Towards Designing Optimal Individualized Placement Tests

Octavio Lerma, Olga Kosheleva, Shahnaz Shahbazova, and Vladik Kreinovich

Abstract To find the current level of a student's knowledge, we use a sequence of problems of increasing complexity; if a student can solve a problem, the system generates a more complex one; if a student cannot solve a problem, the system generates an easier one. To find a proper testing scheme, we must take into account that every time a student cannot solve a problem, he/she gets discouraged. To take this into account, we define an overall effect on a student by combining "positive" and "negative" problems with different weights, and we design a testing scheme which minimizes this effect.

1 Optimal Testing: Formulation of the Problem

Need for a placement test. Computers enable us to provide individualized learning, at a pace tailored to each student. In order to start the learning process, it is important

Octavio Lerma
Computational Science Program, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: lolerma@gmail.com

Olga Kosheleva
Department of Teacher Education, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Shahnaz Shahbazova
Department of Digital Technologies and Applied Informatics
Azerbaijan State University of Economics UNEC, Baku, AZ1001, Azerbaijan
e-mail: shahnaz_shahbazova@unec.edu.az
and
Institute of Control Systems, Ministry of Science and Education of the Republic of Azerbaijan
68 Bakhtiyar Bahabzadeh str., Baku, Azerbaijan, AZ1141, e-mail: shahbazova@cyber.az

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso, 500 W. University
El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

to find out the current level of the student's knowledge, i.e., to place the student at an appropriate level.

Usually, such placement tests use a sequence of N problems of increasing complexity; if a student is able to solve a problem, the system generates a more complex one; if a student cannot solve a problem, the system generates an easier one – until we find the exact level of this student. After this, the actual learning starts.

A seemingly natural idea. A natural tendency is to speed up this preliminary stage and to get to actual learning as soon as possible, i.e., to minimize the number of problems given to a student.

Resulting solution: bisection. The solution to the corresponding optimization problem is a well-known bisection procedure (also known, in the discrete case, as binary search; see, e.g., [1]. To describe this procedure, let us add, to the problems of levels 1 though N, two fictitious "problems":

- a trivial problem that everyone can solve – which will be called level 0; and
- a very complex problem that no one can solve – which will be called level $N+1$.

In the beginning, we know that a student can solve a problem at level 0 (since everyone can solve a problem at this level) and cannot solve a problem of level $N+1$ (since no one can solve problems at this level).

After the tests, we may know that a student can or cannot solve some problems. Let i be the highest level of problems that a student has solved, and let j be the lowest level of problems that a student cannot solve. If $j = i+1$, then we know exactly where the student stands: he or she can solve problems of level i but cannot solve problems of the next complexity level $i+1$.

If $j > i = 1$, we need further testing to find out the exact level of knowledge of this student. In the bisection method, we give the student a problem on level $m \stackrel{\text{def}}{=} (i+j)/2$. Depending on whether a student succeeded in solving this problem or not, we either increase i to m or decrease j to m.

In both cases, we decrease the interval by half. We started with the interval

$$[0, N+1].$$

After s steps, we get an interval of width $2^{-s} \cdot (N+1)$. Thus, when $2^{-s} \cdot (N+1) \leq 1$, we get an interval of width 1, i.e., we have determined the student's level of knowledge. This requires $s = \lceil \log_2(N+1) \rceil$ steps.

The problem with bisection. The problem with bisection is that every time a student is unable to solve a problem, he/she gets discouraged; in other words, such problems have a larger effect on the student than problems which the student can solve. For example, if a student is unable to solve any problem already on level 1, this students will get a negative feedback on all $\approx \log_2(N+1)$ problems – and will be thus severely discouraged.

How to solve this problem: an idea. To take the possible discouragement into account, let us define an overall effect on a student by combining "positive" and "negative" problems with different weights.

In other words, we will count an effect of a positive answer as one, and the effect of a negative answer as $w > 1$. For positive answers, the student simply gets tired, while for negative answers, the student also gets stressed and frustrated. The value w can be determined for each individual student.

