Different Learners Need Different Resubmission Policies in Automatic Assessment Systems

Ville Karavirta
Helsinki University of Technology
Department of Computer Science and Engineering
Finland
vkaravir@cs.hut.fi

Ari Korhonen
Helsinki University of Technology
Department of Computer Science and Engineering
Finland
archie@cs.hut.fi

Lauri Malmi
Helsinki University of Technology
Department of Computer Science and Engineering
Finland
lma@cs.hut.fi

ABSTRACT

Many automatic assessment systems support immediate feedback as well as allow resubmissions to improve the learning outcome based on the feedback. Several strategies exist to implement the resubmission policy. The goal, however, is to improve the learning, and thus the strategies should also aim to prevent the learner from using the resubmission feature irresponsibly. One of the key questions here is how to develop the system and its use in order to cut down the reiteration that does not seem to be worthwhile?

In this paper, we study data gathered from an automatic assessment system that supports resubmissions. We use a clustering technique to draw a distinction among learner groups that seem to behave differently according to their use of the resubmission feature and the points achieved. By comparing these groups with each other, we conclude that for a small minority of learners there is a risk that they use the resubmission inefficiently. Thus, limiting the number of reiterations is justified even in a case where students have to solve the exercise with *new random* input data each time they wish to resubmit the solution.

1. INTRODUCTION

Automatic Assessment systems and strategies [2, 4, 6, 8, 10, 11, 13, 14, 16, 15, 17] have many benefits compared with traditional teaching methods. The most cited features are that the learner can work at any place with an internet access, and submit the accomplishment at any time. However, there are other pros and cons, as well. See, for example, the findings of an international survey of current assessment practices [3].

Compared with traditional teaching, most automatic assessment systems allow the learner to resubmit the answer and receive immediate feedback several times within a single learning session. Thus, even if the answer is incorrect, there is an option to allow the learner to continue working with the exercise straight away and resubmit an improved answer. This feature promotes reflective thinking, and can be a very valuable aid for learning [10, 12].

Allowing resubmissions poses some challenges, as well. If the number of resubmissions is not restricted, some learners might be tempted to try solving the exercise by applying a trial-and-error strategy. They could use a vast number of reiterations without pondering the source of the errors. Such a practice, which could be called *bricolage* [1], is very inefficient when solving, for example, programming exercises, and in the worst case, it could promote bad programming practices in the future as well. In order to prevent this endless trial-and-error method to occur, an automatic assessment system can *limit the number of resubmissions*. This mode of operation is applied in many systems (see, e.g., WWW-TRAKLA [8] and Ceilidh [2]).

Randomization or personalization of exercise data can be used to discourage bricolage. In WWW-TRAKLA, for example, students solve individually tailored algorithm simulation exercises based on randomized initial input data in tracing exercises. This method efficiently prevents plagiarism, because all students have different instances of the same exercise. However, the learners practice with the *same initial data* in each reiteration, and therefore the number of allowed resubmissions is typically limited to 3-5 times. On the other hand, TRAKLA2 [11], which is the follower of WWW-TRAKLA, employs a different strategy that allows *unlimited number of resubmissions* and still puts a damper on the endless trial-and-error method. In this system, the learner has to start over the exercise with *new initial data* in each reiteration. Thus, if the learner iterates, for example, 15 times on the same exercise, he or she actually needs to solve 15 different instances of the same exercise without getting any direct benefit from the previous submissions. WWW-TRAKLA has been used on our course of data structures and algorithms with yearly enrollment of over 500 students in years 1997-2003 until it was replaced by TRAKLA2.

At the beginning, the strategy applied seemed to be very promising to prevent bricolage in TRAKLA2. Our assumption was that if students use the resubmission feature heavily, they also use considerably large amounts of time solving many instances of an exercise task, and thus presumably learn better. We observed, as anticipated, that there exist significant differences among learner groups what comes to the number of resubmissions used. We investigated this issue more and performed a regression analysis that implied that there was only a slight correlation between the total number of resubmissions used and the total time used when solving the exercise task. This made us concerned about the overall time used by the learners that employed heavily the resubmission feature. In order to get a deeper insight into the overall phenomena, we used a clustering technique to identify these learner groups based on their behavior. In this paper, we report the differences among these groups

based on the average number of resubmissions used, points achieved from the exercises, time on task, and examination points.

