Game-Theoretic Question Selection for Tests

Paper #1789

Abstract

Conventionally, the questions on a test are assumed to be kept secret from test takers until the test. However, for tests that are taken on a large scale, particularly asynchronously, this is very hard to achieve. For example, example TOEFL iBT and driver's license test questions are easily found online. This also appears likely to become an issue for Massive Open Online Courses (MOOCs).

In this paper, we take the loss of confidentiality as a fact. Even so, not all hope is lost as the test taker can memorize only a limited set of questions' answers, and the tester can randomize which questions appear on the test. We model this as a Stackelberg game, where the tester commits to a mixed strategy and the follower responds. We provide an exponential-size linear program formulation, prove several NP-hardness results, and give efficient algorithms for special cases.

1 Introduction

Massive Open Online Courses (MOOCs) have emerged quite spectacularly and there is much speculation about how their role in society will develop (for a recent discussion, see [Cooper and Sahami, 2013]). In particular, the issue of how the students' accomplishments can be efficiently validated and certified is often discussed. One approach that is now used is to allow students to take a proctored exam in a testing center. A potential vulnerability of this approach is that the questions used on a test may be leaked, online or otherwise. Certainly, this is already the case for other tests that are taken asynchronously by many people, such as the TOEFL iBT and driver's license tests. A similar issue occurs for interview questions, particularly those with a "puzzle" flavor. A natural approach to addressing this is to generate enough questions that a test taker would be unable to memorize them all, and then to select from these questions randomly on each instantiation of the test.

However, sampling test questions uniformly at random may be suboptimal for several reasons. Some questions may be more effective than others at identifying test takers that do not know the material; should these be used more often? Also, it may be not be optimal to consider questions in isolation; it is likely better to use two questions that test very different parts of the material than ones that are quite similar. At the same time, choosing the test questions deterministically is obviously fatally flawed, as this makes it easy to memorize the questions. Instead, a more intelligent form of randomization seems desired. Game theory provides a natural framework to determine such a randomized strategy. We can model the test as a game between the tester, who chooses questions for the test, and the test taker, who chooses for which questions (on material that he has not mastered) to memorize the answers. This is the approach that we take in this paper.

We model the test game as a Bayesian game, in which there is uncertainty about the type of the test taker. We assume here the availability of detailed statistical data on the different types of test takers, specifically concerning their mastery of various material that is potentially on the test, and their ability to memorize answers. Our primary focus is on the computational problem of, given this data, determining an optimal mixed strategy for the tester to determine what is on the test.

While we have used exams and similar tests as motivating examples in the above, there are other potential applications in which some entity is being examined in a limited way. For example, a team of (say, nuclear) inspectors may be able to visit only a limited number of suspect sites (corresponding to being able to place only a limited number of questions on an exam). The entity being investigated may be able to cover up illicit activity at a limited number of sites (corresponding to memorizing the answers to selected questions), and many of the sites may not have any illicit activity in the first place (corresponding to questions that the test taker can handle without having to memorize the answer).

2 Definitions and Notation

A *test game* $G = (Q, \Theta, p, t, u_1, u_2)$ is a 2-player Bayesian game between the tester (player 1) and the test taker (player 2). The tester has a set of potential test questions $Q = \{q_1, q_2, \ldots, q_n\}$.[1] The test taker has a type $\theta \in \Theta$ ($|\Theta| = L$) that characterizes which questions are hard for the test taker (i.e., which questions he would not be able to answer without

[1] In common parlance, "problem" may be a better word, but this runs the risk of confusion with the computational problems studied in the paper.

memorizing them), and how many questions the test taker can memorize. That is, $\theta = (H_\theta, m_\theta)$, where $H_\theta \subseteq Q$ is the set of questions that are hard for θ, and $m_\theta \in \mathbb{N}$ is the number of questions for which θ would be able to memorize the answer, so that a test taker of type θ has $\binom{|H_\theta|}{m_\theta}$ pure courses of action. (W.l.o.g., $m_\theta \leq |H_\theta|$.) $p : \Theta \to [0, 1]$ is the probability distribution over types. The tester can put t questions on the test, and thus has $\binom{n}{t}$ pure strategies. We use T to denote such a pure strategy, and M_θ to denote the subset of questions that θ memorizes. $u_1(\theta, T, M_\theta)$ denotes the tester's utility for type θ, and $u_2(\theta, T, M_\theta)$ denotes the utility of a type θ test taker.

