Evaluating and Improving Adaptive Educational Systems with Learning Curves

BRENT MARTIN[1], ANTONIJA MITROVIC[1], KENNETH R KOEDINGER[2] and SANTOSH MATHAN[3]

[1]*Intelligent Computer Tutoring Group, Department of Computer Science and Software Engineering, University of Canterbury, Private Bag 4800, Christchurch, New Zealand.*
E-mail: {brent,tanja}@cosc.canterbury.ac.nz

[2]*HCI Institute, Carnegie Mellon University, Pittsburgh, PA 15213.*

[3]*Human Centered Systems Group, Honeywell Labs.*

Abstract. Personalised environments such as adaptive educational systems can be evaluated and compared using performance curves. Such summative studies are useful for determining whether or not new modifications enhance or degrade performance. Performance curves also have the potential to be utilised in formative studies that can shape adaptive model design at a much finer level of granularity. We describe the use of learning curves for evaluating personalised educational systems and outline some of the potential pitfalls and how they may be overcome. We then describe three studies in which we demonstrate how learning curves can be used to drive changes in the user model. First, we show how using learning curves for subsets of the domain model can yield insight into the appropriateness of the model's structure. In the second study we use this method to experiment with model granularity. Finally, we use learning curves to analyse a large volume of user data to explore the feasibility of using them as a reliable method for fine-tuning a system's model. The results of these experiments demonstrate the successful use of performance curves in formative studies of adaptive educational systems.

Keywords: empirical evaluation, intelligent tutoring systems, student modeling, user modelling, domain modelling, learning curves

1 Introduction

Adaptive educational systems such as intelligent tutoring systems (ITS) have user modelling at their core. Developers of such systems strive to maximise their efficacy through intensive evaluation and enhancements. ITS are complex and have many aspects that may affect learning performance, including the interface, pedagogy, adaptive strategies and the domain and student models utilised. Of these aspects the domain and student model are very important because they drive all other aspects of the system. Depending on the approach used the student model may be invoked for some or all of diagnosis, problem selection, feedback or hint selection/generation, argumentation, error correction and student performance evaluation. The student model is typically derived in some way from the domain model, e.g. an overlay, where the student model is considered a subset of the domain model, or a perturbation model, where it additionally contains some representation of the student's buggy concepts (Holt, Dubs et al., 1994).

Performance (*summative*) analysis of adaptive educational systems—such as ITS—is hard because the students' interaction with the system is but one small component of their education. Pre- and post-test comparisons provide the most rigorous means of comparing two systems (or comparing a system to pen-and-paper), but in order to be statistically rigorous they require a significant numbers of students and a sufficiently long learning period. The latter confounds the results unless it can be guaranteed that the students do not undertake any relevant learning outside the system being measured. Typically studies are conducted in a much more controlled manner, such as within a single session spanning one or two hours. With such evaluations useful results can be obtained but the effect size tends to be smaller. In such cases other differences may be found in how students interacted with the system, but they may be too little to give a clear test outcome, e.g. (Ainsworth and Grimshaw, 2004); (Uresti and Du Boulay, 2004). In contrast, Suraweera and Mitrovic (2004) did find significant differences between using their ITS (KERMIT) versus no tutor in a short-term study, but such a result appears rare.

Because of the lack of clear results researchers often measure other aspects of their systems to try to find differences in behaviour. However, these do not always measure learning performance specifically, and the results are in danger of being biased (e.g. Uresti and Du Boulay, 2004; Walker, Recker et al., 2004; Zapata-Rivera and Greer, 2004). Finally, many studies include the use of questionnaires to analyse student attitudes towards the system, but again student perception does not necessarily correlate with learning gain. To date summative analysis of adaptive educational systems remains a non-trivial problem.

We would also like to be able to analyse various components of our system and use the results to improve its performance (*formative* analysis). Whilst this is theoretically possible using pre- and post-tests, when fine-tuning parts of an educational system (such as the domain or student model) a large number of studies may need to be performed, dividing the students into many small experimental groups. Paramythis and Weibelzahl (2005) advocate breaking an adaptive system into layers and evaluating each in isolation which helps alleviate this problem, but this may fail to reveal complex interactions between the layers. In this article we focus primarily on the modelling layer, and in particular on the domain and student models. (An ITS may contain other models, such as pedagogical, curriculum sequencing, etc.) However, our approach is suitable for evaluating any aspect of the system.

In the case of adaptive educational systems the domain model can be very large (up to thousands of knowledge components[1]), and we need to be able to fine-tune this model to maximise learning. For example, we may wish to optimise the structure of the domain model by measuring the performance of each knowledge component in the model; traditional pre- and post-test analysis is infeasible because this would require sufficient groups to isolate the effect of each knowledge component. Another possibility is to plot *learning curves*, i.e. the error rate with respect to the number of times each knowledge component has been invoked. Learning curves thus give us a measure of the amount of learning that is taking place relative to the system's model. They can in theory be applied to any part of the model, from the entire system to individual knowledge components, making them a promising tool for formative evaluation. They also have the potential to enable quantitative comparisons between disparate systems. However, there are problems with such comparisons that need to be overcome, which we later discuss.

In this article we explore learning curves as an evaluation tool and conduct several experiments that seek to determine whether they can be used to refine the design of an ITS. The next section describes the two ITS that were used for these experiments. Section 3 introduces learning curves, while Section 4 describes their use for ITS evaluation and explores their strengths and weaknesses. Section 5 then discusses a study where we used learning curves in conjunction with domain model metadata to explore whether the level of granularity of our student model was appropriate. Section 6 goes further and explores using learning curves as a means of analysing the quality of a domain model when large amounts of data are available. Finally, in Section 7 we discuss the efficacy of using learning curves to shape ITS design and draw conclusions.

2 The experiment tutors

SQL-Tutor is an example of a practice-based ITS, which teaches the SQL database language to university-level students. Fig. 1 shows a screen shot of the tutor. For a detailed discussion of the system, see (Mitrovic, Martin et al., 2002; Mitrovic, 2003). SQL-Tutor consists of an interface, a pedagogical module—which determines the timing and content of pedagogical actions—and a student modeller, which analyses student answers. The system contains definitions of several databases and a set of problems and their ideal solutions. Constraint-Based Modelling (CBM) (Ohlsson, 1994) is used for both the domain and student models. Like all constraint-based ITS feedback is attached directly to the constraints. An example of a constraint is:

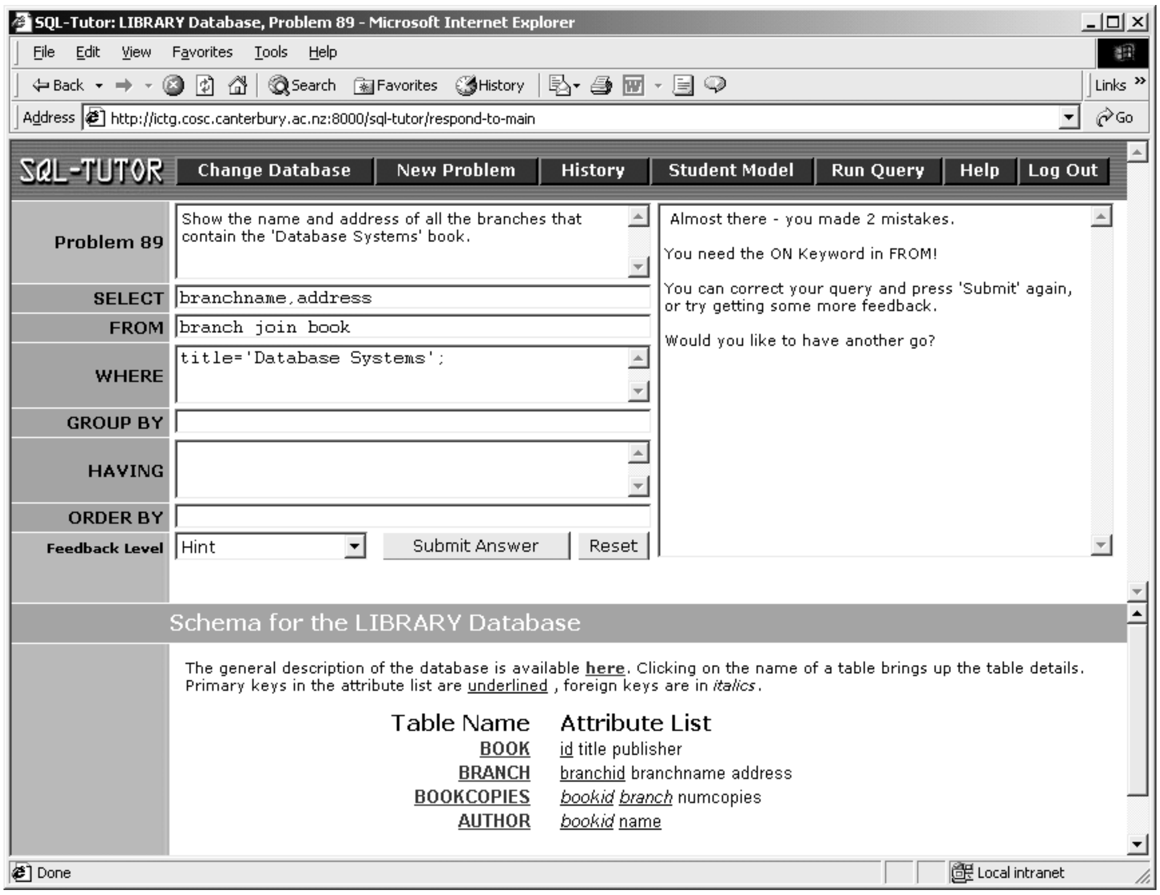
```
(147
"You have used some names in the WHERE clause that are not from this database."

; relevance condition
 (match SS WHERE (?* (^name ?n) ?*))

; satisfaction condition
 (or   (test SS (^valid-table (?n ?t))
       (test SS (^attribute-p (?n ?a ?t))))

; Relevant clause
"WHERE")
```

[1] We use the term "knowledge component" to generalize over different knowledge representation approaches, like schemas, constraints, or production rules (Koedinger, Corbett and Perfetti, submitted) and we show how learning curve analysis can apply despite potential differences in the details of these approaches.


SQL-Tutor: LIBRARY Database, Problem 89 - Microsoft Internet Explorer

Address http://ictg.cosc.canterbury.ac.nz:8000/sql-tutor/respond-to-main

Problem 89	Show the name and address of all the branches that contain the 'Database Systems' book.
SELECT	branchname,address
FROM	branch join book
WHERE	title='Database Systems';
GROUP BY	
HAVING	
ORDER BY	
Feedback Level	Hint

Almost there - you made 2 mistakes.

You need the ON Keyword in FROM!

You can correct your query and press 'Submit' again, or try getting some more feedback.

Would you like to have another go?

Schema for the LIBRARY Database

The general description of the database is available here. Clicking on the name of a table brings up the table details. Primary keys in the attribute list are underlined , foreign keys are in *italics*.

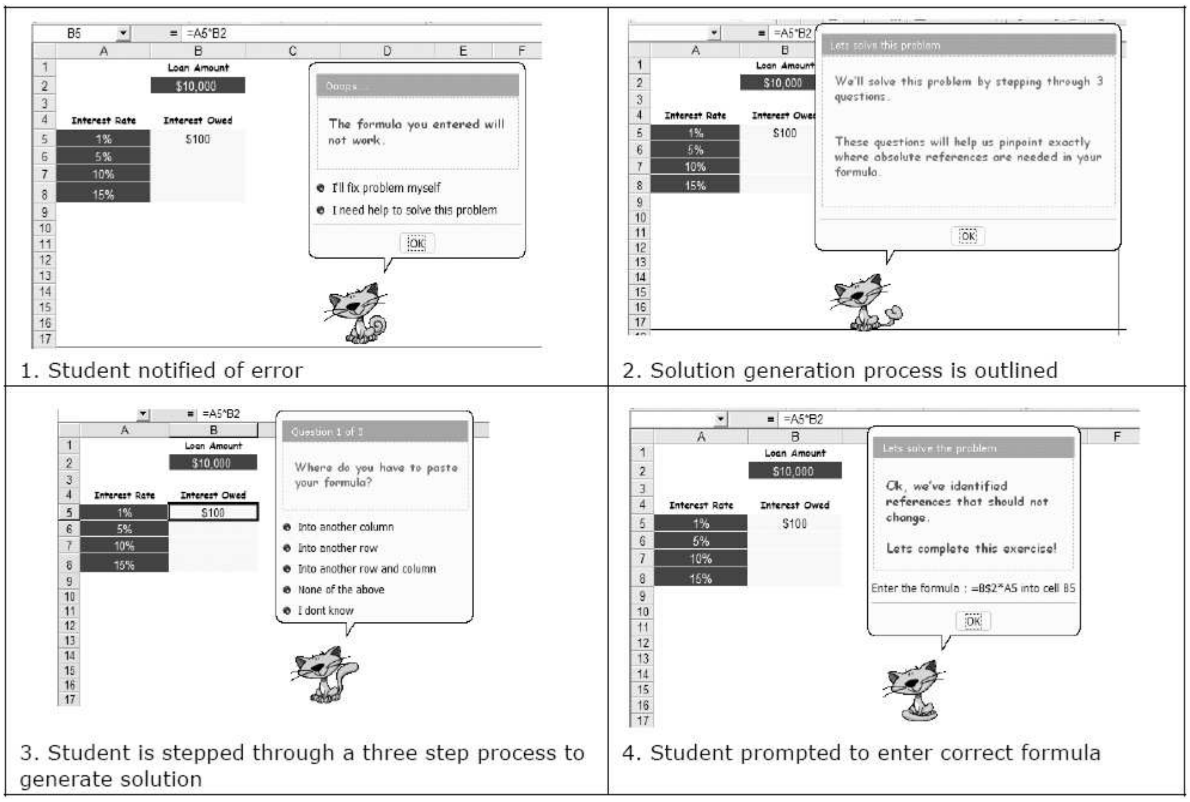
Table Name	Attribute List
BOOK	id title publisher
BRANCH	branchid branchname address
BOOKCOPIES	*bookid* *branch* numcopies
AUTHOR	*bookid* name

Fig. 1 A screen shot of SQL-Tutor.

