ISSN 0005-1179 (print), ISSN 1608-3032 (online), Automation and Remote Control, 2025, Vol. 86, No. 6, pp. 589–597.

Optimization by a Probabilistic Criterion in a Dynamic Test Passing Model

S. V. Ivanov[*,a] **and A. V. Stepanov**[*,b]

[*] *Moscow Aviation Institute (National Research University), Moscow, Russia*
e-mail: [a] *sergeyivanov89@mail.ru,* [b] *rus.fta@yandex.ru*

Received February 23, 2025
Revised April 8, 2025
Accepted April 12, 2025

Abstract—A dynamic model of passing a time-limited test is considered. The problems of finding program and positional strategies that maximize the probability of test passing are stated. A strategy is to complete or not complete a current test task. A positional strategy is defined as a function of the time spent since the test start and the sum of points scored for previous tasks. Bellman's dynamic programming method is used to design a positional strategy. An algorithm based on the branch-and-bound method is proposed to find an optimal program strategy. The results of calculations are presented and compared with those of a similar problem with a time limit (i.e., when the test is considered unpassed if the testee does not meet the limit).

Keywords: testing model, probabilistic criterion, control design

DOI: 10.31857/S0005117925060076

1. INTRODUCTION

With the modern application level of information technologies in education, it is topical to develop, in particular, the theory of computerized adaptive testing (CAT). Its foundations were laid at the end of the twentieth century, e.g., in the works of G. Rasch [1]. A good survey of the state-of-the-art results in this field can be found in [2]. Conceptually CAT involves information about the testee's performance (the results of learning) and passing of a control event in the process of forming the test content and deciding on test completion. (Below testees will be called subjects.) The processing of this information and decision-making are often implemented with contemporary machine learning and artificial intelligence techniques. The same technologies are also used to form individual learning trajectories in learning management systems (LMSs); for example, see [3, 4]. Recent works in this area include [5–7]. Such technologies are commonly applied by test organizers and LMS software developers. However, the testing problem can also be viewed from the subject's standpoint, who needs to develop an individual strategy for passing an (often, time-limited) test. The test completion time is directly related to the subject's response time to test tasks. Many publications have been devoted to probabilistic models of the response time of a subject or an LMS user to a task; for example, we refer to [8–10]. The problem of finding the subject's optimal strategy by the criterion of maximizing the probability of scoring a given sum of points under a time limit was considered in [11]. A similar problem in a quantile formulation (maximizing the sum of points scored during a test under a probabilistic time constraint) was studied in [12]. In the last two papers mentioned, the following stochastic programming problems were solved: before a test (a priori), a subject chooses as an individual strategy a group of tasks he/she will tackle first. Other test tasks are performed by the subject if he/she has enough time after completing the first group of tasks. The above formulations neglect the sequence in which a subject performs test tasks. This

may be natural if a subject receives no information about the correctness of performing a current task and, consequently, about the current sum of points scored. However, very often, especially in LMS testing, e.g., [4], such information is available to a user and can be utilized by him/her during the test. Therefore, it is relevant to consider dynamic testing models with a test passing strategy corrected after each task considered by a subject.

This paper deals with a dynamic model of passing a fixed time-limited test by the criterion of maximizing the probability of the subject's scoring at least a given sum of points. The problems of finding the subject's program and positional strategies are stated. Algorithms are proposed to solve these problems based on stochastic and dynamic programming methods. The resulting solutions are compared with those obtained in [11, 12] for the same initial data.

2. DESCRIPTION OF THE MODEL AND CONTROL DESIGN PROBLEM

Consider a dynamic system describing test passing by a subject (e.g., a student). At the kth step, the subject is asked to perform a task; in case of success, he/she is awarded b_k points, $k = \overline{1,n}$. It is required to score at least φ points to pass the test successfully. The test completion time is limited by $\bar{T}$. The time for performing the kth task is described by a random variable τ_k. The correctness of performing the kth task is described by a random variable X_k : it takes value 1 if the kth task has been performed correctly and value 0 otherwise. A strategy is defined by a set of variables $u = (u_1, \ldots, u_n)$, where $u_k = 1$ if the subject attempts to perform the kth task and $u_k = 0$ otherwise. Now we introduce the state variables of the dynamic system. Let T_k be the time spent to complete the first k tasks and S_k be the sum of points scored during this time. Then the system dynamics are described by the equations