In general, the utility functions could be very rich, depending in a complicated way on the true type and the number (or even precise set) of questions answered correctly. To avoid getting too deeply into the modeling aspects of this, we study only a case that should be a special case of most reasonable models. This strengthens the hardness results that we obtain later in the paper (since these hardness results automatically also apply to the richer models of which our model is a special case), though it weakens our positive results. Specifically, we assume that the only two outcomes of the test are to pass or fail the test taker. Moreover, we assume that the tester does not want to pass any test taker that cannot answer all the questions (without memorizing any). Thus, the tester will pass exactly the agents that answer all the questions on the test correctly. Agents that can answer all the questions without memorizing any have no strategic decisions to make and will always pass; therefore, we do not model them explicitly as a type in the game. (They can be thought of as adding a constant to the tester's utility function that is sufficiently large that the tester indeed wants to pass agents that answer all questions correctly.)

Thus, for the explicitly modeled types θ (which the tester wants to fail), we assume that $u_1(\theta, T, M_\theta) = 0$ if $T \cap (H_\theta \setminus M_\theta) \neq \emptyset$ (i.e., a question is tested that the agent cannot answer, so the the agent fails) and $u_1(\theta, T, M_\theta) = -v_\theta$ otherwise (the cost of passing an agent of type θ). For the test takers, $u_2(\theta, T, M_\theta) = 0$ if $T \cap (H_\theta \setminus M_\theta) \neq \emptyset$, and $u_2(\theta, T, M_\theta) = \omega_\theta$ otherwise. We require $v_\theta, \omega_\theta > 0$.

We believe that a Stackelberg model is most natural for our setting, where the tester first commits to a mixed strategy, and the test takers observe this mixed strategy and best-respond. This is because the test takers can observe and learn the tester's strategy over time. Arguably, however, if the test takers are not able to do such learning, a Nash equilibrium model (i.e., no Stackelberg leadership) also makes sense. In the next section, we show that in fact, both of these models are equivalent to the same zero-sum game.

3 Equivalence to a Zero-Sum Game

For any test game G as defined above, we can modify it into a zero-sum game G' by substituting the following utility function for the test taker: $u_2'(\theta, T, M_\theta) = 0$ if $T \cap (H_\theta \setminus M_\theta) \neq \emptyset$, and $u_2'(\theta, T, M_\theta) = v_\theta$ otherwise.

Proposition 1. *The set of Stackelberg mixed strategies for the tester in G is equal to the set of maximin strategies for the tester in G'. These sets are also equal to the set of Nash equilibrium strategies for the tester in G.*

Proof. First, we argue that the sets of Stackelberg mixed strategies for the tester in G and G' are the same. This follows from the fact that every test taker type simply acts to maximize his probability of passing, whether the utility function is u_2 or u_2'. Second, it is well known that in a 2-player zero-sum game, the set of Stackelberg mixed strategies is equal to the set of maximin strategies for the leader.

Similarly, the sets of Nash equilibrium strategies for the tester in G and G' are the same. This again follows from the fact that every test taker type simply acts to maximize his probability of passing, whether the utility function is u_2 or u_2'. Finally, it is well known that in a 2-player zero-sum game, the set of Nash equilibrium strategies is equal to the set of maximin strategies for the leader (by the minimax theorem [von Neumann, 1928]). □

We note that the equivalence to this zero-sum game is not complete, in the sense that it would *not* hold if the *test taker* is a Stackelberg leader who can commit to a strategy (before learning his type). An example is provided in Appendix A. However, since this reversed Stackelberg model is not of primary interest to us, we proceed with the zero-sum formulation from here on, focusing on G' knowing that its solution will give us a solution to our original problem.

4 General Linear Program (LP) Formulation

Computing an optimal Stackelberg mixed strategy in a general 2-player Bayesian game is NP-hard [Conitzer and Sandholm, 2006] and inapproximable [Letchford *et al.*, 2009]; a general mixed-integer linear program formulation has previously been proposed [Paruchuri *et al.*, 2008]. However, thanks to Proposition 1, we only need to solve for a maximin strategy of a zero-sum game. The following linear programming approach is standard:

$$\begin{aligned} \max \ & \sum_l p(\theta) V_\theta \qquad (1) \\ s.t. \ & (\forall \theta, \forall M_\theta \subseteq H_\theta : |M_\theta| = m_\theta) \\ & \sum_{T \subseteq Q : |T| = t} \mathrm{u}_1(\theta, T, M_\theta) x_T \geq V_\theta; \\ & \sum_{T \subseteq Q : |T| = t} x_T = 1; \\ & (\forall T \subseteq Q : |T| = t) \ x_T \geq 0; \end{aligned}$$

Here, V_θ is the utility that the tester receives from type θ, and x_T is the probability of testing exactly the questions in T.