Constraints are used to critique the students' solutions by checking that the concept they represent is being correctly applied. The relevance condition first tests whether or not this concept is relevant to the problem and current solution attempt. If so, the satisfaction condition is checked to ascertain whether or not the student has applied this concept correctly. If the satisfaction condition is met, no action is taken; if it fails, the feedback message is presented to the student.

In this case the relevance condition checks whether the student has used one or more names in the WHERE clause; if so, the satisfaction condition tests that each name found is a valid table or attribute name. The student model consists of the set of constraints, along with information about whether or not it has been successfully applied, for each attempt where it is relevant. Thus the student model is a trace of the performance of each individual constraint over time.

Constraints are just one representation used for domain and student models. Cognitive tutors (Anderson, Corbett et al., 1995) use *productions*, which are intended to represent the skills acquired by the student during learning. An example of a cognitive tutor is Excel-tutor (Koedinger and Mathan, 2004), which teaches students to use Microsoft Excel. Productions are used to track student inputs to assess whether or not they conform to a valid path or trace of the domain model. The model contains "buggy" productions as well as correct ones, the former being used to provide specific feedback when a student's action matches a common error pattern. The "correct" productions can be used to generate hints or provide an example if the student is unsure what to do next. Fig. 2 illustrates the style of interaction with this system. In this example the student has made an error and is being tutored on how to correct the problem.


1. Student notified of error

Ooops...

The formula you entered will not work.

- I'll fix problem myself
- I need help to solve this problem

2. Solution generation process is outlined

Lets solve this problem

We'll solve this problem by stepping through 3 questions.

These questions will help us pinpoint exactly where absolute references are needed in your formula.

3. Student is stepped through a three step process to generate solution

Question 1 of 3

Where do you have to paste your formula?

- Into another column
- Into another row
- Into another row and column
- None of the above
- I dont know

4. Student prompted to enter correct formula

Lets solve the problem

Ok, we've identified references that should not change.

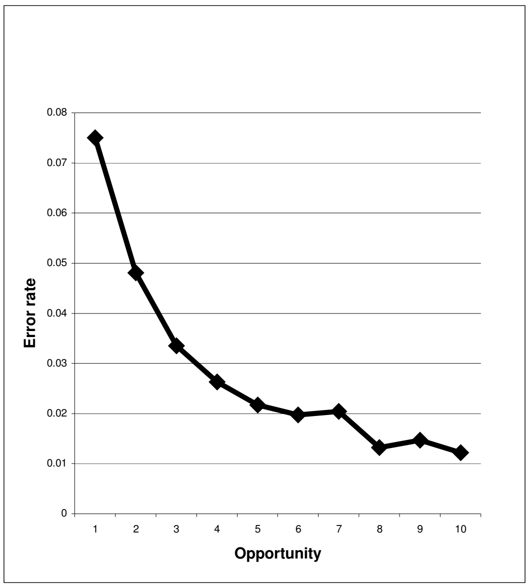
Lets complete this exercise!

Enter the formula : =B$2*A5 into cell B5

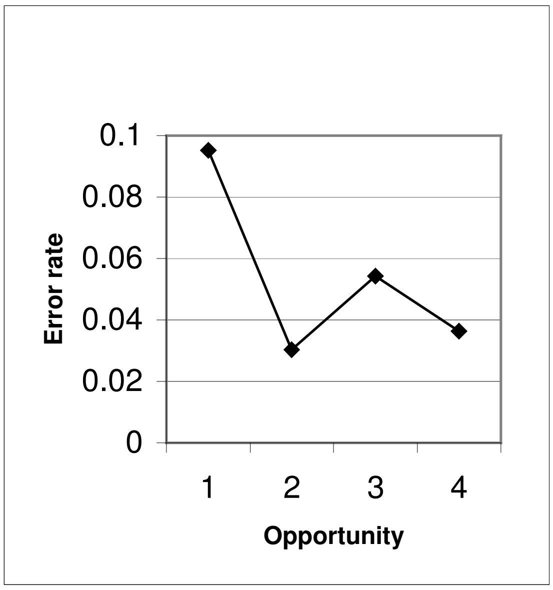
Fig. 2 Sample screenshots of interaction with the Excel Tutor.

3 Introduction to learning curves

A learning curve is a graph that plots performance on a task versus the number of opportunities to practice. Performance can be measured in several ways, with two common ones being time taken to complete the task and probability of making an error. In the case of a simple task such as catching a ball, the performance measure (e.g. number of drops) is measured for the participants for each attempt, giving the proportion of balls dropped for the first throw, then the second, etc. This data can then be plotted to see how, in general, the ability to catch a ball improves with practice. For more complex tasks, such as learning to drive a car, the task can be split up into various skills that are likely to be learned by practice; each skill is then assessed by measuring its performance for each opportunity to practice *that skill*. The resulting data can then be plotted for each skill or, alternatively, the data can be aggregated giving the performance of *any* skill as a function of the number of times it was practiced. If the task is learned with practice we expect to see a decrease in the error rate (or time taken). Fig. 3 shows examples of learning curves for SQL-Tutor; the first curve aggregates data for all students, while the second plots the performance across all knowledge components of an individual student. In both cases the likelihood of incorrectly applying a knowledge component is clearly decreasing with the number of opportunities to apply that component.



(a) Learning curve for all students



(b) Learning curve for an individual student

Fig. 3 Two sample learning curves for SQL-Tutor

3.1 The power law of practice

Learning curves can be analysed to *quantify* learning performance, an idea motivated by Newell and Rosenbloom (1981) who noted the existence of a "power law of practice" with respect to time taken in trials where a single task is repeated multiple times. The law of practice dates back to at least 1926 when Snoddy observed that when motor tasks are repeated (in this case mirror tracing of visual mazes) there is a rapid initial improvement in performance followed by a gradual reduction in the amount of improvement observed (Snoddy, 1926). This pattern has since been observed in many studies. The same power law is observed in industry. For example, the cost of producing an airplane decreases with the number of (identical) units built, presumably as a result of the staff becoming more efficient at their tasks (Wright, 1936).

Newell and Rosenbloom analysed several prior experiments where the task included perceptual-motor skills, perception, motor behaviour, elementary decisions, memory, complex routines and problem solving. For all of these tasks they observe that the time taken to complete the task decreases as a power law. Further, they report on a study by Stevens and Savin in which performance on eight tasks is plotted using a variety of measures including error rate as well as time (Stevens and Savin, 1962). From this analysis they conclude that the power law holds for error rate as well as time, and that the law "holds for practice learning of all kinds" (Newell and Rosenbloom, 1981). They also considered other curves that have been used in cognitive science, namely exponential, hyperbolic and logistic curves. Of these the most interesting is the exponential, because a power law can also be considered an exponential law where the rate of decay is decreasing over time. They tested each curve by subjecting the raw data from many experiments to an appropriate transformation function such that the expected result would be linear (e.g. log-log for a power law, log-linear for exponential curves). By observing any systematic error in each curve they concluded that the power law provided the best fit.

The formula for a power law is:

$$T = BN^{-\alpha} \tag{1}$$

Where *T* is the performance measurement (traditionally time) and *N* is the number of trials. The constant *B* represents the y-axis intercept, which for learning curves is the error rate at x=1, i.e. prior to any practice. *α* depicts the power law slope, equivalent to the linear slope when the data is plotted using a log-log axis. This indicates the steepness of the curve, and hence the speed with which performance is improving. Note however that, unlike an exponential curve (whose linear slope is a constant multiple of *T*), the rate of decay of the curve varies with the number of trials, i.e.

$$\frac{dT}{dN} = -\left(\frac{\alpha}{N}\right)T \tag{2}$$

Finally, the fit (R^2) of the curve is measured to give a quantitative judgement of how well the measured data follows a power law, and therefore the degree of evidence that learning is taking place with respect to the unit being measured. All of these might be used to compare two different approaches or models to determine which is better. They can also be applied to subsets of a model to look for additional trends. For example, Anderson (1993) uses the log-log slope (i.e. *α*) to compare the rate of improvement students exhibit between "old" and "new" productions (old productions are those that had been previously used in a lesson, versus new productions that were being used for the first time in the current session).

3.2 Learning curves and the evaluation of education systems

We can compare student performance between two different learning systems by comparing the parameters of their power laws. In particular, the slope of the power law indicates the speed with which students are improving their performance; a better education system would be expected to result in a "steeper" power law when plotting the average performance of all students. Second, the fit (R^2) measures the reliability of the power law. This latter parameter can be interpreted in two ways. First, if there is no learning taking place as a result of practice we may fail to observe a power law because the students' performance is essentially random with respect to the number of times they have practiced the particular task or skill. (Alternatively we might obtain a power law with zero slope if performance is constant over practice.) Second, a poor power law can indicate that the performance measure we are using does not capture what is being learned. For example, in a complex task such as computer programming the number of syntax errors made is unlikely to decrease via a power law because the complexity of the task will rise during a student's session and new concepts will be introduced. In contrast, student performance with regard to a particular skill (e.g. writing a "main" method in Java) would be expected to improve as a power law. A high degree of fit therefore indicates that the performance measure being used successfully captures student learning performance. Given two alternative models of student performance we can therefore generally say that the one showing the best fit is superior. This issue is discussed further in Section 4. Fig. 5 is an example of learning curves for two variants of an ITS, each exhibiting a good power law. We might use slope and fit to draw conclusions about the relative quality of the two systems.

Anderson (1993) makes extensive use of learning curves to evaluate various tutoring systems in an attempt to draw conclusions about the mechanisms of learning. In the LISP tutor he observes that not only does programming speed decrease with practice as a power law, but the mean error rate also follows a similar function. Anderson analysed the coding time for each individual knowledge component ("production") in his learning model; a regression analysis of coding time found that the strongest factor was the opportunity, i.e. the number of times this production was applicable, and that this relationship was log-log linear. He uses learning curves to assess both differences in behaviour of the tutoring system ("modality") and individual differences between students. In particular, he asserts that the consistency of quality of the learning curves across different tutoring conditions supports his ACT-R conception of the learning process. In other words, the quality of the learning curves is evidence of the quality of the model. This result was replicated for a geometry tutor where a significant log-log relationship was observed for three different tasks (selecting premises, specifying rules and selecting a conclusion). Further, a very important finding he reports is that learning curves for individual productions appear to follow a power law, despite the fact that each production is unlikely to be entirely independent. He uses this result to justify comparing the shape of different learning curves to draw conclusions about the appropriateness of a model's granularity: time taken as a function of individual knowledge components (production rules in the ACT-R model) produces a much steeper curve than time versus the type of inference required to solve a particular problem step (a coarser unit of measurement), and he therefore claims the individual knowledge components form a more accurate model. Based on this result we performed a similar experiment (sections 5 and 6).

From Anderson's work we can see that learning curves can be used to analyse many aspects of an adaptive education system. Some examples are: comparing two versions of the same system to evaluate incremental changes; comparing two disparate modelling approaches; diagnosing problems in a student model by analysing sub-components; comparing student cohorts; comparing models for different education systems (e.g. to observe transfer effects or generic knowledge components). In the specific context of ITS learning curves can be used to evaluate many *functions*, including feedback quality, feedback selection (i.e. conflict resolution when multiple errors exist), problem selection strategy, scaffolding and model correctness. Within the wider field of adaptive systems in general, learning curves might apply to all layers of Paramythis and Weibelzahl's evaluation model if we assume that practice with the system will lead to a measurable improvement in user performance, (typically task completion time) as a result of the user learning from the help the system gives them, the system learning to better support the user, or both, and that each layer can potentially have an impact on the user's performance.

