Journal of Artificial Intelligence Research 59 (2017) 437-462 Submitted 12/16; published 07/17

Game-Theoretic Question Selection for Tests

Yuqian Li YUQIAN@CS.DUKE.EDU
Vincent Conitzer CONITZER@CS.DUKE.EDU
Duke University
Durham, NC 27707 USA

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, 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, as offered for example by Coursera, Udacity, and edX). Specifically, the test result may not reflect the true ability of a test taker if questions are leaked beforehand.

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 to let appear on the test. We model this as a Stackelberg game, where the tester commits to a mixed strategy and the follower responds. Informally, the goal of the tester is to best reveal the true ability of a test taker, while the test taker tries to maximize the test result (pass probability or score). We provide an exponential-size linear program formulation that computes the optimal test strategy, prove several NP-hardness results on computing optimal test strategies in general, and give efficient algorithms for special cases (scored tests and single-question tests). Experiments are also provided for those proposed algorithms to show their scalability and the increase of the tester's utility relative to that of the uniform-at-random strategy. The increase is quite significant when questions have some correlation—for example, when a test taker who can solve a harder question can always solve easier questions.

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 recent discussions, see Cooper & Sahami, 2013; Hew & Cheung, 2014; Reich et al., 2015). 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 so 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 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).

An early version of this paper appeared in the proceedings of IJCAI 2013 (Li & Conitzer, 2013). The following additional material has been added to this journal version. A more general game model is introduced, which also includes scored tests and their zero-sum transformation. Polynomial-time algorithms for solving scored tests are given, and additional experiments with scored tests as well as with non-uniform test games are included. Finally, a more detailed proof of Theorem 3 (Theorem 2 in the conference paper) is included, for which there was insufficient space in the conference proceedings. The abstract and introduction have also been updated.

1.1 Related Work

The computation of Stackelberg mixed strategies has recently been applied to a number of real security domains (Pita, Jain, Ordóñez, Portway, Tambe, & Western, 2009; Tsai, Rathi, Kiekintveld, Ordóñez, & Tambe, 2009; Shieh, An, Yang, Tambe, Baldwin, DiRenzo, Maule, & Meyer, 2012; Yin, Jiang, Johnson, Tambe, Kiekintveld, Leyton-Brown, Sandholm, & Sullivan, 2012; Fang, Jiang, & Tambe, 2013; Xu, Fang, Jiang, Conitzer, Dughmi, & Tambe, 2014). The main reason for using a Stackelberg formulation in these domains is that in many of them, an attacker could observe the defender's actions over time and thereby learn the defender's strategy before acting; the same can be argued for the domain that we are studying here. The Stackelberg model also largely avoids the possible equilibrium selection problems of a simultaneous-move model. Another advantage is that a Stackelberg strategy of a two-player normal-form game can be computed in polynomial time (Conitzer & Sandholm, 2006; von Stengel & Zamir, 2010), whereas computing one Nash equilibrium is

PPAD-complete (Daskalakis, Goldberg, & Papadimitriou, 2009; Chen, Deng, & Teng, 2009) and computing an (even approximately) optimal Nash equilibrium is NP-hard (Gilboa & Zemel, 1989; Conitzer & Sandholm, 2008). However, for Bayesian games (which we study here), computing an optimal (or even approximately optimal) Stackelberg strategy is NP-hard (Conitzer & Sandholm, 2006; Letchford, Conitzer, & Munagala, 2009). Mixed integer program and heuristic branch-and-bound approaches have been used to tackle this problem and have turned out to be reasonably scalable (Jain, Pita, Tambe, Ordóñez, Paruchuri, & Kraus, 2008; Jain, Kiekintveld, & Tambe, 2011). Our approach is very different. We modify the Bayesian Stackelberg game into a Bayesian zero-sum game by exploiting our problem's structure, allowing us to use a much more efficient minimax LP.

Of course, problem structure has also been exploited in previous work on security games. Kiekintveld et al. proposed a quite general (though not all-encompassing) model of security games (Kiekintveld, Jain, Tsai, Pita, Ordóñez, & Tambe, 2009). In it, the defender assigns resources to targets (or sets of targets), and the attacker chooses a single target to attack. The size of the normal form of these games is exponential in the natural representation size, but a more compact MIP/LP formulation can be given whose size is polynomial (Kiekintveld et al., 2009). However, this formulation is correct only under certain conditions (for example, each resource can be assigned only to a single target), and when these do not hold NP-hardness is usually encountered (Korzhyk, Conitzer, & Parr, 2010; Letchford & Conitzer, 2013).

In our domain, when tests have binary outcomes ("pass" or "fail"), we give a polynomial-size LP for the case where only a single question is tested, but show NP-hardness results for multi-question tests. (Our general minimax LP has exponential size in the natural representation of the problem.) On the other hand, when tests have linear outcomes (i.e., agents' utilities are linear in scores), we can solve for the optimal test for any number of questions using a polynomial separation oracle. Hence scored tests are relatively easy compared to binary tests. As a result, we will devote most of the space in later sections to binary tests.

It is not surprising that there are many high-level similarities between our results and previous results on security games. In some ways, the role of the tester is analogous to that of the attacker (with the test-taker "defending" certain questions by memorizing their answers). However, the tester is the natural Stackelberg leader and in that sense resembles the defender in security games. In section 3, we show that if we do make the test taker the Stackelberg leader, the zero-sum equivalence that we demonstrate and rely on no longer holds.

1.2 Contributions

In this paper, we take the loss of confidentiality as a fact and model the test as a 2-player Bayesian Stackelberg game between the tester and the test taker. By exploiting the utility structure, we transform the game into a Bayesian zero-sum game (which only works if the tester is the Stackelberg leader). It allows us to show the equivalence among Nash equilibrium, Stackelberg, and minimax strategies of the transformed zero-sum game. Hence using a minimax linear program, we can compute Stackelberg strategies more efficiently than than we can for general Bayesian Stackelberg games.

However, that linear program is still of exponential size. To completely settle the computational complexity, we further consider two special cases of test games: scored tests and binary tests, where the former one has a linear utility (score), while the latter has a binary utility (pass or fail). We show that scored tests are relatively easy as we can use a separation oracle to compute their optimal strategies in polynomial time. On the other hand, we prove several (co)NP-hardness results for computing binary tests' optimal strategies. Yet, in the special case where the binary test only consists of one question, polynomial algorithms exist. We gave both a linear program approach and a network-flow approach for this special case. The network-flow approach is faster in our experiments, and it can be used to prove an interesting property of the optimal test strategies in one-question binary tests: there always exists an optimal strategy that selects a subset of questions, and asks any one of them with equal probability.