The dual of the above LP is:

$$\begin{aligned} \min \ & U \qquad (2) \\ s.t. \ & (\forall T \subseteq Q : |T| = t) \\ & U \geq \sum_{\theta, M_\theta \subseteq H_\theta : |M_\theta| = m_\theta} u_1(\theta, T, M_\theta) y_{\theta, M_\theta}; \\ & (\forall \theta) \sum_{M_\theta \subseteq H_\theta : |M_\theta| = m_\theta} y_{\theta, M_\theta} = p(\theta); \\ & (\forall \theta, M_\theta \subseteq H_\theta : |M_\theta| = m_\theta) \ y_{\theta, M_\theta} \geq 0; \end{aligned}$$

The variables of the dual LP y_{θ,M_θ} give a strategy for the test taker that maps types to probabilities of memorizing subsets of questions: the probability of memorizing M_θ conditional on having type θ is $y_{\theta,M_\theta}/p(\theta)$. $U = \max_{T\subseteq Q:|T|=t} U_T$ is the best-response utility for the tester, where $U_T = \sum_{\theta,M_\theta\subseteq H_\theta:|M_\theta|=m_\theta} u_1(\theta,T,M_\theta)y_{\theta,M_\theta}$ is the utility of testing T. By linear programming duality, the two LPs have the same optimal solution value (corresponding to the minimax theorem); strategies corresponding to optimal solutions of these LPs constitute an equilibrium of the game.

Linear programs can be solved in time polynomial in their size [Khachiyan, 1979]. However, while the size of the LPs above is polynomial in the size of the game matrix, it is nevertheless exponential in the size of the natural representation of our test games. The tester's pure strategy space is exponential in t (the number of questions to test) and space of pure courses of action for a test taker of type θ is exponential in m_θ (the number of questions whose answers he can memorize). When t and $\max_\theta m_\theta$ are constant, the LPs indeed give us a polynomial-time algorithm. But, can the test game be solved in polynomial time when either the number of questions to test, the number of answers to memorize, or both are not constant?

5 Constant Memory Size

In this section, we study the case where memory size ($\max_\theta m_\theta$) is constant, but test size t is not. We first define a decision variant of our problem:

Definition 1 (The OPTIMAL TEST STRATEGY problem)**.** *Given a test game G and a value u, the* OPTIMAL TEST STRATEGY *problem is to decide whether the tester has a strategy that gives her a utility of at least u (when the test taker best-responds).*

Proposition 2. *When memory size ($\max_\theta m_\theta$) is constant,* OPTIMAL TEST STRATEGY *is in NP.*

Proof. When $\max_\theta m_\theta$ is constant, the number of constraints (not counting the nonnegativity constraints on the variables) in LP (1) is polynomial. Any LP has an optimal solution with a number of nonzero variables that is at most the number of constraints (not counting the nonnegativity constraints)—this follows, for example, from the simplex algorithm. Hence, a subset of the variables of this LP of this size can serve as a certificate: we can solve the LP restricted to these variables in polynomial time, and check whether the optimal solution is at least u. □

We now prove that the problem is in fact NP-hard, even when the test taker cannot memorize any problems!

Theorem 1. *Even if the test taker cannot memorize any questions ($m_\theta = 0$) and $|H_\theta| = 2$ for all $\theta \in \Theta$,* OPTIMAL TEST STRATEGY *is NP-hard.*

Proof. We reduce from the VERTEX COVER problem, in which we are given a graph (V, E) and a number k, and are asked whether there exists a subset of k vertices such that every edge is incident on at least one of these vertices. For any instance of this problem, we construct an OPTIMAL TEST STRATEGY instance as follows. Let $Q = V$. For each edge $e = \{i, j\} \in E$, add one type of test taker θ_e whose hard question set $H_{\theta_e} = e$. Let $p(\theta_e) = 1/|E|, v_{\theta_e} = 1$ and $m_{\theta_e} = 0$ for all $e \in E$. Finally, let $t = k$ and $u = 0$.

If there exists a vertex cover, consider the tester strategy of testing exactly these questions. Then, every type will fail at least one question and hence the test, resulting in a utility of 0 for the tester.

Conversely, if there exists a tester strategy that gives the tester a utility of 0, then every type passes the test with probability 0 under this tester strategy. Thus, consider any $T \subseteq Q$ with $|T| = k$ that gets positive probability; every type must fail this test. This means that T includes at least one endpoint of every edge, i.e., it is a vertex cover. □

6 Constant Test Size

In this section, we study the case where test size is constant, but memory size is not.