When using learning curves to evaluate educational systems we need to select an appropriate performance measure. In the case of ITS a common approach is to measure the proportion of knowledge components in the domain model encountered by the student that have been used incorrectly, or the "error rate" and to plot this parameter as a function of the number of times they have had an opportunity to practice that particular knowledge component. Alternatives exist, such as the number of attempts taken to correct a particular type of error or the time taken to apply a unit of knowledge. The x axis generally represents the number of occasions the knowledge component has been used. This in turn may be determined in a variety of ways: for example, a single time unit may represent each *new problem* the student attempted that was relevant to this knowledge component, on the grounds that repeated attempts within a single problem are benefiting from the user having been given feedback about that particular circumstance and may thus improve from one

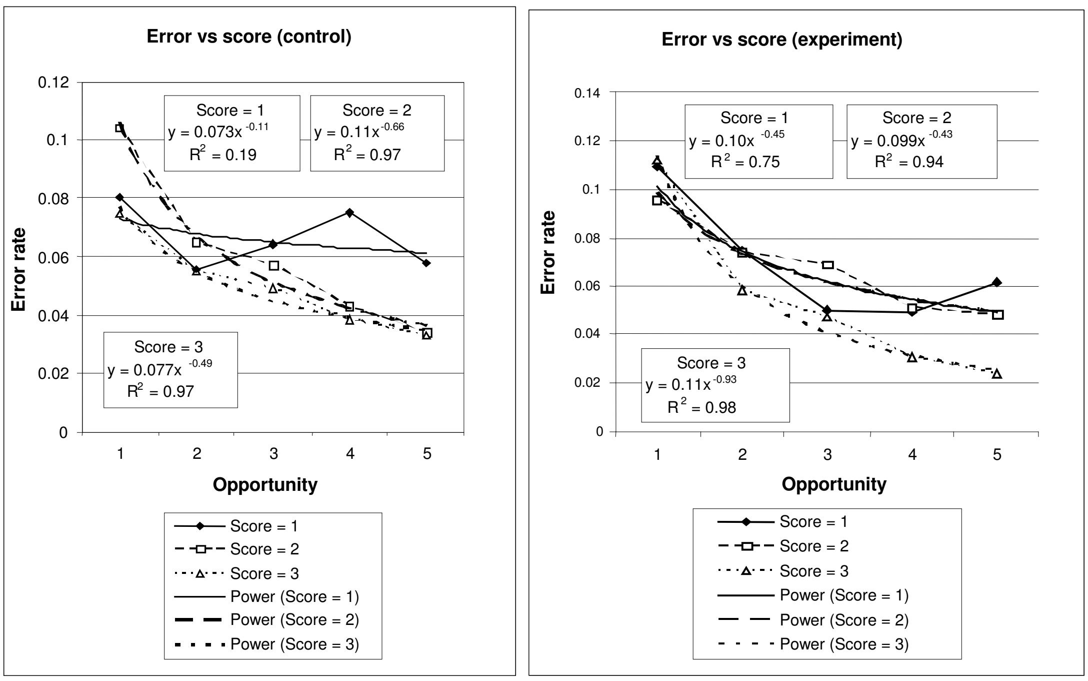


Fig. 4 Learning curves versus score for two versions of an intelligent tutoring system

attempt to the next by simply carrying out the suggestions in the feedback without learning from them.

Fig. 4 illustrates how power law parameters can be used to compare adaptive educational systems. In this experiment the role of adaptive problem selection in an intelligent tutoring system is being investigated, including the degree to which it affects the performance of students of differing initial ability. The knowledge components measured are the same in each group; only the method of computing problem difficulty has been altered (static for the control, dynamic based on the student model for the experimental group). Comparing the overall learning curve for each group showed no significant difference. When each group is further split by ability however, some differences emerge: For the control group, the exponential slope of the curve for a score of 2 (medium ability) is considerably greater than for scores of 1 and 3 (low and high ability, respectively), 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 particularly poorly in terms of both exponential slope and power law fit. Conversely, for the experimental group the low and medium ability curves have similar parameters, while the more advanced students demonstrate a much higher learning speed, as evidenced by an exponential slope more than twice as large as the other two student groups. Using the curve parameters to compare the two conditions, the low-ability students for the experimental group have a significantly higher exponential slope (0.44 versus 0.11) and fit (R^2 = 0.75 versus 0.19) than the control, suggesting more learning is taking place. The high-ability group has also improved its slope substantially (0.93 versus 0.45 for the control). The medium-ability group is relatively unchanged. These difference suggest the dynamic difficulty calculation better serves students far from the mean of initial performance, whereas the simpler static model works for “average” students.

In the sections that follow we use learning curves to measure the performance of ITS domain or student models. In all cases the ITS being tested is of the “learning by doing” kind: students are presented with a problem to solve and they use the system to develop a correct answer. The system observes their behaviour while they work and develops a model of their ability (the student model), which records their ability with respect to a set of knowledge components over time. The system also diagnoses their solution and provides feedback; the outcome of this diagnosis informs the student model. The timing (or modality) of the diagnosis/feedback step can be either immediate (i.e. it is carried out every time the student makes a change to the solution) or on demand. However, the use of learning curves is much more general than this: any system that supports a task for which user performance is expected to improve with practice is a candidate for analysis by learning curves. In the case of educational systems, the system might use other modes of teaching such as worked examples, where the learning curve plots problem completion time versus exposure to examples that introduce each knowledge component. On the other hand, an educational system might be purely for practice and contain no specific problem to solve (for example, simulation systems such as RIDES (Munro, Johnson et al., 1997)); the system can still diagnose the general performance of the student with respect to best practice, and so learning curves are still appropriate. Learning curves have also been used to evaluate educational games (Baker, Habgood et al., 2007; Eagle and Barnes, 2010). An interesting variation is (Nwaigwe, Koedinger et al., 2007), in which learning curves based on criteria other than the domain model are analysed to try to determine which knowledge component caused each error, and thus to build up the individual learning curves for the separate domain model concepts.

Data for learning curves is usually obtained *post-hoc* from student logs. A trace is generated for each student (for each knowledge component) indicating the degree to which the student has correctly applied this knowledge component over time. Each entry may be represented by a continuous value or simply a binary flag (“success” or “failure”). Individual data values for a single knowledge component and student are unlikely to produce a smooth power law because they simply represent too little data. However, the data can be aggregated in several ways to represent useful summaries: data can be grouped for all students by knowledge component (to compare individual elements for efficacy), by student across all elements (to compare students) or over both for comparing different systems (e.g. two different domain models). Power law fit and slope can then be compared between variants of the same system, different systems, different parts of the same system or different student cohorts for the same system. Fig. 5 illustrates this: the two curves represent the learning histories for two populations using different variants of the same ITS, SQL-Tutor (Mitrovic and Ohlsson, 1999). The curve has been limited to the first 10 problems for which each constraint is relevant. This is necessary because aggregated learning curves degrade over time as the number of contributing data points decreases. Both curves exhibit a similar degree of fit and their exponential slopes are similar. However, their y-axis intercepts are markedly different; the y-intercept indicates the initial probability of making an error (i.e. without any feedback). The curve for the experimental group exhibits more than double the initial error rate of the control group.

Learning curves may be used to measure various aspects of an ITS by averaging the raw user data in different ways and comparing the slope and fit of the resulting curves. Some prior summative evaluations we have performed are: overall learning performance of a system; comparison of different student cohorts; feedback versus no feedback; comparison of different types of feedback; comparison of learning performance by ability (as measured by a pre-test) (Mitrovic, Martin et al., 2002). More generally, learning curves can be applied to systems whose main task is not education, but rather that of supporting a user, such as adaptive systems. For example, we would expect adaptive menus to result in a reduction of task time if the adaptations are beneficial. By plotting learning curves representing task completion time versus the number of times a menu is used we can compare alternative adaptation approaches. Within a single system we might also measure the performance of

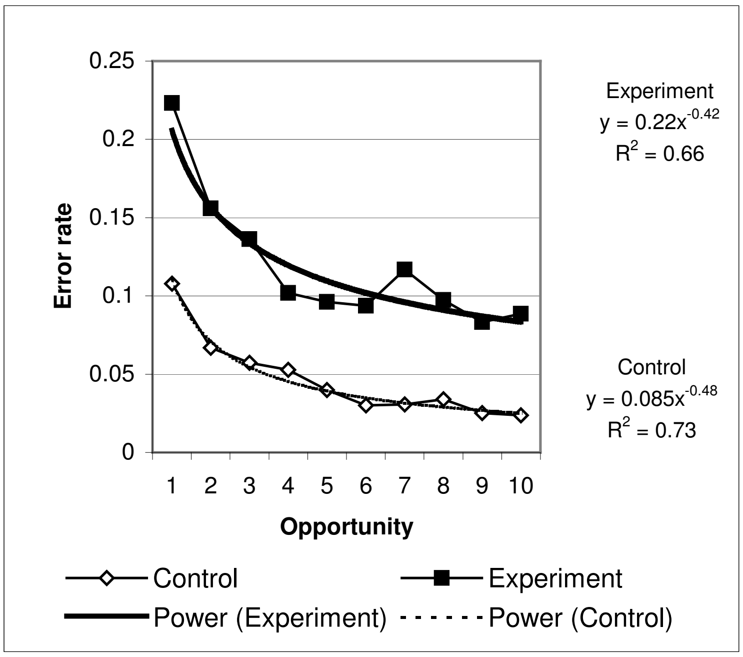


Fig. 5 Learning curves for two variants of SQL-Tutor.

sub-parts, such as heuristics that perform the adaptation or individual menu items, to highlight which are the most effective.

4 Evaluating domain models with learning curves

Whilst learning curves have been compared with one another to look for learning differences (e.g. in (Anderson, 1993)), there are several issues that may arise. First, what parameters of the curve can we reliably use? Second, when are such comparisons valid? Finally, whilst a power law appears to indicate that learning is taking place, does it necessarily indicate optimum performance of an educational system? We now explore these issues further.

4.1 Power law Fit

The residual error of a power law can be used to determine the degree to which any learning is evident with respect to the model; a poor model is unlikely to produce a power law. This in turn suggests that, for two systems with different models whose curves have comparable slope, we might use the power law fit to choose which model is better. However, there is a potential issue: the quality of a power law tends to increase with data set size because the influence of a single data point (i.e. single occurrence of one student using one concept) decreases as the number of aggregated occurrences increases. This has two consequences. First, a larger domain model and/or student sample size is likely to exhibit a better fit than a smaller one, even if the system does not teach the students any better. Typically learning curves plot data that is aggregated across *n* students and *m* knowledge components, so any disparity between the two groups of either the number of students participating or the number of knowledge components in the model will affect the relative power law fits. Whilst it is reasonable to control for the number of students in a sample, doing so for the number of knowledge components being aggregated is more difficult because the number of knowledge components still in use tends to decrease as the number of occurrences (i.e. the X-axis) increases. This latter effect arises because some knowledge components will only have been relevant for a small number of student attempts. It is tempting to try to normalise this effect by selecting only those knowledge components that have been used at least *n* times by *all* students, and then plot a learning curve that is cut off at x=*n*. However, this may unwittingly introduce a further bias: only the knowledge components that most commonly occur will be selected, and these may be the simplest. In practice we have found it better to include all knowledge components and try to determine a suitable cut off point for the curve, either by selecting *a priori* an acceptable reduction in the number of occurrences being aggregated to produce each data point (e.g. 50% of those for x=1) or by examining the resulting curves and making a judgement call on where the curve appears to be deteriorating markedly. The challenge is to find a cut off point that does not bias in favour of one experimental group over another.

For example, consider the curves in Fig. 5. In both cases the learning curves exhibit fairly good power law fits. However, there is some deviation from a power law, particularly for the experimental group, where the curve begins sharply, before *rising* at x=7 and then flattening out. Fig. 6 shows how the slope and fit are affected by the cut-off point: both slope and fit are nearly the same for the two groups up to x=6 after which there is a marked divergence, with the experimental group showing a marked decrease in both measures. Depending on what is being investigated an incorrect conclusion may be drawn if the cut-off point is ill-chosen: in this case selecting for the best slope and fit would suggest a cut-off at x=6, but this may hide a valid difference in learning that does not emerge until later in the learning of each knowledge component.

The problem of disparity in data volume between groups is subtle and needs to be treated even more carefully. For example, in (Koedinger and Mathan, 2004) the learning outcomes associated with two types of feedback were compared in the context of the Excel Tutor. Two versions of the tutor were created: *Expert* and *Intelligent Novice*. In the *Expert* version students were given corrective feedback as soon as they deviated from an efficient solution path. In contrast, students using the *Intelligent Novice* version were permitted to make a limited number of errors and feedback and next-step hints were restructured to guide students through the activities of error detection and correction. Learning curves were

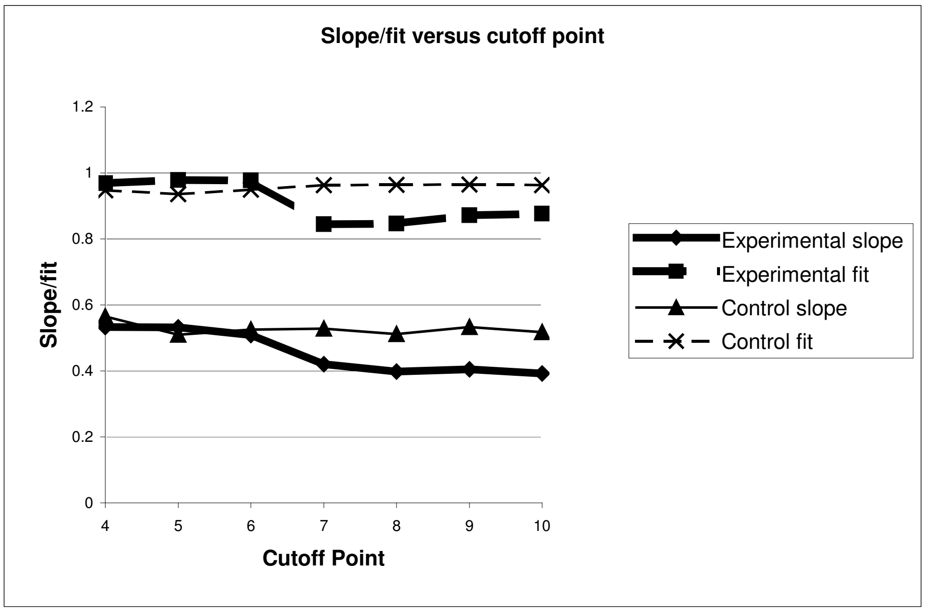


Fig. 6 Slope and fit versus the cut-off point for two learning curves.