$$T_k = T_{k-1} + \tau_k u_k, \tag{1}$$

$$S_k = S_{k-1} + b_k X_k I\{T_k \leqslant \bar{T}\} u_k, \tag{2}$$

$$T_0 = 0,\ S_0 = 0, \quad k = \overline{1,n}, \tag{3}$$

where $I\{\cdot\}$ denotes the indicator of the event in curly brackets, equal to 1 if the condition is satisfied and to 0 otherwise. Thus, b_k points are awarded if the subject has spent no more than $\bar{T}$ units of time on the first k tasks. Assume that all the random variables τ_k are discrete with a finite set $\mathcal{T}_k$ of realizations and let $p_k(t,x) = \mathbf{P}\{\tau_k = t, X_k = x\}$. In this paper, for all $k = \overline{2,n}$, the sigma algebra generated by the random variables X_k and τ_k is supposed to be independent of the sigma algebra generated by the random variables $X_1, \ldots, X_{k-1}$ and $\tau_1, \ldots, \tau_{k-1}$. Such an assumption means that the answer to the current test questions does not affect further test performance. How does the knowledge of the correctness of performing previous tasks influence the correctness of performing subsequent ones? This issue requires additional statistical analysis. Let $S(u)$ be the random sum of points scored, equal to the value S_n under the chosen control action u.

We state the problem of finding the maximum of the probability function

$$P^1_{\varphi,\bar{T}}(u) = \mathbf{P}\{S(u) \geqslant \varphi\} \to \max_{u \in \{0,1\}^n}. \tag{4}$$

This problem is to design a program strategy for choosing tasks to be performed under which the subject will maximize the probability of test passing.

Now assume that the control action is chosen in the class of positional strategies. In other words, when choosing his/her strategy at each step, the subject considers the current time spent and the current sum of points scored for the previous tasks. We denote by $\mathbf{u}_k(T_{k-1}, S_{k-1})$ the subject's strategy at the kth step, represented by the following possible values of some function $\mathbf{u}_k$ of the system's state variables: 1 if the subject tries to perform the kth task and 0 otherwise. Let $\mathbf{u}$ be a

vector composed of the functions $\mathbf{u}_1, \ldots, \mathbf{u}_n$ to describe a positional strategy. We denote by $\mathbf{S}(\mathbf{u})$ the random sum S_n of points defined by equations (1) and (2) when substituting $u_k = \mathbf{u}_k(T_k, S_k)$ therein. Boldface is used here to distinguish the positional strategy $\mathbf{u}$ (a function) from the program strategy u (a set of variables) as well as the sum $\mathbf{S}(\mathbf{u})$ of points scored by the positional strategy from the sum S_n of points scored by the program strategy.

We state the problem of finding the maximum of the probability function under the positional strategy:

$$P^2_{\varphi,\bar{T}}(\mathbf{u}) = \mathbf{P}\{\mathbf{S}(\mathbf{u}) \geqslant \varphi\} \to \max_{\mathbf{u}\in\mathcal{U}}, \tag{5}$$

$$\mathbf{u}^* = (\mathbf{u}^*_1, \ldots, \mathbf{u}^*_n) \in \operatorname{Arg}\max_{\mathbf{u}\in\mathcal{U}} P^2_{\varphi,\bar{T}}(\mathbf{u}), \tag{6}$$

where $\mathcal{U}$ is the set of admissible control functions. Since the random response time for any task is a discrete random variable with a finite number of values and the correct answer is modeled by the Bernoulli distribution, the number of states (T_k, S_k) at the kth step is finite. However, for convenience of optimization, the control action $\mathbf{u}_k$ at the kth step will be chosen as a function on the wider set of states $\mathbb{T}_k \times \mathbb{S}_k$, where $\mathbb{T}_k$ and $\mathbb{S}_k$ are some finite sets containing all possible values of the states T_k and S_k, respectively. For this reason, $\mathcal{U}$ is supposed to be the set of all functions from $\bigotimes_{k=1}^{n} \mathbb{T}_k \times \mathbb{S}_k$ into $\{0,1\}^n$.