The rest of the paper is organized as follows. Sections 2 and 3 describe the learning environment that is used to gather the data, and the clustering method applied in analyzing the data, respectively. Section 4 reports the results, and finally in Section 5 we conclude our findings.

2. TRAKLA2

TRAKLA2 is a web-based learning environment for Data Structures and Algorithms course. The system is capable of distributing automatically assessed *algorithm simulation exercises* in which the learner can practice several important core computer science topics such as basic data structures (stack, queue), sorting algorithms, priority queues (i.e., binary heap), search trees, hashing methods, and graph algorithms. In the exercises, the task is often to show, in terms of manipulating conceptual diagrams of the data structures, how an algorithm changes the given data structures while its execution proceeds. The goal is that the learner simulates the working of the algorithm without writing any code. In order to support this, the system provides an applet that visualizes the data structures used by the algorithm. Through these visualizations, the learner is able to modify the underlying *real data structures* in terms of direct manipulation. Finally, after completing the task, the learner submits the answer to our course database and receives feedback on his or her performance. The feedback is based on comparing the recorded sequence of simulation steps and a model sequence generated by an actual implemented algorithm.

TRAKLA2 has been used at the Helsinki University of Technology since 2003 on the course of data structures and algorithms. Each year about 500 students, comprised of both CS majors and minors[1], take the course. Each student solves some 25-30 different exercises, which is a compulsory part of the course. At least 50% of the maximum points is required to pass the exercise part, but getting more points will increase the exercise grade. Achieving 90% of the maximum points gives the maximum grade of the exercise part. The course includes other tasks, as well, such as a final examination and classroom exercises (CS majors) or a design project (CS minors). The weight of the algorithm simulation exercises in the final course grade is 30%.

The concept of algorithm simulation exercises has been an important addition to aid the learning of the functionality of data structures and algorithms. The following list characterizes the features available in the system.

1. The system can produce exercises with random initial data so that each student solves a different instance of the same exercise. An example: "Insert the following keys one by one into an initially empty AVL tree: J M R G A B T Z K L". For each student, and for each exercise instance the given set of keys is different.

   Personalization of exercises can encourage natural discussion among students without the attempt to copy answers from each other.

2. The exercise is solved in terms of direct manipulation through the available GUI operations, typically drag-and-dropping keys or nodes on the screen, and applying exercise-dependent push buttons.

3. The system can be used both as an off-line training system as well as an on-line submission system where all submissions are recorded to the course database.

4. The system can compare the recorded simulation sequence with a sequence generated by an actual implemented algorithm, and thus perform automatic assessment.

5. The system provides immediate feedback for each solution by showing how many correct simulation steps were performed. However, after getting the feedback, the learner cannot correct the solution and submit it again. Instead, the exercise has to be reset and a new exercise instance is given.

6. The learner can examine the model solutions that are illustrated by means of algorithm animation, i.e., the solution for the exercise instance is shown in a separate window as discrete steps of animation. However, after viewing the model solution, the learner's own solution cannot be submitted any more. Instead, the exercise has to be reset and a new exercise instance is given.

7. The learner can submit a solution to arbitrary number of exercise instances. The number of allowed submissions, however, can be limited by the teacher.

8. The system records student actions (e.g., number of submissions and resets, time on task, model solution steps reviewed, and so on) to produce statistics on students' performance, and to allow deeper analysis of the solution processes.

Figure 1 shows an example of a TRAKLA2 visual algorithm simulation exercise. The learner is supposed to insert (drag and drop) keys from the given array (initial data) – one by one – into the binary search tree. After completing the exercise, the learner can ask the system to grade his or her performance. The feedback received contains the number of correct steps out of the maximum number of steps. The learner can also view the model solution for the exercise as an algorithm animation sequence. The model solution is shown as a sequence of discrete states of the data structures, which can be browsed backwards and forwards. The list of the 26 exercises used in spring 2004 is in Appendix A.