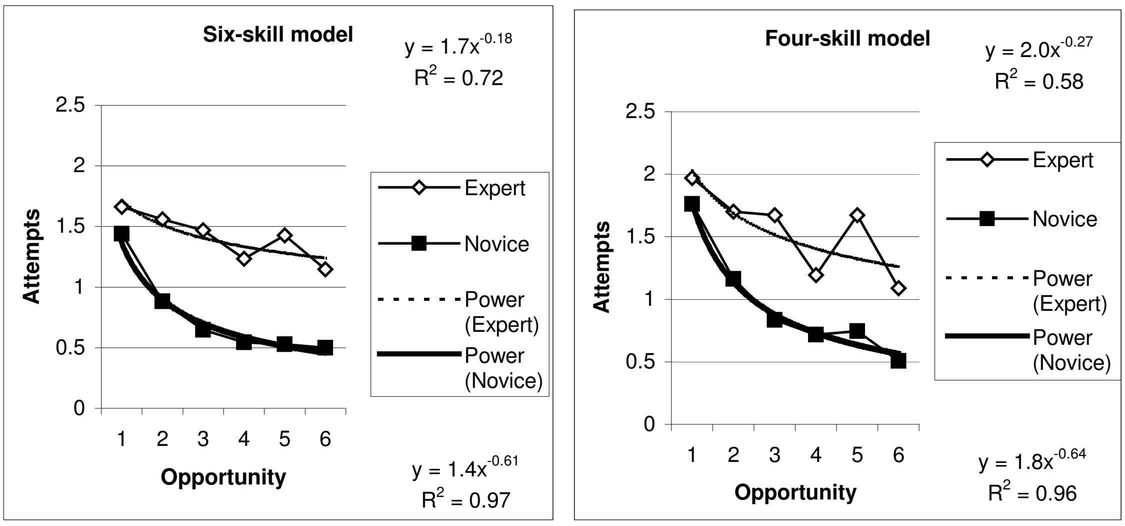


Fig. 7 Learning curves for six- versus four-skill models of the Excel tutor.

plotted and analysed to determine whether students in one condition acquired knowledge in a form that would generalize more broadly across problems (i.e. provide better transfer). The tutor provided opportunities to practice six distinct types of problems. A shallow mastery of the domain would result in the acquisition of a unique rule for each problem type, whereas a deeper understanding of domain principles would help students to see a common abstract structure in problems that may seem superficially different (e.g., see the deep functional commonality in copying a formula across a column or down a row even though these operations have a different perceptual-motor look and feel). Consequently, students exhibiting deeper learning would acquire a smaller set of rules that would generalize across multiple problems. In the case of the spreadsheet tutor it was possible to use a set of four rules to solve the six types of problems represented in the tutor. It was expected that the Intelligent Novice version of the system would lead to deeper learning, and thus that the learning curve for a domain model based on four knowledge components would be superior to the curve for the original model containing six knowledge components.

Two plots were created (Fig. 7), representing these two approaches to the underlying knowledge encoding. The first assumed the student learns a unique rule associated with each of the six types of problems represented in the tutor. Thus, with each iteration through the six types of problems there was a single opportunity to apply each production rule. In contrast, with the four-skill model it is assumed the student learns fewer, more general rules, where there are multiple opportunities to practice each production rule. Fitting power law curves to data plotted with these alternative assumptions about the underlying skill encoding might determine whether or not students were acquiring a skill encoding that would generalize well across problems.

The graphs for both models strongly suggest that the “intelligent novice” system is considerably better than the “expert” version regardless of which underlying model is used– both fit and slope are considerably higher for this variant. However, the difference between the six- and four-skill models is not so clear. For both the expert and novice systems the slope is higher for the four-skill model, suggesting more learning took place: this is particularly true for the “expert” system. However, in both cases the fit (R^2) *decreases*, and again this is more marked in the “expert” system. At first glance these observations appear contradictory: learning is improved but quality of the model (as defined by fit) is lower. However, the four-skill model has 33% fewer knowledge components than the original

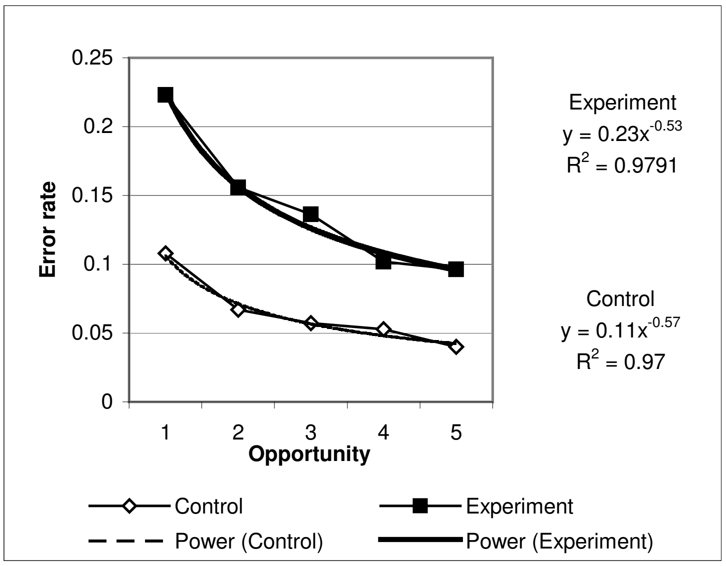


Fig. 8 Two variants of SQL-Tutor with the same domain model but different problem selection strategy.

model, so each of the data points in the curve has been averaged over less student interactions than for the six-skill model, and we would therefore expect the fit to degrade. This means we are unable to make comparisons based on fit in this case. Further, the comparisons of slope now arguably also become less reliable, although in this case the effect is sufficiently large that we might still conclude that the four-skill model is superior. This latter concern could be overcome by plotting individual student curves and testing for a statistically significant difference in the average slopes, as described in Section 4.2.

4.2 Power law slope

Another potential parameter for comparing learning curves is the power law slope, which gives some measure of how rapidly the student's performance is increasing. For example, we might measure the difference between two different feedback presentation modes (e.g. raw text versus a video presentation), where neither the task nor the model has changed. However, a serious issue with the use of power law slope is that it is highly sensitive to changes in the other parameters of the curve, particularly the y-axis intercept, so care is needed to ensure the curves can be reliably compared. For example, changes to average problem difficulty will make such comparisons unreliable; if problem difficulty is greatly increased, students may be overwhelmed and perform more poorly. Conversely, a substantial decrease in difficulty may adversely affect motivation.

If differences related to problem difficulty are what we are trying to measure, the power law slope may not be the best choice. In (Martin and Mitrovic, 2002), we compared two versions of SQL-Tutor that had different problem sets and selection strategies but were otherwise comparable. Fig. 8 shows the learning curves for the two systems trialled on samples of 12 (control) and 14 (experiment) University students. The two curves have similar fit and slope, which might lead us to conclude there is little difference in performance. However, the raw reduction in error suggests otherwise: between x=1 and x=5, the experimental group have reduced their error rate by 0.12, whereas the control group has only improved by 0.7, or about half.

The problem with using learning curve slope here is that it does not measure what we are trying to evaluate. In this study we were looking for differences caused by an improved

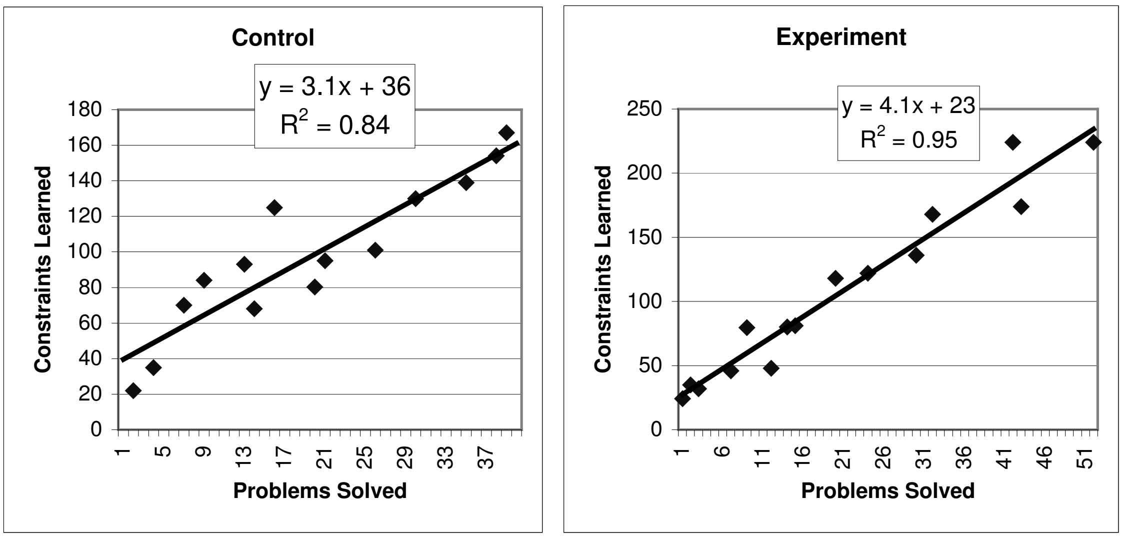


Fig. 9 New constraints learned as a function of problems solved.

problem selection strategy: if the new strategy is better, it should cause the student to learn a greater volume of new knowledge components at a time. The power law slope does not measure this because it measures the *proportional* improvement of the initial error rate, which is more usually what we are interested in (i.e. the extent to which the student corrected their original misconceptions), rather than the absolute proportion of the domain model learned, which is what we would expect a change in problem suitability to affect. In contrast the y-axis intercept *does* in some way reflect this difference, because it measures the size of the initial error rate, but it does not indicate the level of improvement observed. (Note that a discrepancy in the y-axis intercept could also be because of differences in the students' prior knowledge; this possibility is obviated by a pre-test comparison.) A parameter that captures the magnitude of improvement is the local slope, which is given by (Newell and Rosenbloom, 1981):

$$\frac{dT}{dN} = -\alpha B N^{-\alpha-1} \qquad (3)$$

We argued therefore that by comparing the local slope of the curve at N=1, or initial learning rate (equal to $-\alpha B$), we are measuring the reduction in error at the beginning of the curve; this represents how much of the domain the student is learning in absolute terms and better represents what we would like to optimise, namely the learning realised after receiving feedback about a knowledge component *just once*. Ideally the error rate at N=2 will be zero. For the graphs in Fig. 8 we computed the initial learning rate, i.e. the local slope of the power curve at x=1. This yielded initial learning rates of 0.12 for the experimental group and 0.06 for the control group, which correlates with the overall gain for x=5. The advantage of using the initial power law slope (rather than simply calculating the difference between N=1 and N=2 directly) is that the former averages out variance across the x-axis while the latter is a point calculation and is therefore more sensitive to residual error.

In the case just described we were particularly interested in the effect of changing the selection of problems, but in general the discussion highlights that the slope of two power laws is only comparable when the task being performed is broadly the same, because

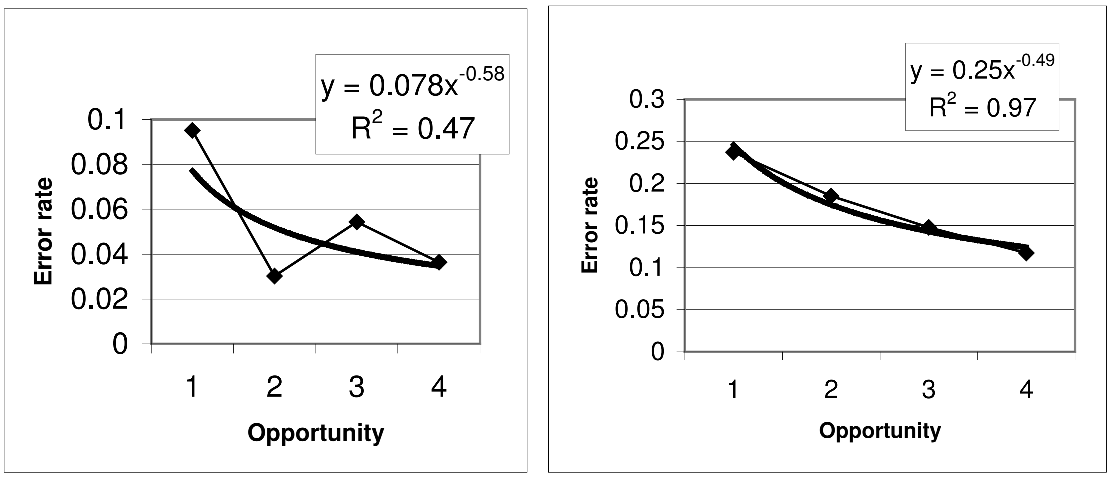


Fig. 10 Two examples of individual student learning curves.

differences in the initial error rate will have an effect. We can further conclude that in general the slope tells us little (in absolute terms) about how well the system is educating the student. For example, a system that presents the student with simple, repetitive tasks might be expected to display a steep power law, whereas one that presents harder problems (or increases the difficulty more quickly) might display a shallower curve, even though the student is arguably learning more. This can be illustrated with respect to the previous study: another absolute measure of learning is the number of knowledge components learned over time. Fig. 9 graphs the number of knowledge components successfully applied (averaged over all students) as a function of the number of problems solved for each of the control and experiment groups. The slope of the linear portion for the experimental group is 1.3 times greater than for the control group, suggesting more is being learned per problem, even though the power law slope for the experimental group was actually *lower* than that of the control.

