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. $p : \Theta \to [0, 1]$ 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.

[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.

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 function for the test taker: $u_2'(\theta, T, M_\theta) = 0$ if $T \cap (H_\theta \setminus M_\theta) \neq \emptyset$, and $u_2'(\theta, T, M_\theta) = v_\theta$ otherwise.

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

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

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

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

4 General Linear Program (LP) Formulation

General 2-player Bayes Stackelberg games are NP-hard in terms of game matrix size so previous work used mixed integer LP (e.g. DOBSS [cite]) to solve them. We model test games as zero-sum so we can bypass the NP-hardness and use the following maximin LP to solve our 2-player zero-sum Bayes Stackelberg games in polynomial time with respect to the game matrix size:

$$\begin{aligned} \max \ & \sum_l p_l V_l \qquad (1) \\ s.t. \ & (\forall l, \forall c_l \in C_l), \\ & \sum_r x_r u^l(r, c_l) \geq V_l \\ & \sum_r x_r = 1 \\ & (\forall r \in R) x_r \geq 0 \end{aligned}$$

where

- V_l: utility that the test taker can get from type l test takers
- C_l: action space of type l test takers, i.e. all combinations of problems that they can memorize.
- R: action space of the tester, i.e. all combinations of problems that he can test.
- x_r: probability to test problem set r

The dual of the above LP is:

$$\begin{aligned} \min \ & U \qquad (2) \\ s.t. & (\forall r) U \geq \sum_{l, c_l} u^l(r, c_l) y_{l, c_l} \\ & (\forall l) \sum_{c_l} y_{l, c_l} = p_l \\ & (\forall l, c_l) y_{l, c_l} \geq 0 \end{aligned}$$

which is equivalent to

$$\begin{aligned} \min \ & \max_r (U_r) \qquad (3) \\ s.t. & (\forall r) U_r = \sum_{l, c_l} u^l(r, c_l) y_{l, c_l} \\ & (\forall l) \sum_{c_l} y_{l, c_l} = p_l \\ & (\forall l, c_l) y_{l, c_l} \geq 0 \end{aligned}$$

The dual LP is as if that all types of test takers share the same goal to lower the tester's best testing utility ($\max_r(U_r)$ where U_r is the utility of testing r) as much as possible by memorizing problems and revealling their memorizing strategies y_{l,c_l} to the tester. This also shows that a Stackelberg strategy is also a Nash equilibrium if test takers play the dual minimax strategy.

However, even if the LPs above are in polynomial size of the game matrix size (or the action space size), they are exponential to the input of our test games: the tester's action space is exponential to t (the number of problems to test) and the type-l test taker's action space is exponential to m_l (the number of problems he can memorize). So those LPs only give us polynomial algorithms when m_l, t are constant. Our next question is whether the test game in general can be solved in polynomial time in terms of the input size.

4.1 Hardness on Test Size

In order to formally characterize *test game*'s hardness, define

Definition 1 (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 Stackelberg strategy with at least u utility.*

It's easy to show

Theorem 1. *Even if test takers cannot memorize any problems ($m_l = 0$),* OPTIMAL TEST STRATEGY *is NP-hard when test size t is non-constant.*

Proof. Reduce a VERTEX COVER instance to a OPTIMAL TEST STRATEGY instance as follows. Given a graph $G = (V, E)$, construct a test game G' with problem set $A = V$. For each edge $e = \{i, j\} \in E$, add one type of test takers l_e whose unsolvable problem set $B_{l_e} = e$. Let $p_{l_e} = 1/|E|, v_{l_e} = 1$ and $m_{l_e} = 0$ for all l_e. Graph G has a vertex cover of size k if and only if the tester has a strategy to test $t = k$ problems and gets $u = 1$ utility (i.e. all test takers will fail for sure) in test game G'. □

4.2 Hardness on Memorize Capacity

Theorem 2. *Even if test size is* 2, OPTIMAL TEST STRATEGY *is coNP-hard when memorize capacity is non-constant.*

Proof. We reduce INDEPENDENT SET instances to test games with test size $t = 2$ and ask the complementary question: is u the highest utility that the tester can get. That question has a yes answer if and only if the dual minimax LP [reference] has a feasible solution with objective at most u. For a graph $G = (V, E)$, let V be our test problem set A. Add $L = |E| + |V| + 2$ types of test takers as follows:

- For each edge $e = (i, j) \in E$, add one type of test takers l_e with $v_{l_e} = 1$, $B_{l_e} = \{i, j\}$ and $m_{l_e} = 0$.
- For each vertex i, add one type of test takers l_i with $v_{l_i} = d_G - d(i), B_{l_i} = \{i\}$ and $m_{l_i} = 0$. Here d_G is the maximum vertex degree in G and $d(i)$ is the degree of vertex i.
- Add one auxiliary type of test takers l_α with $v_{l_\alpha} = \alpha\varepsilon, m_{l_\alpha} = 2$ and $B_{l_\alpha} = V$ where $\alpha = \binom{|V|}{2} - |E| - \binom{k}{2}$ and $\varepsilon \ll 1$.
- Finally, add one type of test takers l_k with $v_{l_k} = \varepsilon, B_{l_k} = V$ and $m_{l_k} = k$.

Let all types of test takers occur with uniform probability $p = 1/L$. Then we show that deciding whether $u = (2d_G + a \cdot \varepsilon)/L$ is an upper-bound for the tester's utility, or equivalently whether we can find a dual solution with objective as low as $u = (2d_G + a \cdot \varepsilon)/L$ in our dual minimax LP, is equivalent to finding a size-k independent set in G. Recall that a dual solution is a strategy for test takers to memorize problems. If no one memorizes, the tester's utility U_r to test a problem set r is $U_r = (2d_G + a \cdot \varepsilon + \varepsilon)/L$ if $r \notin E$ and $U_r = (2d_G - 1 + a \cdot \varepsilon + \varepsilon)/L$ if $r \in E$. To lower the objective, test takers l_α, l_k should memorize problem sets that bring the maximum of U_r down. No matter what the strategy is (as long as they only memorize unsolvable problems), $\sum_r U_r$ is a constant. And because U_r for $r \notin E$ is much larger than U_r for $r \in E$ as $\varepsilon \ll 1$, it's better to only decrease U_r for $r \notin E$. The only way to do that is to let l_α test takers only memorize pairs of problems that are not edges and let l_k test takers only memorize size-k indepedent sets of problems. If there's a size-k independent set, let l_k test takers memorize one size-k independent set all the time and let l_α test takers memorize all pairs of problems that are not covered by that independent set with uniform probability. That will make $U_r = (2d_G + a \cdot \varepsilon)/L$ for $r \notin E$ and $U_r = (2d_G - 1 + a \cdot \varepsilon + \varepsilon)/L$ for $r \in E$. This is the best they can achieve and this can only be achieved when a size-k independent set exists in G.

□

4.3 Summary on General Test Games

Theorem 3. *The* OPTIMAL TEST STRATEGY *for a* test game *can be computed in polynomial time when test size and memorize capacity are constant. It's NP-complete when test size is non-constant and coNP-complete when memorize capacity is non-constant. Hence it's both NP-hard and coNP-hard when both test size and memorize capacity are non-constant.*

Proof. When test size and memorize capacity are both non-constant, OPTIMAL TEST STRATEGY is both NP-hard and coNP-hard by Theorem 1 and 2. LP (1) shows that OPTIMAL TEST STRATEGY can be computed in polynomial time when test size and memorize capacity are constant. When memorize capacity is constant or test size is constant, either the primal LP (1) or the dual LP (3) has constant number of constraint. Such LP has a polynomial size optimal solution even if the number of variables is exponential. That optimal solution's feasibility and objective can be checked in polynomial time. Therefore OPTIMAL TEST STRATEGY is in NP when memorize capacity is constant and it is in coNP when test size is constant. Hence they are NP-complete and coNP-complete respectively by Theorem 1 and 2. □

[HARDNESS TABLE]

5 One-Problem Test Game

We showed the hardness of test games when either one of test size or memorize capacity is non-constant. When the test size is non-constant, there is little we can improve because it is NP-hard even if the memorize capacity is zero. When the memorize capacity is non-constant, our proof for Theorem 2 only showed that NP-hardness for testing 2 problems. Therefore an efficient algorithm to compute optimal one-problem test may exist. A quick evidence for the existence of such polynomial algorithm is constraint generation. Though there are exponential many constraints, we can find the most violated constraint in polynomial time by: enumerate over all types of test takers; for type-l test takers select top m_l problems M_l according to x_a that maximize $\sum_{a \in M_l} x_a$; finally see whether $V_l - \sum_{a \in B_l \setminus M_l} x_a$ updates the most violated constraint.

