Game-Theoretic Question Selection for Tests

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, 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

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 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 test taker of type θ.

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 special 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. Naturally, we require $v_\theta > 0$ and $\omega_\theta > 0$.

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 func-

[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.

tion for the test taker: $u'_2(\theta, T, M_\theta) = 0$ if $T \cap (H_\theta \backslash M_\theta) \neq \emptyset$, 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 [**?**]). $\square$

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 [**?**] (in fact, inapproximable [**?**]); a general mixed-integer linear program formulation has previously been proposed [**?**]. 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 [**?**]. 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. $\square$

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.*

Proof. 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, as follows. Let $Q = V$. Construct $|E| + |V| + 2$ test taker types, as follows:

- For each edge $e = (i, j) \in E$, add one test taker type θ_e with $v_{\theta_e} = |\Theta|$, $H_{\theta_e} = \{i, j\}$, and $m_{\theta_e} = 0$.
- For each vertex i, add one test taker type θ_i with $v_{\theta_i} = |\Theta|(d_{\max} - d(i)), H_{\theta_i} = \{i\}$, and $m_{\theta_i} = 0$. Here $d(i)$ is the degree of vertex i and $d_{\max} = 1 + \max_i d(i)$.
- Add a single test taker type θ^* with $v_{\theta^*} = |\Theta|\alpha\varepsilon, m_{\theta^*} = 2$ and $H_{\theta^*} = V$, where $\alpha = \binom{|V|}{2} - |E| - \binom{k}{2}$ and $\varepsilon < 1$.
- Finally, add a single test taker type θ^{**} with $v_{\theta^{**}} = |\Theta|\varepsilon, H_{\theta^{**}} = V$, and $m_{\theta^{**}} = k$.

Let the test taker types occur with uniform probability $p = 1/|\Theta|$, let $t = 2$, and let

$$u = (2 - |V|)d_{\max} + |E| - \frac{\binom{V}{2} - |E| - 1}{\binom{V}{2} - |E|}\varepsilon$$

First, suppose there exists an independent set of size k. We will construct a strategy for the test taker, that is, a feasible solution to LP 2, that results in a utility of at most $(2 - |V|)d_{\max} + |E| - \varepsilon < u$ for the tester. Let type θ^{**} choose (memorize the answers to) the questions in the independent set (with probability 1). Let type θ^* choose a pair of questions that do not have an edge between them, and of which at least one is outside the independent set (and choose such a pair uniformly at random). The other types have memory size 0.

If, in response to this strategy, the tester chooses a pair of questions with an edge between them ($(i, j) \in E$)), then the types that fail are $\theta_i, \theta_j, \theta^*, \theta^{**}$, and all the θ_e where $e \in E, e \cap \{i, j\} \neq \emptyset$. This results in a savings of $2d_{\max} - d(i) - d(j) + \alpha\varepsilon + \varepsilon + d(i) + d(j) - 1 = 2d_{\max} + (\alpha + 1)\varepsilon - 1$ relative to passing everyone. If the tester chooses i and j with $(i, j) \notin E$ and at least one of i and j outside the independent set, then the types that fail with probability 1 are $\theta_i, \theta_j, \theta^{**}$, and all the θ_e where $e \in E, e \cap \{i, j\} \neq \emptyset$; type θ^* fails with probability $\frac{\alpha-1}{\alpha}$. This results in a savings of $2d_{\max} - d(i) - d(j) + \varepsilon + d(i) + d(j) + \frac{\alpha-1}{\alpha}\alpha\varepsilon = 2d_{\max} + \alpha\varepsilon$ (which is greater than the previous case). Finally, if the tester chooses i and j that are both in the independent set, then the types that fail are $\theta_i, \theta_j, \theta^*$, and all the θ_e where $e \in E, e \cap \{i, j\} \neq \emptyset$. This results in a savings of $2d_{\max} - d(i) - d(j) + \alpha\varepsilon + d(i) + d(j) = 2d_{\max} + \alpha\varepsilon$ (which is the same as the previous case). Because the utility to the tester of passing everyone is $-|E| - |V|d_{\max} + 2|E| - \alpha\varepsilon - \varepsilon$, a savings of $2d_{\max} + \alpha\varepsilon$ results in a utility of $(2 - |V|)d_{\max} + |E| - \varepsilon < u$ for the tester. So the answer to the OPTIMAL TEST STRATEGY instance is "no."