The immediate feedback, based on the automatic assessment of the submitted exercises, is a particularly important feature here [9, 12, 13]. During the last 15 years, we have applied several different *resubmission strategies*. Until 1996, the old TRAKLA system allowed only one submission per exercise, which corresponded to the practice used in manual assessment. Thereafter, the WWW-TRAKLA system employed a strategy in which the number of resubmissions was restricted to 3–5 per exercise. In this case, a learner was allowed to continue working with the exercise with the same initial data after each submission. However, the personalized model solutions were available only after the deadline had passed. Finally, with the TRAKLA2 system, we introduced the strategy with an unlimited number of resubmissions, but this time the learner had to start over with new initial data in each reiteration. In this way, we can give the

[1] Minors are students from other engineering curricula.

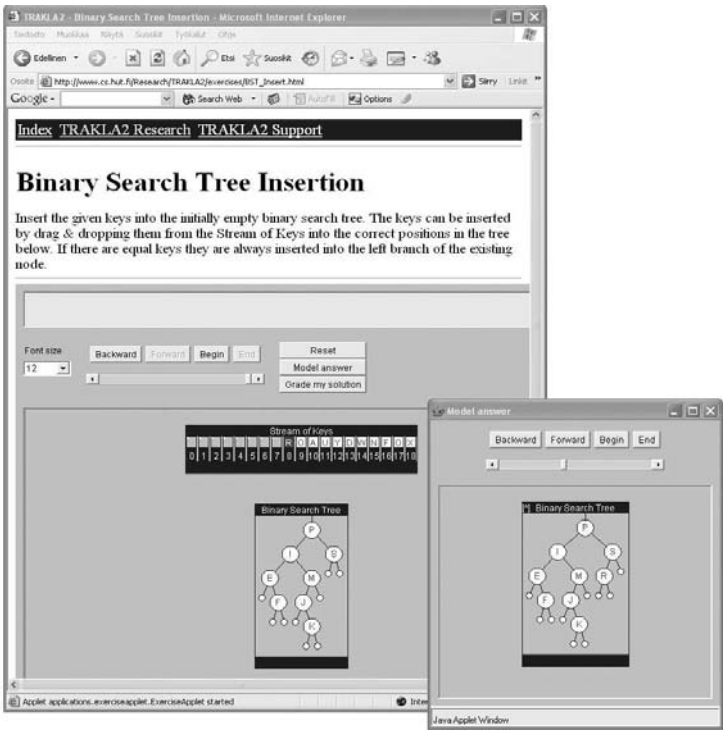

Figure 1: TRAKLA2 exercise applet. Exercise window includes data structures and push buttons. The model solution window is open in the front.

learner the possibility to view the personalized model solution between two consecutive submissions. This practice provides valuable feedback. In addition, it supports learning from one's own mistakes.

The distribution of points awarded for the algorithm simulation exercises changed during the move from TRAKLA to TRAKLA2. In the following, we have divided the students into four groups based on the points received. First, those who achieved less than 50% of the maximum points possible and who consequently failed the exercise part. Second, the students who received 50-90% of maximum points and passed. Third, those who received more than 90% of the maximum points and received the maximum grade for the exercises, and finally those students who solved all the exercises correctly. The most interesting figure is the number of students in the fourth group gaining the maximum points. In year 2003, the proportion of students were about 10% as in 2004 it was over 30%. By applying the χ^2-test, we compared the grade distributions of years 2003 and 2004 to see whether they were homogeneous. The results show that the differences in the distributions are statistically very significant ($p < 0.001$).

3. METHOD

Based on the observations that students apply the resubmission option in very different ways, we decided to perform a deeper analysis. It seemed obvious that the student population contains several different groups with different characteristic behaviors. We chose to apply cluster analysis techniques to identify these groups. In this section, we will introduce the *clustering algorithm* used in our analysis. We will also describe the data we have gathered up and how we applied the clustering algorithm to the data.

The aim of a clustering algorithm is to *classify* similar inputs to belong to same category, *i.e. cluster*. As a result of the clustering, similar data points are within the same cluster that distinctively differs from the other clusters. *Competitive learning* is a well-known neural network method for clustering represented in neural network literature (see, for example, [7]). In competitive learning, for each input vector a *winning-unit* is calculated by finding the unit closest to the input vector with regard to some criteria. All of the output units compete to be the winner that is adapted, thus the name competitive learning.