Proposition 3. *When the test size (t) is constant,* OPTIMAL TEST STRATEGY *is in coNP.*

Proof. When t is constant, the number of constraints (not counting the nonnegativity constraints on the variables) in LP (2) is polynomial. As in the proof of Proposition 2, this implies that a subset of the variables of this LP of the requisite size can serve as a certificate that u cannot be achieved: we can solve the LP restricted to these variables in polynomial time, and check whether the optimal solution is strictly less than u. If so, then by weak duality, it is impossible for the tester to obtain u or more (and moreover, by strong duality, if it is not possible for the tester to obtain u or more, such a certificate must exist). □

Theorem 2. *Even if the test size t is 2 and there are only two types that can memorize any answers,* OPTIMAL TEST STRATEGY *is coNP-hard.*

Sketch of Proof. In the full version of this paper, we reduce from the INDEPENDENT SET problem, in which we are given a graph (V, E) and a number k, and are asked whether there exists a subset of k vertices such that no two of these vertices have an edge between them. For any instance of this problem, we construct an OPTIMAL TEST STRATEGY instance that has a "no" answer if and only if the INDEPENDENT SET instance has a "yes" answer. The vertices correspond to questions. We create several types that cannot memorize anything; these ensure that the tester is best off testing exactly the pairs of questions that are *not* edges. Then, we create two more types for which all questions are hard and that can memorize k answers and 2 answers, respectively. The tester will be worst off if the former type memorizes an independent set (thereby covering $\binom{k}{2}$ non-edge pairs of questions) and the latter spreads his mass equally across all the other non-edge pairs. □

7 Tests of Size One

Theorem 1 shows that when we do not bound the test size, our problem is hard even if test takers cannot memorize anything. But, if we do not bound the memory size, Theorem 2

only shows that our problem is hard if the tester can test two questions simultaneously. This still leaves open whether an efficient algorithm exists when the test size is restricted to 1. In this section, we show that this is in fact the case. Our interest in this special case derives not only from it being left open by our hardness results, but also from potentially important applications. For one, an interviewer often only has time to ask an applicant a single question. Moving away from interpreting questions literally, a (for example) nuclear inspections team may be able only to inspect a single site (within a particular time frame).

In this special case, LP (1) has polynomially many variables, but an exponential number of constraints. A quick proof of the polynomial-time solvability of this case is given by establishing that there is a polynomial-time separation oracle for finding a violated constraint. This separation oracle is straightforward: simply select, for every type θ, the m_θ problems in H_θ that get the highest probability x_q in the current solution (breaking ties arbitrarily), and see whether the corresponding constraint is violated. However, in this section, we develop more direct approaches that do not require generating violated constraints and that give more insight into the structure of the solution.

7.1 Linear Program

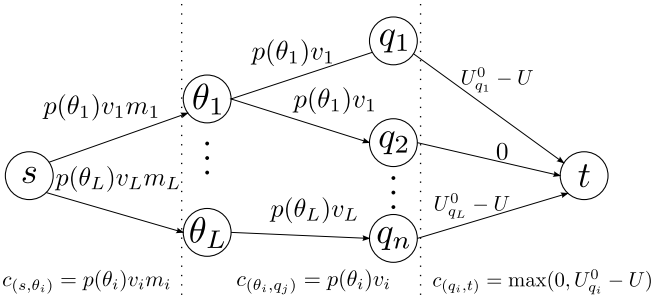
We first present a linear program that can be thought of as a polynomial-size version of LP (2). Instead of variables y_{θ,M_θ}, this linear program has a variable $z_{\theta,q}$ for every individual $q \in H_\theta$. These variables correspond to the *marginal* probability that θ memorizes q (in this case, conditional on θ being the type). In the case where the tester tests one question, these marginal probabilities are all that matter. (This is not true when the tester tests more than one question, because in that case, for example, if the tester always tests two questions together, a test taker may be best off memorizing the answers either to both questions or to neither question.) Let U_q^0 be the utility that the tester obtains when she tests q and nobody memorizes q, i.e., $U_a^0 = \sum_{\theta:q\notin H_\theta} p(\theta)(-v_\theta)$.

$$\begin{aligned} &\min\ U \qquad (3)\\ &s.t.\ (\forall q \in Q)\ U \geq U_q^0 - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}\\ &\qquad (\forall \theta \in \Theta)\ \sum_{q\in H_\theta} z_{\theta,q} \leq m_\theta\\ &\qquad (\forall \theta \in \Theta, q \in H_\theta)\ 0 \leq z_{\theta,q} \leq 1 \end{aligned}$$

Solving LP (3) gives us an equilibrium test taker strategy.[2]

[2]To be precise, we have to find a test taker strategy that has these marginal probabilities. This is quite straightforward in this context: for example, one can, for each question in sequence, flip a coin with the appropriate probability to determine whether to memorize it, and subsequently renormalize the remaining questions' probabilities to ensure that the (expected) number of memorized questions stays the same. This strategy, however, has an exponential-size support and so the support cannot be listed explicitly. Alternatively, we can find the mixed strategy explicitly using the Dulmage-Halperin algorithm [Dulmage and Halperin, 1955; Chang *et al.*, 2001] for finding the Birkhoff-von Neumann decomposition [Birkhoff, 1946]; similar techniques are used in the context of security games [Korzhyk *et al.*, 2010].