Conversely, suppose that no independent set of size k exists. Consider the tester mixed strategy that chooses a pair of questions i, j with $(i, j) \notin E$ uniformly at random (there are $\binom{|V|}{2} - |E|$ such pairs). We will show that this strategy guarantees the tester an expected utility of at least u. Again, we will reason in terms of the savings to the tester relative to passing everyone. We first consider the savings from types θ_e with $e \in E$ and θ_i with $i \in V$. For these types (which do not have a choice to make), it is easier to reason about the savings resulting from testing a specific pair of questions i, j with $(i, j) \notin E$ with probability $\frac{1}{\binom{|V|}{2} - |E|}$. This savings is $\frac{d(i)+d(j)+2d_{\max}-d(i)-d(j)}{\binom{|V|}{2}-|E|} = \frac{2d_{\max}}{\binom{|V|}{2}-|E|}$. Because there are $\binom{|V|}{2} - |E|$ such pairs, we have a total savings of $2d_{\max}$. The best response for type θ^* is to memorize any pair $\{i, j\}$ where $(i, j) \notin E$, resulting in a probability of $\frac{\binom{|V|}{2}-|E|-1}{\binom{|V|}{2}-|E|}$ that this type fails, so the total savings corresponding to this type is $\alpha\varepsilon\frac{\binom{|V|}{2}-|E|-1}{\binom{|V|}{2}-|E|} = \alpha\varepsilon(1 - \frac{1}{\binom{|V|}{2}-|E|})$ Finally, because by assumption, no independent set of size k exists, no matter which set of k questions θ^{**} memorizes, the probability that θ^{**} fails is at least $\frac{\binom{|V|}{2}-|E|-\binom{k}{2}+1}{\binom{|V|}{2}-|E|} = \frac{\alpha+1}{\binom{|V|}{2}-|E|}$, so the total savings corresponding to this type is at least $\varepsilon\frac{\alpha+1}{\binom{|V|}{2}-|E|}$. Summing the savings for θ^* and θ^{**}, we get $\alpha\varepsilon + \frac{\varepsilon}{\binom{|V|}{2}-|E|}$. Adding all these savings to the utility to the tester of passing everyone, which is $-|E| - |V|d_{\max} + 2|E| - \alpha\varepsilon - \varepsilon$, we get that the tester's utility is $(2-|V|)d_{\max}+|E|+\frac{\varepsilon}{\binom{|V|}{2}-|E|}-\varepsilon = (2-|V|)d_{\max}+|E|-\frac{\binom{V}{2}-|E|-1}{\binom{V}{2}-|E|}\varepsilon = u$. So the answer to the OPTIMAL TEST STRATEGY instance is "yes." □

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., $U_a^0 = \sum_{\theta:q\notin H_\theta} p(\theta)(-v_\theta)$.

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

Solving LP (3) gives us an equilibrium test taker strategy.[2] An equilibrium tester strategy can be read off from the corresponding solution to the dual of this linear program. ?????APPENDIX???????

We can also explicitly construct the tester's optimal strategy from the dual solution using Algorithm 1, which always return a strategy that tests a subset of questions uniformly.

Algorithm 1 Compute the optimal one-problem test strategy

```
1:  function OPTIMALONEPROBLEMTEST($Q, \Theta, p$)
2:      Solve LP (3) and let the solution be $U, z_{\theta,q}$
3:      $T \leftarrow \{q \mid U_q^0 \geq U\}$
4:      $S \leftarrow \{\theta : \sum_{q\in H_\theta \cap T} z_{\theta,q} < m_\theta\}$
5:      while $S$ has an unmarked element do
6:          $\theta \leftarrow$ get and mark an unmarked element from $S$
7:          for all $q \in H_\theta \cap T$ and $z_{\theta,q} < 1$ do
8:              $T \leftarrow T \setminus \{q\}$
9:              $S \leftarrow S \cup \{\theta' : q \in H_{\theta'} \wedge z_{\theta',q} > 0\}$
10:         end for
11:     end while
12:     return Strategy that test $T$ uniformly
13: end function
```

We prove the correctness of Algorithm 1 by firstly introducing the following two lemmas.

Lemma 1. *T returned by Algorithm 1 is non-empty.*

Proof. T is non-empty before we delete problems from T on line 8 of Algorithm 1. Denote that initial T as T_0. If all problems in T_0 are deleted, then for every q in T_0 there is a θ that is in S and $z_{\theta,q} < 1$. Note that for every θ in Q, it has a potential to memorize more either because $\sum_{q\in H_\theta\cap T_0} z_{l,a} < m_l$ (on line 4) or because there is an unnecessary memorizing

[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 [**?**; **?**] for finding the Birkhoff-von Neumann decomposition [**?**]; similar techniques are used in the context of security games [**?**].