Remark 1. In the class of program strategies, the problem of maximizing the probability of scoring a required sum of points under a time limit was solved in [11]. In the current notation, this problem has the form

$$\tilde{P}^1_{\varphi,\bar{T}}(u) = \mathbf{P}\{S(u) \geqslant \varphi,\ T(u) \leqslant \bar{T}\} \to \max_{u\in\{0,1\}^n}, \tag{7}$$

where $T(u)$ is the random test completion time under the program strategy u. Within problem (7), a test is considered unpassed if the subject fails to meet the time limit even when scoring the required sum of points. There is no requirement to meet the time limit in problem (4), but no points are awarded for performing tasks after the time $\bar{T}$. Meanwhile, the advantage of (7) is the independence of the optimal choice of tasks from their order in the test. In problems (4) and (5), the optimal set of tasks to be performed depends on their order. Note that the optimal value of the objective function in problem (7) is a lower bound for those of the objective functions in problems (4) and (5) for any task order in the test.

Remark 2. Problem (7) can be generalized to the case of positional strategies. As a result, we obtain the problem

$$\tilde{P}^2_{\varphi,\bar{T}}(\mathbf{u}) = \mathbf{P}\{\mathbf{S}(\mathbf{u}) \geqslant \varphi,\ \mathbf{T}(\mathbf{u}) \leqslant \bar{T}\} \to \max_{\mathbf{u}\in\mathcal{U}}, \tag{8}$$

where $\mathbf{T}(\mathbf{u})$ is the random test completion time T_n defined by equations (1) when substituting $u_k = \mathbf{u}_k(T_k, S_k)$ therein. As will be shown below, problem (5) turns out to be completely equivalent to problem (8), in contrast to the pair of problems (4) and (7).

3. POSITIONAL CONTROL DESIGN

To solve the positional control design problem (5), we apply Bellman's dynamic programming method. Let us define the Bellman function for $k = \overline{1,n}$ by the rule

$$B_k(T_{k-1}, S_{k-1}) = \max_{\mathbf{u}_k,\ldots,\mathbf{u}_n} \mathbf{P}\{\mathbf{S}(\mathbf{u}) \geqslant \varphi \mid T_{k-1}, S_{k-1}\}. \tag{9}$$

At the last step, the Bellman function is defined by the equality

$$B_{n+1}(T_n, S_n) = I\{S_n \geqslant \varphi\}. \tag{10}$$

Note the absence of dependence on T_n at the last step: it has been introduced for the convenience of writing the dynamic programming relations. Obviously,

$$\max_{\mathbf{u}\in\mathcal{U}} P^2_{\varphi,\bar{T}}(\mathbf{u}) = B_1(T_0, S_0).$$

We obtain the dynamic programming relations using the formula of total probability:

$$B_k(T_{k-1}, S_{k-1}) = \max_{u_k=\mathbf{u}_k(T_{k-1},S_{k-1})} \max_{\mathbf{u}_{k+1},\dots,\mathbf{u}_n} \sum_{t\in\mathcal{T}_k,\, x\in\{0,1\}} \mathbf{P}\{\mathbf{S}(\mathbf{u})$$

$$\geqslant \varphi \mid S_k = S_{k-1} + b_k x I\{T_{k-1} + tu_k \leqslant \bar{T}\}u_k,\ T_k = T_{k-1} + tu_k\}p_k(t,x)$$

$$= \max_{u_k\in\{0,1\}} \sum_{t\in\mathcal{T}_k,\, x\in\{0,1\}} B_{k+1}(T_{k-1} + tu_k, S_{k-1} + b_k x I\{T_{k-1} + tu_k \leqslant \bar{T}\}u_k)p_k(t,x). \tag{11}$$

The control action $\mathbf{u}^*_k(T_{k-1}, S_{k-1})$ will be determined by maximizing the Bellman function at the kth step.

Since the random variables have discrete distributions, all the functions in the dynamic programming method are measurable, which ensures the correctness of this method.

Thus, the problem can be solved using the following algorithm, which implements dynamic programming.

Algorithm 1 (positional control design).

1. Set $k := n + 1$; for all $S_n \in \mathbb{S}_n$ calculate the value $B_{n+1}(T_n, S_n)$.
2. If $k > 1$, set $k := k - 1$; otherwise, proceed to Step 4.
3. For all $T_{k-1} \in \mathbb{T}_{k-1}$ and $S_{k-1} \in \mathbb{S}_{k-1}$ calculate the value $B_k(T_{k-1}, S_{k-1})$ and determine $\mathbf{u}^*_k(T_{k-1}, S_{k-1})$.
4. Calculate the value $B_1(0,0)$ and determine the control action $\mathbf{u}^*_1(0,0)$ of the first step.