4.3 Reliability of comparisons

The fact that we have averaged the results across both all knowledge components and students (in a sample group) may raise questions about the validity of the result. Further, whilst plotting and comparing curves for two groups illustrates any difference in aggregated performance, it does not provide any evidence of the significance of the difference. Statistical significance can be obtained by plotting curves for *individual students*, calculating the learning rates and comparing the means for the two populations using an independent samples T-test. Fig. 10 shows examples of individual student curves. In general the quality of curves is poor because of the low volume of data, although some students exhibit high-quality curves. For the experiment described this yielded similar results to the averaged curves (mean initial learning rate = 0.16 for the experimental group and 0.07 for the control group). Further, the T-test indicated that this result was statistically significant ($p<0.01$). We can therefore be more confident that the experimental group really did exhibit faster learning of the domain model, and that this is not just a random outcome or an artefact resulting from the averaging of data across multiple students.

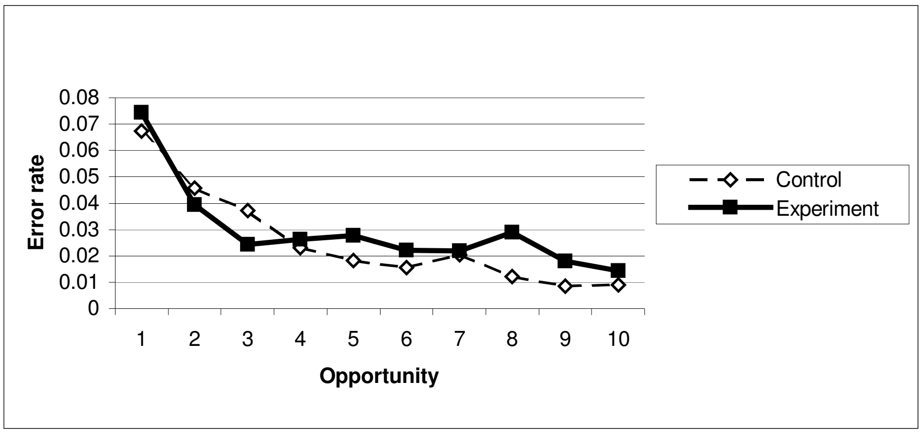


Fig. 11 Comparison of domain models with differing feedback granularity

4.4 Is a power law appropriate?

When evaluating learning curves we assume that the power law of practice holds, and that the students' error rate will therefore trend towards zero errors in a negatively accelerated curve. However, there are arguably *two* power laws superimposed: the first is caused by simple practice, and should ideally trend to zero, although this may take a very long time. The second is caused by the feedback the system is giving: as long as this feedback is effective the student will improve, probably following a power law. However, we do not know how the effect of the feedback will vary with time: if it becomes less effective, the overall curve will "flatten" over time and thus deviate from a power curve. The graph will therefore appear to be a power law trending to a y-asymptote greater than 0.

Fig. 11 illustrates this point. In this study we compared two different types of feedback in SQL-Tutor on samples of 23 (control) and 24 (experiment) second year University students (Martin and Mitrovic, 2006). Over the length of the curves the amount of learning appears comparable between the two systems. However, the absolute gain for the *first two times the feedback was given* (i.e. the difference in y-value between x=1 and x=3) is different for the two systems: For the control group the gain is around 0.03, while for the experimental group it is 0.05. We also notice that the curve for the experimental group appears to abruptly flatten off after this, suggesting that the feedback is only effective for the first two times it is viewed; after that it no longer helps the student[2]. We could use the initial learning rate again to measure the early gain, but this is unlikely to be useful because of the way the curve flattens off and therefore deviates from the initial trend. (We could cut off the curve at x=3 but this is dubious since it is too few data points.) In this case we used the raw improvement as described in the previous paragraph. We obtained learning curves for individual students and performed an independent samples T-test on the value of error(x=3)-error(x=1) for each student. The results were similar to those from the aggregated graphs (mean error reduction was 0.058 for the experimental group and 0.035 for the control group), and the difference was significant ($p<0.01$). This suggests the difference in learning observed early in the graphs is a real phenomenon and not a result of aggregation.

Having determined that a difference in either slope or initial slope is significant, does it really mean anything concrete with regard to learning performance? We have been unable to find any examples of "calibration" of learning curves, mainly because they tend to be

[2] Possible reasons for this are discussed in Section 5.

used as a *substitute* for formal testing methods (e.g. pre-/post-tests) owing to the impracticality of their use for classroom studies. As seen in the previous section, the number of knowledge components learned is one piece of evidence that the effect is more than a statistical anomaly. The Excel-Tutor provides some further evidence that learning curves reflect real effects; recall that the "intelligent novice" interface yielded a steeper curve; the slope for this power law was around three times that of the "expert" version. Post-test results for the two groups (after correcting for covariate parameters of computer experience, conceptual and coding pre-test scores, and math ability) all indicated that the intelligent novice group performed significantly better. The ratio of test error for the two groups ranged from 1.46 to 2.95 (all in the intelligent novice group's favour), with an overall ratio of 1.7. All results were significant (e.g. overall performance ANCOVA: $F(1,44)=6.10$, $p<0.02$). This suggests the difference in learning curves is reflected in student performance, although without a control we have no way of determining the reliability of this inference. For further details see (Mathan, 2003).

4.5 Power versus exponential curves

As stated earlier, the power law is not the only model that has been put forward for learning. Recently exponential curves have begun to be used by some instead of a power law for plotting learning performance. An exponential curve has the form:

$$Y = Ae^{-Bx} \tag{4}$$

Heathcote, Brown et al. (2000) argue that the reason a power law is the best fit is because of bias introduced by averaging many trials, and that when dealing with individuals and a single problem-solving strategy an exponential curve is more appropriate. However, for our studies we are always interested in aggregated (averaged) results; either we are averaging learning across students, or across knowledge components, or both. The "smallest" curves that are informative are all (or groups of) knowledge components for a single student, or the performance of a single knowledge component across multiple students. We would therefore expect a power law to be the better choice. We tested this assumption by fitting power and exponential curves to the learning curves plotted for individual students in a small study of 16 participants using SQL-Tutor. In 12 of the 16 cases a power curve yielded a better fit. The power curve also had a slightly better average fit (0.52, SD=0.28 versus 0.47, SD=0.23) although this is not statistically significant. Interestingly, for the four cases where an exponential curve was superior the curves were very poor in terms of both slope and fit ($R^2 < 0.3$ for a power law) suggesting an exponential curve may be a better fit to *poor learning evidence*. This is consistent with exponential curves being better for measuring learning gains for a single problem solving strategy and student (which is represented by a very small amount of data) and warrants further investigation.

4.6 Summary

The power law parameters of slope and fit can be used for comparison but with certain restrictions: power law fit is valid when the amount of data being compared is comparable; additionally, slope is appropriate only if the task being performed does not differ significantly (and so the initial error rate is approximately the same). Finally, neither slope

nor fit measures the amount of learning taking place with respect to the domain model; the initial local slope may be a more useful parameter for this.
Learning curves are a potentially useful tool for formative evaluation of adaptive educational systems. In the next section we report our experiences using them to try to both evaluate and improve the modelling performance of SQL-Tutor.

5 Using learning curves to analyse and improve domain model structure

A key to good performance in an ITS is its ability to provide the most effective feedback possible. Feedback in an ITS is usually very specific. However, in some domains there may be low-level generalisations that can be made where the generalised knowledge component is more likely to match what the student is learning. For example, for the Excel Tutor previously described the smaller four-skill model was based on the assumption that the concept of relative versus fixed indexing is independent of the direction the information is copied (between rows versus between columns) and thus is a better measure of what is learned than having two separate knowledge components.

Some systems use Bayesian student models to represent students' knowledge at various levels (e.g. (Zapata-Rivera and Greer, 2004)) and so theoretically they can dynamically determine the best level to provide feedback, but this is difficult and potentially error-prone: building Bayesian belief networks requires the large task of specifying the prior and conditional probabilities. We are interested in whether it is possible to infer a set of high-level knowledge components that represent the concepts being learned while avoiding the difficulty of building a belief network, by analysing past student model data to determine significant subgroups of a system's knowledge components that represent such concepts.

When analysing Excel-Tutor two substantially different models were compared in their entirety. According to Anderson the power law appears to hold for individual knowledge components even when they are not entirely independent (Anderson, 1993). Taking advantage of this fact, we can analyse a model's knowledge components at a finer granularity (than the whole model) by plotting learning curves for various subsets of the domain model. We can then compare the resulting curves to try to determine which groups of knowledge components appear to perform well when treated as a single concept. For each student we maintain a log of how the student performed with respect to each relevant knowledge component each time they submit a solution. We can then extract all evidence for a given knowledge component to get a trace of how the student performs over time with respect to that component, from which we can plot a power law. However, we can also simulate the effect of substituting a single general knowledge component for a group of the original (specific) ones as follows: extract from the student log the evidence for *any* of the knowledge components of interest, then use these to create a single trace of the performance of the group of knowledge components over time, and use this to plot a single learning curve for the high-level concept. A good power law fit suggests a new generalised knowledge component represents some concept being learned. Further, the parameters of this curve (slope and fit) may exceed those of the original group of knowledge components, indicating that the generalised concept better represents what the student is actually learning. We might further hypothesise that delivering feedback in a manner consistent with these more general knowledge components would improve learning.

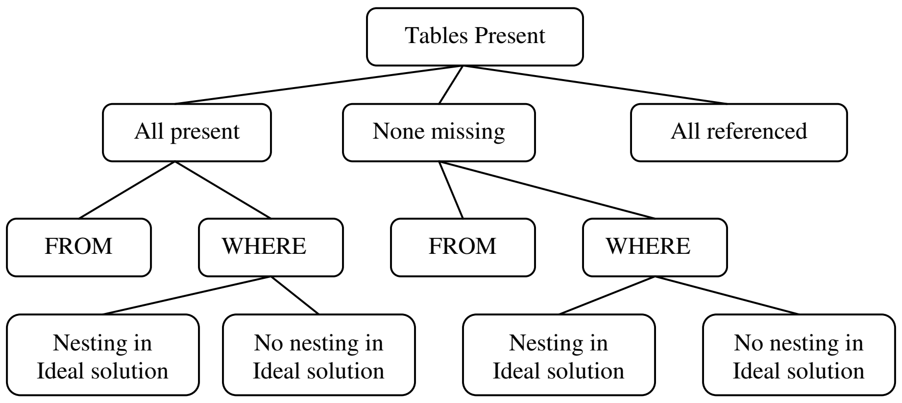

Tables Present
All present
None missing
All referenced
FROM
WHERE
FROM
WHERE
Nesting in Ideal solution
No nesting in Ideal solution
Nesting in Ideal solution
No nesting in Ideal solution

Fig. 12 Example sub tree from the SQL–Tutor domain taxonomy.

5.1 Exploring the domain model structure of SQL-Tutor

The goal of this experiment was to investigate whether we can predict the effectiveness of different levels of feedback by observing how well a group of knowledge components representing a general concept results in a power law, and thus appears to measure a single concept being learned (Martin, Koedinger et al., 2005). To investigate this possibility we performed an experiment in the context of SQL-Tutor. We hypothesised that some groupings of knowledge components (constraints) would represent the concepts the student was learning better than the (highly specialised) constraints themselves. We then further reasoned that for such a grouping learning might be more effective if students were given feedback about the general concept, rather than more specialised feedback about the specific context in which the concept appeared (represented by the original constraint). To evaluate the first hypothesis we analysed data from a previous study of SQL-Tutor on a similar population, namely second year students from a database course at the University of Canterbury, New Zealand. To decide which constraints to group together we used a taxonomy of the SQL-Tutor domain model that we had previously defined (Martin, 1999). This taxonomy is very fine-grained, consisting of 530 nodes to cover the 650 constraints[3], although many nodes only cover a single constraint. The deepest path in the tree is eight nodes, with most paths being five or six nodes deep. Fig. 12 shows the sub tree for the concept "Correct tables present". Whilst developing such a hierarchy is a non-trivial task, in practice this can actually aid construction of the domain model (Mizoguchi and Bourdeau, 2000; Suraweera, Mitrovic et al., 2004).

We grouped constraints according to each node in the taxonomy, and rebuilt the student models as though these were real constraints that the system had been tracking. For example, if a node N1 in the taxonomy covers constraints 1 and 2, and the student has applied constraint 1 incorrectly, then 2 incorrectly, then 1 incorrectly again, then 2 correctly, the original model would be:

```
(1 FAIL FAIL)
(2 FAIL SUCCEED)
```

[3] As SQL-Tutor is continually refined the number of constraints increases; as of January 2010 there are more than 700 constraints.