Figure 2 represents a simple neural network with the input layer and one output layer. The input x_t is an m-by-1 vector $x_t = [x_{t,1}, \ldots, x_{t,m}]^T$. Related to every unit on the output layer are weights $w_{t,i,j}$, where i refers to the input unit and j to the output unit. Some of these weights are marked in Figure 2 to clarify the numbering. Thus, the weights of the output unit j are $w_{t,j} = [w_{t,1,j}, \ldots, w_{t,m,j}]^T$.

In the simple competitive learning (SCL) algorithm, there

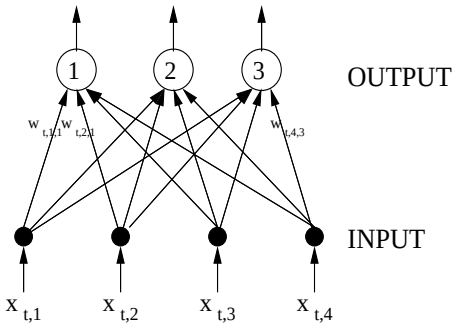

Figure 2: Simple neural network.

is only one layer of output units (as in Figure 2). In the following, we will denote x_t and $w_{t,j}$ the input vector and the codevectors (weights) at time t, respectively.

In SCL, the codevectors are adapted to better classify the input data. The adaptation $w_{t,j}$ at time t $\Delta w_{t,j}$ is calculated using Equation 1.

$$\begin{array}{ll} \Delta w_{t,j} = \alpha_t(x_t - w_{t,j}) & if \quad \|x_t - w_{t,j}\| = min_k\|x_t - w_{t,j}\| \\ \Delta w_{t,j} = 0 & for \quad k \neq i \end{array} \tag{1}$$

In Equation 1, α_t is the learning rate that controls the input vectors influence over the codevectors. Usually α_t is a monotonically decreasing function. $\|x_t - w_{t,j}\|$ is the *Euclidean distance* between the points x_t and $w_{t,j}$. The codevectors for the next input (at time t+1) are calculated using Equation 2.

$$w_{t+1,j} = w_{t,j} + \Delta w_{t,j} \tag{2}$$

Equations 1 and 2 state that only the value of the winning codevector $w_{t,k}$ updates.

Naturally, the codevectors need to have some initial values. They can be initialized in several ways. For instance, the values can be randomized from the target value space, or picked randomly from the input vectors.

In this paper, the clustering is based on the following two dimensions (input units): the number of resubmissions on the whole course, and the total points gained from the TRAKLA2 exercises. The number of resubmissions was a natural choice since we are studying the effect of resubmissions on learning. TRAKLA2 points were used because students using equal amounts of resubmissions get very different grades. Thus, this is a good way to separate the weak students from the good ones.

We performed several tests to identify, how many clusters could be found, and finally selected the number of clusters to be five. Thus the number of codevectors (output units) was also five.

4. RESULTS

The analysis presented below is based on the results of the course in spring 2004. We also analyzed the data from spring 2005 and found out that the results were similar, thus we do not present the comparison here.

The five clusters produced by the SCL algorithm form the basis of our model that classifies the students. The results of the analysis are shown in Figure 3. The X axis denotes the number of submissions during the whole course, and the Y axis denotes the total number of achieved points (maximum 64 points). Each point corresponds to one student in the target population. The clusters are indicated by different symbols. We identified the cluster characteristics, hereafter labeled as PASSERS, ORDINARIES, ITERATORS, AMBITIOUS and TALENTED. The labels describe our hypothesis on the learning style, motivation, and skills of the students in the particular group. These features are discussed more below.

PASSERS can be found in the lower left corner of the diagram. They get clearly the lowest exercise points, and do not reiterate much to improve their grade. Our assumption is that they do not put much effort in the course. Instead, they aim at passing the course with minimum effort.

ORDINARIES are found above PASSERS on the left side of the of diagram. Thus, they achieve considerably better results. However, they apparently do not aim at the best grades since they do not exploit the resubmission feature compared with, for example, the AMBITIOUS. They seem to be satisfied with a reasonably good grade that is achieved without too much effort.

ITERATORS are found in the top right corner. They use the resubmission feature heavily, much more than any other group. They get finally good exercise points, but not all of them reach full points regardless of the large number of submissions.