Note that with additionally calculating $B_1(t,s)$ at Step 4 of Algorithm 1, we will maximize the objective functional $P^2_{\varphi-s,\bar{T}-t}(\mathbf{u})$. These calculations can be carried out if the set of reachable states of the dynamic system with the initial conditions $T_0 = t$, $S_0 = s$ is a subset of $\mathbb{T}_k \times \mathbb{S}_k$ at each kth step. Thus, it is possible to solve the problem for several values of φ and $\bar{T}$ at once.

The above solution method allows proving the equivalence of problems (5) and (8).

Proposition 1. *The optimal values of the objective functionals in problems* (5) *and* (8) *are equal. There exist strategies optimal in both problems, namely, those satisfying the condition* $\mathbf{u}^*_k(T_{k-1}, S_{k-1}) = 0$ *for* $S_{k-1} \geqslant \varphi$ *or* $T_{k-1} > \bar{T}$.

Proof. To solve problem (8), we also apply the dynamic programming method. For this purpose, it is necessary to define the Bellman function at the last step by the rule $\tilde{B}_{n+1}(T_n, S_n) = I\{T_n \leqslant \bar{T},\ S_n \geqslant \varphi\}$. The dynamic programming relations for solving problem (8) will be the same as (11), with the functions $\tilde{B}_k$ written instead of B_k.

Let us analyze the resulting dynamic programming relations. Note that for $T_{k-1} > \bar{T}$ (the time limit is exhausted) or $S_{k-1} \geqslant \varphi$ (the required sum of points is scored), when computing $B_k(T_{k-1}, S_{k-1})$ and $\tilde{B}_k(T_{k-1}, S_{k-1})$, we can assign $\mathbf{u}^*_k(T_{k-1}, S_{k-1}) = 0$ (the condition formulated in Proposition 1).

Now let us show the following result by induction: $B_k(T_{k-1}, S_{k-1}) = \tilde{B}_k(T_{k-1}, S_{k-1})$ for $T_{k-1} \leqslant \bar{T}$, with the presence of coincident strategies among those on which the maximum is reached, and if $S_{k-1} < \varphi$, then the maximum is reached on the same strategies. For $k = n$ this assertion is true by the definition of the above Bellman functions. Assuming its truth for $B_{k+1}(T_k, S_k)$, we establish it for $B_k(T_{k-1}, S_{k-1})$. If $S_{k-1} \geqslant \varphi$ and $T_{k-1} \leqslant \bar{T}$, then $B_k(T_{k-1}, S_{k-1}) = \tilde{B}_k(T_{k-1}, S_{k-1}) = 1$ (as noted

above, we can take $\mathbf{u}_k^*(T_{k-1}, S_{k-1}) = 0$). Two cases may arise when calculating $B_k(T_{k-1}, S_{k-1})$ under the conditions $S_{k-1} < \varphi$ and $T_{k-1} \leqslant \bar{T}$: 1) If $T_k > \bar{T}$, then the equality $S_k = S_{k-1}$ holds and the value $B_k(T_k, S_{k-1}) = \tilde{B}_k(T_k, S_{k-1}) = 0$ is considered. 2) If $T_k \leqslant \bar{T}$, then the value $B_k(T_k, S_{k-1}) = \tilde{B}_k(T_k, S_{k-1})$ is considered (equality is valid by the inductive hypothesis). In both cases, we have $B_k(T_{k-1}, S_{k-1}) = \tilde{B}_k(T_{k-1}, S_{k-1})$ and the maximum in the definitions of Bellman functions is achieved on the same strategies. Thus, the proof of Proposition 1 is complete.

Remark 3. Among the optimal strategies in problem (5), there may be strategies satisfying the condition $\mathbf{u}_k^*(T_{k-1}, S_{k-1}) = 1$ for $S_{k-1} \geqslant \varphi$, which are not optimal ones in problem (8). Indeed, when the required sum of points has been scored, performing additional tasks reduces the probability of not exceeding the time limit, decreasing the value of the objective functional in problem (8) compared to the value of that in problem (5).

4. PROGRAM CONTROL DESIGN

First, we describe a method for calculating the objective functional in (4) for a fixed $u \in \{0,1\}^n$. Let us define the following functions:

$$B_{n+1}^u(T_n, S_n) = I\{S_n \geqslant \varphi\},$$
$$B_k^u(T_{k-1}, S_{k-1}) = \sum_{t \in \mathcal{T}_k,\, x \in \{0,1\}} B_{k+1}^u(T_{k-1} + tu_k, S_{k-1} + b_k x I\{T_{k-1} + tu_k \leqslant \bar{T}\}u_k) p_k(t,x), \tag{12}$$

where $k = \overline{1,n}$. The value $B_1^u(0,0)$ yielded by these formulas is equal to that of the probability functional $P_{\varphi,\bar{T}}^1(u)$. This fact follows from the definition of the Bellman function (9) since the relations (12) are similar to the Bellman equations but without maximization and with the control strategy u_k at the kth step.