Resulting optimization problem. For each testing scheme, the resulting effect on each student can be computed as the number of problems that this student solved plus w multiplied by the number of problems that this student did not solve. This effect depends on a student: for some students it may be smaller, for other students it may be larger. As a measure of quality of a testing scheme, let us consider the worst-case effect, i.e., the largest effect over all possible students.

Our objective is to find a testing scheme which places all the students while leading to the smallest effect on a student, i.e., for which the worst-case effect is the smallest possible.

2 Optimal Testing: Analysis of the Problem and the Resulting Algorithm

Testing scheme: a general description. A general testing scheme works as follows. First, we ask a student to select a problem of some level n between 1 and N. Depending on whether a student succeeds or not, we ask the student to solve a problem of some other level $n' \neq n$:

- if the student succeeded in solving the problem, and $n < N$, we assign a level $n' > n$;
- if the student succeeded in solving the problem and $n = N$, we finish the process;
- if the student did not succeed in solving the problem, and $n = 1$, we finish the process;
- if the student did not succeed in solving the problem and $n > 1$ is not the lowest possible level, we take $n' < n$.

Depending on whether the student succeeds in solving the second problem (at level n'), we ask the student to solve a problem at some other level n'':

- if the student succeeded in solving both problems and $n' < N$, we assign a level
$$n'' > n';$$
- if the student succeeded in solving both problems and $n' = N$, we finish the process;
- if the student succeeded in solving the first problem but did not succeed in solving the second problem – and $n' - n > 1$ – we take n'' for which $n < n'' < n'$;
- if the student succeeded in solving the first problem but did not succeed in solving the second problem – and $n' - n = 1$ – we finish the process;
- if the student did not succeed in solving both problems and $n' = 1$, we finish the process;

- if the student did not succeed in solving both problems and $n' > 1$, we take
$$n'' < n'.$$

This procedure continues until it finishes. As a result, we get the knowledge level of a student, i.e., we get the level i for which the student can solve the problems on this level but cannot solve problems on the next level $i+1$. This level i can take any of the $N+1$ values from 0 to N.

Deriving the main formula. Let $e(x)$ denote the smallest possible effect needed to find out the knowledge level of a student in a situation with $x = N+1$ possible student levels.

In the beginning, we know that a student's level is somewhere between 0 and N. In the optimal testing scheme, we first ask a student to solve a problem of some level n. Let us consider both possible cases: when the student succeeds in solving this problem and when the student doesn't.

If the student successfully solved the level n problem, this means that after providing a 1 unit of effect on the student, we know that this student's level is somewhere between n and N. In this case, we must select among $N-n+1 = x-n$ possible student levels. By definition of the function $e(x)$, the remaining effect is equal to $e(x-n)$. Thus, in this case, the total effect on a student is equal to $1+e(x-n)$.

If the student did not solve the problem of level n, this means that after producing w units of effect on the student, we learn that the student's level is somewhere between 0 and $n-1$. The remaining effect to determine the student's level is equal to $e(n)$. Thus, the total effect on the student is equal to $w+e(n)$.

The worst-case effect $e(x)$ is, by definition, the largest of the two effects $1+e(x-n)$ and $w+e(n)$: $e(x) = \max(1+e(x-n), w+e(n))$. In the optimal method, we select n (from 1 to $N = x-1$) for which this value is the smallest possible. Thus, we conclude that

$$e(x) = \min_{1 \le n < x} \max(1+e(x-n), w+e(n)). \qquad (1)$$

The value $n(x)$ corresponding to x can be determined as the value for which the right-hand side of the expression (1) attains its minimum.

Comment. It is worth mentioning that similar formulas appear in other situations; see, e.g., [2, 3]. Because of this similarity, in this paper, we have used – after a proper modification – some of the mathematics from [2, 3].

Towards the optimal testing scheme. For $x = 1$, i.e., for $N = 0$, we have $e(1) = 0$. We can use the formula (1) to sequentially compute the values $e(2)$, $e(3)$, $\ldots$, $e(N+1)$ by using formula (1); while computing these values, we also compute the corresponding minimizing values $n(2)$, $n(3)$, $\ldots$, $n(N+1)$.