TALENTED are students found at the top left corner. They use fewest iterations and still achieve the best results. It is likely that they are brilliant students that do well whatever they do, or that they have well developed learning practices in which they first think the problem through and act only thereafter. Thus, they seem to know what they are doing.

AMBITIOUS is the final group, which is located at the top of the diagram in the middle, thus they fall between the TALENTED and ITERATORs. They seem to aim at very good exercise points (and overall course grade), which they also achieve. However, they use considerably fewer resubmissions than ITERATORS. In addition, they use the resubmission feature more than ORDINARIES to get better results.

It should be noted, that altough many of the PASSERS get less than the minimum amount of TRAKLA2 points (32 points) needed to pass the exercises, they still belive in passing the course since they take the examination. In addition, they have the possibility to make up the missing points after the deadline. These points, however, are not included in this analysis due to their different grading policy.

A deeper analysis of the student groups revealed more evidence that the clusters really differ from each other. We studied the groups in respect of two independent variables: total points received in the final examination, and the number of CS majors within the group. In addition, we had one

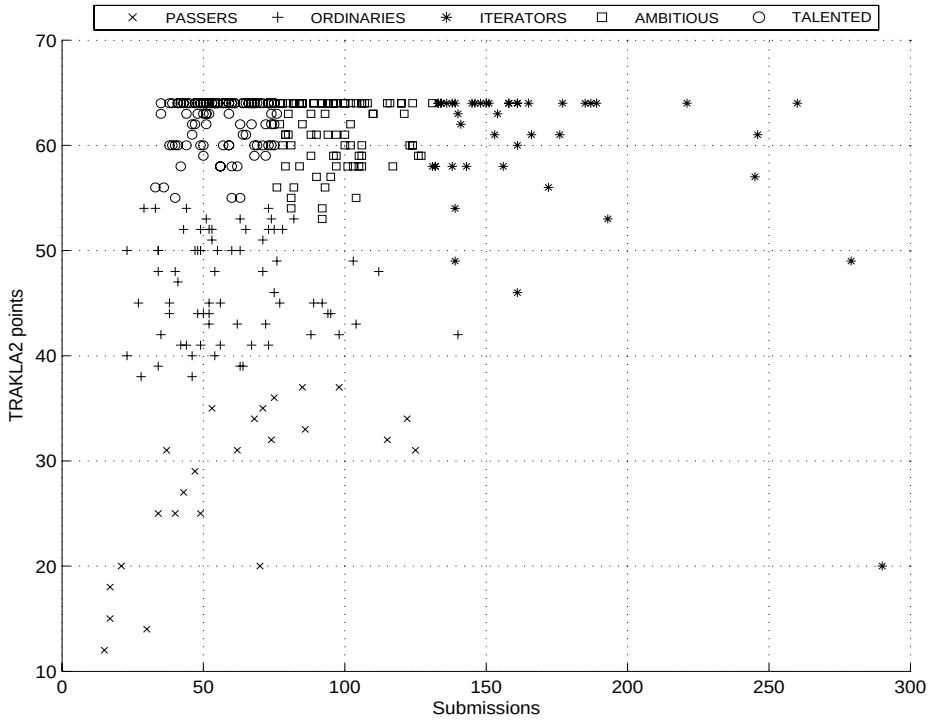

Figure 3: Learners clustered into five groups. The number of learners is 356, number of exercises is 26, and maximum TRAKLA2 points 64.

dependent variable *time-on-task*, *i.e.*, the time spent on the last two exercise rounds[2].

The average points in the final examination after the course[3] are shown in Table 1. The examination takes place in a controlled lecture hall, and students are not allowed to have any extra material with them. The examination contains several different types of assignments: definitions of concepts, explaining the working of algorithms, small algorithm analysis and design tasks, as well as questions about choosing algorithms for some application. There are algorithm simulation tasks, too, but they are a small part of the whole examination.