We also investigated such tests by conducting experiments using LP (1). Surprisingly, no matter how input varies, the strategy we get always has the following format: among a subset T of problems A, test any one of them with uniform probability. That is counter-intuitive as some problems in T seem to be obviously more superior than others so intuitively they should be tested with more probability. For example, when n problems can be sorted by difficulty, a test taker who can solve a harder problem can always solve an easier problem. In that case, it seems that the hardest problem should be tested strictly more likely than the second hardest problem. But in most cases, the optimal strategy will test one of them with equal probability.

[EXAMPLE TABLE]

Inspired by that uniform test property, we present an algorithm to compute optimal strategies and use it to prove uniform test property consequently. Let U_a^0 be the utility of testing a without memorizing: $U_a^0 = \sum_{B_l \ni a} v_l p_l$. The algo-

rithm's first part is based on the following LP:

$$\begin{aligned} &\min\ U \\ &s.t.\ (\forall a \in A)\ U \geq U_a^0 - \sum_{B_l \ni a} p_l v_l z_{l,a} \\ &(\forall l)\ \sum_{a \in B_l} z_{l,a} \leq m_l \\ &(\forall l, a)\ 0 \leq z_{l,a} \leq 1 \end{aligned} \tag{4}$$

It modifies LP (2) by:

1. Change joint probability y_{l,c_l} to marginal probability $z_{l,a}$ of memorizing one problem a. We did this modification because marginal probability is more meaningful for one-problem test and we can always restore a joint probability from a marginal probability[reference].
2. Relax equality $\sum_{a \in B_l} z_{l,a} = m_l$ to inequality $\sum_{a \in B_l} z_{l,a} \leq m_l$ (allow using less memorize capacity). We can do this because $v_l \geq 0$ (thus $u^l, V_l \geq 0$) and this modification is essentially adding $V_l \geq 0$ to the primal LP (1). Intuitively, memorizing less problems won't hurt tester's utility.

Our algorithm is shown in Algorithim 1.

