Tighter Circuit Lower Bounds for MA/1 With Efficient PCPs

Joshua Cook
Joint work with Dana Moshkovitz
Trust Can’t Buy Time***

An Alternate Title
## Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
<th>Expect New Resource To Help Solve Some Problems</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Advice</strong></td>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
<td></td>
</tr>
<tr>
<td><strong>Untrusted, Adaptive</strong></td>
<td>NTIME[T]</td>
<td>MATIME[T]</td>
<td></td>
</tr>
<tr>
<td><strong>Trusted, Unadaptive</strong></td>
<td>SIZE[T]^*</td>
<td>BPTIME[T]/T</td>
<td></td>
</tr>
<tr>
<td><strong>Untrusted, Unadaptive</strong></td>
<td>ONTIME[T]</td>
<td>OMATIME[T]</td>
<td></td>
</tr>
</tbody>
</table>

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of TIME[T]/T
### Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Advice</td>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
</tr>
<tr>
<td>Untrusted, Adaptive</td>
<td>NTIME[T]</td>
<td>MATIME[T]</td>
</tr>
<tr>
<td>Trusted, Unadaptive</td>
<td>SIZE[T]*</td>
<td>BPTIME[T]/T</td>
</tr>
<tr>
<td>Untrusted, Unadaptive</td>
<td>ONTIME[T]</td>
<td>OMATIME[T]</td>
</tr>
</tbody>
</table>

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of TIME[T]/T

Expect New Resource To Help Solve Some Problems

Suspect some problems can’t be sped up with these resources.
# Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Advice</strong></td>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
</tr>
<tr>
<td><strong>Untrusted, Adaptive</strong></td>
<td>NTIME[T]</td>
<td>MATIME[T]</td>
</tr>
<tr>
<td><strong>Trusted, Unadaptive</strong></td>
<td>SIZE[T]*</td>
<td>BPTIME[T]/T</td>
</tr>
<tr>
<td><strong>Untrusted, Unadaptive</strong></td>
<td>ONTIME[T]</td>
<td>OMATIME[T]</td>
</tr>
</tbody>
</table>

Expect New Resource To Help Solve Some Problems

Suspect some problems can’t be sped up with these resources.

TIME\[n^4\] \(\subseteq\) NTIME\[n\]

---

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of TIME\[T]/T

---

Can All Statements Be Verified Faster than Computed?
## Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th>No Advice</th>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Untrusted, Adaptive</th>
<th>NTIME[T]</th>
<th>MATIME[T]</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Trusted, Unadaptive</th>
<th>SIZE[T]'</th>
<th>BPTIME[T]/T</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Untrusted, Unadaptive</th>
<th>ONTIME[T]</th>
<th>OMATIME[T]</th>
</tr>
</thead>
</table>

Expect New Resource To Help Solve Some Problems

Suspect some problems can’t be sped up with these resources.

\[
\text{TIME}[n^4] \subseteq \text{NTIME}[n]
\]

Can All Statements Be Verified Faster than Computed?

\[
\text{TIME}[n^4] \subseteq \text{SIZE}[n]
\]

Can fixed instance sizes be hard coded to faster, short programs?

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of \(\text{TIME}[T]/T\)
### Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Advice</td>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
</tr>
<tr>
<td>Untrusted, Adaptive</td>
<td>NTIME[T]</td>
<td>MATIME[T]</td>
</tr>
<tr>
<td>Trusted, Unadaptive</td>
<td>SIZE[T]*</td>
<td>BPTIME[T]/T</td>
</tr>
<tr>
<td>Untrusted, Unadaptive</td>
<td>ONTIME[T]</td>
<td>OMATIME[T]</td>
</tr>
</tbody>
</table>

Expect New Resource To Help Solve Some Problems

Suspect some problems can’t be sped up with these resources.

\[ \text{TIME}[n^4] \subseteq \text{NTIME}[n] \]

\[ \text{TIME}[n^4] \subseteq \text{SIZE}[n] \]

\[ \text{NTIME}[n^4] \subseteq \text{SIZE}[n] \]

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of TIME[T]/T

---

Can All Statements Be Verified Faster than Computed?

Can fixed instance sizes be hard coded to faster, short programs?