Statistical analysis showed that PASSERS' average points are significantly lower (t-test, $p < 0.01$) than the average points for groups ORDINARIES, ITERATORS and AMBITIOUS. Moreover, the average points of the TALENTED group is significantly higher (t-test, $p < 0.01$) compared with all the three other groups. On the other hand, there are no statistical differences among the rest of the groups (t-test, $0.33 < p < 0.84$). The box plot in Figure 4 illustrates other differences among the groups by showing the medians and 25% as well as 75% quartiles. The linear correlation between TRAKLA2 points and examination points for all groups is very significant ($\rho = 0.38, p < 0.001$).

Table 1: The examination points in each cluster. The number of maximum points is 34.

Student group	N	**Avg**
PASSERS	24	10.0
ORDINARIES	72	16.2
ITERATORS	45	15.9
AMBITIOUS	98	16.9
TALENTED	117	20.6
ALL	356	17.4

Although there are no significant differences in the examination points among the groups ORDINARIES, ITERATORS and AMBITIOUS, there are clear differences in time-on-task as depicted in Figure 5. ITERATORS use most time and AMBITIOUS use somewhat less, but much more than ORDINARIES. The differences among these groups are significant (t-test, $p < 0.01$). However, there are no significant differences among the groups ORDINARIES, TALENTED and PASSERS. That is, these groups use about the same to-

[2]The time was recorded only for the last two rounds due to the fact that the necessary logging feature was not implemented at the beginning of the course.

[3]Students are allowed to take the final examination several times during the year. However, the vast majority of them take it immediately after the course. The results are from the first examination.

tal time to achieve results which they consider satisfactory. PASSERS seem to use slightly less time than the others. It should be noted, however, that only the time-on-task in the two last exercise rounds was logged, thus it is possible that PASSERS use relatively less time compared with the previous three rounds. They might have calculated the point limit to just pass the course and therefore the difference between them and the other groups could be larger at the end of the course.

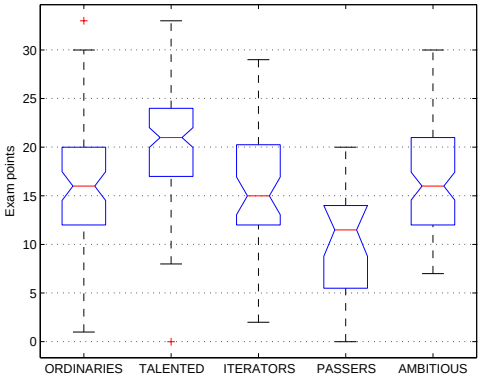

Figure 4: Boxplots for examination points of each group. The box has lines at the lower quartile, median, and upper quartile values. The whiskers show the extent of the data. Values beyond the whiskers are considered outliers. The notches represent the 95% data range.

The correlation between the number of resubmissions and the total time used is evident. If we consider all clusters, the correlation ($\rho = 0.5682, p < 0.001$) is significant. However, there are significant differences among the groups if we look at the average time per one resubmission. Table 2 depicts the times used for the last two exercise rounds R4 and R5. We can see that TALENTED spent approximately 40% more time per resubmission than ITERATORS, but approximately as much as PASSERS. This suggests that ITERATORs resubmit their work more carelessly than the others.

Table 2: Overall time (in minutes) spent, the number of resubmissions, and the average time on task (in minutes) in rounds R4-R5.

Student group	Time	Resubm. (avg.)	Subm. time (avg.)
PASSERS	139	14.8	9.35
ORDINARIES	173	20.6	8.39
ITERATORS	347	50.7	6.85
AMBITIOUS	269	33.1	8.12
TALENTED	177	18.4	9.60
ALL	220	26.8	8.24

Finally, we considered the difference between the results of

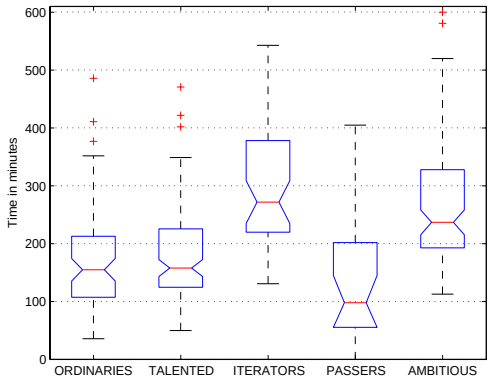

Figure 5: Boxplot of time used for each group in rounds R4 and R5. Four outliers from group ITERATORS and one from group TALENTED have been left out of the picture.