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. (Without loss of generality, we assume $m_\theta \leq |H_\theta|$.) The function $p : \Theta \to [0, 1]$ is the probability distribution over types ($\sum_{\theta \in \Theta} p(\theta) = 1$). 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 the two cases that occur most frequently in our lives.

1. **Scored Tests.** Each question $q \in Q$ has a score s_q. If a test taker fails to solve q, score s_q is deducted. Most exams at schools fall into this category. The test taker wants to maximize the score, which is equivalent to minimize the score deducted. Hence the test taker's utility is $u_2(\theta, T, M_\theta) = -\omega_\theta \sum_{q \in T \cap (H_\theta \setminus M_\theta)} s_q$ where ω_θ denotes how much θ cares about the test.

   We also assume that the tester wants to find out as many questions as possible, in terms of the sum of their scores, that the test takers cannot solve. Hence $u_1(\theta, T, M_\theta) = v_\theta \sum_{q \in T \cap (H_\theta \setminus M_\theta)} s_q$, where v_θ is how much the tester cares about type θ.

2. **Binary Tests.** The only two outcomes of a binary test are to pass or fail the test taker. An example of this is driver's license tests. Moreover, we assume that the tester

1. In common parlance, "problem" may be a better word, but this runs the risk of confusion with our *computational* problems.

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.[2]

In both cases, ω_θ can be seen as θ's perception of the test's importance, and v_θ can be seen as the tester's perception of θ's importance. 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.[3] 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.

Definition 1 (Nash Equilibrium)**.** *A strategy profile $\sigma = (\sigma_1, \sigma_2)$ is a* Nash equilibrium *if and only if for each player i, her strategy σ_i is a best response to the other: $\sigma_1 \in BR^1(\sigma_2), \sigma_2 \in BR^2(\sigma_1)$.*

Definition 2 (Stackelberg Strategy)**.** *A* Stackelberg strategy *σ_1^* of the leading player 1 is one that maximizes her utility when the following player 2 plays a best response strategy $\sigma_2 \in BR^2(\sigma_1^*)$. (If player 2 has multiple best responses ($|BR^2(\sigma_1^*)| > 1$), we assume that he will break the tie in favor of player 1, as is commonly done in this line of research.)*

3. Equivalence to a Zero-Sum Game

For any test game G as defined above (both scored tests and binary tests), we can modify it into a zero-sum game G' by substituting the following utility function for the test taker: $u_2'(\theta, T, M_\theta) = \frac{v_\theta}{\omega_\theta} u_2(\theta, T, M_\theta)$.

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

2. It is instructive to consider the case where we have only a single type θ ($p(\theta) = 1$). In this case, it is optimal for the test taker to choose questions uniformly at random from H_θ. We face more interesting tradeoffs when there are more types. For example, a question that is hard for multiple types is more appealing than one that is hard only for a single type. However, if it is completely predictable that we will put the former question on the test, then all types will memorize it.
3. It is observable even if only a fraction of test takers leak the tests (especially when there are many test takers).

for test taker θ_1	memorize q_1	memorize q_2
test q_1	-100, 1	0, 0
test q_2	0, 0	-100, 1

for test taker θ_2	memorize q_1	memorize q_2
test q_1	-1, 100	0, 0
test q_2	0, 0	-1, 100

Table 1: Normal-form representation of our counterexample to zero-sum equivalence when the test taker (in a binary test) is the Stackelberg leader.

Proof. From each test taker's perspective, we just multiply their utility by a constant $\frac{v_\theta}{\omega_\theta}$. Hence their best responses remain the same. Therefore the tester's Stackelberg mixed strategies and the set of Nash equilibrium strategies remain the same.

Finally, G' is a zero-sum game, and 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, see von Neumann, 1928), which is equal to the set of Stackelberg mixed strategies (in a 2-player zero-sum game). □

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 below in section 3.1. 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. We also note that there is no harm for the tester if the test taker does not play the equilibrium strategy of G': the tester's utility (which is the same in both G and G') could only weakly increase.

3.1 Counterexample to Zero-Sum Equivalence when the Test Taker (in a Binary Test) is the Stackelberg Leader

Consider a binary test where $Q = \{q_1, q_2\}$, $\Theta = \{\theta_1, \theta_2\}$, $H_{\theta_1} = H_{\theta_2} = Q$, $m_{\theta_1} = m_{\theta_2} = 1$, $p(\theta_1) = p(\theta_2) = 1/2$, $t = 1$, $v_{\theta_1} = 100$, $v_{\theta_2} = 1$, $\omega_{\theta_1} = 1$, and $\omega_{\theta_2} = 100$. Table 1 is a normal-form representation of this game.

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.

4. Scored Tests

Computing an optimal Stackelberg mixed strategy in a general 2-player Bayesian game is NP-hard (Conitzer & Sandholm, 2006) and inapproximable (Letchford et al., 2009); a general mixed-integer linear program formulation has previously been proposed (Paruchuri, Pearce, Marecki, Tambe, Ordóñez, & Kraus, 2008). However, thanks to Proposition 1, we only need to solve for a maximin strategy of a zero-sum game. With that, we show how to solve scored tests in polynomial time in this section.[4]

Theorem 1. *Finding the optimal test strategies for scored tests is in P.*

Proof. Due to Proposition 1, we only focus on solving zero-sum game G', which has the same optimal test strategies as G. The linear program (1) below is essentially a maximin LP for zero-sum game G'. Inequality (1a) ensures that the test takers are best responding and equation (1b) lets the tester only test t questions. It has an exponential number of constraints but only a polynomial number of variables. (The number of constraints is exponential because there are $\binom{|H_\theta|}{|M_\theta|}$, an exponential number, actions for the test taker.) In order to obtain only polynomially many variables (rather than exponentially many), we use *marginal* probabilities x_q instead of pure action probabilities x_T. Here, variable V_θ is the utility that the tester receives from type θ (V_θ could be nonnegative or negative), x_q is the marginal probability of testing question q, and x_T is the probability of testing exactly the questions in T.