Owing to the use of dynamic relations, the above procedure is much more efficient than the direct calculation of probability by enumerating all possible realizations of the random variables X_k and τ_k.

The optimal program control can be found by enumerating all possible control actions $u \in \{0,1\}^n$. In this case, it is possible to eliminate the control actions for which $\sum\limits_{k=1}^{n} b_k u_k < \varphi$: such actions do not ensure test passing for any realizations of the random variables with a nonzero probability. Also, we eliminate the control actions u with $u_n = 0$ because they are no worse than the corresponding ones with $u_n = 1$. Indeed, trying to perform the last task cannot reduce the probability of test passing.

We propose a procedure for significantly reducing the number of program control actions enumerated. Assume that the values of the Bellman function $B_k(S_{k-1}, T_{k-1})$ are known. (They can be calculated using the algorithm from the previous section.) Let us introduce the notation

$$C_{k,k}^{u_1,\dots,u_k}(T_{k-1}, S_{k-1}) = \sum_{t \in \mathcal{T}_k,\, x \in \{0,1\}} B_{k+1}(T_{k-1} + tu_k, S_{k-1} + b_k x I\{T_{k-1} + tu_k \leqslant \bar{T}\}u_k) p_k(t,x),$$

$$C_{l,k}^{u_1,\dots,u_k}(T_{l-1}, S_{l-1}) = \sum_{t \in \mathcal{T}_l,\, x \in \{0,1\}} C_{l+1,k}^{u_1,\dots,u_k}(T_{l-1} + tu_l, S_{l-1} + b_l x I\{T_{l-1} + tu_l \leqslant \bar{T}\}u_l) p_l(t,x), \tag{13}$$
$$l = k-1, k-2, \dots, 1, \quad C_k(u_1, \dots, u_k) = C_{1,k}^{u_1,\dots,u_k}(T_0, S_0).$$

The value $C_{l,k}^{u_1,\dots,u_k}(T_{k-1}, S_{k-1})$ describes the minimum loss starting from the kth step under the control actions $u_1, \dots, u_k$ applied at the first k steps. These values are calculated from the dynamic programming relations with fixed control actions at the first k steps. The values $C_k(u_1, \dots, u_k)$ can be used as upper bounds for the objective function.

Proposition 2. *The following relations hold for the values given by* (13)*:*

1) $C_k(u_1,\dots,u_k) \geqslant B_1^u(0,0) = P^1_{\varphi,\bar{T}}(u) = C_n(u_1,\dots,u_n)$.

2) $C_k(u_1,\dots,u_k) \geqslant C_{k+1}(u_1,\dots,u_{k+1})$ *for all* $k = \overline{1,n-1}$.

Proof. Equations (13) are the dynamic Bellman relations with the control actions $u_1,\dots,u_k$ at the first k steps. Therefore, the value $C_k(u_1,\dots,u_k)$ coincides with that of the objective functional in problem (5) with positional control, where the control vector $\mathbf{u}$ has fixed components equal to $u_1,\dots,u_k$ at the first k steps and the optimally chosen components at the subsequent steps (with respect to the state achieved in k steps). Hence, for any program control u with fixed $u_1,\dots,u_k$, we have the inequality $C_k(u_1,\dots,u_k) \geqslant B_1^u(0,0)$. In addition, the control vector with fixed components $u_1,\dots,u_{k+1}$ gives a smaller value of the performance functional in (5) than that with fixed components $u_1,\dots,u_k$. Therefore, we obtain $C_k(u_1,\dots,u_k) \geqslant C_{k+1}(u_1,\dots,u_{k+1})$ for all $k = \overline{1,n-1}$. The equality $B_1^u(0,0) = P^1_{\varphi,\bar{T}}(u)$ has been established above. The proof of Proposition 2 is complete.