In the beginning, we know that a student's level ℓ is between 0 and N, i.e., that $0 \le \ell < N+1$. At each stage of the testing scheme, we know that the student's level ℓ is between some numbers i and j: $i \le \ell < j$, where i is the largest of the levels for which the student succeeded in solving the problem, and j is the smallest level for which the student was unable to solve the corresponding problem. In this case, we have $j-i$ possible levels i, $i+1$, $\ldots$, $j-1$. In accordance with the above algorithm, we should thus ask a question corresponding to the $n(j-i)$-th of these levels. If we count from 0, this means the level $i+n(j-i)$. Thus, we arrive at the following algorithm.

Resulting optimal testing scheme. First, we take $e(1)=0$, and sequentially compute the values $e(2)$, $e(3)$, $\ldots$, $e(N+1)$ by using the main formula (1), while simultaneously recording the corresponding minimizing values $n(2)$, $\ldots$, $n(N+1)$.

At each stage of testing, we keep track of the bounds i and j for the student's level. In the beginning, $i=0$ and $j=N+1$. At each stage, we ask the student to solve a problem at level $m=i+n(j-i)$.

- If the student succeeds in solving this problem, we replace the original lower bound i with the new bound m.
- If the student did not succeed in solving the problem on level m, we replace the original upper bound j with the new bound m.

We stop when $j=i+1$; this means that the student's level is i.

Numerical example 1. To make our algorithm clearer, we provide two easy-to-follow numerical examples. In the first example, we consider the case when $N=3$ and $w=3$. In this example, we need to compute the values $e(2)$, $e(3)$, and $e(4)$.

- We take $e(1)=0$.
- When $x=2$, the only possible value for n is $n=1$, so

$$e(2)=\min_{1\le n<2}\{\max\{1+e(2-n),3+e(n)\}\}=$$

$$\max\{1+e(1),3+e(1)\}=\max\{1,3\}=3.$$

  Here, $e(2)=3$, and $n(2)=1$.
- To find $e(3)$, we must compare two different values $n=1$ and $n=2$:

$$e(3)=\min_{1\le n<3}\{\max\{1+e(3-n)),3+e(n)\}\}=$$

$$\min\{\max\{1+e(2),3+e(1)\},\max\{1+e(1),3+e(2)\}\}=$$

$$\min\{\max\{4,3\},\max\{1,6\}\}=\min\{4,6\}=4.$$

  Here, the minimum is attained when $n=1$, so $n(3)=1$.
- To find $e(4)$, we must consider three possible values $n=1$, $n=2$, and $n=3$, so

$$e(4)=\min_{1\le n<4}\{\max\{1+e(4-n),3+e(n))\}\}=$$

$$\min\{\max\{1+e(3),3+e(1)\},\max\{1+e(2),3+e(2)\},$$

$$\max\{1+e(1),3+e(3)\}\}=$$

$$\min\{\max\{5,3\},\max\{4,6\},\max\{1,7\}\}=\min\{5,6,7\}=5.$$

Here, the minimum is attained when $n = 1$, so $n(4) = 1$.

So here, the optimal testing procedure is as follows. First, we have $i = 0$ and $j = N+1 = 4$, so we ask a student to solve a problem of level $m = i+n(j-i) = 1$.

If a student did not succeed in solving this level 1 problem, we replace the original upper bound j with the new value $j = 1$. Now, $j = i+1$, so we conclude that the student is at level 0.

If the student succeeds in solving the level 1 problem, we take $i = 1$ (and keep $j = 4$ the same). In this case, the next problem is of level $m = i+n(j-i) = 2$.

If the student fails to solve the level 2 problem, then we replace the original upper bound j with the new value $j = m = 2$. Here, $j = i+1$, so we conclude that the student is at level 1.

If the student succeeds is solving the problem at level 2, then we replace the previous lower bound i with the new bound $i = m = 2$. Now, we give the student the next problem of level $i+n(j-i) = 2+n(4-2) = 2+1 = 3$.

If the student fails to solve this problem, then we replace the original upper bound j with the new value $j = m = 3$. Here, $j = i+1$, so we conclude that the student is at level 2.

