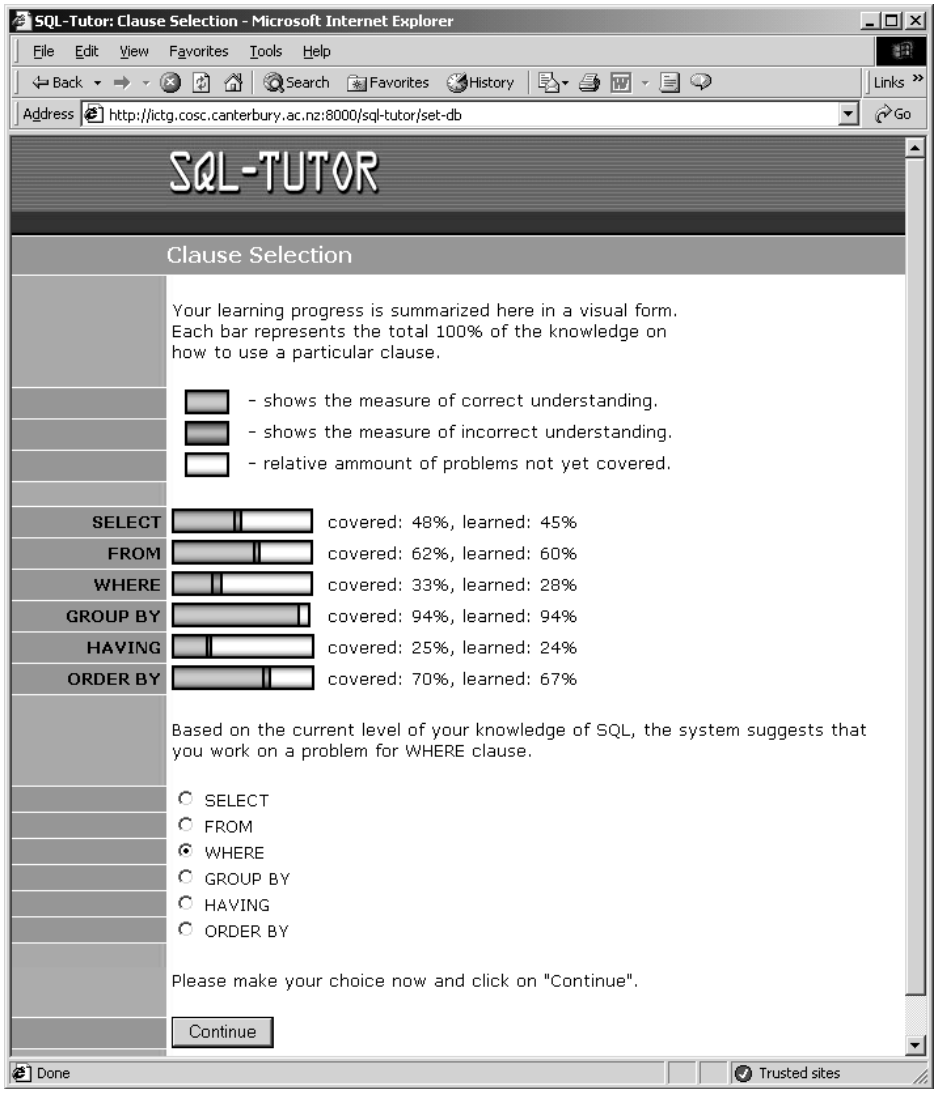

Fig. 1. The clause selection page from SQL-Tutor

problems with are shown in red (second segment of the bars). The total shows the coverage of a particular constraint.

SQL-Tutor suggests the type of problem the student should work on. In Figure 1 for example, the system suggests that the student works on the WHERE clause. To suggest a clause, the system checks the student level, which ranges from 1 to 9 and is proportional to the number of constraints the student knows. If the student level is less than 3, the system selects one of the initial three clauses (SELECT, FROM and WHERE), for which the problems are easier. For students whose level exceeds this threshold, the system selects any one of all six clauses. SQL-Tutor then selects the candidate clause that the student has had most problems with. This is based on a simple measure: we find all constraints relevant for a clause and average the probabilities that the student knows these constraints.