Thus, if $\psi < \max\limits_{u\in\{0,1\}^n} P^1_{\varphi,\bar{T}}(u)$ (the known lower bound for the optimal value of the objective function) and $C_k(u_1,\dots,u_k) \leqslant \psi$ for some k, then by Proposition 2 we have $P^1_{\varphi,\bar{T}}(u) < \max\limits_{u\in\{0,1\}^n} P^1_{\varphi,\bar{T}}(u)$ and, consequently, any strategy u with the corresponding first k components is not optimal. Therefore, Algorithm 2 based on the branch-and-bound method (see below) can be proposed to find the program strategy. This algorithm traverses a binary tree with the following properties: the root corresponds to the variable u_1; the outgoing edges of the root, to values 1 and 0 of this variable; the adjacent vertices of the root, to the variable u_2; the outgoing edges of these vertices, to the values of the variable u_2, and so on. The vertices corresponding to the variable u_n have two outgoing edges corresponding to the values of the variable u_n and connecting them to the leaves. A depth-first search is applied to traverse the tree; the search can be terminated if its further execution does not yield an optimal strategy. To describe the termination time, we introduce a function $L(u_1,\dots,u_k)$: it takes value 0 if no further depth-first search is performed and value 1 otherwise. The variables ψ and u^* modified when executing the function $L(u_1,\dots,u_k)$ are global.

Algorithm 2 (program control design).

0. Define the function $L(u_1,\dots,u_k)$ by the following rule:

0a. Set $\tilde{u} = (u_1,\dots,u_k,1,\dots,1)$. If $\sum\limits_{k=1}^{n} b_k\tilde{u}_k \geqslant \varphi$ or ($k = n$ and $u_n = 1$), calculate $C_k(u_1,\dots,u_k)$; otherwise, return $L(u_1,\dots,u_k) = 0$.

0b. If $C_k(u_1,\dots,u_k) \leqslant \psi$, return $L(u_1,\dots,u_k) = 0$.

If $k = n$ and $C_n(u_1,\dots,u_n) > \psi$, assign $\psi := C_n(u_1,\dots,u_n)$ and $u^* := (u_1,\dots,u_n)$ and return $L(u_1,\dots,u_k) = 0$.

If $k = n$ and $C_n(u_1,\dots,u_n) \leqslant \psi$, return $L(u_1,\dots,u_k) = 0$;

otherwise return $L(u_1,\dots,u_k) = 1$.

1. Set $k := 1$, $\psi := 0$, $u^* := (0,\dots,0)$, and $u_1 := 1$.

2. Calculate $L(u_1,\dots,u_k)$.

3. If $L(u_1,\dots,u_k) = 1$, set $k := k+1$ and $u_k := 1$ and get back to Step 2.

If $L(u_1,\dots,u_k) = 0$ and $u_k = 1$, set $u_k := 0$ and get back to Step 2.

If $L(u_1,\dots,u_k) = 0$ and $u_l = 0$ for all $l = \overline{1,k}$, terminate execution of the algorithm;

otherwise, assign $k := \max\{l \mid u_l = 1, l < k\}$ and $u_k := 0$ and get back to Step 2.

Algorithm 2 yields the optimal control strategy u^* and the corresponding optimal value of the objective function $P^1_{\varphi,\bar{T}}(u^*) = \psi$.

The depth-first search of the algorithm starts from the variable values $u_k = 1$: in this case, the first strategy to be considered is "perform all tasks," which usually corresponds to a large value of the probability of test passing.

Remark 4. Algorithm 2 can be applied to solve problem (7) as well. For this purpose, we need to set $B_{n+1}(T_n, S_n) = I\{T_n \leqslant \bar{T},\ S_n \geqslant \varphi\}$ when determining the values $C_k(u_1, \ldots, u_k)$. In addition, at Step 0a, it is necessary to eliminate the check ($k = n$ and $u_n = 1$) since the optimal strategy may have $u_n = 0$.

5. NUMERICAL RESULTS

The two problems posed were successfully solved for the data from [11]. In the test under consideration, there are 10 tasks with the points $b_1 = \ldots = b_5 = 1$, $b_6 = b_8 = 2$, $b_7 = b_{10} = 3$, and $b_9 = 4$ scored for performing them. A subject needs to score $\varphi = 11$ points. The probabilities of correct answers to each task are known. Each task is associated with three possible realizations of the time spent on a correct answer and its three realizations for an incorrect answer; their conditional probabilities are known and were given in [11].

Table 1 presents the solutions of problems (4) and (5) depending on the parameter $\bar{T}$. Here, the values of $\bar{T}$ are different fractions of $T^m = 3830$, the maximum possible time for performing all tasks. The run times of Algorithms 2 and 1 to find program (prog.) and positional (pos.) strategies, respectively, are indicated. The last column shows the time for solving problem (4) when enumerating all possible program strategies, except for the obviously suboptimal ones, using formulas (12).