Can any verifiable problem on fixed instance sizes be hard coded into a faster, short program.
### Untrusted Advice Vs Trusted Advice

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Randomized</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No Advice</strong></td>
<td>TIME[T]</td>
<td>BPTIME[T]</td>
</tr>
<tr>
<td><strong>Untrusted, Adaptive</strong></td>
<td>NTIME[T]</td>
<td>MATIME[T]</td>
</tr>
<tr>
<td><strong>Trusted, Unadaptive</strong></td>
<td>SIZE[T]^*</td>
<td>BPTIME[T]/T</td>
</tr>
<tr>
<td><strong>Untrusted, Unadaptive</strong></td>
<td>ONTIME[T]</td>
<td>OMATIME[T]</td>
</tr>
</tbody>
</table>

* Is some gap between circuit size program size. Circuit size is more commonly studied, so used instead of TIME[T]/T

Expect New Resource To Help Solve Some Problems

Suspect some problems can’t be sped up with these resources.

\[
\text{TIME}[n^4] \subseteq \text{NTIME}[n]
\]

\[
\text{TIME}[n^4] \not\subseteq \text{SIZE}[n]
\]

\[
\text{NTIME}[n^4] \subseteq \text{SIZE}[n]
\]

\[
\text{ONTIME}[n^4] \not\subseteq \text{SIZE}[n]
\]

*Can All Statements Be Verified Faster than Computed?*

*Can fixed instance sizes be hard coded to faster, short programs?*

*Can any verifiable problem on fixed instance sizes be hard coded into a faster, short program?*

*Can trusted programs always run faster than untrusted programs?*
Santhanam: \( \forall k > 1: \quad \text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)
Santhanam: $\forall k > 1$: $\text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)]$
Santhanam: \( \forall k > 1: \text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)

Murray-Williams: \( \forall k > 1: \text{MATIME}[n^{ck^2}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)
Santhanam: $\forall k>1: \text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)]$

Murray-Williams: $\forall k>1: \text{MATIME}[n^{ck^2}]/1 \not\subseteq \text{SIZE}[O(n^k)]$
Santhanam: $\forall k>1: \text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)]$

Murray-Williams: $\forall k>1: \text{MATIME}[n^{ck^2}]/1 \not\subseteq \text{SIZE}[O(n^k)]$

Our result: $\exists a>1: \forall k<a: \text{MATIME}[n^{k+o(1)}]/1 \not\subseteq \text{SIZE}[O(n^k)]$
Santhanam: \( \forall k>1: \ MATIME[n^{O(k)}]/1 \not\subseteq SIZE[O(n^k)] \)

Murray-Williams: \( \forall k>1: \ MATIME[n^{ck^2}]/1 \not\subseteq SIZE[O(n^k)] \)

Our result: \( \exists a>1: \forall k<a: \ MATIME[n^{k+o(1)}]/1 \not\subseteq SIZE[O(n^k)] \)
Santhanam: \( \forall k > 1: \text{MATIME}[n^{O(k)}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)

Murray-Williams: \( \forall k > 1: \text{MATIME}[n^{ck^2}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)

Our result: \( \exists a > 1: \forall k < a: \text{MATIME}[n^{k+o(1)}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)
\( \forall k > 1: \text{MATIME}[n^{ak+o(1)}]/1 \not\subseteq \text{SIZE}[O(n^k)] \)

Win-Win if \( a \) is small
There exists randomized programs with one bit of trusted advice and a long, untrusted program advice that cannot be solved much faster with trusted advice.

Santhanam: $\forall k>1: \text{OMATIME}[n^{O(k)}]/1 \not\subseteq \text{BPTIME}[O(n^k)]/O(n^k)$

Murray-Williams: $\forall k>1: \text{OMATIME}[n^{ck^2}]/1 \not\subseteq \text{BPTIME}[O(n^k)]/O(n^k)$

Our result: $\exists a>1: \forall k<a: \text{OMATIME}[n^{k+o(1)}]/1 \not\subseteq \text{BPTIME}[O(n^k)]/O(n^k)$

$\forall k>1: \text{OMATIME}[n^{ak+o(1)}]/1 \not\subseteq \text{BPTIME}[O(n^k)]/O(n^k)$

Win-Win if $a$ is small
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

IVTIME[T]: Arthur time T.
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

IVTIME[T]: Arthur time T.
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

$\text{IVTIME}[T]$: Arthur time $T$. 
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

IVTIME[T]: Arthur time T.
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

$\text{IVTIME}[T]$: Arthur time $T$. 
Interactive Proofs (IPs)?

Untrusted Merlin
Randomized Arthur.

Many Questions and Answers.

$\text{IVTIME}[T]$: Arthur time $T$. 
How powerful is IP?

Shamir 92 proved IP = PSPACE!

\[
\text{SPACE}[n] \subseteq \text{IVTIME}[n^2] \\
\text{IVTIME}[n] \subseteq \text{SPACE}[n]
\]

Prover’s for IP also small space!