Figure 1: The network that is used to solve LP (3).



An equilibrium tester strategy can be read off from the corresponding solution to the dual of this linear program.

7.2 A Network Flow Approach

We now show that LP (3) can also be solved using a network flow approach. Specifically, given a value U for the objective, we can compute a feasible solution that attains this objective value (if it exists), as follows. We construct a network consisting of a directed graph (V, E) and capacities c_e on the edges, as follows (see also Figure 1).

Definition 2 (Test network)**.** *Given an instance of the test game and a value U, construct a network as follows. Let $V = \{s\} \cup \Theta \cup Q \cup \{t\}$ and $E = (\{s\} \times \Theta) \cup (\bigcup_{\theta\in\Theta}\{\theta\} \times H_\theta) \cup (Q \times \{t\})$. For each edge (s, θ), let its capacity be $c_{(s,\theta)} = p(\theta)v_\theta m_\theta$. For each edge (θ, q) (with $q \in H_\theta$), let its capacity be $c_{(\theta,q)} = p(\theta)v_\theta$. Finally, for each edge (q, t), let its capacity be $c_{(q,t)} = \max(0, U_q^0 - U)$.*

Proposition 4. *The test network has a feasible flow that saturates all the edges in $Q \times \{t\}$ if and only if LP (3) has a feasible solution with value U.*

Proof. If the test network has a feasible flow f that saturates all the edges in $Q \times \{t\}$, then consider the solution to LP (3) where $z_{\theta,q} = f_{(\theta,q)}/c_{(\theta,q)}$, so that clearly $0 \leq z_{\theta,q} \leq 1$. Because the $Q \times \{t\}$ edges are saturated, we have that for each $q \in Q$, $\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = \max(0, U_q^0 - U) \geq U_q^0 - U$, which implies $U \geq U_q^0 - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}$. For each $\theta \in \Theta$, because of the capacity constraint on edge (s, θ), we have $\sum_{q\in H_\theta} z_{\theta,q} = (1/(p(\theta)v_\theta))\sum_{q\in H_\theta} f_{(\theta,q)} \leq (1/(p(\theta)v_\theta))f_{(s,\theta)} \leq (1/(p(\theta)v_\theta))p(\theta)v_\theta m_\theta = m_\theta$. Hence, we have a feasible solution to LP (3) with objective value U.

Conversely, if we have a feasible solution to LP (3) with objective value U, then set: (1) $f_{(\theta,q)} = 0$ if $U_q^0 \leq U$; (2) otherwise, set $f_{(\theta,q)} = \frac{U_q^0 - U}{\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}} z_{\theta,q}c_{(\theta,q)}$. To satisfy the flow constraints, this implies $f_{(s,\theta)} = \sum_{q\in H_\theta} f_{(\theta,q)}$ and $f_{(q,t)} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)}$. We now check that the capacity constraints hold. By the first constraint in the LP, we have $\frac{U_q^0 - U}{\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}} \leq 1$ so $f_{(\theta,q)} \leq z_{\theta,q}c_{(\theta,q)}$,

and because $z_{\theta,q} \leq 1$ we have that the capacity constraints on the $\bigcup_{\theta\in\Theta}\{\theta\} \times H_\theta$ edges are satisfied. Then, we have $f_{(s,\theta)} = \sum_{q\in H_\theta} f_{(\theta,q)} \leq \sum_{q\in H_\theta} z_{\theta,q}c_{(\theta,q)} = p(\theta)v_\theta \sum_{q\in H_\theta} z_{\theta,q} \leq p(\theta)v_\theta m_\theta = c_{(s,\theta)}$ by the second constraint of the LP. Finally, for any $q \in Q$, there are two cases: (1) if $U_q^0 \leq U$, then we have $f_{(q,t)} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = 0$; (2) if $U_q^0 > U$, we have $f_{(q,t)} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = \frac{U_q^0-U}{\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}} \sum_{\theta:q\in H_\theta} z_{\theta,q}c_{(\theta,q)} = U_q^0 - U$. So the $Q \times \{t\}$ edges are exactly saturated. □

We can do a binary search for the optimal value of U to the desired level of precision.[3] The proof of Proposition 4 shows us how to find the test taker's strategy corresponding to a particular flow.

7.3 A Direct Algorithm for Identifying the Tester's Strategy

We now give a direct algorithm (Algorithm 1) that, given an equilibrium strategy for the test taker, computes an equilibrium strategy for the tester. This also allows us to prove that there always exists such a strategy where the tester uniformly randomizes over a subset of the questions (rather than, for example, placing high probability on a question that is hard for many types and low, but nonzero, probability on a question that is hard for fewer types).

Algorithm 1 Input: A test game with $t = 1$ and an optimal primal solution $(U, (z_{\theta,q}))$ to LP (3).