The entry for the new constraint is:

(N1 FAIL FAIL FAIL SUCCEED)

Note that several constraints from N1 might be applied for the same problem. In this case we calculated the proportion of such constraints that were violated. We performed this operation for all non-trivial nodes in the hierarchy (i.e. those covering more than one constraint) and plotted learning curves for *each* of the resulting 304 generalised constraints. We then compared the curve for each generalised constraint to one obtained by aggregating the results for all of the participating constraints, based on their individual models. Note that these curves were for the first four occurrences only: the volume of data in each case is low, so the curves deteriorate relatively quickly after that. Overall the results showed that the more general the grouping is, the worse the learning curve (either a poorer fit or a lower slope), which is what we might expect. However, there were eight nodes from the hierarchy for which the generalised constraint had superior power law fit and slope compared to the average for the individual constraints, and thus appeared to better represent the concept being learned, and a further eight that were comparable. Whilst this may seem to be weak evidence, it should be noted that attempting to combine knowledge components that are *unrelated* in this fashion is expected to *always* degrade the curve. For example, consider two completely unrelated knowledge components whose learning performance with respect to a sample of students is identical. Combining these two curves when the two knowledge components are alternatively exposed to the student (the best case scenario) lengthens the x-axis by a factor of two resulting in a modest reduction in the degree of fit and a much lower slope. If the two knowledge components follow each other consecutively (the worst case scenario), both slope and fit are greatly adversely affected. In contrast, if the two knowledge components in fact represent the same concept (and therefore being exposed to *either* leads to the same reduction in error for both constraints), combining them results in lower slope but the same degree of fit for the alternating case (again resulting from the change in scale) and has no effect on either slope or fit for the sequential case. Two or more knowledge components can therefore be suspected of modelling the same concept if combining their traces yields a comparable curve to that which treats them separately.

From this result we tentatively concluded that some of our constraints might be at a lower level than the concept that is actually being learned, because it appears that there is "crossover" between constraints in a group. In the example above, this means that exposure to constraint 1 appears to lead to some learning of constraint 2, and vice versa. This supports our first hypothesis. The results of this exploration were sufficiently compelling to warrant researching whether we could use this information to further improve the performance of the tutoring system. The next section describes an experiment with feedback generality based on this analysis.

5.2 Does generalised feedback work?

Constraints are intended to model the domain at a sufficiently low level to capture all errors a student may make and provide effective feedback to help them master the domain. Sometimes this means the constraints will model sub-parts of a concept being learned. The analysis of learning curves confirmed this; for some constraints learning is better modelled as an aggregation of several constraints. Taking this one step further, it is possible that some constraints might be modelling the domain at too low a level to provide optimal feedback, and that, conversely, providing feedback at the more general level would improve learning for those high-level constraints that exhibited superior learning curves. Although the evidence was reasonably weak (just eight cases where learning the concept represented by a node in the taxonomy appeared clearly superior to the individual constraints) we considered this sufficient to test the hypothesis that overall the domain modelling (and hence feedback) in SQL-Tutor might be too specific (Martin and Mitrovic, 2006). (In making this inference we are assuming the low data volume explains why the evidence is not stronger).

5.2.1 Experiment

We produced a set of 63 new constraints that were one or two levels up the taxonomy from the individual constraints. This new constraint set covered 468 of the original 650 constraints, with membership of each generalised constraint varying between 2 and 32 constraints, with an average of 7 members (SD=6). Note that this is not a direct superset of the generalised constraints from the previous study. In some cases where there was positive evidence for combining the constraints for a node at the bottom level of the hierarchy, it appeared intuitively that the parent node would be a feasible generalisation, and there was no evidence to the contrary (i.e. one or more siblings also had positive evidence or, at worst, none of the siblings provided negative evidence); in such a case we opted for the more general grouping. This is discussed further when we analyse the results.

For each new constraint we produced a tuple that described its membership, and included the feedback message that would be substituted in the experimental system for that of the original constraint. An example of such an entry is:

```
(N5 "Check that you are using the right operators in numeric
comparisons." (462 463 426 46 461 427 444 517 445 518 446 519 447 520
404 521 405 522))
```

This example covers all individual constraints that perform some kind of check for the presence of a particular numeric operator. Students for the experimental group thus received this new feedback, while the control group were presented with the more specific feedback from each original constraint concerning the particular operator involved. To evaluate this second 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 six lectures on SQL and had two laboratories on the Oracle RDBMS. SQL-Tutor was demonstrated to students in a lecture on September 20, 2004. The experiment was performed in scheduled laboratories during the same week. The 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, which required the student to identify correct definitions of concepts in the domain, or to specify whether a given SQL statement is appropriate for the given context.

The experimental version of SQL-Tutor was identical to the control, except feedback was now provided for the high-level concept instead of for the constraints themselves. Students were randomly allocated to one of the two versions of the system. A post-test was administered at the conclusion of a two-hour session with the tutor, 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.

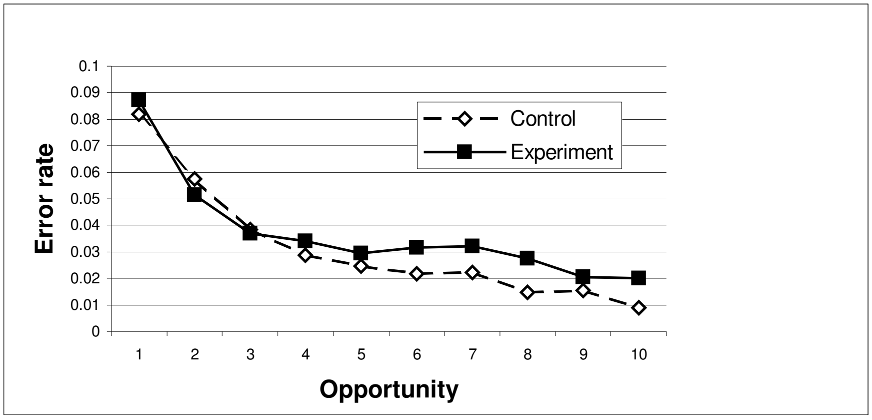


Fig. 13 Learning curves for the two groups.

5.2.2 Results

Of the 124 students enrolled in the course, 100 students logged on to SQL-Tutor at least once. However, some students looked at the system only briefly. We therefore excluded the logs of students who did not attempt any problems. The logs of the remaining 78 students (41 in the control, 37 in the experimental group) were then analysed. The mean score for the pre-test for all students was 2.17 out of 4 (SD=1.01). The students were randomly allocated to one of the two versions of the system. A T-test showed no significant differences between the pre-test scores for the two groups (mean=2.10 and 2.24 for the control and experimental groups respectively, standard deviation for both=1.01, p>0.5).

Fig. 13 plots the learning curves for the control and experimental groups. Note that the unit measured for both groups is the *original* constraints because this ensures there are no differences in the unit being measured, which might alter the curves and prevent their being directly compared as described in Section 4. Only those constraints that belonged to one or more generalised constraints were included. These curves are comparable over the range of ten observations of each constraint, and give similar power curves, with the experimental group being slightly worse (slope = -0.57, R^2 = 0.93, compared to slope = -0.86, R^2 = 0.94 for the control). However, the experimental group appears to fare better between the first and second time each constraint is encountered indicating that they have learned more from the first time they receive feedback for a constraint. In fact, the experimental curve appears to follow a smooth power law up to n=4, then abruptly plateaus. We measured this early learning effect by adjusting the y-asymptote for each group to give the best power law fit over the first four problems, giving a y-asymptote of 0.0 for the control group and 0.02 for the experimental group.

Having made this adjustment the exponential slope for this portion of the graph was –0.75 for the control group (R^2 = 0.9686) and –1.17 for the experiment group (R^2=0.9915), suggesting that the experimental group learned each concept faster for the first few problems for which it applied, but then failed to learn any more. In contrast, the control group learned more steadily, without this plateau effect. Note that this graph does *not* indicate how this feedback is spread over the student session: for example, the first four times a particular constraint was relevant might span the 1st, 12th, 30th and 35th problems attempted. However, this is still a weak result.

Although the generalised constraints used were loosely based on the results of the initial analysis, they also contained generalisations that appeared feasible, but for which we had no

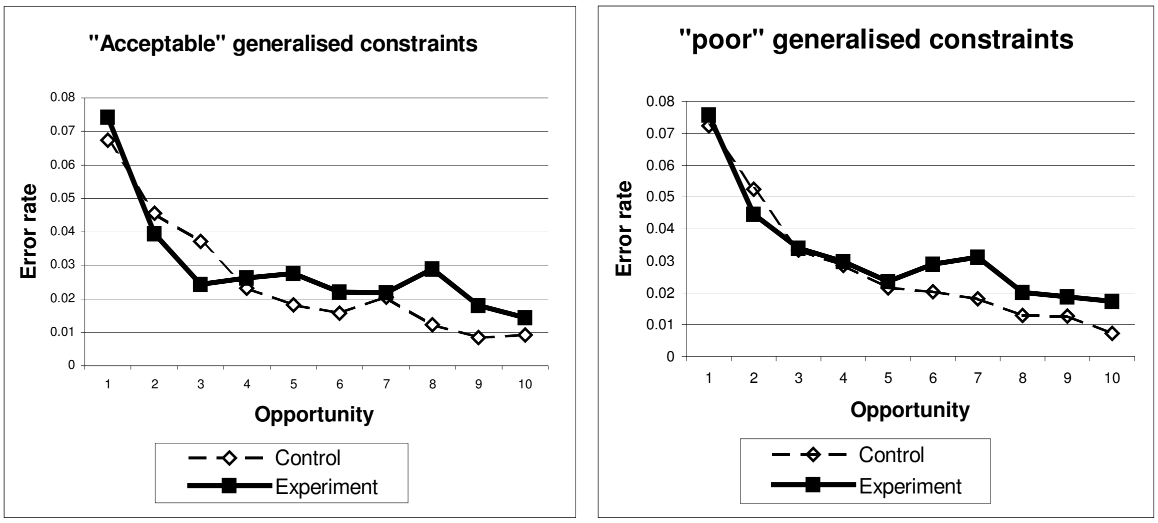


Fig. 14 Power curves based on predictions of goodness.

evidence that they would necessarily be superior to their individual counterparts. The experimental system might therefore contain a mixture of good and bad generalisations. We measured this by plotting, for the control group, individual learning curves for the generalised constraints and comparing them to that of the member constraints when traced individually, the same as was performed for the *a priori* analysis. The cut-off point for these graphs was at n=4, because the volume of data is low and so the curves rapidly degenerate, and because the analysis already performed suggested that differences were only likely to appear early in the constraint histories. Of the 63 generalised constraints, six appeared to clearly be superior to the individual constraints, a further three appeared to be equivalent, and eight appeared to be significantly worse. There was insufficient data about the remaining 46 to draw conclusions. As mentioned earlier there is not a one-to-one mapping between the meta-constraints of the previous study and those for the current system, however some comparison can still be made. Of the nine that were identified in the previous study as good generalisations, two were used again unchanged in the current study, and the rest were represented within five more general groupings. The two that were directly represented again had superior power laws to their constituent constraints. Of the five more general groupings one was superior (checking the condition of the "like" predicate"), two had insufficient evidence (checking that the FROM clause has the required tables; checking the numeric constants of comparisons) and two were worse (checking the presence of correct string constants; checking the presence of correct integer constants), suggesting we were too aggressive when we generalised the results from the initial analysis. We then plotted curves for two subsets of the constraints: those that were members of the generalised constraints considered better, the same or having insufficient data (labelled "acceptable"), and those that were worse (labelled "poor"). Fig. 14 shows the curves for these two groups.

For the "acceptable" generalised constraints the experimental group appears to perform considerably better for the first three problems, but then plateaus; for the "poor" generalised constraints the experimental group performs better for the first two problems only, and the effect is weaker. In other words, for the "acceptable" generalisations the feedback is more helpful than the standard feedback during the solving of the first two problems in which it is encountered (and so students do better on the second and third one) but is less helpful after that; for the "poor" group this is true for the first problem only.

We tested the significance of this result by computing the error reduction between n=1 and n=3 for each student and comparing the means. For the "acceptable" meta-constraints the experimental group had a mean error reduction of 0.058 (SD=0.027), compared to 0.035 (SD=0.030) for the control group. In an independent-samples T-test the difference was significant ($p<0.01$). In contrast there was no significant difference in the means of error reduction for the "poor" group (experimental mean=0.050, SD=0.035; control mean=0.041, SD=0.028; $p>0.3$).