Once the type of problem (i.e. the clause) has been selected, SQL-Tutor searches for problems of the appropriate type. Out of all problems relevant to the chosen clause the system selects ones that are at the appropriate level for the student. These are the problems whose levels equal or exceed the current student ability level.

The level of the problem differs in the two versions of the system used in this study. The control version uses the static, pre-defined problem complexity, which is determined by the domain expert. In contrast, the experimental version computes the problem difficulty dynamically based on the student model. The problem difficulty ranges from 1 to 9, and is computed as the (scaled) weighted sum of probability of the student having already learned each constraint relevant for the problem. The weight of each constraint depends on the number of tests it contains; the probability that the student has learned the constraint is simply the proportion of correct applications of the constraint for last five times that it was relevant. This is similar to, but simpler than, a method we have previously used in 2001 [4]. The 2001 study suggested that such an approach might be more effective than using static problem difficulty, but the result was confounded by the two systems having a different problem set.

Figure 2 shows the page with all problems from the PRODUCTS database relevant for the WHERE clause. The problems are sorted according to the complexity/difficulty, and one of the problems is highlighted as the preferred problem (e.g. problem 105 in Figure 2). The student selects the problem by clicking the problem number. The student is therefore free to either accept the suggested problem or select any of the other available problems, including previously solved ones. The order of the problems gives them help in making their selection.

3. Experiment Design

We hypothesized that problem selection based on the dynamically computed problem difficulty would be superior to that based on static problem complexities. To evaluate this hypothesis, we performed an experiment with the students enrolled in an introductory database course at the University of Canterbury. Participation in the experiment was voluntary. Prior to the study, students attended four lectures on SQL and had two labs on the Oracle RDBMS. There were two additional lectures on SQL during the experiment, and a series of three more labs. SQL-Tutor was demonstrated to students in a lecture on 15th September 2003. The