If the student succeeds in solving the problem at level 3, then we replace the previous lower bound i with the new bound $i = m = 3$. Here, $j = i+1$, so we conclude that the student is at level 2.

Comment. In this case, the optimal testing scheme is the most cautious one, when we increase the level by one every time. This way, we are guaranteed that a tested student has no more than one negative experience.

Numerical example 2. Let us now consider an example when $N = 3$ and $w = 1.5$.

- We take $e(1) = 0$.
- When $x = 2$, then

$$e(2) = \min_{1\le n<2} \{\max\{1+e(2-n), 3+e(n)\}\} =$$

$$\max\{1+e(1), 1.5+e(1)\} = \max\{1, 1.5\} = 1.5.$$

  Here, $e(2) = 1.5$, and $n(2) = 1$.
- To find $e(3)$, we must compare two different values $n = 1$ and $n = 2$:

$$e(3) = \min_{1\le n<3} \{\max\{1+e(3-n)), 1.5+e(n)\}\} =$$

$$\min\{\max\{1+e(2), 1.5+e(1)\}, \max\{1+e(1), 1.5+e(2)\}\} =$$

$$\min\{\max\{2.5, 1.5\}, \max\{1, 3\}\} = \min\{2.5, 3\} = 2.5.$$

  Here, the minimum is attained when $n = 1$, so $n(3) = 1$.
- To find $e(4)$, we must consider three possible values $n = 1$, $n = 2$, and $n = 3$, so

$$e(4) = \min_{1\le n<4} \{\max\{1+e(4-n), 1.5+e(n))\}\} =$$

$$\min\{\max\{1+e(3),1.5+e(1)\},\max\{1+e(2),1.5+e(2)\},$$
$$\max\{1+e(1),1.5+e(3)\}\} =$$
$$\min\{\max\{3.5,1.5\},\max\{2.5,3\},\max\{1,4\}\} = \min\{3.5,3,4\} = 3.$$

Here, the minimum is attained when $n = 2$, so $n(4) = 2$.

So here, the optimal testing procedure is as follows. First, we have $i = 0$ and $j = N+1 = 4$, so we ask a student to solve a problem of level $m = i+n(j-i) = 2$.

If a student did not succeed in solving the level 2 problem, we replacing the original upper bound j with the new value $j = 2$. Now, we ask the student to solve a problem on level $m = i+n(j-i) = 1$. If a student succeeds, his/her level is 1; if the student does not succeed, his/her level is 0.

If the student succeeds in solving the level 2 problem, we take $i = 2$ (and keep $j = 4$ the same). In this case, the next problem is of level $m = i+n(j-i) = 3$. If a student succeeds, his/her level is 3; if the student does not succeed, his/her level is 2.

Comment. In this case, the optimal testing scheme is the bisection.

Computational complexity. For each n from 1 to N, we need to compare n different values. So, the total number of computational steps is proportional to

$$1+2+\ldots+N = O(N^2).$$

Additional problem. When N is large, N^2 may be too large. In some applications, the computation of the optimal testing scheme may takes too long. For this case, we have developed a faster algorithm for producing a testing scheme which is only asymptotically optimal.

3 A Faster Algorithm for Generating an Asymptotically Optimal Testing Scheme

Description of the algorithm. First, we find the real number $\alpha \in [0,1]$ for which $\alpha + \alpha^w = 1$. This value α can be obtained, e.g., by applying bisection [1] to the equation $\alpha + \alpha^w = 1$.

Then, at each step, once we have the lower bound i and the upper bound j for the (unknown) student level ℓ, we ask the student to solve a problem at the level $m = \lfloor \alpha \cdot i + (1-\alpha) \cdot j \rfloor$.

Comments. This algorithm is similar to bisection, expect that bisection corresponds to $\alpha = 0.5$. This makes sense, since for $w = 1$, the equation for α takes the form $2\alpha = 1$, hence $\alpha = 0.5$. For $w = 2$, the solution to the equation $\alpha + \alpha^2 = 1$ is the well-known golden ratio $\alpha = \dfrac{\sqrt{5}-1}{2} \approx 0.618$.