Here we can use the marginal probability x_q because the test utility is linear for test takers so their best-responses only depend on x_q. Given those marginal probabilities x_q, we can find the mixed strategy explicitly using the Dulmage-Halperin algorithm (Dulmage & Halperin, 1955; Chang, Chen, & Huang, 2001) for finding the Birkhoff-von Neumann decomposition (Birkhoff, 1946); similar techniques are used in the context of security games (Korzhyk et al., 2010).

$$\max_{x,V} \sum_{\theta\in\Theta} p(\theta)V_\theta \tag{1}$$

$$s.t.\ (\forall\theta\in\Theta, \forall M_\theta \subseteq H_\theta : |M_\theta| = m_\theta)$$

$$v_\theta \sum_{q\in H_\theta\setminus M_\theta} s_q x_q \geq V_\theta; \tag{1a}$$

$$\sum_{q\in Q} x_q = t; \tag{1b}$$

$$(\forall q)\ \ 0 \leq x_q \leq 1; \tag{1c}$$

Given any vector of x_q, we can find out in polynomial time whether a constraint is violated as follows. There are only a polynomial number of inequalities (1b) and (1c), so checking them is trivial. For inequality (1a), we enumerate over all types. For each type θ, sort questions $q \in H_\theta$ by $s_q x_q$, and select the top m_θ questions as θ's best response M_θ. If

4. We thank Yang Cai for his helpful advice about separation oracles in general multi-row/column zero-sum games.

inequality (1a) holds for this M_θ, it also holds for any other M_θ. Hence we either report violation of this constraint, or report that no violation is found after checking all θ.

With this polynomial separation oracle[5], we can solve the LP and find the optimal strategy in polynomial time using the ellipsoid algorithm (Grötschel, Lovász, & Schrijver, 1981) (one can solve a LP with an exponential number of constraints as long as there is a polynomial separation oracle). □

5. Binary Tests

We now turn our attention to binary tests, where our algorithmic contributions will be more technically demanding. As before, we will only focus on solving zero-sum game G' due to Proposition 1.

5.1 General Linear Program (LP) Formulation

Similar to LP (1), the following maximin linear programming approach is standard. However, unlike LP (1), we have to use pure action probabilities x_T instead of marginal probabilities x_q since the outcome of a test is no longer linear in the questions missed.

$$\begin{aligned}
\max_{x,V}\ & \sum_{\theta\in\Theta} p(\theta)V_\theta \\
s.t.\ & (\forall\theta, \forall M_\theta \subseteq H_\theta : |M_\theta| = m_\theta) \\
& \sum_{T\subseteq Q:|T|=t} \mathrm{u}_1(\theta, T, M_\theta)x_T \geq V_\theta; \\
& \sum_{T\subseteq Q:|T|=t} x_T = 1; \\
& (\forall T \subseteq Q : |T| = t)\ x_T \geq 0;
\end{aligned} \tag{2}$$

The dual of the above LP is:

$$\begin{aligned}
\min_{y,U}\ & U \\
s.t.\ & (\forall T \subseteq Q : |T| = t) \\
& U \geq \sum_{\theta, M_\theta\subseteq H_\theta:|M_\theta|=m_\theta} u_1(\theta, T, M_\theta)y_{\theta,M_\theta}; \\
& (\forall\theta) \sum_{M_\theta\subseteq H_\theta:|M_\theta|=m_\theta} y_{\theta,M_\theta} = p(\theta); \\
& (\forall\theta, M_\theta \subseteq H_\theta : |M_\theta| = m_\theta)\ \ y_{\theta,M_\theta} \geq 0;
\end{aligned} \tag{3}$$

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

5. An algorithm that for a candidate solution produces a constraint that that candidate solution violates, or returns that there is no such constraint, is a separation oracle.

conditional on having type θ is $y_{\theta,M_\theta}/p(\theta)$.[6] The objective $U = \max_{T \subseteq Q:|T|=t} U_T$ 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. (Note that we are now considering zero-sum game G' in Proposition 1 so the Nash equilibrium, minimax, and Stackelberg strategies are all equivalent.)

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 the pure strategy space 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 2 and 3 indeed give us a polynomial-time algorithm. But, can the binary 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?

5.2 Constant Memory Size

In this subsection, 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 3 (The Optimal Binary Test Strategy problem)**.** *Given a binary test game G and a value u, the* Optimal Binary 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 Binary 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 (2) 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 (Vanderbei, 2001, p. 38). 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 questions!

Theorem 2. *Even if the test taker cannot memorize any questions ($m_\theta = 0$) and $|H_\theta| = 2$ for all $\theta \in \Theta$,* Optimal Binary 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

6. Without loss of generality, we assume $p(\theta) > 0$, otherwise we can just ignore that type.

construct an Optimal Binary 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 $u = 0$.[7]

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. □

5.3 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 Binary Test Strategy *is in coNP.*

Proof. When t is constant, the number of constraints (not counting the nonnegativity constraints on the variables) in LP (3) 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. Moreover, if it is not possible for the tester to obtain u or more, then by strong duality, there exists a set of variables (of the requisite size, by the same argument as in the proof of Proposition 2) of the LP that certifies this. □

Theorem 3. *Even if the test size t is 2 and there are only two types that can memorize any answers,* Optimal Binary 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 Binary 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 $L = |E| + |V| + 2$ test taker types, as follows:

- For each edge $e = \{i, j\} \in E$, add one test taker type θ_e with $v_{\theta_e} = L$ ($|\Theta| = L$), $H_{\theta_e} = \{i, j\}$, and $m_{\theta_e} = 0$.
- For each vertex i, add one test taker type θ_i with $v_{\theta_i} = L(d_{\max} - d(i)), H_{\theta_i} = \{i\}$, 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 $v_{\theta^*} = L\alpha\varepsilon, m_{\theta^*} = 2$ and $H_{\theta^*} = V$, where $\alpha = \binom{|V|}{2} - |E| - \binom{k}{2}$ and $\varepsilon < 1$.

7. The value of ω_θ is irrelavant due to our zero-sum transformation; for completeness, however, one can always assume $\omega_\theta = 1$ in our game constructions.

- Finally, add a single test taker type θ^{**} with $v_{\theta^{**}} = L\varepsilon, H_{\theta^{**}} = V$, and $m_{\theta^{**}} = k$.

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

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

We now prove both directions of the equivalence.

1. 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 (3), that results in a utility of at most $(2 - |V|)d_{\max} + |E| - \varepsilon < u$ for the tester.[8] 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.

   We then consider different pairs of questions that the tester may choose in response to this test taker strategy.

   (a) If 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.

   (b) 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).

   (c) 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 Binary Test Strategy instance is "no."

2. 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.

8. Note that this is only a feasible strategy, not an optimal strategy, so the player may not be best responding; in other words, this is only a feasible solution, not an optimal solution to our LP.

(a) 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 $2d_{\max}$.

(b) 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 $\alpha\varepsilon\frac{\binom{|V|}{2}-|E|-1}{\binom{|V|}{2}-|E|} = \alpha\varepsilon(1 - \frac{1}{\binom{|V|}{2}-|E|})$

(c) 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 Binary Test Strategy instance is "yes."