CS majors and minors. Formally, these groups have separate parallel courses. However, the lectures and the TRAKLA2 assignments are the same for both courses. The distributions of students between the two courses is shown in Table 3. The group TALENTED contains considerably more, whereas the AMBITIOUS group has less CS major students than the average percentage for all groups would imply. It is probable that prior knowledge on the topic has some role to play here.

Table 3: Amount of CS majors and CS minors in the courses. The percentages imply the part of majors/minors within each student group.

Student group	N	CS Majors	CS Minors
PASSERS	24	10 (41.7%)	14 (58.3%)
ORDINARIES	72	25 (34.7%)	47 (65.3%)
ITERATORS	45	14 (31.1%)	31 (68.9%)
AMBITIOUS	98	29 (29.6%)	69 (70.4%)
TALENTED	117	57 (48.7%)	60 (51.3%)
ALL	356	135 (37.9%)	221 (62.1%)

5. CONCLUSIONS

The clustering method used in this research provides an instrument to predict and identify several learner groups in computer science education. The data analysis shows that there are two extreme groups that can be identified quite early on the course: PASSERS and TALENTED. Their performance in the examination is very different. The data supports the view that TALENTED have the skill and the motivation to do well, whereas PASSERS probably just wish to pass the compulsory course with minimum effort.

On the other hand, the three other groups, ITERATORS, AMBITIOUS, and ORDINARIES, do equally well in the examination although their behavior in solving the exercises

differs considerably. ITERATORS put a lot of time and effort for the exercises, but this does not seem to improve their examination grade. AMBITIOUS have similar motivation for getting a good exercise grade, but they think more before they submit an exercise anew. The average time between two consecutive resubmissions is considerably higher for them than for ITERATORS. Finally, it is interesting that the ordinary students seem to optimize their effort the best. They get equal grades in the examination without aiming at the best grades in the exercises. The exercise grade has only 30% weight on the final course grade, thus they need not get the best exercise grade to be able to get a satisfactory final grade.

Our experience during the transition from WWW-TRAKLA (limited number of submission) to TRAKLA2 (unlimited resubmissions) was that the average number of resubmissions increased considerably. At the same time, the percentage of students gaining maximum points increased significantly. We did not worry about the increased number of resubmissions since the exercise is reinitialized with new random data in each reiteration. We thought that they would learn better due to the fact that they have to rethink the exercise before they can resubmit. This analysis has revealed that our assumption does not hold for all the learners: ITERATORS work inefficiently[4]. Fortunately, they are a minority in this course (about 13% of the population). However, this result suggests that we really should guide their learning process by limiting the number of resubmissions, at least to some extent. The other groups are not affected by this limit because they use fewer resubmissions in the first place. However, ITERATORS need encouragement to rethink more before resubmitting their solution. Identifying them in the early phase of the course, however, is not an easy task. They use less time per iteration, but the difference to students from other groups grows significant only during the course. Our conclusion is that instead of trying to identify them among other students, we should just give strict limits for resubmissions, either setting a separate limit for each exercise or limiting the total number of allowed submissions. The statistics gathered here gives us guidelines where the limits should be set. About 10 allowed resubmissions per TRAKLA2 exercise would not probably affect much the other groups. However, most ITERATORS should reconsider their working style.

In the future, there are many interesting options to continue the research work and refine the clustering instrument. First, we need more research to justify our assumptions on the behavior of students in each cluster. We could interview a number of students as well as use questionnaires to find out whether our hypothesis of the motivations are correct. Second, we could gather more background information about student groups, such as learning styles [5], goal orientations, resubmission behavior in other programming courses and so on. These studies could reveal new pieces of information on student behavior. Finally, adjusting the resubmission policy is not the only alternative to guide the students into a better learning style. Improving the feedback may also have a role to play. Thus, the system could better adapt to the learners' needs and in this way reduce the inefficient reiteration.

[4]We believe that ITERATORS use trial-and-error strategy to solve the exercises. Another explanation for the large amount of submissions could be, for example, that the learners are practicing for the examination.

6. ACKNOWLEDGMENTS

This work was supported by the Academy of Finland under grant number 210947.