Similarly, Table 2 provides the solutions of problems (7) and (8). The run times of Algorithms 2 and 1 are indicated as well. All calculations were carried out on an Acer Aspire A315-54K laptop (Intel Core i5-6300U 2.4 GHz CPU, 8Gb RAM).

Table 1. The solutions of problems (4) and (5) depending on the parameter $\bar{T}$ for $\varphi = 11$

$\bar{T}$	Optimal program strategy u^*	$P^1_{\varphi,\bar{T}}(u^*)$	$\max\limits_{\mathbf{u}\in\mathcal{U}} P^2_{\varphi,\bar{T}}(\mathbf{u})$	Run time, prog./pos. (s)	Run time, enumeration (s)
$0.4T^m$	$(0,0,0,0,1,1,1,1,0,1)$	0.1447	0.1689	0.83 / 0.08	1.69
$0.5T^m$	$(1,1,1,0,1,1,1,1,0,1)$	0.3583	0.4220	0.45 / 0.07	1.78
$0.6T^m$	$(1,0,1,1,1,1,1,1,0,1)$	0.5244	0.6130	0.53 / 0.07	1.93
$0.7T^m$	$(1,1,1,0,1,1,1,1,1,1)$	0.6906	0.7242	0.28 / 0.07	2.17
$0.8T^m$	$(1,1,1,1,1,1,1,1,1,1)$	0.7815	0.7876	0.11 / 0.08	2.34
$0.9T^m$	$(1,1,1,1,1,1,1,1,1,1)$	0.8006	0.8007	0.15 / 0.07	2.64

Table 2. The solutions of problems (7) and (8) depending on the parameter $\bar{T}$ for $\varphi = 11$

$\bar{T}$	Optimal program strategy u^*	$\tilde{P}^1_{\varphi,\bar{T}}(u^*)$	$\max\limits_{\mathbf{u}\in\mathcal{U}} \tilde{P}^2_{\varphi,\bar{T}}(\mathbf{u})$	Run time, prog./pos. (s)	Run time, enumeration (s)
$0.4T^m$	$(0,0,0,0,1,1,1,1,0,1)$	0.1447	0.1689	0.99 / 0.08	3.47
$0.5T^m$	$(1,0,1,0,1,1,1,1,0,1)$	0.2460	0.4220	3.02 / 0.07	3.45
$0.6T^m$	$(1,1,1,0,1,1,1,1,0,1)$	0.4963	0.6130	1.49 / 0.07	3.45
$0.7T^m$	$(1,1,1,1,1,1,1,1,0,1)$	0.6059	0.7242	0.64 / 0.07	3.46
$0.8T^m$	$(1,1,1,0,1,1,1,1,1,1)$	0.7216	0.7876	0.63 / 0.07	3.41
$0.9T^m$	$(1,1,1,1,1,1,1,1,1,1)$	0.7965	0.8007	0.14 / 0.07	3.42

According to these tables, positional strategies have an advantage over program ones, not only in terms of the optimal value of the objective functional but also in terms of the speed of computation. Comparing problem (4) with the one from [11], we note that, as a rule, the solution of (4) implies a larger number of subproblems and a larger value of the objective functional. This is due to the absence of a time limit in problem (4).

As mentioned earlier, the solution of (4) and (5) depends on the order of test tasks. Therefore, additional calculations of the strategies for passing the same test, but with the reverse order of tasks, were performed: the first task of the test was made the tenth, the second the ninth, and so on. The points awarded for the tasks and the probabilities of performing the tasks remained the same. The results of solving this problem are described in Table 3. For convenience of comparing the tables, the components of the optimal program strategies are given in reverse order. The run times of the algorithms are omitted here because they insignificantly differ from the ones in Table 1. Direct comparison of the solutions shows that the values of the objective functions changed slightly with the reverse order of the tasks, and the program strategy differs for the time limits $0.4T^m$ and $0.6T^m$.