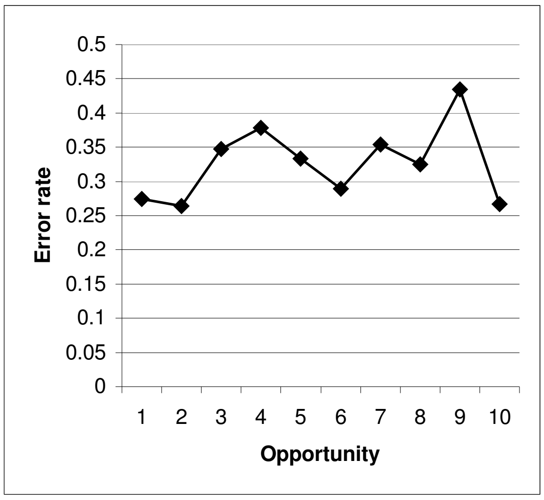
5.2.3 Discussion

There are several tentative conclusions we can infer from these results. First, generalised feedback (when the generalisation is valid) may be more effective in the early stages of learning a new concept (the first two problem instances for each concept), but then the effect disappears, and the students actually do *worse*. This might because as problems become more difficult more detail on the particular context is required, the student having already learned how to apply the concept in general. It is possible that a dynamic approach might work best. For example, a conservative method might be to use generalised feedback only for the first problem (for a given concept), and then revert to more specialised feedback. Alternatively, we might measure the performance of each generalisation: when it appears to be losing its effectiveness the system could switch to specific feedback. However, the small amount of data available makes this a difficult task. More general feedback may also increase the generality of what is learned, thus leading to better knowledge transfer for different types of problems.

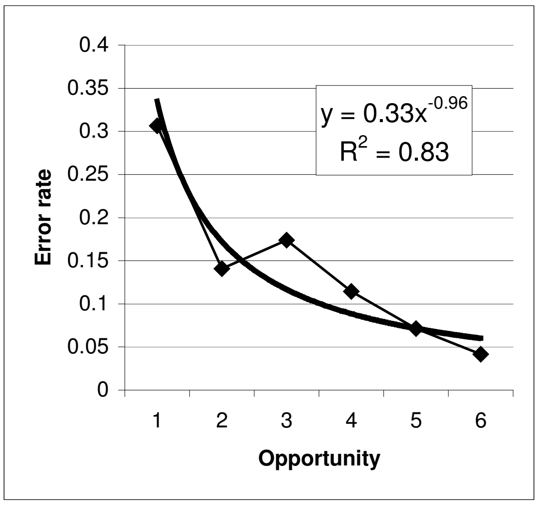
Despite the small amount of data (and thus poor quality of the curves) the learning curves for each of the generalised concepts did appear to be predictive in some sense (i.e. the knowledge component groupings that had the strongest *a priori* support produced the best results). This suggests a system might be able to tailor feedback on the fly if it considers all of the student models when making decisions rather than individual models. This holds promise for increased adaptability in Intelligent Tutoring Systems, and may allow a system to quickly tailor its feedback responses to the current student population. However, the data volume may be too small to individually tailor feedback in this way, so other measures may need to be employed. We conducted a study in October 2005 that compared adaptive feedback granularity with SQL-Tutor's standard feedback, but the results were inconclusive and further research is needed.

6 Learning curves for quality control

The presence of a power law indicates the underlying objects being measured represent concepts being learned; the quality of the power law (measured by slope and fit) thus gives us a measure of the appropriateness of the underlying model; if the power law fit is poor, we are probably not modelling individual concepts being learned, although there are other possible explanations. For example, the slip rate for learned concepts may increase if problem difficulty rises significantly during the student session. For example, in 2002 the first author developed and trialled LBITS (Language Builder ITS), a constraint-based ITS for teaching spelling and grammar to middle school children (Martin and Mitrovic, 2002). The constraints were mainly based on rules of teaching spelling that group words according to the most important letter groupings, such as "ough", "ate" and "ei". However, we also added a constraint for each letter of the alphabet to detect when a letter was missing from the student's answer; the rationale was that students might be unfamiliar with certain letters



(a) All constraints



(b) Main constraints

Fig. 15 LBITS learning curves.

that are used infrequently. After trialling the system on nine students we plotted the learning curve for the system as a whole, as shown in Fig. 15(a). This curve does not exhibit a power law, and suggests that in general no learning took place. We then removed the 26 constraints for checking the presence of the individual letters of the alphabet; the resulting curve, Fig. 15(b), is a typical power law with a good degree of fit (R^2=0.83).

The LBITS experience shows how learning curves can be used to detect problems with a model if the problem is sufficiently severe. It is also feasible that, given enough data (e.g. thousands of users), we can detect individual poor constraints by plotting curves for each constraint separately; if a constraint shows a very poor power law fit, it is probably a poor constraint. Further, in Section 5.2 we inferred that if a group of constraints gave a better power law fit when treated as a single constraint, this probably implied that the constraints were related sufficiently to warrant issuing a higher level feedback message that describes some concept the group has in common. With a much larger amount of data we might take this one step further: if a group of constraints exhibits a significantly superior power law when treated as a single constraint, the existing constraints are probably too specific (i.e. they represent different views of the same fundamental concept) and could be merged.

6.1 Analysing the SQL-Tutor domain model

SQL-Tutor has been commercialised on Addison Wesley's Database Place website[4]. At the time we captured the logs (end of 2007) more than 7600 students had registered to use SQL-Tutor of which 3818 had attempted at least one problem. First we plotted learning curves for all constraints individually, plus one for each of the nodes in the manually created hierarchy. Curves for individual constraints should resemble a power law provided there is sufficient data. From our analysis 415 of the 673 constraints (62%) did not have sufficient evidence to pass any judgement, i.e. sufficient students who used that constraint three or more times. The remaining 258 constraints were grouped by quality into three groups (*high*, *low* and *negative*) as follows: Constraints in the high category produced a power law with a negative slope and a fit (R^2) of 0.5 or greater; those labelled low had a negative slope but a lower fit; constraints with negative learning evidence had a positive

[4] http://www.aw-bc.com/databaseplace

Table 1. Constraint performance.

Learning evidence	Number of constraints
High	123 (48%)
Low	54 (21%)
Negative	81 (31%)

slope, i.e. the number of violations increased with the number of times the constraint was encountered.

Table 1 lists the number of constraints that fell into each category. On face value these results are surprising; nearly a third of the constraints analysed exhibit negative learning performance. However, an analysis of these constraints reveals why this is so. First, the constraint set must act both as the model and the diagnostic rules (including providing feedback), and this can be difficult to achieve. For example, it may be pedagogically useful to test for students using syntactic constructs in the *wrong clause*, e.g. using "distinct" in places other than the SELECT clause. In the version of SQL-Tutor analysed this constraint was always relevant[5] (it is theoretically possible at any time to make this mistake) but in the majority of cases the student would not make this error because they would not use the "distinct" keyword at all. However, there are some more advanced problems (i.e. those requiring unique values to be returned) for which the probability of violating this constraint rises significantly, hence the learning curve for this constraint has positive slope. 18 of the 81 constraints exhibiting negative learning curves were of this type.

We can generalise this reasoning to all constraints that check for the absence of superfluous artefacts in the solution. For example, constraint 380 checks for superfluous attributes in the SELECT clause. Whilst this constraint is important for diagnosis, it is debatable whether or not a student learns not to put additional attributes in the SELECT clause; more likely this constraint is violated as the result of slips, which may increase as problems become more complex. Generally speaking, whilst catching such errors is important, it is probably true to say that such constraints have a low probability of being learned by practice. Such constraints account for 32 of the 81 constraints with negative learning performance (40%).

Another possible reason for poor constraint performance is that the constraint models more than one concept. Six of the constraints in the "poor" set tested for the presence of all tables. Whilst this seems pedagogically valid, there is a significant difference between problems requiring a single table and those requiring multiple. The poor curves for these constraints suggests they may need to be further split such that constraints that test multiple tables are not relevant in situations where only one table is required. On the other hand, some constraints may be more specific than the concept being learned, and therefore be blind to other interactions that are contributing to the learning of this constraint. For modelling purposes such constraints might need to be merged. To test for this we repeated the analysis method of Section 5.2, i.e. computed the learning curves for each node in SQL-Tutor's manually authored constraint hierarchy, and compared the slope and fit to that of the constituent constraints. Of the 306 nodes, just 12 exhibited power laws clearly superior to their constituent constraints, i.e. both the slope and fit were higher than that for the individual constraints. The rest of the hierarchy nodes either yielded similar or worse learning performance, or had insufficient data to draw any conclusions (61 fell into this latter category).

[5] This problem has since been corrected and SQL-Tutor no longer has constraints that are always relevant, with the exception of two that check the two mandatory query clauses (SELECT and FROM).

Table 2. Performance of meta-constraints.

Id	Concept	Slope New (old)	Fit (R^2) New (old)	Constraints	Previous study
M272	Use of semicolon in WHERE	-0.64 (-0.45)	0.80 (0.55)	2	-
M99	Argument of aggregate functions	-0.53 (-0.33)	0.92 (0.49)	5	Better slope, worse fit
M5	More tables (nested select present in IS)	-0.51 (-0.27)	0.95 (0.88)	2	Better
M100	Argument of count()	-0.50 (-0.30)	0.92 (0.42)	3	Better slope, worse fit
M171	Comparison operator for string constant	-0.48 (-0.33)	0.53 (-0.26)	8	-
M296	Brackets for in, exists, any, all (where/having)	-0.43 (-0.33)	0.87 (0.79)	2	-
M130	Operator for aggregate function in HAVING (all same feedback)	-0.17 (-0.11)	0.73 (0.29)	3	-
M265	All atts/expr in select (includes *)	-0.15 (+0.02)	0.74 (0.02)	4	Same
M32	Use of "=" in comparison with numbers	-0.12 (-0.05)	0.35 (0.08)	4	-
M303	Name qualification needed (each clause)	-0.09 (+0.01)	0.65 (0.05)	6	Better slope, comparable fit
M295	Syntax, arguments of IN, ANY, EXISTS, ALL	-0.36 (-0.09)	0.98 (0.80)	13	Same
M102	Need for/superfluous Aggregate functions	-0.43 (-0.20)	0.89 (0.70)	12	WORSE

Table 2 details the results: "Id" is the meta-constraint identifier; "Concept" summarises the concept this meta-constraint represents; "Slope" and "Fit" give the power slope and fit respectively for both the new meta-constraint and the original constraints; "Constraints" indicates how many of the original constraints are subsumed by this meta-constraint. Finally, "Previous study" indicates what evidence existed for each grouping in the original (small) study. Seven of the 12 high performing meta-constraints were from the lowest level of the hierarchy, with a further three from the next level up. The last two were from the third level of the taxonomy (M295 and M102). In some cases (e.g. M171 – string constant comparison operator) the feedback from all of the child constraints was the same, so effectively SQL-Tutor already treats all of these constraints as though they were one concept; it is only the modelling that might be changed to better reflect learning progress. However the domain/student model also drives pedagogical processes such as problem selection, so combining these constraints for modelling purposes might improve the performance of the system overall. In the case of aggregate functions (M102) it appears that students learn to recognise that they need *any* aggregate function more than they learn to use a particular one. It might therefore seem desirable to modify the feedback to be more general in this case, but this could have a detrimental effect on helping students solve a particular problem, and could lead to reduced performance as seen in the previous section. Finally, it appears from M295 that students learn the syntax of the EXISTS, IN, ANY and ALL predicates together, since they are conceptually very similar. This suggests these constraints might be better grouped into one that covers all aspects of the syntax of these predicates. Another pattern that was observed was that constraints whose only difference was the clause they apply to performed better when grouped together as a single constraint.

For example, many constraints are duplicated for the WHERE and HAVING clauses. It may be worth combining these constraints in some cases (e.g. those that test for correct syntax). Another example is M303 (need to qualify attribute names): there are six constraints covering the six clauses, but the evidence suggests the concept of name qualification is learned independently of the clause involved.

6.2 Discussion

From this analysis it appears that the constraint set performs well, and there is not much evidence that it should be changed, apart from a few small changes as described. However, the analysis has yielded some interesting insight into how some minor improvements might be made to improve the system's performance. The large size of the experimental data set also allows us to determine the reliability of the results in Section 5. Recall that we hypothesised that the lower two levels of the taxonomy might represent concepts being learned better than individual constraints based on a small number of constraints for which there was evidence of this, and we assumed the other nodes had insufficient data to draw any conclusions. However, the Addison-Wesley data strongly suggests this hypothesis was not well-founded: Of 135 nodes from the lowest level of the taxonomy for which sufficient evidence was available only seven actually proved superior to the individual constraints. This highlights the need for a sizeable volume of data if such inferences are to be reliably made, and for caution to be exercised when making inferences beyond effects for which there is compelling evidence. On a more positive note, when considering just the observed evidence the results appear reasonably robust: of the seven meta-constraints in table 2 for which we had evidence from both studies, four are well-supported by data from the small study, a further two have weaker support (i.e. the previous study suggested the grouping was at least as good as the individual constraints), and only one was contradicted. Performing the reverse analysis, of the nine groupings with either superior slope or fit (or both) in the small study, three showed superior performance for the large data set, a further three showed comparable performance, one lacked evidence, and just two were found to give worse performance in the large data set. Further, of five meta-constraints that in the original study had the same performance as the underlying constraints, four also showed the same performance in the large study, and the remaining meta-constraint proved superior to the underlying constraints in the large data set. Despite the poor quality of curves in the small dataset therefore, the results nonetheless appear to have some consistency.

7 Conclusions and future work

As Intelligent Tutoring Systems become mature and move from the lab to the classroom it is essential that we continue to strive to make them as effective as possible, and the problems are quickly found and eradicated. This requires the ability to analyse such systems on the fly, and to draw inferences about what is working and what could be improved. Traditional methods of evaluation such as pre-post testing are not feasible because they are too coarse-grained and take too long to run. On the other hand, most (if not all) ITS maintain logs of student process that can be accessed at any time without interrupting the system. Such logs typically contain a wealth of information about learners' performance and hence, tell us much about the performance of the system.