□

5.4 Tests of Size One

Theorem 2 shows that when we do not bound the test size, our problem is NP-hard even if test takers cannot memorize anything. But, if we do not bound the memory size, Theorem 3 only shows that our problem is coNP-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 while the memory size is not limited. 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 (2) 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 questions 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.

5.4.1 Linear Program

We first present a linear program that can be thought of as a polynomial-size version of LP (3). Likewise, it has a single variable U for the objective. But 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., $U_q^0 = \sum_{\theta:q\notin H_\theta} p(\theta)(-v_\theta)$. The first set of inequalities below corresponds to the minimax condition, and the remaining inequalities guarantee that z constitutes a valid test taker strategy (memory size is limited to m_θ, and each probability is in $[0,1]$).

$$\begin{aligned} &\min\ U \\ &s.t.\ (\forall q \in Q)\ U \geq U_q^0 - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q} \\ &\qquad (\forall \theta \in \Theta)\ \sum_{q\in H_\theta} z_{\theta,q} \leq m_\theta \\ &\qquad (\forall \theta \in \Theta, q \in H_\theta)\ 0 \leq z_{\theta,q} \leq 1 \end{aligned} \tag{4}$$

Solving LP (4) gives us an equilibrium test taker strategy.[9]

5.4.2 A Network Flow Approach

We now show that LP (4) 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 4 (Test network)**.** *Given an instance of the binary 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 $c_{(q,t)} = \max(0, U_q^0 - U)$.*

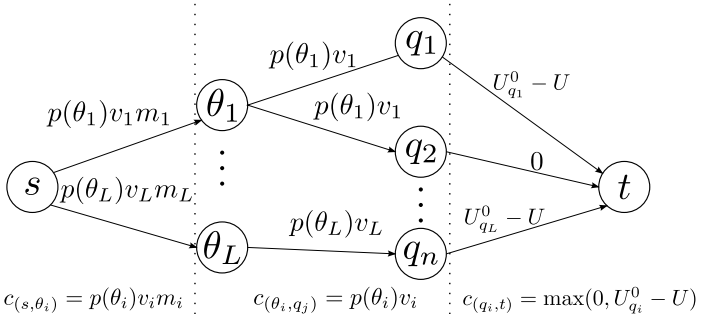
Proposition 4. *The test network has a feasible flow that saturates all the edges in $Q \times \{t\}$ if and only if LP (4) has a feasible solution with value U.*

Proof. We prove both directions of the equivalence.

1. If the test network has a feasible flow f that saturates all the edges in $Q \times \{t\}$, then consider the solution to LP (4) where $z_{\theta,q} = f_{(\theta,q)}/c_{(\theta,q)}$, so that clearly $0 \leq z_{\theta,q} \leq 1$.

9. See the proof of Theorem 1 for how we can translate marginal probabilities into a mixed strategy.

Figure 1: The network that is used to solve LP (4).



Because the $Q \times \{t\}$ edges are saturated, we have that for each $q \in Q$,

$$\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = \max(0, U^0_q - U) \geq U^0_q - U$$

which implies $U \geq U^0_q - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}$. 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 (4) with objective value U.

2. Conversely, if we have a feasible solution to LP (4) with objective value U, then set: (1) $f_{(\theta,q)} = 0$ if $U^0_q \leq U$; (2) otherwise, set $f_{(\theta,q)} = \frac{U^0_q - U}{\sum_{\theta':q\in H_{\theta'}} p(\theta')v_{\theta'} z_{\theta',q}} z_{\theta,q} c_{(\theta,q)}$. To satisfy the flow constraints, let $f_{(s,\theta)} = \sum_{q\in H_\theta} f_{(\theta,q)}$ 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 $\frac{U^0_q - U}{\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}} \leq 1$ 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 $U^0_q \leq U$, then we have $f_{(q,t)} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = 0$; (2) if $U^0_q > U$, we have

$$f_{(q,t)} = \sum_{\theta:q\in H_\theta} f_{(\theta,q)} = \frac{U^0_q - U}{\sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q}} \sum_{\theta:q\in H_\theta} z_{\theta,q} c_{(\theta,q)} = U^0_q - U$$

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.[10] The proof of Proposition 4 shows us how to find the test taker's strategy corresponding to a particular flow.

10. 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

5.4.3 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).

The high level idea of this algorithm is to first construct T, the set of questions that provide the tester the highest utility U. It will not suffice to directly test all those questions uniformly at random because some test taker types may have unused memory capacity that can further lower the utility to test some questions. We put those test takers into S, and use a while loop to gradually expand S and shrink T.

Algorithm 1 Input: A binary test game with $t = 1$ and an optimal primal solution $(U, (z_{\theta,q}))$ to LP (4).

$T \leftarrow \{q \mid U_q^0 - \sum_{\theta:q\in H_\theta} p(\theta)v_\theta z_{\theta,q} = U\}$
$S \leftarrow \{\theta : \sum_{q\in H_\theta\cap T} z_{\theta,q} < m_\theta\}$
let all θ be unmarked
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 questions 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

some distinguished set of edges $E' \subseteq E$. To do so, first (assuming w.l.o.g. $U_{q_1}^0 \le U_{q_2}^0 \le U_{q_n}^0$) 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 $U_q^0 \ge U_{q_i}^0$; 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.

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 $\sum_{q \in H_\theta} z'_{\theta,q} < m_\theta$ (intuitively, S is the set of test taker types that have unused memory capacity). 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 $\epsilon = \min\{1 - z_{\theta,q}, m_\theta - \sum_{q \in H_\theta} z'_{\theta,q}\}/2$. 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 (every new type θ' that enters S is guaranteed to have unused memory). 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 $\sum_{\theta \in \Theta : q \in H_\theta} p(\theta)v_\theta z'_{\theta,q} > \sum_{\theta \in \Theta : q \in H_\theta} p(\theta)v_\theta z_{\theta,q}$ (because it was eliminated). □

Lemma 2. *Let $(z_{\theta,q})$ be the solution to LP (4) 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.*[11]

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. □

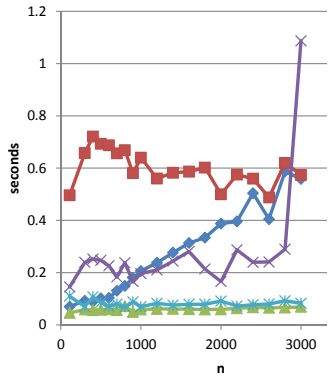
Theorem 4. *Let $(z_{\theta,q})$ be the solution to LP (4) and (x_q) the uniform distribution over the set T returned by Algorithm 1. Then $((x_q), (z_{\theta,q}))$ constitute an equilibrium.*[12]

