Identifying an Honest $\text{EXP}^{\text{NP}}$ Oracle Among Many

Shuichi Hirahara

The University of Tokyo
Overview

Our Contributions

1. We formulate the notion of selector.
   ➢ ∃ Selector ⇔ Able to remove short advice
2. We prove the existence of a selector for EXP^{NP} - complete languages.
Background: Instance checker

- Introduced by Blum & Kannan (1989).
- An instance checker for a function $f$ checks if a given oracle correctly computes $f(x)$ on input $x$ in polynomial time.

<table>
<thead>
<tr>
<th>Instance checker for $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries</td>
</tr>
<tr>
<td>Given: input $x$</td>
</tr>
</tbody>
</table>
| Is the program buggy or correct on $x$? | | }
Background: Instance checker

✓ There exist instance checkers for $P^{\#P}$-, $PSPACE$-, $\text{EXP}$-complete languages.  
   [LFKN92, Sha92, BFL91]

✓ Any languages with an instance checker must be in $\text{NEXP} \cap \text{coNEXP}$.
   (Note: $\text{NEXP} \subseteq \text{EXP}^{\text{NP}}$)

Instance checker for $f$

Queries  

Answers

Possibly buggy program  
(modeled by black-box access to an oracle)

Is the program buggy or correct on $x$?
(Probabilistic) Selector for SAT

Given: 1. An input $\varphi$, and
2. access to two oracles
one of which is honest.

Task:
compute $\text{SAT}(\varphi)$
with the help of the oracles

Given: input $\varphi$
(Probabilistic) Selector for SAT

Given:
1. An input $\varphi$, and
2. access to two oracles one of which is honest.

Task:
compute $\text{SAT}(\varphi)$ with the help of the oracles

Queries:
Is $\psi$ satisfiable?

Given: input $\varphi$

Selector for SAT

Is $\varphi$ satisfiable?
(Probabilistic) Selector for SAT

**Given:**
1. An input \( \varphi \), and
2. access to two oracles one of which is honest.

**Task:**
compute \( \text{SAT}(\varphi) \) with the help of the oracles

**Dishonest oracle**
- \( \psi \) is not satisfiable!
- Queries:
  - Is \( \psi \) satisfiable?
  - Arbitrary answers

**Honest oracle**
- \( \psi \) is satisfiable!
- Correct answers

**Is \( \varphi \) satisfiable?**

**Given:** input \( \varphi \)
Why is it called “selector”?

Given:
1. An input $\varphi$, and
2. access to two oracles one of which is honest.

Task:
compute $\text{SAT}(\varphi)$ with the help of the oracles

Dishonest oracle

Honest oracle

Is $\varphi$ satisfiable?

Correct answers

Arbitrary answers

Given: input $\varphi$

Selector for SAT
Why is it called “selector”?

Given:
1. An input $\varphi$, and
2. access to two oracles one of which is honest.

Task:
compute $\text{SAT}(\varphi)$ with the help of the oracles

Is $\varphi$ satisfiable?

Given: input $\varphi$

Selector for SAT

The answer must be YES!!
Why is it called “selector”?

Given:
1. An input $\phi$, and
2. access to two oracles one of which is honest.

Essential Task:
determine which is honest when they disagree on $\phi$.

Dishonest oracle

Honest oracle

Is $\phi$ satisfiable?

Arbitrary answers

Correct answers

Given: input $\phi$

Selector for SAT

Which is honest?
Definition of (Probabilistic) Selector

A selector $S$ for a language $L$ is a polynomial-time probabilistic oracle machine such that

$$A_0 = L \text{ or } A_1 = L \implies \Pr[S^{A_0,A_1}(x) = L(x)] \geq 0.99$$

(for any $A_0, A_1 \subseteq \{0,1\}^*, x \in \{0,1\}^*$)

Selector $S$ for a language $L$
(polynomial-time probabilistic oracle TM)
Definition of **Deterministic Selector**

**Definition (Deterministic Selector)**

A **deterministic selector** $S$ for a language $L$ is a polynomial-time deterministic oracle machine such that

$$A_0 = L \text{ or } A_1 = L \implies S^{A_0, A_1}(x) = L(x)$$

(for any $A_0, A_1 \subseteq \{0,1\}^*, x \in \{0,1\}^*$)

---

**Oracle $A_0$**

**Arbitrary answers**

**Queries:**

$L(q) = ?$

**Oracle $A_1$**

**Correct answers (YES/NO)**

**Selector $S$ for a language $L$**

(polynomial-time deterministic oracle TM)
Selector for $\mathbb{P}^{\mathbb{NP}}$-complete languages (1/2)

Def. (Lexicographically Maximum Satisfying Assignment)

**Input**: a Boolean formula $\varphi: \{0, 1\}^n \rightarrow \{0, 1\}$ and an index $k$

**Output**: the $k^{th}$ bit of the lexicographically maximum satisfying assignment of $\varphi$.

**Goal**: to construct a deterministic selector for this language.

**Given**: an input $(\varphi, k)$ and two oracles 🤔 🤔.
Selector for \(P^{NP}\)-complete languages (2/2)

- Make queries \((\varphi, 1), \ldots, (\varphi, n)\) to the oracles:
  
  The lexicographically maximum satisfying assignment of \(\varphi\) is...