$T \leftarrow \{q \mid U_q^0 - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q} = U\}$
$S \leftarrow \{\theta : \sum_{q\in H_\theta\cap T} z_{\theta,q} < m_\theta\}$
while S has an unmarked element **do**
  $\theta \leftarrow$ an unmarked element from S
  for all $q \in H_\theta \cap T$ and $z_{\theta,q} < 1$ **do**
    $S \leftarrow S \cup \{\theta' \in \Theta : q \in H_{\theta'} \wedge z_{\theta',q} > 0\}$
    $T \leftarrow T \setminus \{q\}$
  end for
  mark θ
end while
return the uniform distribution over T

To prove the correctness of Algorithm 1, we first introduce the following two lemmas.

Lemma 1. *At every point in Algorithm 1, T is nonempty.*

Proof. T is non-empty initially, because otherwise U would clearly be suboptimal. Let T_0 denote that initial T. Suppose that at some point, all problems in T_0 are deleted. We will show how to construct an alternative solution $(z'_{\theta,q})$ such that for each $q \in T_0$, we have $\sum_{\theta\in\Theta:q\in H_\theta} p(\theta)v_\theta z'_{\theta,q} > \sum_{\theta\in\Theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}$. This would then allow us to reduce U, contradicting its optimality.

We initialize $z'_{\theta,q} = z_{\theta,q}$ for all $\theta, q \in T_0 \cap H_\theta$, and $z'_{\theta,q} = 0$ otherwise. Then, we will adjust the $z'_{\theta,q}$ for $q \in T_0$ in the same order in which these q were eliminated from T. We maintain the property that, if q is the latest question for which we have adjusted the $z'_{\theta,q}$, then for all $\theta \in S_q$ (where S_q is the subset S in the algorithm after eliminating q), we have $\sum_{q\in H_\theta} z'_{\theta,q} < m_\theta$. This property holds initially by the initialization of S. When we reach $q \in T_0$, because it was eliminated, there is a $\theta \in S$ such that $z'_{\theta,q} < 1$. Let $\epsilon = (1 - z'_{\theta,q})/2$. Then, let $z'_{\theta,q} \leftarrow z'_{\theta,q} + \epsilon$ and, for every $\theta' \in \Theta$ such that $q \in H_{\theta'}$ and $z'_{\theta',q} > 0$, let $z'_{\theta',q} \leftarrow z'_{\theta',q} - \min(z'_{\theta',q}, \frac{\epsilon}{2|\Theta|}\frac{p(\theta)v_\theta}{p(\theta')v_{\theta'}})$, thereby maintaining the property. As a result, $\sum_{\theta''\in\Theta:q\in H_{\theta''}} p(\theta'')v_{\theta''}z'_{\theta'',q}$ will have increased by at least $\epsilon p(\theta)v_\theta - \sum_{\theta'\in\Theta} p(\theta')v_{\theta'}\frac{\epsilon}{2|\Theta|}\frac{p(\theta)v_\theta}{p(\theta')v_{\theta'}} = \epsilon p(\theta)v_\theta - |\Theta|\frac{\epsilon}{2|\Theta|}p(\theta)v_\theta > 0$. Hence, at the end, for each $q \in T_0$, we have $\sum_{\theta\in\Theta:q\in H_\theta} p(\theta)v_\theta z'_{\theta,q} > \sum_{\theta\in\Theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}$. □

Lemma 2. *Let $(z_{\theta,q})$ be the solution to LP (3) and let T be the final subset that Algorithm 1 returns. Then for each type θ, either $\sum_{q\in H_\theta\cap T} z_{\theta,q} = m_\theta$ (θ memorizes as much of $T \cap H_\theta$ as possible) or $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$ (θ memorizes every question in $T \cap H_\theta$ with probability 1 and will certainly pass the test). Therefore all types of test taker are best-responding.*

Proof. We will show that the algorithm maintains the following properties after initialization of T and S. (1) Any type $\theta \in \Theta$ for which $\sum_{q\in H_\theta\cap T} z_{\theta,q} < m_\theta$ is in S. (2) All marked types θ satisfy $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$. From this, the lemma follows, because at the end of the algorithm, the condition for the **while** loop must be false, so every type must be either not in S, in which case the first condition in the lemma holds, or marked, in which case the second condition holds.