$z_{\theta,q} > 0$ that can be freed (on line 9). So all questions in T_0 being deleted means that we can bring U down by using those potentials to memorize all questions in T_0 more (recall $\forall q \in T_0, \exists \theta, z_{\theta,q} < 1$), a contradiction to that U is minimized. □

Lemma 2. *Let $z_{\theta,q}$ be the solution of LP (3) and T be the final subset that Algorithm 1 returns. Then for each type θ, either $\sum_{q \in H_\theta \cap T} z_{\theta,q} = m_\theta$ (they always memorize the maximal subset of $T \cap H_\theta$) or $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$ (they memorize any question in $T \cap H_\theta$ with probability 1 so they will pass the test for sure). Therefore all test takers are best responding.*

Proof. Let T_0 be the initial T before any deletion on line 8 of Algorithm 1. Initially, for any θ such that $\sum_{q \in H_\theta capT_0} z_{\theta,q} < m_\theta$, we push it to S. After that any $q \in H_\theta$ where $z_{\theta,q} < 1$ is deleted from T on line 8. Therefore $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$ must hold for that θ. Afterwards, more θ may violate $\sum_{q \in H_\theta capT} z_{l,a} = m_l$ because some q are deleted from T. But for those H_θ, we must have pushed them into S on line 9. So they will satisfy $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$ afterwards. In the end, every $\theta \in S$ will satisfy $\forall q \in H_\theta \cap T, z_{\theta,q} = 1$ and every other $\theta \notin Q$ will satisfy $\sum_{q \in H_\theta \cap T} z_{\theta,q} = m_l$. □

Then we show our strategy is optimal by complementary slackness theorem, which in our context is essentially: 1) the tester only tests problems (non-zero primal variables) that gives him best utility (tight dual constraints); 2) the test taker only memorizes problems (non-zero dual variables) that gives him best utility (tight primal constraints).[3]

Theorem 3. *Uniformly testing T returned by Algorithm 1 is optimal.*

Proof. First of all, T is non-empty by Lemma 1 so we can test T uniformly. Then since memorizing strategy $z_{\theta,q}$ makes U_q (the utility to test a problem q) decrease to U if $U_q^0 > U$ and we only pick T from problems that $U_q^0 > U$, every problem that we might test gives the tester best utility. That is the first condition for the complementary slackness theorem. Finally, Lemma 2 shows that one type of test takers either pass the test for sure or memorize the maximal subset in $H_\theta \cap T$ all the time which is optimal for uniform test, the second condition of complementary slackness theorem holds. □

7.2 A Network Flow Approach

Our algorithm is shown in Algorithim 1.

Finally, instead of using general LP algorithm like simplex [cite] to solve LP (3), we can alternatively use binary search plus max flow to solve it as shown in Algorithm 2. With a given U, we use max flow to check its feasibility. Intuitively, a U is feasible if and only if there is a strategy $z_{l,a}$ such that for every a where $U_a^0 > U$, that strategy decreases it down to U. This is achievable if and only if the max flow can saturate all edges from A to t. Given a max flow f, we can restore the strategy by setting $z_{l,a} = f_{(B_l,a)}/c_{(B_l,a)}$ which is the flow of an edge divded by the capacity of an edge. If we use Push-Relabel algorithm [cite] to solve the max flow in $O((n+L)^3)$ time, the whole Algorithm 1 runs in $O((n+L)^3 + \sum_l |B_l|)$ time as the remaining part can run in linear time with repsect to the input size (assuming we only binary search for constant times to achieve a constant accuracy).

[3]This is essentially saying that in a zero-sum game, a minimax, or maximin state is reached, if and only if the Nash equilibrium is reached.