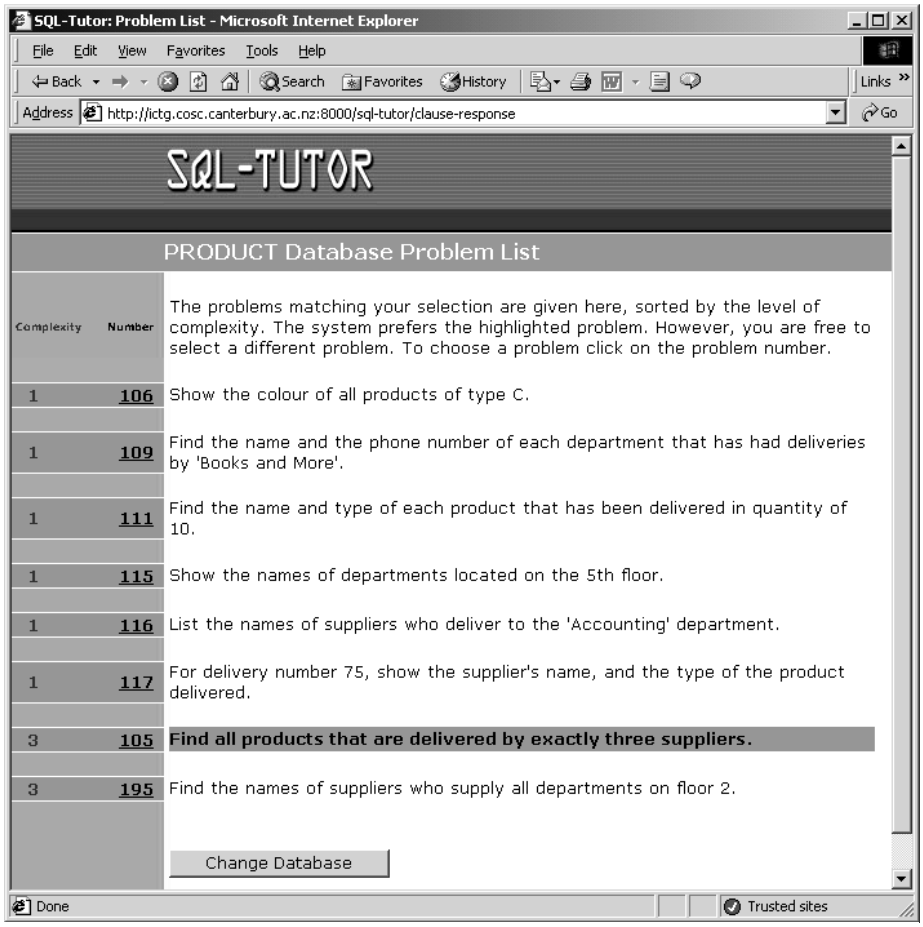

Fig. 2. The problem selection page from SQL-Tutor

experiment required the students to sit a pre-test, which was administered online the first time students accessed SQL-Tutor. The pre-test consisted of four multi-choice questions. Two questions contained the text of a problem for which students were asked to select the correct SQL query. The other two questions asked about SQL constructs.

The students were randomly allocated to one of the two versions of the system. The course involved a test on SQL on 16th October 2003, which provided additional motivation for students to practise with SQL-Tutor. The post-test was administered online the first time a student logged on to the system on or after 15th October 2003, and consisted of four questions of similar nature and complexity as the questions in the pre-test. The maximum mark for the pre/post tests was 4.

As well as analysing the pre- and post-test results, we also analysed the learning rates students exhibited while using the two systems, to determine what dependencies there were between prior ability (as defined by the pre-test scores) and performance while using the tutor.

Table 1. Some statistics about the groups

	Pre-test	Post-test	Lab test	Sessions	Solved	Time
Control (n=24)	64.58 (23.21)	67.86 (34.50)	61.87 (19.53)	3.08 (2.45)	15.87 (13.29)	182 (166)
Exper. (n=20)	50 (18.14)	54.17 (24.58)	59.25 (15.75)	2.6 (1.85)	14.2 (12.18)	136 (131)

4.Results

Of the 110 students enrolled in the course, 88 students logged on to SQL-Tutor at least once. The mean score for the pre-test for all students was 54.75% (sd=25.02%). The students were randomly allocated to one of the versions of the system, forming groups of similar size. A t-test showed no significant differences between the pre-test scores for the two groups. However, some students looked at the system only briefly. We therefore excluded the logs of students who did not attempt any problems. Further, we noted that some students had logged on to SQL-Tutor on the last day of the study, and therefore some of them submitted the post-tests before solving any problems. Such students have not benefited by working on the system, so we removed them also. The remaining logs were then analysed.

Table 1 gives the number of students in each group, their scores on the various tests (pre-, post- and the lab test), and some additional information about their logs. The maximal number of solved problems was 49 for the control, and 36 for the experimental group students. As can be seen, the performance of students who learnt with the system has improved on the post-test (although not significantly). There was also a significant difference (p=0.011) on the lab test-performance between all students who used SQL-Tutor before the last day of the study (mean=60.68%) and the rest of the class (mean=52.10%). However, this has to be taken with caution, as the study was voluntary, and the more motivated students who used the system might be more able as well.

We also compared the effect the two systems had on learning while the students were using them. Figure 3 plots "learning curves," indicating the proportion of