- If \(v_0 = v_1\), then output the \(k^{th}\) bit of \(v_0 (= v_1)\).
- Else, we have \(v_0 \neq v_1\). We assume \(v_0 < v_1\).
  Evaluate \(\varphi(v_1)\).
- We trust \(\varnothing\) if \(\varphi(v_1) = 1\) and trust \(\varnothing\) otherwise.

(i.e. output the \(k^{th}\) bit of \(v_1\))

(i.e. output the \(k^{th}\) bit of \(v_0\))

Q.E.D.
Identifying an Honest $\text{EXP}^\text{NP}$ oracle

**Theorem (Main Result)**
There exists a selector for $\text{EXP}^\text{NP}$-complete languages.

**Proof sketch:**

**Def. (An $\text{EXP}^\text{NP}$-complete language)**

*Input:* a succinctly described Boolean formula $\Phi: \{0, 1\}^{2^n} \rightarrow \{0, 1\}$ and an index $k$

*Output:* the $k^{\text{th}}$ bit of the lexicographically maximum satisfying assignment of $\Phi$. 
Proof Sketch of the Main Theorem

- Proof strategy: the same with $\mathsf{P}^{\mathsf{NP}}$-complete languages

We are given a (succinctly described) exponential-sized formula $\Phi$ and two oracles $V_0 \in \{0, 1\}^{2n}$, $V_1 \in \{0, 1\}^{2n}$.

**Step 1:** Which is the larger? (i.e. compute $V_0 < V_1$)

- Binary search & Polynomial identity testing

**Step 2:** Is $V_1$ a satisfying assignment of $\Phi$?

- Can be done in the same way with $\mathsf{MIP} = \mathsf{NEXP}$.

[Babai, Fortnow, Lund (1991)].
Instance Checker vs. Selector

Counterexample: EXP^NP-complete languages (unless EXP^{NP} = \text{NEXP})

✓ The task of selectors is strictly easier than instance checking.
Motivation: Removing short advice

[Karp & Lipton (1980)]
\[ \text{SAT} \in \text{P/log} \implies \text{SAT} \in \text{P} \]

[Trevisan & Vadhan (2002)]
\[ \text{EXP} \subseteq \text{BPP/log} \implies \text{EXP} \subseteq \text{BPP} \]
✓ This follows from the instance checkability of EXP-complete languages.

Q. When can we remove short advice?

A. When we have a selector.
∃ Selector \[\Rightarrow\text{Able to remove 1-bit advice}\]

1. Suppose \(L\) is computable with 1-bit advice.

\(i.e.\) ∃ machine \(M\) such that, given advice “0” or “1”, \(M\) computes \(L\) correctly.

\[M(q, 0) = L(q)\text{ or } M(q, 1) = L(q)\text{ for any } q \in \{0,1\}^{l}\]

2. Define two oracles as follows:

One of these oracles is honest!

\[\Rightarrow \text{The selector can compute } L \text{ correctly} \text{ (without any advice).}\]
Key Lemma: “Among Many”

Identifying an honest oracle among two

Identifying an honest oracle among polynomially many

On input $x$, outputs $L(x)$

✓ Able to remove advice of 1 bit

✓ Able to remove advice of $O(\log n)$ bits
Our Results

➢ The notion of selector provides a general framework to remove short advice:

Thm. (∃ Selector ⇔ Able to remove short advice)

For any paddable language $L$, the following are equivalent:
1. There exists a deterministic selector for $L$.
2. $L \in \mathbf{P}/\log$ implies $L \in \mathbf{P}$ under any relativized world.
   In other words,
   $$L \in \mathbf{P}^R/\log \implies L \in \mathbf{P}^R \quad (\forall R: \text{oracle})$$

✓ The converse direction ($2 \implies 1$) also holds in this sense!
Our Results

- The notion of selector provides a general framework to remove short advice:

<table>
<thead>
<tr>
<th>Thm. (∃ Selector ↔ Able to remove short advice)</th>
</tr>
</thead>
<tbody>
<tr>
<td>For any paddable language $L$, the following are equivalent:</td>
</tr>
<tr>
<td>1. There exists a probabilistic selector for $L$.</td>
</tr>
<tr>
<td>2. $L \in \text{BPP}/\text{log}$ implies $L \in \text{BPP}$ under any relativized world.</td>
</tr>
<tr>
<td>In other words,</td>
</tr>
<tr>
<td>$L \in \text{BPP}^R/\text{log} \implies L \in \text{BPP}^R$ (∀$R$: oracle)</td>
</tr>
</tbody>
</table>