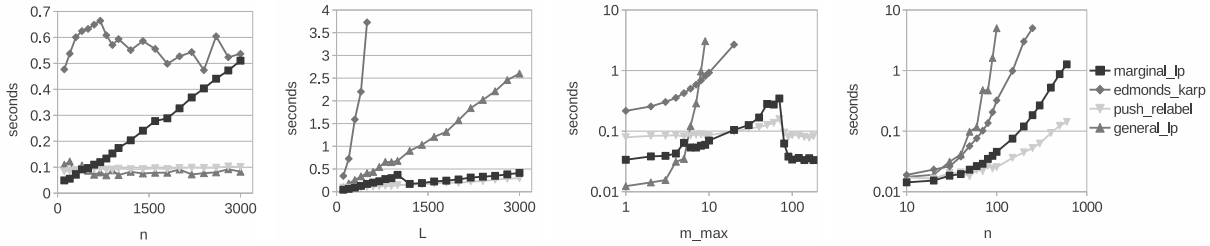
Clearly (1) and (2) hold after initialization. For any θ, the only way in which $\sum_{q\in H_\theta\cap T} z_{\theta,q}$ can decrease is if some $q \in H_\theta$ with $z_{\theta,q} > 0$ is removed from T; but in that case, in the preceding line of the algorithm, θ will have been added to S. This proves (1) is maintained. When some θ is marked, all $q \in H_\theta$ such that $z_{\theta,q} < 1$ have just been removed from T. This proves (2) is maintained. □

Theorem 3. *Let $(z_{\theta,q})$ be the solution to LP (3) and (x_q) the uniform distribution over the set T returned by Algorithm 1. Then $((x_q), (z_{\theta,q}))$ constitute an equilibrium.*[4]

Proof. By Lemma 1, (x_q) is a valid strategy. By the initialization of T, T can ever only include questions that give the

[3] If an exact solution is desired, we can use a similar construction to reduce to the minimax network flow problem [Han, 1997], where the goal is to compute a maximum flow that minimizes $\max_{e\in E'} f_e$ for some distinguished set of edges $E' \subseteq E$. To do so, first (assuming w.l.o.g. $U_{q_1}^0 \leq U_{q_2}^0 \leq U_{q_n}^0$) we use our algorithm above to find out in which interval $[U_{q_i}^0, U_{q_{i+1}}^0]$ the optimal U lies. Then, we modify the network by setting $c_{(q,t)} = \max(0, U_q^0 - U_{q_{i+1}}^0)$ and find a flow that saturates these edges. We consider the residual graph consisting of the remaining capacities, and again adjust the capacities $c_{(q,t)}$ from 0 to $U_{q_{i+1}}^0 - U_{q_i}^0$ whenever $U_q^0 \geq U_{q_i}^0$; finally, we can call a minimax network flow solver on this network with $E' = Q \times \{t\}$. However, as we are not aware of any such solvers, in our experiments we focus on the approach based on binary search.

[4] By complementary slackness, this also means they constitute optimal primal and dual solutions to our LPs.



(a) Increasing n with $L = 150$ and $m_{\max} = 6$.

(b) Increasing L with $n = 150$ and $m_{\max} = 6$.

(c) Increasing $m_{\max}$ with $n = L = 150$.

(d) Increasing n with $L = n$ and $m_{\max} = \lfloor \sqrt{n} \rfloor$.

Figure 2: Runtime for solving an optimal tester strategy in one-question tests.

tester the maximum utility U. Finally, Lemma 2 shows that all test taker types are best-responding. □

8 Experiments

In this section, we describe experiments that we performed to see how different algorithms scale. We restrict our attention to one-question test games so that we can evaluate all algorithms at once. We consider four different algorithms. We use CPLEX (out-of-the-box) to solve (1) the general LP (1) and (2) the one-question marginal-probabilities LP (3). Also, we use the network-flow approach from Definition 2 with binary search on U to a precision of 10^{-8} using (3) Edmonds-Karp [Edmonds and Karp, 1972] and (4) Push-Relabel [Goldberg and Tarjan, 1988], in each case combined with Algorithm 1 to compute the tester's optimal strategy.

In particular, we use CPLEX 10.010 and the boost 1.46.1 C++ library for Edmonds-Karp and Push-Relabel. Our machine has an Intel i7-2600 3.40GHz CPU and 8GB memory.

For each experimental data point, we specify three parameters: the number of questions n, the number of types L ($|\Theta| = L$), and the maximum memory size $m_{\max}$. We always set $b_{\max}$, the maximum size of any H_θ, to $2m_{\max}$. Given those parameters, a test game instance is randomly generated as follows: for each $\theta \in \Theta$, draw m_θ uniformly from 1 to $m_{\max}$; draw $|H_\theta|$ uniformly from m_θ to $b_{\max}$; generate H_θ by drawing $|H_\theta|$ elements from Q uniformly; and draw $w_\theta = p(\theta)v_\theta$ uniformly from $[0, 1]$ (these two factors always appear together). For each data point, we generate 5 test game instances and compute the average running time for each algorithm. We set a timeout of 5 seconds for each instance.