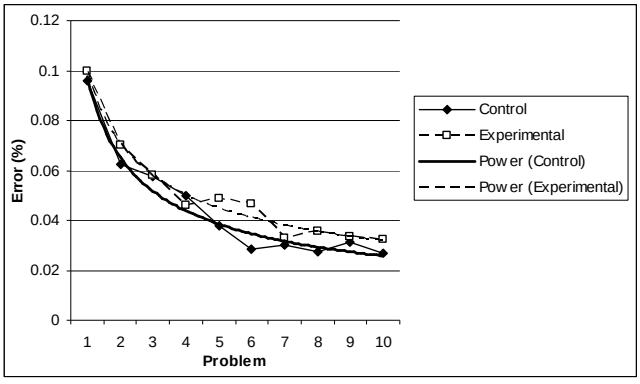

Fig. 3. Learning curves for the two groups

Table 2. Groups used for learning speed analysis

Pre-test score	N (control group)	N (experimental group)
1	9	9
2	11	17
3	9	4

times the constraints were violated for the n^{th} problem for which the student encountered that particular constraint. Such curves give a measure of how well the learners improved their performance with respect to the constraint set over time. While the curve is slightly steeper for the control group, it must be remembered this group had a higher average pre-test score, and may therefore represent more able learners.

We then considered whether the two systems might affect the learning rate differently for students of different ability, and hypothesised that dynamic problem difficulty might benefit students of a wider range of abilities. We therefore divided the two groups further according to individual pre-test scores and used those groups where there were a reasonable number of students with that score. Table 2 summarises the resulting groups.

Figure 4 shows the resulting learning curves for the two systems for each pre-test score. Note that the curves have been cut off at N=5 to counter the effects of having small sample sizes, because the number of participating constraints decreases with N. For the control group, the slope of the curve for a score of 2 is considerably greater than for scores of 1 and 3, suggesting that the static problem difficulty is more suited to intermediate learners than those with lower or higher initial ability, with lower ability learners faring very poorly. Conversely, for the experimental group students with a score of 1 or 2 perform about the same, while