Table 3. The solutions of problems (4) and (5) depending on the parameter $\bar{T}$ for $\varphi = 11$: the reverse order of tasks

$\bar{T}$	Optimal program strategy $(u_{10}^*, \ldots, u_1^*)$	$P^1_{\varphi,\bar{T}}(u^*)$	$\max\limits_{\mathbf{u}\in\mathcal{U}} P^2_{\varphi,\bar{T}}(\mathbf{u})$
$0.4T^m$	$(1,0,1,0,1,1,1,1,0,1)$	0.1639	0.1787
$0.5T^m$	$(1,1,1,0,1,1,1,1,0,1)$	0.3526	0.3834
$0.6T^m$	$(1,1,1,1,1,1,1,1,0,1)$	0.5340	0.5718
$0.7T^m$	$(1,1,1,0,1,1,1,1,1,1)$	0.6781	0.7003
$0.8T^m$	$(1,1,1,1,1,1,1,1,1,1)$	0.7824	0.7847
$0.9T^m$	$(1,1,1,1,1,1,1,1,1,1)$	0.8007	0.8007

6. CONCLUSIONS

In this paper, we have found program and positional strategies for passing a time-limited test in which a testee (subject) knows the current sum of points scored after performing each task. Algorithms for solving these problems have been proposed and their effectiveness has been demonstrated. The results have been compared with those of the problem statement considered by one of the authors previously. According to the analysis, with the current sum of points being known to a subject, there is a test passing strategy under which the subject will exceed the required level with a higher probability.

Future research may deal with various modifications of this model. In particular, the dependence of the results of performing test tasks on the current sum of points scored can be considered. Such a dependence arises due to the psychological peculiarities of some subjects. The possibility of modifying the Bellman equations (11) in this case needs to be investigated. Also, it seems interesting to describe the response time of subjects (their answers to test tasks) by continuous random variables and to study the solution accuracy in the discretization case.

REFERENCES

1. Rasch, G., *Probabilistic Models for Some Intelligence and Attainment Tests*, Chicago: The University of Chicago Press, 1980.

2. Xiao, J. and Bulut, O., Item Selection with Collaborative Filtering in On-the-fly Multistage Adaptive Testing, *Appl. Psychol. Meas.*, 2022, vol. 46, no. 8, pp. 690–704.

3. Naumov, A.V., Dzhumurat, A.S., and Inozemtsev, A.O., Distance Learning System for Mathematical Disciplines CLASS.NET, *Herald of Computer and Information Technologies*, 2014, no. 10, pp. 36–44.

4. CLASS.NET. The Distance Learning System of Moscow Aviation Institute.
URL: https://distance.kaf804.ru// (Accessed October 12, 2024.)

5. Kuravsky, L.S., Margolis, A.A., Marmalyuk, P.A., et al., A Probabilistic Model of Adaptive Training, *Appl. Math. Sci. (Ruse)*, 2016, vol. 10, no. 48, pp. 2369–2380.

6. Bosov, A.V., Martyushova, Ya.G., Naumov, A.V., and Sapunova, A.P., Bayesian Approach to the Construction of an Individual User Trajectory in the System of Distance Learning, *Informatics and Applications*, 2020, vol. 14, no. 3, pp. 86–93.

7. Bosov, A.V., Application of Self-Organizing Neural Networks to the Process of Forming an Individual Learning Path, *Informatics and Applications*, 2022, vol. 16, no. 3, pp. 7–15.

8. van der Linden, W.J., Scrams, D.J., and Schnipke, D.L., Using Response-Time Constraints to Control for Differential Speededness in Computerized Adaptive Testing, *Appl. Psych. Meast.*, 1999, vol. 23, no. 3, pp. 195–210.

9. Bosov, A.V., Mkhitaryan, G.A., Naumov, A.V., and Sapunova, A.P., Using the Model of Gamma Distribution in the Problem of Forming a Time-Limited Test in a Distance Learning System, *Informatics and Applications*, 2019, vol. 13, no. 4, pp. 11–17.

10. Naumov, A.V., Mkhitaryan, G.A., and Cherygova, E.E., Stochastic Statement of the Problem of Generating Tests with Defined Complexity with the Minimization of Quantile of Test Passing Time, *Herald of Computer and Information Technologies*, 2019, no. 2, pp. 37–46.

11. Naumov, A.V., Stepanov, A.E., and Ustinov, A.E., On the Problem of Maximizing the Probability of Successful Passing of a Time-Limited Test, *Autom. Remote Control*, 2024, vol. 85, no. 1, pp. 64–72.

12. Martyushova, Ya.G., Naumov, A.V., and Stepanov, A.E., Optimization of the Strategy of Passing the Time-Limited Test According to the Quantile Criterion, *Informatics and Applications*, 2024, vol. 18, no. 4, pp. 44–51.

This paper was recommended for publication by A.I. Kibzun, a member of the Editorial Board