Proof. By Lemma 1, (x_q) is a valid strategy. By the initialization of T, T can ever only include questions that give the tester the maximum utility U. Finally, Lemma 2 shows that all test taker types are best-responding. □

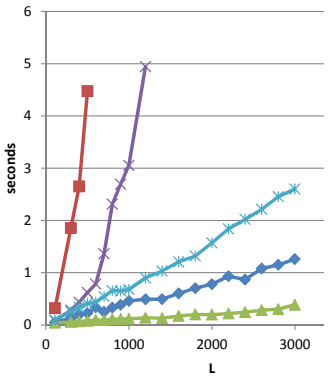
6. Experiments

In this section, we describe experiments that we performed to see how different algorithms scale. We also show how the optimal test strategy outperforms simple test strategies, such as drawing questions uniformly at random. In order to compare scored test algorithms with

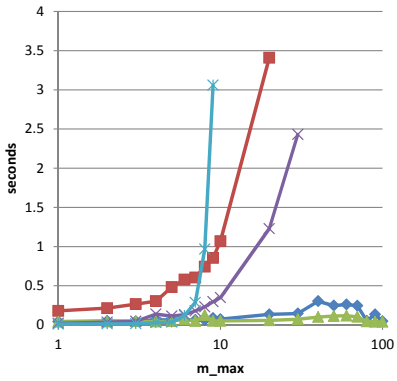
11. Note that we test each question in T with equal probability, so memorizing as much as possible from $T \cap H_\theta$ is a best-response.
12. By complementary slackness, this also means they constitute optimal primal and dual solutions to our LPs.



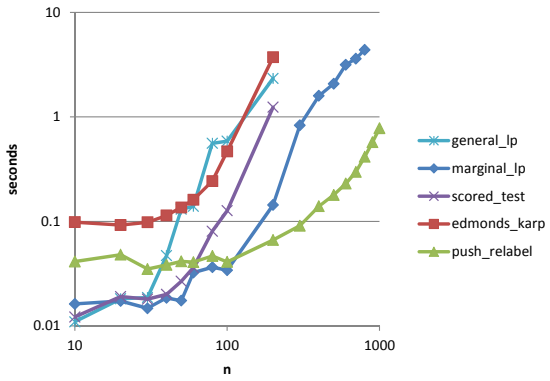
(a) Increasing n with $L = 150$ and $m_{\max} = 6$.



(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 for an optimal tester strategy in one-question tests.

binary test algorithms, we assume that the score of each question is $s_q = 1$ throughout this section. Hence, a binary test game instance is translated to a scored one by setting $s_q = 1$ without changing any other parameters.

6.1 Tests of Size 1: Scalability

We first restrict our attention to single-question test games, in which we can evaluate all our algorithms at once. We consider five different algorithms. We use CPLEX (out-of-the-box) to solve (I) the general LP (2) for binary tests, (II) the one-question marginal-probability-of-memorizing LP (4) for binary tests, and (III) the scored test LP (1) with constraint generation.[13] Also, we use the network-flow approach from Definition 4 with binary search on U to a precision of 10^{-8} using (IV) Edmonds-Karp (Edmonds & Karp, 1972) and (V) Push-Relabel (Goldberg & Tarjan, 1988), in each case combined with Algorithm 1 to compute the tester's optimal strategy for binary tests.

In particular, we use CPLEX 12.6.0.0 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. The scored test algorithm has a spike at $n = 3000$ because it takes more time when t becomes smaller compared to n. In this case, $t = 1$ (single-question tests) becomes relatively smaller as we grow n. In fact, when we change to $t = 2$ for that $n = 3000$ instance, the running time suddenly drops and the spike disappears. Later we will discuss this more in the multi-question experiments.

Edmonds-Karp and the scored test algorithm do not scale very well in L and $m_{\max}$,[14] and the general LP scales particularly poorly in $m_{\max}$. However, the general LP outperforms the scored test algorithm in L. This seems surprising as we argued that scored tests are easier to solve. But when $t = 1$, the general LP (2) is exactly the same as the scored test LP (1). The only difference is that the scored test algorithm uses constraint generation while the general LP feeds all constraints to CPLEX at once. When the number of constraints is not

13. Specifically for (III), we wrote a C++ program that interacts with CPLEX: we first submit an LP with a small number of constraints; whenever CPLEX returns a solution, we detect whether it violates a constraint that we did not submit to CPLEX using the algorithm we described in the proof of Theorem 1; if no, we return the solution; if yes, we add it to the LP and let CPLEX solve it again. In our implementation, the initial number of constraints is $|Q| + |\Theta| + 1$: all $|Q|$ constraints of (1c); the one constraint (1b); and only $|\Theta|$ constraints of (1a), one for each type θ where M_θ is simply the first m_θ elements of H_θ.
14. Although Edmonds-Karp is a polynomial algorithm, its $O(VE^2)$ complexity is quite high for large graphs.

exponential, feeding all constraints at once is faster. That is the case when we scale in n or L.

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.

6.2 Tests with Multiple Questions: Scalability

When the binary test can contain more than one question, the only algorithm that we have provided for binary tests is our general LP (2). We compare this with the standard solver for Bayesian Stackelberg games, DOBSS (Paruchuri et al., 2008), and our scored test algorithm (using the same version of CPLEX).[15] The machine we use and the way we generate test game instances (other than the number of questions) remain the same as in the previous subsection. (Note that the scored test algorithm solves a different problem.) However, since the distribution of the running time is less concentrated, we generate 50 random instances for each case and show the full box plot of its distribution.

Figure 3 shows that DOBSS scales quite poorly in all parameters, especially in $m_{\max}$ and t. As might be expected when solving MIPs, the runtime varies dramatically even for instances of the same size, as indicated by the many outliers in the plots. Our general LP (2), on the other hand, scales much better in n, L. It does still struggle when scaling $m_{\max}$ and t (though it outperforms DOBSS). This is expected because the size of the LP grows exponentially in those parameters. Finally, as expected, the scored test algorithm scales well in all parameters. One interesting phenomenon is that the scored test algorithm runs faster when t increases. We believe that this is due to the constraint generation. As t grows, the number of variables and the total number of constraints do not change. But for greater t, more questions can be covered in a single test. Hence fewer constraints need to be added to reach the solution.