Computational complexity. At each step, we end up with either an interval $[i,m]$ whose width is $1-\alpha$ from the original size, or with the interval $[m,j]$ whose width is α from the original size. Since $\alpha \geq 1-\alpha$, the worst-case decrease is decrease by a factor of α. In k steps, we decrease the width N to $\leq N \cdot \alpha^k$. Thus, we stop for sure when $N \cdot \alpha^k \leq 1$, i.e., after $k = O(\log(N))$ problems.

At each level, we need a constant number of computation steps to compute the next level, so the overall computation time is $O(\log(N))$.

In what sense the resulting testing scheme is asymptotically optimal. We will prove that for this scheme, there is a constant C such that for every N, the worst-case effect from this scheme differs from the worst-case effect of the optimal testing scheme by no more than C.

Proof that the resulting testing scheme is indeed asymptotically optimal. Let us denote the optimal effect by $e(N)$ and the worst-case effect corresponding to our procedure by $e_0(N)$. Let us also denote $K = 2^{-\alpha}$. To prove our result, we will prove that there exist constants $C > 0$ and $C_1 > 0$ such that for every N, we have $K \cdot \log_2(N) \leq e(N)$ and $e_0(N) \leq K \cdot \log_2(N) + C - \dfrac{C_1}{N}$. By definition, $e(N)$ is the smallest worst-case effect of all possible testing schemes, thus, $e(N) \leq e_0(N)$. So, if we prove the above two inequalities, we will indeed prove that our algorithm is asymptotically optimal.

Proof of the first inequality. Let us first prove the first inequality by induction over N. The value $N = 1$ represents the induction base. For this value, $K \cdot \log_2(1) = 0 = e(1)$, so the inequality holds.

Let us now describe the induction step. Suppose that we have already proved the inequality $K \cdot \log_2(n) \leq e(n)$ for all $n < N$. Let us prove that $K \cdot \log_2(N) \leq e(N)$.

Due to our main formula, $e(N)$ is the smallest of the values

$$\max\{1 + e(x-n), w + e(n)\}$$

over $n = 1, 2, \ldots, N-1$. So, to prove that $K \cdot \log_2(N)$ is indeed the lower bound for $e(N)$, we must prove that $K \cdot \log_2(N)$ cannot exceed each of these values, i.e., that

$$K \cdot \log_2(N) \leq \max\{1 + e(N-n), w + e(n)\}$$

for every $n = 1, 2, \ldots, N-1$. For these n, we have $n < N$ and $N - n < N$, so for all these values, we already know that $K \cdot \log_2(n) \leq e(n)$ and $K \cdot \log_2(N-n) \leq e(N-n)$. Therefore,

$$1 + K \cdot \log_2(N-n) \leq 1 + e(N-n),$$

$$w + K \cdot \log_2(n) \leq w + e(n),$$

and

$$\max\{1 + K \cdot \log_2(N-n), w + K \cdot \log_2(n)\} \leq \max\{1 + e(N-n), w + e(n)\}.$$

So, to prove the desired inequality, it is sufficient to prove that

$$K \cdot \log_2(N) \le \max\{1 + K \cdot \log_2(N-n), w + K \cdot \log_2(n)\}.$$

We will prove this inequality by considering two possible cases: $n \le (1-\alpha) \cdot N$ and $n \ge (1-\alpha) \cdot N$.

- When $n \le (1-\alpha) \cdot N$, we have $N - n \ge \alpha \cdot N$ and therefore, $1 + K \cdot \log_2(N-n) \ge z$, where

  $$z \overset{\text{def}}{=} 1 + K \cdot \log_2(\alpha \cdot N) = 1 + K \cdot \log_2(N) + K \cdot \log_2(\alpha).$$

  Here, by definition of $K = 2^{-\alpha}$, we have $\log_2(\alpha) = -1/K$, hence

  $$1 + K \cdot \log_2(\alpha) = 0,$$

  and so $z = K \cdot \log_2(N)$. In this case,

  $$K \cdot \log_2(N) \le z = 1 + K \cdot \log_2(N-n) \le \max\{1 + K \cdot \log_2(N-n), w + K \cdot \log_2(n)\}.$$