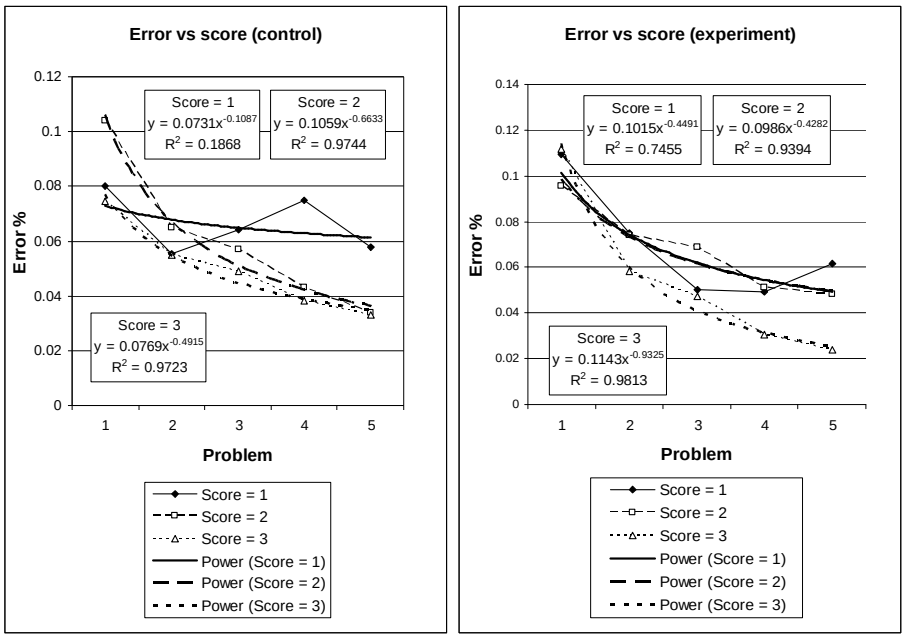

Fig. 4. Learning curves versus score for each group

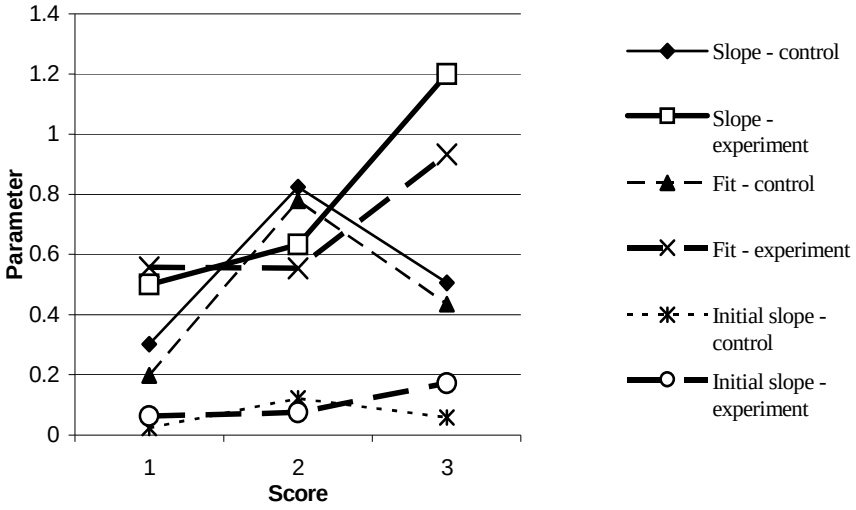

Fig. 5. Curve parameters versus score for each group

the more advanced students demonstrate a much higher learning speed.

We tested the statistical significance of these results by plotting individual curves for each student and comparing the average power curve slopes. We also computed the average *initial* slope and power curve *fit*. The initial slope gives a measure of the absolute decrease in errors after encountering a constraint once: the higher this value, the more learning is taking place. The power curve fit is a further indicator of how much learning is being achieved, by showing how well the students' performance fits the expected model of learning [1]. Figure 5 plots these values as a function of pre-test score for the two groups, while table 3 summarises those comparisons that displayed statistical significance at p=0.05. We also performed one-way ANOVA tests on the parameters for each of the two main groups, which indicated that the curve fit varied significantly for the control group (p=0.003) while the slope did for the experiment group (p=0.027).

The analysis of the power curves across scores suggests that the two methods differ in how they fit students of different abilities. The static problem difficulty appears to suit medium students (pre-test score = 2) much better than either beginner or advanced students; both the curve slope and fit peaks for this score. Conversely, the dynamic difficulty appears to be best suited to advanced students, while serving medium students moderately well, and improving the performance of

Table 3. Significant differences between control and experiment

Comparison	Control Mean	Control SD	Exper. Mean	Exper. SD	Significance
Initial slope, score=1	0.024	0.015	0.063	0.039	P=0.04
Curve fit, score = 1	0.20	0.14	0.56	0.31	P=0.03
Curve fit, score=3	0.44	0.36	0.93	0.03	P = 0.04

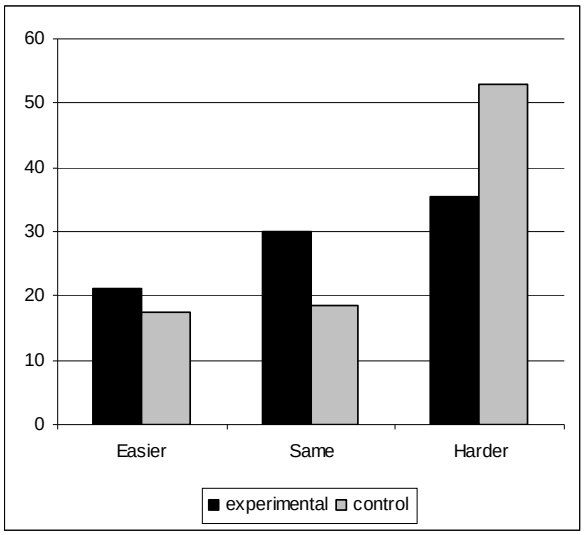

Fig. 6. Percentage of abandoned problems

lower ability students with respect to the static system. For the experiment group, the learning rate (slope) trends upwards with ability, suggesting that the system may possibly be advancing each group at a rate proportional to their ability.