Circuit bounds for SPACE apply to IP!
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Main Idea

Use a Circuit as Merlin in IP.

Merlin Gives a Circuit
Arthur Uses it to run IP
Santhanam’s Proof: Lower Bound From IP=PSPACE
Santhanam’s Proof: Lower Bound From IP=PSPACE
Santhanam’s Proof: Lower Bound From IP=PSPACE

PSPACE \subseteq \text{?} P/poly

- PSPACE \nsubseteq \text{SIZE}[n^k] \text{ (PSPACE can search outside } \text{SIZE}[n^k]).
- PSPACE=MA \text{ (MA guesses prover circuit for IP).}
Santhanam’s Proof: Lower Bound From IP=PSPACE

• PSPACE $\not\subseteq$ SIZE[$n^k$] (PSPACE can search outside SIZE[$n^k$]).
• PSPACE=MA (MA guesses prover circuit for IP).

YES

PSPACE $\subseteq$ ? P/poly

NO

• PSPACE-Complete L not in P/poly.
• Suppose L circuit size $T>\text{poly}(n)$.
• Pad so $T$ just above $h^f$ (advice ensures padding right).
• MA guesses prover circuit.
Santhanam’s Proof: Lower Bound From IP=PSPACE

**PSPACE** \( \subseteq ? \) **P/poly**

- **PSPACE** \( \not\subseteq \text{SIZE}[n^k] \) (**PSPACE** can search outside \( \text{SIZE}[n^k] \)).
- **PSPACE**=**MA** (**MA** guesses prover circuit for **IP**).

**YES**

- **PSPACE**-Complete \( L \) not in **P/poly**.
- Suppose \( L \) circuit size \( T > \text{poly}(n) \).
- Pad so \( T \) just above \( n^k \) (advice ensures padding right).
- **MA** guesses prover circuit.

**NO**

\[ T(m) \sim n^k \]
Santhanam’s Proof: Lower Bound From $\text{IP=PSPACE}$

- To simulate verifier-prover interaction need time polynomially larger than prover circuit size.

- $\text{PSPACE} \not\subseteq \text{SIZE}[n^k]$ ($\text{PSPACE}$ can search outside $\text{SIZE}[n^k]$).
- $\text{PSPACE}=\text{MA}$ ($\text{MA}$ guesses prover circuit for $\text{IP}$).

\[
T(m) \sim n^k
\]
Santhanam’s Proof: Lower Bound From IP=PSPACE

- To simulate verifier-prover interaction need time polynomially larger than prover circuit size.
- Idea: Use PCP to minimize verifier time, queries, interaction.

- **PSPACE \not\subseteq \text{SIZE}[n^k]** (PSPACE can search outside \text{SIZE}[n^k]).
- **PSPACE=\text{MA}** (MA guesses prover circuit for IP).

- **PSPACE** \subseteq ? \text{P/poly}

- **PSPACE-Complete L not in P/poly.**
- Suppose L circuit size \(T>\text{poly}(n)\).
- Pad so \(T\) just above \(n^k\) (advice ensures padding right).
- MA guesses prover circuit.

\(T(m) \sim n^k\)
New PCP Theorem

For Time-Space\([T,S]\) there is PCP verifier with:

1. Verifier time \(O\sim(n+\log T)\).
2. Prover space \(O\sim(S+n)\).
3. Queries \(O(\log n + \log \log T)\).
4. Answer size \(O(\log \log \log T)\).
New PCP Theorem

For Time-Space \([T,S]\) there is PCP verifier with:

1. Verifier time \(O\sim(n + \log T)\).
2. Prover space \(O\sim(S+n)\).
3. Queries \(O(\log n + \log \log T)\).
4. Answer size \(O(\log \log \log T)\).

Think of \(T=2^n\) and \(S=n\).
New PCP Theorem

For Time-Space $[T,S]$ there is PCP verifier with:

1. Verifier time $O(\sim(n+\log T))$.
2. Prover space $O(\sim(S+n))$.
3. Queries $O(\log n + \log \log T)$.
4. Answer size $O(\log \log T)$.

Think of $T=2^n$ and $S=n$

As opposed to $\text{polylog} T$ [BGHSV05, ...]
New PCP Theorem

For Time-Space $[T,S]$ there is PCP verifier with:

1. Verifier time $O\sim(n+\log T)$.
2. Prover space $O\sim(S+n)$.
3. Queries $O(\log n + \log \log T)$.
4. Answer size $O(\log \log T)$.

Think of $T=2^n$ and $S=n$

As opposed to $\text{polylog}T$ [BGHSV05,...]