6.3 Tester Utility

One may wonder whether computing a game-theoretically optimal strategy is worth the effort; maybe a simple heuristic performs almost as well. To assess this, we have compared our optimal test strategies to (1) the optimal strategy under the assumption that test takers do not memorize anything, and (2) choosing t questions uniformly at random from all questions. We generated all the test game instances as before except for H_θ. Besides generating H_θ uniformly at random, we also generate structured instances, where the questions $q_1, q_2, \ldots$ are sorted by difficulty (starting with the most difficult), and a type is defined by how deep into the list it needs to go before being able to answer questions. That is, we have $H_\theta = \{q_1, q_2, \ldots, q_{|H_\theta|}\}$ for every θ. In all cases, (1) performed exceedingly poorly, many

15. For both DOBSS and our LPs, we generate the LP/MILP using GLPK in CPLEX LP format, and let CPLEX solve it using a single "optimize" command without any initialization.

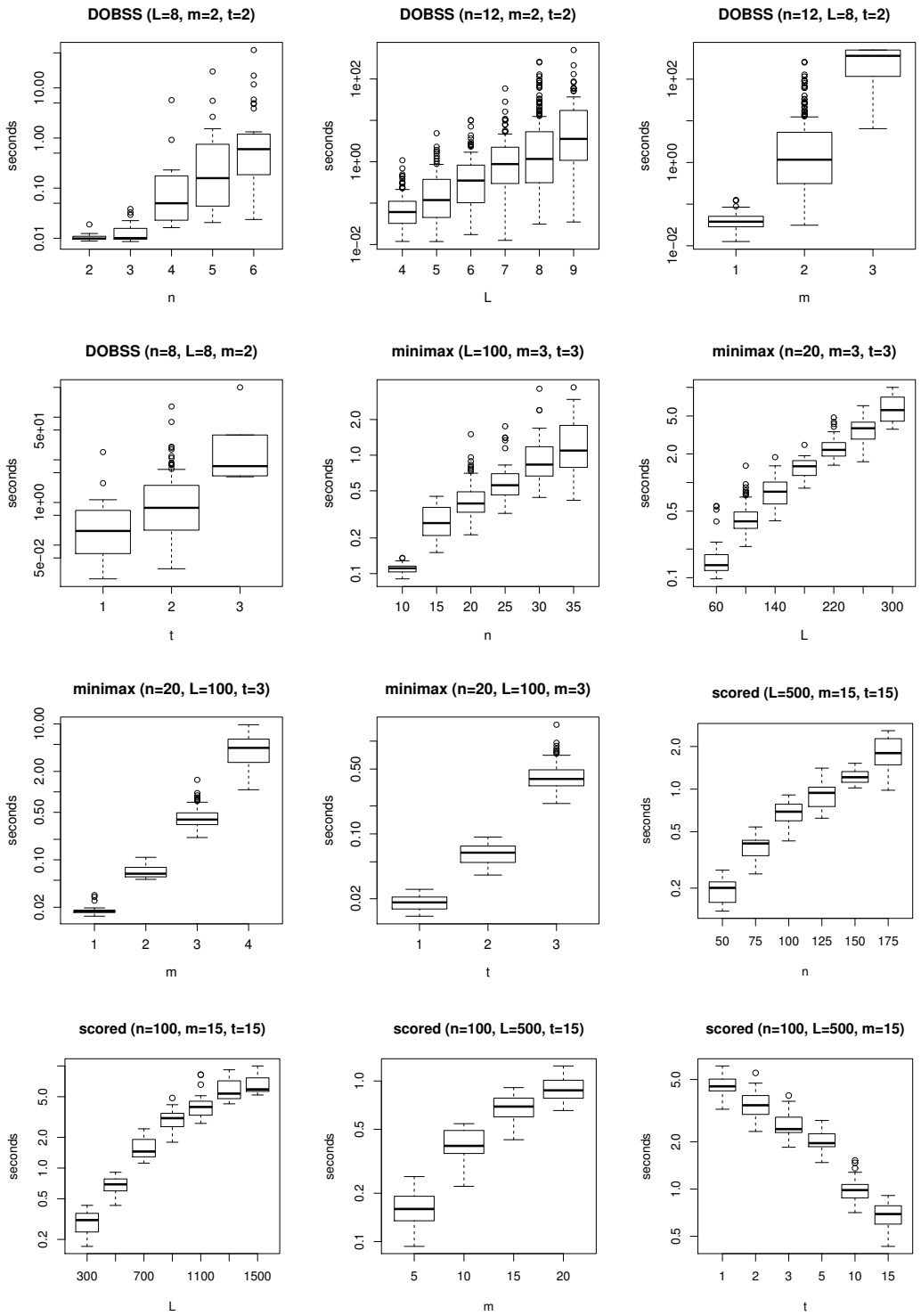


Figure 3: Running time for solving for an optimal tester strategy in multi-question tests. For brevity, the parameter $m_{\max}$ is denoted as m in the title of each graph. The minimax algorithm in the second and the third row refers to LP (2).