Finally, we compared the students' behaviour when selecting a new problem to work on. The experimental group selected a total of 396 problems (an average of 20.84 each), while the control group had 530 selections (23.04). Both groups typically adopted the problem suggested by SQL-Tutor or selected another problem on the same level (67.93/70.75% for experimental/control groups). However, the control group students asked for more difficult problems in preference to easier ones (19.24/12.1%), while those in the experimental group were twice as likely to select an *easier* problem (17.93% versus 8.68%). From this analysis it seems that dynamic problem difficulties may cause more complex problems to be selected. Further, the success rate for a selected problem (Figure 6) increases fairly slowly relative to the difficulty of the problem selected (easier, same as or harder than the system selection) for the experimental group, whereas for the control group the proportion of problems abandoned varies considerably, with more than half of the more difficult problems not being completed. This suggests that dynamic problem difficulties are better matched to the student's level, in that the problems just outside the suggested difficulty are only moderately easier/harder, whereas for the statically ordered problems the student may find them considerably less/more of a challenge.

5. Conclusions

This experiment aimed to determine whether adaptive problem selection could improve the learning experience of students using an Intelligent Tutoring System. SQL-Tutor was modified to calculate problem difficulty based on individual knowledge elements (constraints) in the student model, and this was used to

adaptively select the next problem. The modified system was then evaluated against the standard SQL-Tutor using a class of University students.

There is a significant difference between the two problem selection strategies in the way they suit users of differing ability: dynamically computed problem difficulty performed well for a wide range of students, whereas static problem complexity performed well for students of intermediate ability, but fared badly for beginners and advanced students. The results were statistically significant. This suggests problem selection benefits from being more adaptive.

The experimental system described included a moderate level of adaptation: problem choice was guided by how each problem was mapped onto the student model, and this mapping had a fairly large granularity (there were only 9 difficulty ratings). However, there are other ways the system could be made more adaptive. For example, the student's level (against which each problem is compared) is derived from their coverage of the domain model, which could be problematic if there is a mis-match between this value and the calculated problem difficulties: the system may repeatedly give the students problems that are too hard or too easy. In [4] we used an adaptive student level that increased each time the student answered a problem correctly the first time, and decreased when they got an answer wrong. Such an approach may further improve the match between student and problem.

To be effective, Intelligent Tutoring systems need to be well matched to their students. Adaptive problem selection is just one area where this match can be strengthened. The results of this study indicate that such an approach has merit.

References

1. Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Erlbaum.
2. Brusilovsky, P. (1992) Intelligent Tutor, Environment and Manual for Introductory Programming. *Educational and Training Technology International*, 29(1), 26-34.
3. Brusilovsky, P. (1992) A Framework for Intelligent Knowledge Sequencing and Task Sequencing. In: C. Frasson, G. Gauthier and G. McCalla (eds) *Proc. ITS 1992*, Berlin: Springer-Verlag, pp. 499-506.
4. Martin, B. and Mitrovic, A., Automatic Problem Generation in Constraint-Based Tutors. In: S. Cerri, G. Gouarderes and F. Paraguacu (eds.) *Proc. 6th Int. Conf on Intelligent Tutoring Systems ITS 2002*, Biarritz, France, LCNS 2363, 2002: 388-398.
5. Mitrovic, A. (2003) An Intelligent SQL Tutor on the Web. *Artificial Intelligence in Education*, 13(2-4), 173-197.
6. Mitrovic, A., Martin, B. (2002) Evaluating the Effects of Open Student Models on Learning. In: P. de Bra, P. Brusilovsky and R. Conejo (eds) *Proc. AH 2002*, Springer-Verlag LCNS 2347, pp. 296-305.
7. Mitrovic, A., Martin, B., Mayo, M. (2002) Using Evaluation to Shape ITS Design: Results and Experiences with SQL-Tutor. *User Modeling and User-Adapted Interaction*, 12(2-3), 243-279.
8. Mitrovic, A., Ohlsson, S. (1999) Evaluation of a Constraint-based Tutor for a Database Language. *Artificial Intelligence in Education*, 10(3-4), 238-256.
9. Ohlsson, S. (1994) Constraint-based Student Modeling. In: Greer, J.E., McCalla, G (eds): *Student Modeling: the key to Individualized Knowledge-based Instruction*, 167-189.