- When $n \ge (1-\alpha) \cdot N$, we have $w + K \cdot \log_2(n) \ge t$, where

  $$t \overset{\text{def}}{=} w + K \cdot \log_2((1-\alpha) \cdot N) = w + K \cdot \log_2(N) + K \cdot \log_2(1-\alpha).$$

  By definition of α, we have $1 - \alpha = \alpha^w$, so $\log_2(1-\alpha) = w \cdot \log_2(\alpha)$ and thus, $w + K \cdot \log_2(1-\alpha) = w \cdot (1 + K \cdot \log_2(\alpha)) = 0$. Hence, $t = K \cdot \log_2(N)$. So, in this case,

  $$K \cdot \log_2(N) \le t = w + K \cdot \log_2(n) \le \max\{1 + \log_2(N-n), w + K \cdot \log_2(n)\}.$$

In both cases, we have the desired inequality. The induction step is proven, and so, indeed, for every N, we have $K \cdot \log_2(N) \le e(N)$.

Proof of the second inequality. Let us now prove that there exist real numbers $C > 0$ and $C_1 > 0$ for which, for all N, we have $e_0(N) \le K \cdot \log_2(N) + C - \frac{C_1}{N}$. To prove this inequality, we will pick a value N_0, prove that this inequality holds for all $N \le N_0$, and then use mathematical induction to show that it holds for all $N > N_0$ as well.

Induction basis. Let us first find the conditions on C, C_1, and N_0 under which for all $N \le N_0$, we have $e_0(N) \le K \cdot \log_2(N) + C - \frac{C_1}{N}$. Subtracting $K \cdot \log_2(N)$ and adding $\dfrac{C_1}{N}$ to both sides of the this inequality, we get

$$C \ge \frac{C_1}{N} + e_0(N) - K \cdot \log_2(N)$$

for all N from 1 to N_0. So, to guarantee that this inequality holds, if we have already chosen N_0 and C_1, we can choose

$$C = \max_{1 \le N \le N_0} \left(\frac{C_1}{N} + e_0(N) - K \cdot \log_2(N) \right).$$

Induction step. Let us assume that for all $n < N$ (where $N > N_0$), we have proven that

$$e_0(n) \le K \cdot \log_2(n) + C - \frac{C_1}{n}.$$

We would like to conclude that

$$e_0(N) \le K \cdot \log_2(N) + C - \frac{C_1}{N}.$$

According to the definition of $e_0(N)$, we have

$$e_0(N) = \max\{1 + e_0(N-n), w + e_0(n))\},$$

where $n = \lfloor (1-\alpha) \cdot N \rfloor$. Due to induction hypothesis, we have

$$e_0(n) \le K \cdot \log_2(n) + C - \frac{C_1}{n}$$

and

$$e_0(N-n) \le K \cdot \log_2(N-n) + C - \frac{C_1}{N-n}.$$

Therefore,

$$e_0(N) \le \max\left\{ 1 + K \cdot \log_2(N-n) + C - \frac{C_1}{N-n}, w + K \cdot \log_2(n) + C - \frac{C_1}{n} \right\}.$$

Thus, to complete the proof, it is sufficient to conclude that this maximum does not exceed

$$K \cdot \log_2(N) + C - \frac{C_1}{N}.$$

In other words, we must prove that

$$1 + K \cdot \log_2(N-n) + C - \frac{C_1}{N-n} \le K \cdot \log_2(N) + C - \frac{C_1}{N}$$

and that

$$w + K \cdot \log_2(n) + C - \frac{C_1}{N-n} \le K \cdot \log_2(N) + C - \frac{C_1}{N}. \tag{2}$$

Without losing generality, let us show how we can prove the second of these two inequalities. Since $n = \lfloor (1-\alpha) \cdot N \rfloor$, the left-hand side of the inequality (2) can be rewritten as

$$W_1 + K \cdot \log_2((1-\alpha) \cdot N) + K \cdot (\log_2(n) - \log_2((1-\alpha) \cdot N)) + C - \frac{C_1}{n}.$$