Holmgren-Rothblum `18 could give $O\sim(n+\log T)$ verifier time, but $O(\log T)$ queries
What Goes Into New PCP: Ultra-Efficient Query Reduction

“Aggregation Through Curves”: How to evaluate an \( m \)-variate low degree polynomial on \( k \) points using a prover?
What Goes Into New PCP: Ultra-Efficient Query Reduction

“Aggregation Through Curves”: How to evaluate an $m$-variate low degree polynomial on $k$ points using a prover?
What Goes Into New PCP: Ultra-Efficient Query Reduction

“Aggregation Through Curves”: How to evaluate an \( m \)-variate low degree polynomial on \( k \) points using a prover?

1. Pass degree-\( k \) curve through \( k \) points and random point.
2. Ask prover for the restriction of polynomial to curve.
3. Check restriction on random point.
What Goes Into New PCP: Ultra-Efficient Query Reduction

“Aggregation Through Curves”: How to evaluate an $m$-variate low degree polynomial on $k$ points using a prover?

1. Pass degree-$k$ curve through $k$ points and random point.
2. Ask prover for the restriction of polynomial to curve.
3. Check restriction on random point.

Time to compute curve $\sim km$, instead of $\sim k+m$. 
What Goes Into New PCP: Ultra-Efficient Query Reduction

“Aggregation Through Curves”: How to evaluate an $m$-variate low degree polynomial on $k$ points using a prover?

1. Pass degree-$k$ curve through $k$ points and random point.
2. Ask prover for the restriction of polynomial to curve.
3. Check restriction on random point.

Time to compute curve $\sim km$, instead of $\sim k+m$.

Idea: need linear transformation of $k$ points in time $\sim k+m$. Possible for related points.
For Which $k$ Prove $\text{MATIME}[n^{k+o(1)}]/1 \not\subset \text{SIZE}[n^k]$?

Have three cases:

1. $\text{PSPACE} \not\subset \text{P/poly}$
2. $\text{SPACE}[n] \subseteq \text{SIZE}[n^{1+o(1)}]$
3. $\exists a > 1: \text{SPACE}[n] \subseteq \text{SIZE}[n^{a+o(1)}] - \text{SIZE}[n^{a-o(1)}]$
For Which \( k \) Prove \( \text{MATIME}[n^{k+o(1)}]/1 \not\subset \text{SIZE}[n^k] \)?

Have three cases:

1. \( \text{PSPACE} \not\subset \text{P/poly} \)
2. \( \text{SPACE}[n] \subset \text{SIZE}[n^{1+o(1)}] \)
3. \( \exists a > 1: \text{SPACE}[n] \subset \text{SIZE}[n^{a+o(1)}] - \text{SIZE}[n^{a-o(1)}] \)

All \( k \). Santhanam's IP works, part of input running IP on shrinks very quickly, poly overhead shrinks.
For Which $k$ Prove $\text{MATIME}[n^{k+o(1)}]/1 \not\subset \text{SIZE}[n^k]$?

Have three cases:

1. $\text{PSPACE} \not\subset \text{P/poly}$
2. $\text{SPACE}[n] \subseteq \text{SIZE}[n^{1+o(1)}]$
3. $\exists a > 1: \text{SPACE}[n] \subseteq \text{SIZE}[n^{a+o(1)}] - \text{SIZE}[n^{a-o(1)}]$
For Which \( k \) Prove \( \text{MATIME}[n^{k+o(1)}]/1 \not\subset \text{SIZE}[n^k] \)?

Have three cases:

1. \( \text{PSPACE} \not\subset \text{P/poly} \)
2. \( \text{SPACE}[n] \subseteq \text{SIZE}[n^{1+o(1)}] \)
3. \( \exists a > 1 : \text{SPACE}[n] \subseteq \text{SIZE}[n^{a+o(1)}] - \text{SIZE}[n^{a-o(1)}] \)

All \( k \). Santhanam's IP works, part of input running IP on shrinks very quickly, poly overhead shrinks.

All \( k \). Space \sim Size. From our PCP Space \sim Prover Space \sim Prover size.

\( k < a \). For \( k = a \), Space[\( n \)] \not\subset \text{Size}[n^a] \), but Prover Space[\( n \)] \sim \text{Size}[n^{a+o(1)}]. \) So OMA time is about Size[\( n^{a+o(1)} \)]. Pad inputs for \( k < a \).

For \( k > a \), need something stronger than Space[\( n \)] for hard problem. Space hardness might stall, may need Space[\( n^k \)], but then prover requires Space [\( n^k \)], may need Size[\( n^{ka} \)].
Citations