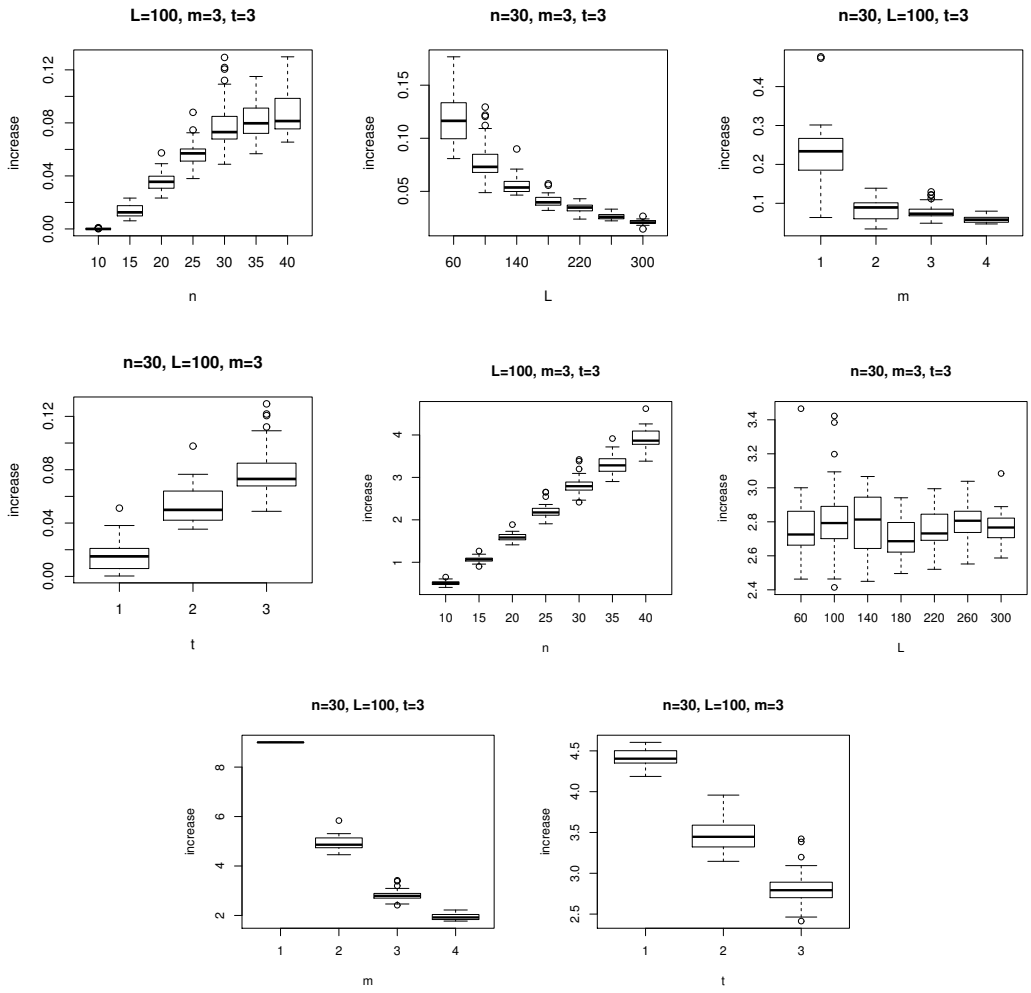


Figure 4: The increase in tester utility that the optimal test strategy gives as compared to the uniform test strategy in *binary* tests. The hard question sets H_θ in the first 4 plots are sampled uniformly at random. In the last 4 plots, the questions are sorted by difficulty so H_θ is always $\{q_1, q_2, \ldots, q_{|H_\theta|}\}$. Letting the optimal utility be u^* and the uniform test's utility be u_0, the number we show for the performance increase is $u^*/u_0 - 1$. For brevity, the parameter $m_{\max}$ is denoted as m in the title of each graph.

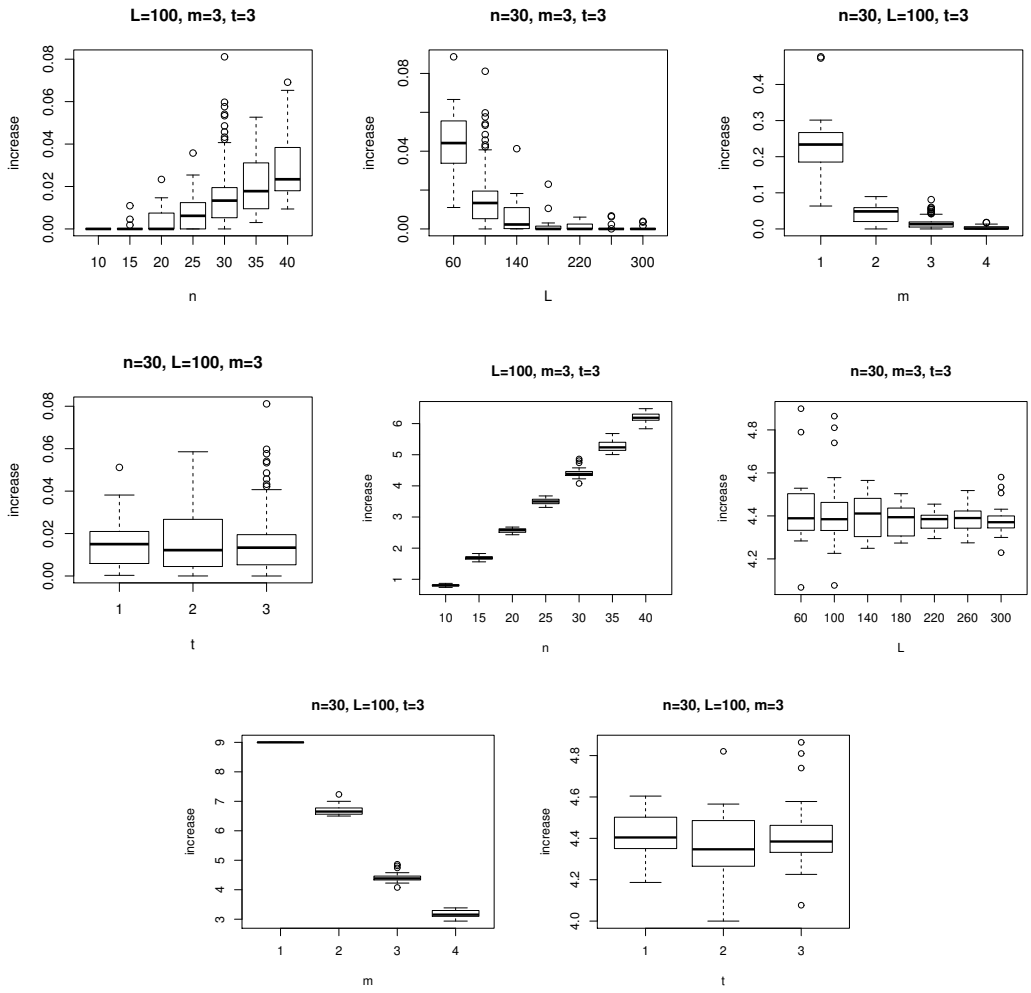


Figure 5: The increase in tester utility that the optimal test strategy gives as compared to the uniform test strategy in *scored* tests. The hard question set H_θ in the first 4 plots are sampled uniformly at random. In the last 4 plots, the questions are sorted by difficulty so H_θ is always $\{q_1, q_2, \ldots, q_{|H_\theta|}\}$. Letting the optimal utility be u^* and the uniform test's utility be u_0, the number we show for the performance increase is $u^*/u_0 - 1$. For brevity, the parameter $m_{\max}$ is denoted as m in the title of each graph.

	constant $m_{\max}$	non-constant $m_{\max}$
$t = 0, 1$	P	P
constant $t \geq 2$	P	coNP-c
non-constant t	NP-c	coNP-h and NP-h

Table 2: Optimal Binary Test Strategy's complexity ($t, m_{\max}$ are the test size and max memory size respectively)

factors worse than the optimal strategy; (2) performed decently and so we only present results comparing to (2).

When H_θ is drawn uniformly at random (the first 4 plots of Figure 4 and 5), the performance increase drops when L and m increase, and it increases when n and t increase. This is a natural consequence of the way we generate the test game instances. The larger L and m are, the more equivalent subsets of questions are likely to be as the hardness sets start to cover the questions more uniformly; on the other hand, the opposite is true for n (for example, with large n, some questions may not be hard for any type and so a waste to test) and t (presumably because as t grows it becomes more important to pick questions that expose weaknesses of different types). All in all, the performance of the uniform strategy is quite decent when H_θ is drawn uniformly at random.