Technical Remark:

This works for any type of advice for BPP, because we can remove the most powerful advice (i.e. BPP//log).
Key Lemma to remove $O(\log n)$ advice

Identifying an honest oracle among **two**

- Able to remove advice of 1 bit

Identifying an honest oracle among **polynomially many**

- Able to remove advice of $O(\log n)$ bits

On input $x$, outputs $L(x)$

(Honest)
Proof of the Key Lemma (1/2)

• We have a selector $\mathcal{S}$ that identifies an honest oracle among two.
• Given input $x$ and many oracles
• Ask them about $x$: $L(x)$ is
• Divide them into two teams: We claim $L(x) = 0$.
  Idea: “Tournament”
• We claim $L(x) = 1$.
  (Honest)
  (Honest)
Proof of the Key Lemma (2/2)

We claim $L(x) = 0$.

(Honest)

We claim $L(x) = 1$.  

(Honest)
Proof of the Key Lemma (2/2)

We claim $L(x) = 0$.

We claim $L(x) = 1$.

(Honest)
Proof of the Key Lemma (2/2)

We claim \( L(x) = 0 \).

(Honest)

We claim \( L(x) = 1 \).

I doubt !!
We claim $L(x) = 0$.

We claim $L(x) = 1$.

(Honest)
Proof of the Key Lemma (2/2)

We claim $L(x) = 0$.

We claim $L(x) = 1$.

(Honest)
We claim $L(x) = 0$.

We claim $L(x) = 1$.

(Honest)
Proof of the Key Lemma (2/2)

We claim \( L(x) = 0 \).

We claim \( L(x) = 1 \).

(Honest)
Proof of the Key Lemma (2/2)

We claim $L(x) = 0$. We claim $L(x) = 1$.

(Honest)
We claim $L(x) = 0$. 

We claim $L(x) = 1$. 

(Honest) 

1 

I doubt 😳!!
Proof of the Key Lemma (2/2)

We claim $L(x) = 0$.

We claim $L(x) = 1$.

(Honest) I trust  , so the answer is 1!

✓ The honest oracle always wins!
Proof of the Key Lemma (summary)

• Pick arbitrary two oracles from each team.

  • Run the selector on input $x$.

  • If outputs 0, then we doubt (the oracle loses!); otherwise we doubt .

• Continue it until one of the teams wins and trust the team.

Q.E.D.
Removing advice of size $O(\log n)$
Removing advice of size $O(\log n)$

Advice “00”

Advice “01”

Advice “10”

Advice “11”

Corollary (Selector $\Rightarrow$ Removing short advice)

Suppose that there exists a selector for a paddable language $L$. Then, $L \in \text{P/}\log$ implies $L \in \text{P}$ (under any relativized world).
∃ selector ⇐ Removing short advice

Claim (∃ Selector ⇐ Able to remove short advice)
Suppose \( L \in \mathbf{P}/\log \) implies \( L \in \mathbf{P} \) under any relativized world. Then, there exists a deterministic selector for \( L \).

- We prove the contraposition:
  - If no selector exists, then for some relativized world, even 1-bit advice cannot be removed:
    \[ \exists R \text{ s.t. } L \in \mathbf{P}^R/1 \& L \notin \mathbf{P}^R \]
  - Constructing such an oracle \( R \) by diagonalization.
  - 1-bit advice can tell which is honest. (i.e. \( L \in \mathbf{P}^R/1 \))
Other Results: Deterministic Selector

Theorem (Deterministic Selector)
1. There exists a deterministic selector for PSPACE-complete languages.
2. Any languages with a deterministic selector sit within PSPACE.

Deterministic selectors are in PSPACE
Other Results: Probabilistic Selector

Theorem (Upper Bound for Probabilistic Selector)

Any languages with a selector sit within $S_2^{exp}$.

SELECTORS ARE IN $S_2^{exp}$
Conclusions

✓ The existence of a selector $\iff$ A property of removing short advice under any relativized world.

✓ There exists a selector for $\text{EXP}^\text{NP}$-complete languages.

  ➢ We can efficiently identify an honest $\text{EXP}^\text{NP}$-complete oracle among many.
Future Work

• Closing the gap between $\text{EXP}^\text{NP}$ and $S_2^{\text{exp}}$
  
  Does there exist a selector for promise-$S_2^{\text{exp}}$-complete languages?

• Does there exist a selector for $\text{NEXP}$-complete languages?
  
  Although $\text{NEXP} \subseteq \text{EXP}^\text{NP}$, the set of languages with a selector is not closed under downward reduction.

• What about removing advice of polynomial size?
  
  e.g. $\{ L \mid \forall R, L \in \text{P}^R / \text{poly} \Rightarrow L \in \text{MA}^R \}$