Algorithm 2 Use binary search and max flow to solve LP (3)

```
1: $U_{lower} \leftarrow 0$, $U_{upper} \leftarrow \max_{a \in A} U_a^0$
2: while $U_{upper} - U_{lower} > \epsilon$ do                ▷ Binary search
3:     $U \leftarrow (U_{upper} + U_{lower})/2$              ▷ Objective utility
4:     $V \leftarrow \{s\} \cup \mathbb{B} \cup A \cup \{t\}$              ▷ Vertex set
5:     $E \leftarrow \emptyset$                              ▷ Initialize edge set
6:     for all $a \in A$ do
7:         if $U_a^0 > U$ then
8:             $E \leftarrow E \cup \{(a,t)\}$
9:             $c_{(a,t)} \leftarrow U_a^0 - U$               ▷ Edge capacity
10:        end if
11:    end for
12:    for all $B_l \in \mathbb{B}$ do
13:        $E \leftarrow E \cup \{B_l\} \times B_l$
14:        $c_{\{B_l\} \times B_l} \leftarrow p_l v_l$
15:        $E \leftarrow E \cup \{(s, B_l)\}$
16:        $c_{(s,B_l)} \leftarrow p_l m_l v_l$
17:    end for
18:    $f \leftarrow \textsc{MaxFlow}(G = (V,E), c)$
19:    if $f$ saturates all edges in $A \times \{t\}$ then
20:        $U_{upper} \leftarrow U$
21:    else
22:        $U_{lower} \leftarrow U$
23:    end if
24: end while
25: $(\forall l, a), z_{l,a} \leftarrow f_{(B_l,a)}/c_{(B_l,a)}$
26: return $U, z_{l,a}$
```

7.3 Experiments

In this section, we conduct experiments to see how different approaches scale on one-problem test games. The first approach is using CPLEX to solve our general LP (1) which is exponential to the memorize capacity. The second one is using CPLEX to directly solve the marginal version LP (3). The third and fourth use Algorithm 2 to solve LP (3). They are using Edmonds-Karp [cite] and Psh-Relabel[4] [cite] max flow algorithm in C++ boost library respectively.

For each test case, we specify four parameters n, L, m_{max}, b_{max}, the number of problems, the number of test taker types, the maximum memorize capacity and the maximum size of B_l. Given those parameters, a test game instance is generated as follows: for each $1 \leq l \leq L$, generate m_l uniformly from 1 to m_{max}, generate $|B_l|$ uniformly from m_l to b_{max}, generate B_l by selecting $|B_l|$ elements from A uniformly and generate $w_l = v_l p_l$ (weight of type-l) from $[0, 1]$ uniformly. Here we only generate w_l because v_l and p_l always appear together as $v_l p_l$. For each parameter setting, we generate 5 test game instances and compute the average running time for each algorithm. During each run, we set timeout to 5 seconds.

[4]We modified its is_flow function for floating number issues.

Figure 1: Average running time (timeout after 5 seconds) for different one-problem test algorithms over the number of problems n. For each n, we set the other three parameters $L = n, m_{max} = \lfloor\sqrt{n}\rfloor$ and $b_{max} = 2m_{max}$ (see Table 1).

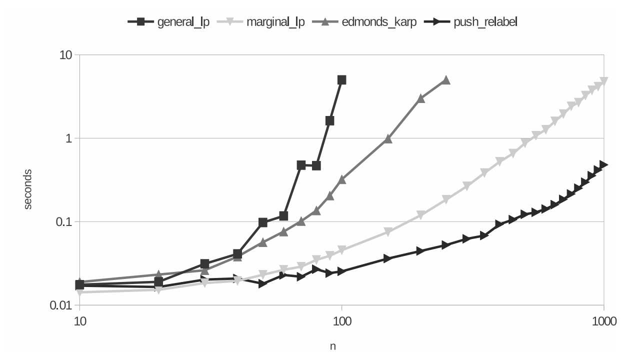

Table 1: Parameter settings for the benchmark

n	L	m_{max}	b_{max}
10	10	3	6
50	50	7	14
100	100	10	20
300	300	17	34
500	500	22	44
700	700	26	52
1000	1000	31	62

In our benchmark, we increase n and set the remaining three parameters L, m_{max}, b_{max} as following: $L = n, m_{max} = \lfloor\sqrt{n}\rfloor$ and $b_{max} = 2m_{max}$. Note that our L, b_{max} are quite small (they can be as large as $b_{max} = n/2, L = \binom{n}{b_{max}}$). That's becuase for large L, b_{max}, our uniform random input will almost always make the trivial strategy to uniformly test among all n problems optimal. So we make them small to avoid such trivial solutions. The realized parameter settings are shown in Table 1. The benchmark is shown in Figure 1. As expected, the general LP is exponential so it times out before n reaches 100. Unexpectedly, using CPLEX to directly solve LP (3) is much faster than Edmonds-Karp implementation. Finally, Push-Relabel version outperforms all others significantly. It solves $n = 1000$ case in about 0.5 second while the second best CPLEX version uses almost 5 seconds.

[Machine, GLPK and GPLEX spec]

8 Conclusion

A Counterexample to Zero-Sum Equivalence When the Test Taker Is the 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.

B Dual of the linear program and interpretation

References