7. REFERENCES

[1] M. Ben-Ari. Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1):45–73, 2001.

[2] S. Benford, E. Burke, E. Foxley, N. Gutteridge, and A. M. Zin. Ceilidh: A course administration and marking system. In *Proceedings of the 1st International Conference of Computer Based Learning*, Vienna, Austria, 1993.

[3] J. Carter, J. English, K. Ala-Mutka, M. Dick, W. Fone, U. Fuller, and J. Sheard. ITICSE working group report: How shall we assess this? *SIGCSE Bulletin*, 35(4):107–123, 2003.

[4] J. English and P. Siviter. Experience with an automatically assessed course. In *Proceedings of The 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'00*, pages 168–171, Helsinki, Finland, 2000. ACM Press, New York.

[5] R. M. Felder and L. K. Silverman. Learning styles and teaching styles in engineering education. *Engineering Education*, 78(7):674–681, 1988.

[6] G. E. Forsythe and N. Wirth. Automatic grading programs. *Communications of the ACM*, 8(5):275–278, 1965.

[7] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Publishing Company, 1991.

[8] A. Korhonen and L. Malmi. Algorithm simulation with automatic assessment. In *Proceedings of The 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'00*, pages 160–163, Helsinki, Finland, 2000. ACM Press, New York.

[9] A. Korhonen, L. Malmi, P. Myllyselkä, and P. Scheinin. Does it make a difference if students exercise on the web or in the classroom? In *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*, pages 121–124, Aarhus, Denmark, 2002. ACM Press, New York.

[10] A. Korhonen, L. Malmi, J. Nikander, and P. Tenhunen. Interaction and feedback in automatically assessed algorithm simulation exercises. *Journal of Information Technology Education*, 2:241–255, 2003.

[11] A. Korhonen, L. Malmi, and P. Silvasti. TRAKLA2: a framework for automatically assessed visual algorithm simulation exercises. In *Proceedings of Kolin Kolistelut / Koli Calling – Third Annual Baltic Conference on Computer Science Education*, pages 48–56, Joensuu, Finland, 2003.

[12] L. Malmi and A. Korhonen. Automatic feedback and resubmissions as learning aid. In *Proceedings of 4th IEEE International Conference on Advanced Learning Technologies, ICALT'04*, pages 186–190, Joensuu, Finland, 2004. IEEE.

[13] L. Malmi, A. Korhonen, and R. Saikkonen. Experiences in automatic assessment on mass courses and issues for designing virtual courses. In *Proceedings of The 7th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'02*, pages 55–59, Aarhus, Denmark, 2002. ACM Press, New York.

[14] P. Naur. Automatic grading of students' ALGOL programming. *BIT 4*, pages 177–188, 1964.

[15] M. J. Rees. Automatic assessment aids for pascal programs. *SIGPLAN Notices*, 17(10):33–42, 1982.

[16] R. Saikkonen, L. Malmi, and A. Korhonen. Fully automatic assessment of programming exercises. In *Proceedings of The 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'01*, pages 133–136, Canterbury, UK, 2001. ACM Press, New York.

[17] V. Tscherter, R. Lamprecht, and J. Nievergelt. Exorciser: Automatic generation and interactive grading of exercises in the theory of computation. In *Fourth International Conference on New Educational Environments*, pages 47–50, 2002.

APPENDIX

Exercises Used in Spring 2004

- Elementary searching
  - binary search, and
  - interpolation search.
- Binary tree traversal algorithms
  - inorder,
  - postorder,
  - levelorder, and
  - preorder with stack.
- Binary heap
  - bottom-up heap construction, and
  - heap manipulation.
- Sorting
  - Quicksort, and
  - Radix-exchange-sort.
- Binary search tree
  - insertion,
  - deletion, and
  - faulty binary search tree.
- Balanced search trees
  - single rotation,
  - double rotation,
  - AVL-tree insertion, and
  - Red-Black tree insertion.
- Binary tries
  - Digital search tree insertion, and
  - Radix search tree insertion.
- Hashing
  - Linear probing,
  - Quadratic probing, and
  - Double hashing.
- Graph algorithms
  - DFS,
  - BFS,
  - Prim's algorithm, and
  - Dijkstra's algorithm.