We already know that $w+K\cdot\log_2((1-\alpha)\cdot N)=K\cdot\log_2(N)$. Thus, the left-hand side of (2) takes the simpler form

$$K\cdot\log_2(N)+K\cdot(\log_2(n)-\log_2((1-\alpha)\cdot N))+C-\frac{C_1}{n}.$$

Substituting this expression into (2) and canceling the terms $K\cdot\log_2(N)$ and C in both sides, we get an equivalent inequality

$$K\cdot(\log_2(n)-\log_2((1-\alpha)\cdot N))-\frac{C_1}{n}\le-\frac{C_1}{N}. \qquad (3)$$

Let us further simplify this inequality. We will start by estimating the difference $\log_2(n)-\log_2((1-\alpha)\cdot N)$. To estimate this difference, we will use the intermediate value theorem, according to which, for every smooth function $f(x)$, and for arbitrary two values a and b, we have $f(a)-f(b)=(a-b)\cdot f'(\xi)$ for some $\xi\in[a,b]$. In our case, $f(x)=\log_2(x)=\dfrac{\ln(x)}{\ln(2)}$, $a=n$, and $b=(1-\alpha)\cdot N$. Here, $f'(\xi)=\dfrac{1}{\xi\cdot\ln(2)}$, so $f'(\xi)\le\dfrac{1}{n\cdot\ln(2)}$; also, $|a-b|\le 1$, so, the difference $\log_2(n)-\log_2((1-\alpha)\cdot N)$ can be estimated from above by:

$$\log_2(n)-\log_2((1-\alpha)\cdot N)\le\frac{1}{n\cdot\ln(2)}.$$

Hence, the above inequality holds if the following stronger inequality holds:

$$\frac{K}{n\cdot\ln(2)}-\frac{C_1}{n}\le-\frac{C_1}{N},$$

or, equivalently,

$$\frac{C_1}{N}\le\frac{C_1-K/\ln(2)}{n}. \qquad (4)$$

Here, $n\ge(1-\alpha)\cdot N-1$, i.e.,

$$\frac{n}{N}\ge(1-\alpha)-\frac{1}{n}.$$

When $N\to\infty$, we have $n\to\infty$ and $\dfrac{1}{n}\to 0$. Thus, for every $\varepsilon>0$, there exists an N_0 starting from which $\dfrac{1}{n}\le\varepsilon$ and hence, $n\ge(1-\alpha-\varepsilon)\cdot N$. For such sufficiently large N, the inequality (4) can be proven if we have

$$\frac{C_1}{N}\le\frac{C_1-K/\ln(2)}{(1-\alpha-\varepsilon)\cdot N},$$

i.e., if we have

$$C_1 \le \frac{C_1 - K/\ln(2)}{1-\alpha-\varepsilon}. \tag{5}$$

Since $0 \le \alpha \le 1$, for sufficiently large C_1, this inequality is true. For such C_1, therefore, the induction can be proven and thus, the second inequality is true.

The statement is proven.

4 What If We Also Know Probabilities

Formulation of the problem. In some cases, we also know the frequencies p_0, $p_1, \ldots, p_N$ with which students are at the corresponding levels. These frequencies can be alternatively described by the corresponding cumulative distribution function $F(i) \stackrel{\text{def}}{=} \text{Prob}(\ell < i) = p_0 + p_1 + \ldots + p_{i-1}$. In this situation, instead of the *worst-case* effect, we can alternatively consider the *average* effect – and look for a testing scheme which minimizes the average effect.

Towards a scheme which minimizes the average effect. Let $e(i,j)$ be the smallest possible conditional average effect under the condition that a student's actual level is between i and j, i.e., that the student has successfully solves a problem at level i and was unable to solve the problem at level j. (The original situation corresponds to $i = 0$ and $j = N+1$.)

In this situation, we ask the student to solve a problem at some level $n \in (i,j)$. Let us consider both possible cases: when the student was able to solve this problem, and when a student was unable to solve this problem.

If a student was able to solve the problem at level n, this means that the student's level is in between n and j. By definition of a function $e(\cdot,\cdot)$, the expected remaining effect is equal to $e(n,j)$. Thus, in this case, the overall expected effect on the student is equal to $1 + e(n,j)$. The conditional probability of this case can be obtained by dividing the probability $F(j) - F(n)$ that the student's level is between n and j by the original probability $F(j) - f(i)$ that the student's level is between i and j. Thus, this probability is equal to $\dfrac{F(j)-F(n)}{F(j)-F(i)}$.