Figures 2(a), 2(b), and 2(c) show how the algorithms scale in n, L, and $m_{\max}$, respectively, holding the other parameters fixed. (Note the logarithmic scales on Figure 2(c).) None of the algorithms have trouble scaling in n alone. Edmonds-Karp does not scale very well in L and $m_{\max}$, and the general LP scales particularly poorly in $m_{\max}$. The marginal LP and particularly push-relabel always scale very well.

In fact, these experiments indicate increasing a single parameter by itself never leads to truly hard instances. For n eventually many of the questions become irrelevant (not hard for any type). For L and $m_{\max}$, eventually it becomes optimal to randomize uniformly over all questions. Thus, to identify more challenging instances, multiple parameters need to increase simultaneously, as we do in Figure 2(d) (note again the logarithmic scales). Push-relabel performs best.

	$0 \le m_{\max} \le$ const.	$m_{\max} >$ const.
$t = 0, 1$	P	P
$2 \le t \le$ const.	P	coNP-c
$t >$ const.	NP-c	coNP-h and NP-h

Table 1: OPTIMAL TEST STRATEGY's complexity ($t, m_{\max}$ are the test size and max memory size respectively)

9 Conclusion

Table 1 summarizes our complexity results. Our work is only a small first step in the design of algorithms for game-theoretically optimal design of tests. Future research could focus on identifying other tractable cases. For example, in practice, one would expect the H_θ sets to exhibit structure that may be exploited algorithmically. Another direction is to generalize beyond tests whose only outcomes are pass and fail. From a practical perspective, it is also important to develop methodology to obtain the statistical information about test taker types that is needed as input to our algorithms. Perhaps even better than a two-phase approach, in which we first estimate this information and then run our algorithm, would be a true online-learning approach, where we update our testing strategy as additional test takers take our tests.

A Counterexample to Zero-Sum Equivalence when the Test Taker is Stackelberg Leader

Let $Q = \{q_1, q_2\}$, $\Theta = \{\theta_1, \theta_2\}$, $H_{\theta_1} = H_{\theta_2} = Q$, $m_{\theta_1} = m_{\theta_2} = 1$, $p(\theta_1) = p(\theta_2) = 1/2$, $t = 1$, $v_{\theta_1} = 100$, $v_{\theta_2} = 1$, $\omega_{\theta_1} = 1$, and $\omega_{\theta_2} = 100$. If the tester is the leader, then it is optimal for her place probability $1/2$ on each question (and so, by Proposition 1, this is also optimal in the zero-sum version of the game). This results in an expected utility of 0.5 for a test taker of type θ_1, and one of 50 for a test taker of type θ_2—so an expected utility of 25.25 for the test taker *ex ante*.

However, if the test taker can commit to a strategy in the Bayesian game *ex ante* (before learning his type), then he can commit to memorize conditionally on the type as follows: $M_{\theta_1} = \{q_1\}$ and $M_{\theta_2} = \{q_2\}$. Because the tester cares more about failing θ_1, she will test q_2. This results in an *ex ante* expected utility of $(1/2) \cdot 100 = 50$ for the test taker, which is more than the 25.25 from the strategy in the regular version.

References

[Birkhoff, 1946] Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumn Rev, Ser. A, no. 5*, pages 147–151, 1946.

[Chang *et al.*, 2001] Cheng-Shang Chang, Wen-Jyh Chen, and Hsiang-Yi Huang. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Communications*, 49(7):1145–1147, July 2001.

[Conitzer and Sandholm, 2006] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 82–90, Ann Arbor, MI, USA, 2006.

[Cooper and Sahami, 2013] Steve Cooper and Mehran Sahami. Reflections on Stanford's MOOCs. *Communications of the ACM*, 56(2):28–30, February 2013.

[Dulmage and Halperin, 1955] Lloyd Dulmage and Israel Halperin. On a theorem of Frobenius-Konig and J. von Neumann's game of hide and seek. *Trans. Roy. Soc. Canada III*, 49:23–29, 1955.

[Edmonds and Karp, 1972] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, April 1972.

[Goldberg and Tarjan, 1988] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, October 1988.

[Han, 1997] C.-C. Han. A fast algorithm for the minimax flow problem with 01 weights. *Applied Mathematics Letters*, 10(1):11 – 16, 1997.

[Khachiyan, 1979] Leonid Khachiyan. A polynomial algorithm in linear programming. *Soviet Math. Doklady*, 20:191–194, 1979.

[Korzhyk *et al.*, 2010] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 805–810, Atlanta, GA, USA, 2010.

[Letchford *et al.*, 2009] Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *Proceedings of the Second Symposium on Algorithmic Game Theory (SAGT-09)*, pages 250–262, Paphos, Cyprus, 2009.

[Paruchuri *et al.*, 2008] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 895–902, Estoril, Portugal, 2008.

[von Neumann, 1928] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.