Algorithm 1 Compute the optimal one-problem test strategy

```
1: function OptimalOneProblemTest(A, 𝔹, m, v, p)
2:     Solve LP (4) and let the solution be U, z_{l,a}
3:     T ← {a | U_a^0 ≥ U}
4:     Q ← all B_l such that Σ_{a∈B_l∩T} z_{l,a} < m_l
5:     for all B_l ∈ Q do        ▷ Potential Memorizing Power
6:         G(B_l) ← m_l − Σ_{a∈B_l∩T} z_{l,a}
7:     end for
8:     while Q has an unmarked element do
9:         B_l ← get and mark an unmarked element from Q
10:        for all a ∈ B_l ∩ T and z_{l,a} < 1 do
11:            T ← T\{a}
12:            for all B_{l'} ∋ a and z_{l',a} > 0 do
13:                G(B_{l'}) ← min(z_{l',a}, G(B_l)/(|B_l|L))
14:                Push B_{l'} into Q
15:            end for
16:        end for
17:    end while
18:    return Strategy that test T uniformly
19: 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 11 of Algorithm 1. Denote that initial T as T_0. If all problems in T_0 are deleted, then for every a in T_0 there is a B_l that is in Q and $z_{l,a} < 1$. Note that for every B_l in Q, it has a positive potential memorizing power $G(B_l)$ either because $\sum_{a \in B_l} z_{l,a} < m_l$ (on line 6) or because there is an unnecessary memorizing $z_{l,a} > 0$ that can be freed (on line 13). This means that we can bring U down by using those potential memorizing powers to memorize every problem in T_0 more (recall that $z_{l,a} < 1$), a contradiction to that U is minimized. □

Lemma 2. *Let $z_{l,a}$ be the solution of LP (4) and T be the final subset that Algorithm 1 returns. Then for each type of test takers B_l, either $\sum_{a \in B_l \cap T} z_{l,a} = m_l$ (they memorize a subset of T that are unsolvable all the time) or $\forall a \in B_l \cap T, z_{l,a} = 1$ (they memorize unsolvable problems in T all the time so they will pass the test for sure).*

Proof. Let T_0 be the initial T before any deletion on line 11 of Algorithm 1. Initially, for any B_l such that $\sum_{a \in B_l capT_0} z_{l,a} < m_l$, we push it to Q. After that any $a \in B_l$ where $z_{l,a} < 1$ is deleted from T on line 11. Therefore $\forall a \in B_l \cap T, z_{l,a} = 1$ must hold for that B_l. Afterwards, more B_l may violate $\sum_{a \in B_l capT} z_{l,a} < m_l$ because some a are deleted from T. But for those B_l, we must have pushed them into Q on line 14. So they will satisfy $\forall a \in B_l \cap T, z_{l,a} = 1$ afterwards. In the end, every $B_l \in Q$ will satisfy $\forall a \in B_l \cap T, z_{l,a} = 1$ and every other $B_l \notin Q$ will satisfy $\sum_{a \in B_l \cap T} z_{l,a} = m_l$. □

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).[2] We will need two more lemmas as follows to use the complementary slackness theorem.

Theorem 4. *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 $z_{l,a}$ makes U_a (the utility to test a problem a) decrease to U if $U_a^0 > U$ and we only pick T from problems that $U_a^0 > U$, 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 a subset in T all the time which is optimal for uniform test, the second condition of complementary slackness theorem holds. □

Finally, instead of using general LP algorithm like simplex [cite] to solve LP (4), 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 $O((n+L)^3)$

[2] 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.

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).

Algorithm 2 Use binary search and max flow to solve LP (4)

$U_{lower} \leftarrow 0,\ U_{upper} \leftarrow \max_{a \in A} U_a^0$
while $U_{upper} - U_{lower} > \epsilon$ **do** ▷ Binary search
  $U \leftarrow (U_{upper} + U_{lower})/2$ ▷ Objective utility
  $V \leftarrow \{s\} \cup \mathbb{B} \cup A \cup \{t\}$ ▷ Vertex set
  $E \leftarrow \emptyset$ ▷ Initialize edge set
  for all $a \in A$ **do**
    if $U_a^0 > U$ **then**
      $E \leftarrow E \cup \{(a,t)\}$
      $c_{(a,t)} \leftarrow U_a^0 - U$ ▷ Edge capacity
    end if
  end for
  for all $B_l \in \mathbb{B}$ **do**
    $E \leftarrow E \cup \{B_l\} \times B_l$
    $c_{\{B_l\} \times B_l} \leftarrow p_l v_l$
    $E \leftarrow E \cup \{(s, B_l)\}$
    $c_{(s,B_l)} \leftarrow p_l m_l v_l$
  end for
  $f \leftarrow \textsc{MaxFlow}(G = (V, E), c)$
  if f saturates all edges in $A \times \{t\}$ **then**
    $U_{upper} \leftarrow U$
  else
    $U_{lower} \leftarrow U$
  end if
end while
$(\forall l, a), z_{l,a} \leftarrow f_{(B_l,a)}/c_{(B_l,a)}$
return $U, z_{l,a}$

6 Experiment

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 (4). The third and fourth use Algorithm 2 to solve LP (4). They are using Edmonds-Karp [cite] and Psh-Relabel[3] [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 $1 \le l \le L$, 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.

[3]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).

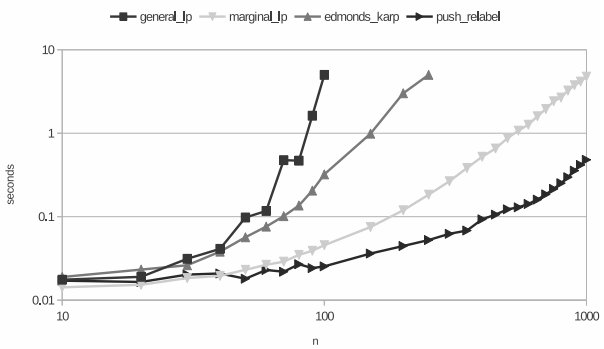

Table 1: Parameter settings for the benchmark

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 (4) 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]

7 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

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