If a student was unable to solve the problem at level n, this means that the student's level is in between i and n. By definition of a function $e(\cdot,\cdot)$, the expected remaining effect is equal to $e(i,n)$. Thus, in this case, the overall expected effect on the student is equal to $w + e(i,n)$. The conditional probability of this case can be obtained by dividing the probability $F(n) - F(i)$ that the student's level is between i and n by the original probability $F(j) - f(i)$ that the student's level is between i and j. Thus, this probability is equal to $\dfrac{F(n)-F(i)}{F(j)-F(i)}$.

Thus, we have the expected effect $1 + e(n,j)$ with probability $\dfrac{F(j)-F(n)}{F(j)-F(i)}$, and the expected effect $w + e(i,n)$ with probability $\dfrac{F(n)-F(i)}{F(j)-F(i)}$. So, the overall expected effect is equal to

$$\frac{F(j)-F(n)}{F(j)-F(i)} \cdot (1+e(n,j)) + \frac{F(n)-F(i)}{F(j)-F(i)} \cdot (w+e(i,n)).$$

Since we want to minimize the average effect, we select n for which this value is the smallest possible. Thus, we arrive at the following formula:

Main formula: average case.

$$e(i,j) = \min_{i \le n < j} \left(\frac{F(j)-F(n)}{F(j)-F(i)} \cdot (1+e(n,j)) + \frac{F(n)-F(i)}{F(j)-F(i)} \cdot (w+e(i,n)) \right). \quad (6)$$

Towards the optimal testing scheme. When $j = i+1$, we know that the student's level is i, so no additional testing is needed and the effect is 0: $e(i,i+1) = 0$. We can start with these values and sequentially use the formula (6) to compute the values $e(i,i+2)$, $e(i,i+3)$, etc. In each case, we find $n(i,j)$ for which the minimum is attained.

Resulting optimal testing scheme. First, we take $e(i,i+1) = 0$ for all i, and use the formula (6) to sequentially compute the values $e(i,i+2)$, $e(i,i+3)$, $\ldots$, until we cover all possible values $e(i,j)$. For each i and j, we record the value $n(i,j)$ for which the right-hand side of the formula (6) is the smallest.

At each stage of the testing, we keep track of the bounds i and j for the student's level. In the beginning, $i = 0$ and $j = N+1$. At each stage, we ask the student to solve a problem at level $m = n(i,j)$.

- If the student succeeds in solving this problem, we replace the original lower bound i with the new bound m.
- If the student did not succeed in solving the problem on level m, we replace the original upper bound j with the new bound j.

We stop when $j = i+1$; this means that the student's level is i.

Computational complexity. For each of $O(N^2)$ pairs $i < j$ of numbers from 0 to N, we need to compare $j - i = O(N)$ different values. So, the total number of computational steps is proportional to $O(N^2) \cdot O(N) = O(N^3)$.

Comment. For large N, this computation time may be too large. It would be nice – similarly to the worst-case optimization – to come up with a faster algorithm even if it generates only an asymptotically optimal testing scheme.

Acknowledgments

This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

The authors are thankful to all the participants of the Third World Conference on Soft Computing (San Antonio, Texas, December 16–18, 2013) and to the anonymous referees for valuable suggestions.

References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.
2. H. T. Nguyen, V. Kreinovich, and E. Kamoroff, “Asymmetric Information Measures: How to Extract Knowledge From an Expert So That the Expert’s Effort is Minimal”, *International Journal of Automation and Control (IJAAC)*, 2008, Vol. 2, No. 2/3, pp. 153–177.
3. R. A. Trejo, J. Galloway, C. Sachar, V. Kreinovich, C. Baral, and L. C. Tuan, “From Planning to Searching for the Shortest Plan: An Optimal Transition”, *International Journal of Uncertainty, Fuzziness, Knowledge-Based Systems (IJUFKS)*, 2001, Vol. 9, No. 6, pp. 827–838.