We suspect, though, that the way we construct those uniform-at-random test game instances, which assumes that each question has the same probability to be hard for a test taker, favors the uniform strategy. In reality, we expect there to be some correlation—that is, a question that is hard for one type is more likely to be hard for another type as well—and presumably the uniform strategy is less effective in this context.

For example, in the last 4 plots of Figure 4 and 5 when the questions can be sorted by difficulty (so H_θ is structured), the optimal strategy significantly outperforms the uniform strategy. The performance increase varies according to n, L, m similarly to the uniform-at-random cases, but not for t. As t increases, it becomes much easier for the uniform strategy to catch test takers in a binary test, hence the performance increase drops.

7. Conclusion

In this paper, we proposed two general classes of test games: scored tests and binary tests. Our goal is to compute the optimal test strategies when confidentiality might be lost. The optimal scored test strategies can be efficiently computed in polynomial time while the optimal binary test strategies are generally harder to compute. Table 2 summarizes our complexity results for binary tests.

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. 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.

Acknowledgments

We thank ARO and NSF for support under grants W911NF-12-1-0550, W911NF-11-1-0332, IIS-1527434, IIS-0953756, CCF-1101659, and CCF-1337215.

References

Birkhoff, G. (1946). Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, *5*, 147–151.

Chang, C.-S., Chen, W.-J., & Huang, H.-Y. (2001). Coherent Cooperation Among Communicating Problem Solvers. *IEEE Transactions on Communications*, *49*(7), 1145–1147.

Chen, X., Deng, X., & Teng, S.-H. (2009). Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM*, *56*(3).

Conitzer, V., & Sandholm, T. (2006). Computing the Optimal Strategy to Commit to. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pp. 82–90 Ann Arbor, MI, USA.

Conitzer, V., & Sandholm, T. (2008). New Complexity Results about Nash Equilibria. *Games and Economic Behavior*, *63*(2), 621–641.

Cooper, S., & Sahami, M. (2013). Reflections on Stanford's MOOCs. *Communications of the ACM*, *56*(2), 28–30.

Daskalakis, C., Goldberg, P., & Papadimitriou, C. H. (2009). The Complexity of Computing a Nash Equilibrium. *SIAM Journal on Computing*, *39*(1), 195–259.

Dulmage, L., & Halperin, I. (1955). On a theorem of Frobenius-Konig and J. von Neumann's game of hide and seek. *Trans. Roy. Soc. Canada III*, *49*, 23–29.

Edmonds, J., & Karp, R. M. (1972). Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *Journal of the ACM*, *19*(2), 248–264.

Fang, F., Jiang, A. X., & Tambe, M. (2013). Protecting moving targets with multiple mobile resources. *Journal of Artificial Intelligence Research*, *48*, 583–634.

Gilboa, I., & Zemel, E. (1989). Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, *1*, 80–93.

Goldberg, A. V., & Tarjan, R. E. (1988). A new approach to the maximum-flow problem. *Journal of the ACM*, *35*(4), 921–940.

Grötschel, M., Lovász, L., & Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, *1*(2), 169–197.

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

Hew, K. F., & Cheung, W. S. (2014). Students and instructors use of massive open online courses (MOOCs): Motivations and challenges. *Educational Research Review*, *12*, 45–58.

Jain, M., Kiekintveld, C., & Tambe, M. (2011). Quality-bounded Solutions for Finite Bayesian Stackelberg Games: Scaling up. In *Proceedings of the Tenth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 997–1004 Taipei, Taiwan.

Jain, M., Pita, J., Tambe, M., Ordóñez, F., Paruchuri, P., & Kraus, S. (2008). Bayesian Stackelberg games and their application for security at Los Angeles International Airport. *SIGecom Exch.*, *7*(2), 1–3.

Khachiyan, L. (1979). A polynomial algorithm in linear programming. *Soviet Math. Doklady*, *20*, 191–194.

Kiekintveld, C., Jain, M., Tsai, J., Pita, J., Ordóñez, F., & Tambe, M. (2009). Computing Optimal Randomized Resource Allocations for Massive Security Games. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 689–696 Budapest, Hungary.

Korzhyk, D., Conitzer, V., & Parr, R. (2010). Complexity of Computing Optimal Stackelberg Strategies in Security Resource Allocation Games. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 805–810 Atlanta, GA, USA.

Letchford, J., & Conitzer, V. (2013). Solving Security Games on Graphs via Marginal Probabilities. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pp. 591–597 Bellevue, WA, USA.

Letchford, J., Conitzer, V., & Munagala, K. (2009). Learning and Approximating the Optimal Strategy to Commit to. In *Proceedings of the Second Symposium on Algorithmic Game Theory (SAGT-09)*, pp. 250–262 Paphos, Cyprus.

Li, Y., & Conitzer, V. (2013). Game-Theoretic Question Selection for Tests. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 254–262 Beijing, China.

Paruchuri, P., Pearce, J. P., Marecki, J., Tambe, M., Ordóñez, F., & Kraus, S. (2008). 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)*, pp. 895–902 Estoril, Portugal.

Pita, J., Jain, M., Ordóñez, F., Portway, C., Tambe, M., & Western, C. (2009). Using game theory for Los Angeles airport security. *AI Magazine*, *30*(1), 43–57.

Reich, J., et al. (2015). Rebooting MOOC research. *Science*, *347*(6217), 34–35.

Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., & Meyer, G. (2012). PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States. In *Proceedings of the Eleventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* Valencia, Spain.

Tsai, J., Rathi, S., Kiekintveld, C., Ordóñez, F., & Tambe, M. (2009). IRIS - A Tool for Strategic Security Allocation in Transportation Networks. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 37–44 Budapest, Hungary.

Vanderbei, R. (2001). *Linear Programming: Foundations and Extensions* (2 edition). Springer.

von Neumann, J. (1928). Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, *100*, 295–320.

von Stengel, B., & Zamir, S. (2010). Leadership Games with Convex Strategy Sets. *Games and Economic Behavior*, *69*, 446–457.

Xu, H., Fang, F., Jiang, A., Conitzer, V., Dughmi, S., & Tambe, M. (2014). Solving Zero-Sum Security Games in Discretized Spatio-Temporal Domains. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 1500–1506 Québec City, Québec, Canada.

Yin, Z., Jiang, A., Johnson, M., Tambe, M., Kiekintveld, C., Leyton-Brown, K., Sandholm, T., & Sullivan, J. (2012). Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *Proc. of the 24th Conference on Innovative Applications of Artificial Intelligence, IAAI*.