Learning curves have been used for some time to perform summative evaluations of educational systems, including comparing multiple versions of a system to evaluate whether new features are beneficial or detrimental to learning performance. Used in such a way they enable developers to continually improve their systems with some assurance that their modifications are leading to improved learning outcomes. In this paper we have explored some of the issues surrounding learning curves and given examples of situations where care needs to be taken to avoid spurious results.

The ability to analyse domain model performance in detail is of great value when building intelligent tutoring systems. Are the knowledge components at the appropriate level of generality? Are they correctly encoded or have errors crept in? Domain modelling is a demanding task requiring a multitude of skills including the ability to not only identify the pedagogically significant concepts of the domain, but also to correctly encode them using whatever representation is appropriate to the modelling technique being used. It is very easy to make mistakes when doing this (Martin and Mitrovic, 2003). In Sections 5 and 6 we have shown how learning curves may be enlisted to check the correctness of a domain model, not only overall but also at the level of individual constraints. Given enough data, learning curves can be plotted on a per-constraint basis, enabling under-performing constraints to be identified and remedied. They can also be plotted for groups of constraints to determine whether a model's granularity is appropriate.

Methods have been developed to search over a space of possible alternative domain models to find the model that best fits the learning curve data without over-fitting (e.g. Cen, Koedinger, & Junker, 2006; Pavlik, Cen, & Koedinger, 2009). Given increasing availability of tutor log data (see ; Koedinger et al, submitted), wider application of such discovery methods and comparison of alternative approaches are fruitful areas for future research.

In Section 5 we used learning curves to evaluate the performance of a domain/student model by measuring the apparent performance of groups of constraints that represented nodes in a (human-authored) taxonomy. This analysis suggested that some, but not all, generalisations at the lower two levels of the taxonomy represent a single learned concept better than the individual constraints; because some constraints may be measuring sub-parts of a fundamental concept, rather than the whole concept, they may be optimal pedagogically but not for the purpose of modelling student knowledge. While this is useful, it relies on the quality of the taxonomy used. Further, for many domains there is no canonical "best" taxonomy. SQL-tutor is no exception: for example, in addition to the concepts used in the taxonomy in Section 5, there are other candidate concepts that span multiple nodes in the taxonomy we used. An example of such a concept is "multiple tables"; a problem requiring more than one table to be accessed potentially has implications for all of the SQL clauses. Similarly, whether or not table columns need to be disambiguated (e.g. *actor.lastname*, *director.lastname*) affects many of the concepts in the hierarchy used. An alternative approach is to try to infer the best granularity of a domain from a large body of student performance data using learning curves. We might try to find appropriate high-level concepts by discovering groups of constraints that, when presumed to be an individual knowledge component, result in the optimal learning curves. Unlike in Section 5.1 however, we would perform a greedy search of all combinations of constraints, rather than using those represented by the manually created hierarchy. We might also employ feature selection methods from machine learning to reduce the search space (Witten and Frank, 2005).

Finally, there is still the issue of learning curve calibration. Learning curves would be of far greater value if differences in their parameters (slope, fit, initial slope) could be correlated with external measures of improvement such as pre-/post-test gain. Ideally a mapping would be found that is robust to differences in domain model size, task difficulty etc. Extensive experimentation is needed in controlled conditions to determine such a mapping, if one exists.

Formative evaluation of adaptive systems is hard but essential. We believe performance curves are a valuable tool for analysing and improving such systems, and have given examples from our experience using learning curves when developing adaptive educational systems. We hope our results will be of benefit to other researchers of adaptive systems and give them fresh ideas for evaluating their systems.

References

Holt, P., Dubs, S., Jones, M. and Greer, J.: 1994, The State of Student Modeling. In: J. Greer and G. McCalla (eds.), *Student Modeling: The Key to Individualized Knowledge-Based Instruction*. New York, Springer-Verlag, pp. 3-39.

Ainsworth, S. E. and Grimshaw, S.: 2004, Evaluating the REDEEM Authoring Tool: Can Teachers Create Effective Learning Environments? *International Journal of Artificial Intelligence in Education* **14**(3), 279-312.

Uresti, J. and Du Boulay, B.: 2004, Expertise, Motivation and Teaching in Learning Companion Systems. *International Journal of Artificial Intelligence in Education* **14**, 67-106.

Suraweera, P. and Mitrovic, A.: 2004, An Intelligent Tutoring System for Entity Relationship Modelling. *International Journal of Artificial Intelligence in Education* **14**(3), 375-417.

Walker, A., Recker, M., Lawless, K. and Wiley, D.: 2004, Collaborative Information Filtering: a review and an educational application. *International Journal of Artificial Intelligence in Education* **14**(1), 3-28.

Zapata-Rivera, J. D. and Greer, J. E.: 2004, Interacting with Inspectable Bayesian Student Models. *International Journal of Artificial Intelligence in Education* **14**(2), 127-163.

Paramythis, A. and Weibelzahl, S.: 2005, A Decomposition Model for the Layered Evaluation of Interactive Adaptive Systems. In Ardissono, L., Brna, P. and Mitrovic, A. (Eds.) *10th International Conference on User Modeling (UM2005)*, Edinburgh, Scotland, LNCS, vol. 3538, pp. 438-442.

Mitrovic, A., Martin, B. and Mayo, M.: 2002, Using evaluation to shape ITS design: Results and experiences with SQL-Tutor. *User Modelling and User Adapted Interaction* **12**(2-3), 243-279.

Mitrovic, A.: 2003, An Intelligent SQL Tutor on the Web. *International Journal of Artificial Intelligence in Education* **13**(2-4), 173-197.

Ohlsson, S.: 1994, Constraint-Based Student Modeling. In: J. Greer and G. McCalla (eds.), *Student Modeling: The Key to Individualized Knowledge-Based Instruction*. New York, Springer-Verlag, pp. 167-189.

Koedinger, K. R., Corbett, A. C., & Perfetti, C.: (submitted). The Knowledge-Learning-Instruction (KLI) framework: Bridging the science-practice chasm to enhance robust student learning.

Anderson, J. R., Corbett, A. T., Koedinger, K. R. and Pelletier, R.: 1995, Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences* **4**(2), 167-207.

Koedinger, K. R. and Mathan, S.: 2004, Distinguishing qualitatively different kinds of learning using log files and learning curves. *ITS 2004 Log Analysis Workshop*, Maceio, Brazil, pp. 39-46.

Newell, A. and Rosenbloom, P. S.: 1981, Mechanisms of skill acquisition and the law of practice. In: J. R. Anderson (eds.), *Cognitive skills and their acquisition*. Hillsdale, NJ, Lawrence Erlbaum Associates, pp. 1-56.

Snoddy, G. S.: 1926, Learning and Stability. *Journal of Applied Psychology* **10**, 1-36.

Wright, T. P.: 1936, Factors affecting the cost of airplanes. *Journal of Aeronautical Science* **3**, 122-128.

Stevens, J. C. and Savin, H. B.: 1962, On the form of learning curves. *Journal of the Experimental Analysis of Behavior* **5**(1), 15-18.

Anderson, J. R.: 1993, Rules of the Mind. Hillsdale, NJ, Lawrence Erlbaum Associates.

Munro, A., Johnson, M. C., Pizzini, Q. A., Surmon, D. S., Towne, D. M. and Wogulis, J. L.: 1997, Authoring Simulation-Centred Tutors with RIDES. *International Journal of Artificial Intelligence in Education* **8**, 284-316.

Baker, R. S. J. D., Habgood, M. P. J., Ainsworth, S. E. and Corbett, A. T.: 2007, Modeling the Acquisition of Fluent Skill in Educational Action Games. In Conati, C., McCoy, K. and Paliouras, G. (Eds.) *UM2007*, Corfu, LNCS, vol. 4511, pp. 17-26.

Eagle, M. and Barnes, T.: 2010, Intelligent Tutoring Systems, Educational Data Mining, and the Design and Evaluation of Video Games. In Aleven, V., Kay, J. and Mostow, J. (Eds.) *ITS2010*, Pittsburgh, USA, LNCS, vol. 6094, pp. 215-217.

Nwaigwe, A., Koedinger, K. R., VanLehn, K., Hausmann, R. and Weinstein, A.: 2007, Exploring Alternative Methods for Error Attribution in Learning Curves Analysis in Intelligent Tutoring Systems. *AIED2007*, Los Angeles, pp. 246-253.

Mitrovic, A. and Ohlsson, S.: 1999, Evaluation of a Constraint-Based Tutor for a Database Language. *International Journal of Artificial Intelligence in Education* **10**, 238-256.

Martin, B. and Mitrovic, A.: 2002, Automatic Problem Generation in Constraint-Based Tutors. *Sixth International Conference on Intelligent Tutoring Systems*, Biarritz, pp. 388-398.

Martin, B. and Mitrovic, A.: 2006, The effect of adapting feedback generality in ITS. In
Wade, V., Ashman, H., Smyth, B. (Eds.) *AH2006*, Dublin, Ireland, LNCS, vol. 4018, pp. 192-202.

Mathan, S. (2003). Recasting the Feedback Debate: Benefits of Tutoring Error Detection and Correction Skills. PhD thesis, School of Computer Science, Human-Computer Interaction Institute. Pittsburgh, PA, Carnegie Mellon University**:** 130p.

Heathcote, A., Brown, S. and Mewhort, D. J.: 2000, The power law repealed: the case for an exponential law of practice. *Psychonomic Bulletin and Review* **7**(2), 185-207.

Martin, B., Koedinger, K. R., Mitrovic, A. and Mathan, S.: 2005, On using learning curves to evaluate ITS. *AIED 2005*, Amsterdam, pp. 419-426.

Martin, B.: 1999, Constraint-Based Modelling: Representing Student Knowledge. *New Zealand Journal of Computing* **7**(2), 30-38.

Mizoguchi, R. and Bourdeau, J.: 2000, Using Ontological Engineering to Overcome Common AI-ED Problems. *International Journal of Artificial Intelligence in Education* **11**, 107-121.

Suraweera, P., Mitrovic, A. and Martin, B.: 2004, The role of domain ontology in knowledge acquisition for ITS. In Lester, J.C., Vicari, R., Paraguacu, F. (Eds.) *Seventh international conference on Intelligent Tutoring Systems*, Maceio, Brazil, LNCS, vol. 3220, pp. 207-216.

Martin, B. and Mitrovic, A.: 2002, Authoring web-based tutoring systems with WETAS. *International conference on computers in education*, Auckland, pp. 183-187.

Martin, B. and Mitrovic, A.: 2003, ITS Domain Modelling: Art or Science? *International Conference on Artificial Intelligence in Education, AIED2003*, Sydney, Australia, pp. 183-190.

Cen, H., Koedinger, K. R., & Junker, B.: 2006, Learning Factors Analysis: A general method for cognitive model evaluation and improvement. In M. Ikeda, K. D. Ashley, T.-W. Chan (Eds.) *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, Jhongli, Taiwan, LNCS, vol. 4053, pp. 164-175.

Pavlik Jr., P. I., Cen, H., & Koedinger, K. R.: 2009, Learning factors transfer analysis: Using learning curve analysis to automatically generate domain models. In T. Barnes, M. Desmarais, C. Romero & S. Ventura (Eds.), *Proceedings of the 2nd International Conference on Educational Data Mining* , Cordoba, Spain, pp. 121-130.

Witten, I. H. and Frank, E.: 2005, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman.

Authors' Vitae

Dr Brent Martin is a Senior Lecturer of Computer Science at the University of Canterbury, New Zealand, and is the director of the Computational Intelligence Research lab at Canterbury. He received his BSc (Computer Science) and MSc (Machine Learning) from the University of Waikato and his PhD (Artificial Intelligence in Education) from Canterbury. Dr Martin has worked in several areas of artificial intelligence including authoring of intelligent tutoring systems, machine learning, natural language parsing for information retrieval and intelligent agents. He has applied artificial intelligence to a range of domains including automated drug prescription, horticulture management and container port optimisation.

Dr Antonija (Tanja) Mitrovic is a full professor and the Head of the Department of Computer Science and Software Engineering at the University of Canterbury, Christchurch, New Zealand. She is the leader of ICTG (Intelligent Computer Tutoring Group). Dr Mitrovic received her PhD in Computer Science from the University of Nis, Yugoslavia in 1994. Dr Mitrovic's primary research interests are in Artificial Intelligence in Education. ICTG has developed a number of constraint-based intelligent tutoring systems in a variety of domains, which have been thoroughly evaluated in real classrooms, and proven to be highly effective. These systems provide adaptive support for acquiring both problem-solving skills and meta-cognitive skills (such as self-explanation and self-assessment). Although most of the ITSs developed by ICTG support students learning individually in areas such as database querying (SQL-Tutor), database design (EER-Tutor and ERM-Tutor), data normalization (NORMIT) and language learning (CAPIT), there are also constraint-based tutors for object-oriented software design and collaborative skills, various engineering topics (thermodynamics, mechanics), training to interpret medical images and language-learning. ICTG has also done a lot of research on authoring, resulting in ASPIRE, a full authoring and deployment environment for constraint-based tutors. Recent research includes affect-aware tutors and motivational tutors.

Dr Kenneth R. Koedinger (to follow)

Dr Santosh Mathan (to follow)