Computational Complexity of Games

Kyle Burke

October 23, 2017
The Big Question

- Does Left have a winning move going first?
The Big Question

- Does Left have a winning move going first?
- i.e. $G \in \mathcal{N} \cup \mathcal{L}$?
The Big Question

- Does Left have a winning move going first?
- i.e. \( G \in \mathcal{N} \cup \mathcal{L} \) ?
- CS: Fastest algorithm to answer this for entire ruleset?
Computational Counting

- Count time as algorithmic steps.
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
  - $10n^2 + 37.4n + 124$
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
  - $10n^2 + 37.4n + 124 \rightarrow 10n^2$
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
  - $10n^2 + 37.4n + 124 \rightarrow 10n^2 \rightarrow n^2$
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
  - $10n^2 + 37.4n + 124 \rightarrow 10n^2 \rightarrow n^2$
  - $O(101n^2) = O(n^2)$
Computational Counting

- Count time as algorithmic steps.
- Cheat: use Big-O notation
  - $10n^2 + 37.4n + 124 \rightarrow 10n^2 \rightarrow n^2$
  - $O(101n^2) = O(n^2)$
  - $O(n^3 + n^2 + n + 50) = O(n^3)$
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
- $O(n^{88})$ ✓
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
- $O(n^{88})$ ✓
- $O(2^n)$ ⊗

Note: No fixed-size rulesets!
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
- $O(n^{88})$ ✓
- $O(2^n)$ ⊘
- $O(4^n)$ ⊘
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2m)$ ✓
- $O(n^2m^6p^{1.5})$ ✓
- $O(n^{88})$ ✓
- $O(2^n)$ ☹
- $O(4^n)$ ☹
- $O(n!)$ ☹
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2m) \checkmark$
- $O(n^2m^6p^{1.5}) \checkmark$
- $O(n^{88}) \checkmark$
- $O(2^n) \bigcirc$
- $O(4^n) \bigcirc$
- $O(n!) \bigcirc$

- Top 3: P (Not $\mathcal{P}$)
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
- $O(n^{88})$ ✓
- $O(2^n)$ ⊗
- $O(4^n)$ ⊗
- $O(n!)$ ⊗

- Top 3: P (Not P)
- Bottom 3: EXPTIME (P ⊊ EXPTIME)
What is Tractable ("Easy")?

Runs in steps polynomial in input size (parameters).

- $O(n^2 m)$ ✓
- $O(n^2 m^6 p^{1.5})$ ✓
- $O(n^{88})$ ✓
- $O(2^n)$ ☹
- $O(4^n)$ ☹
- $O(n!)$ ☹

- Top 3: P (Not $\mathcal{P}$)
- Bottom 3: EXPTIME ($\mathcal{P} \subsetneq \text{EXPTIME}$)

Note: No fixed-size rulesets!
Some Games are Easy!

- **Brussels Sprouts**

  - \( \text{P} \iff n \text{ is even} \)
  - \( O(n \log(m)) \text{ steps} \).
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
Some Games are Easy!

- Brussels Sprouts
  - $n$ crosses, $k$ edges
  - $\mathcal{P} \iff n$ is even

- Nim
  - $n$ piles, each up to $m$ sticks.
  - $\mathcal{P} \iff$ Nim-sum is zero.
  - $O(n \log(m))$ steps.
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
  - $P \iff n$ is even
  - $O(1)$ steps
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
  - $P \iff n$ is even
  - $O(1)$ steps

- **Nim**
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
  - $P \iff n$ is even
  - $O(1)$ steps

- **Nim**
  - $n$ piles, each up to $m$ sticks.
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
  - $\mathcal{P} \iff n$ is even
  - $O(1)$ steps

- **Nim**
  - $n$ piles, each up to $m$ sticks.
  - $\mathcal{P} \iff$ Nim-sum is zero.
Some Games are Easy!

- **Brussels Sprouts**
  - $n$ crosses, $k$ edges
  - $P \iff n$ is even
  - $O(1)$ steps

- **Nim**
  - $n$ piles, each up to $m$ sticks.
  - $P \iff$ Nim-sum is zero.
  - $O(n \log(m))$ steps.
Some Initial Positions Are Easy!

- **Chomp**

- $G \in \mathbb{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$.

- $O(1)$ if I know it's start.

- $O(nm)$ otherwise.

- **Cram**

- $G \in \mathbb{P}$ for any $2 \times 2$ rectangle.

- $O(1)$ if I know it's start.

- $O(4nm) = O(nm)$ otherwise.

- **Sprouts** (maybe)

- **Sprouts Conjecture:**

- $n \mod 6 \leq 2 \iff P$

- $O(\log(n))$ if I know it's a start.

- (otherwise unknown)
Some Initial Positions Are Easy!

- **Chomp**
  - \( G \in \mathcal{N} \) for any \( n \times m \) rectangle bigger than \( 1 \times 1 \)
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **Cram**

- **Sprouts** (maybe)

- **Sprouts Conjecture:** $n \mod 6 \leq 2 \iff P$

  - $O(\log(n))$ if I know it’s a start
  - (otherwise unknown)
Some Initial Positions Are Easy!

- **Chomp**
  1. $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  2. $O(1)$ if I know it’s start
  3. $O(nm)$ otherwise

- **Cram**
  1. $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.

- **Sprouts** (maybe)
  1. Sprouts Conjecture: $n \mod 6 \leq 2 \iff P$
  2. $O(\log(n))$ if I know it’s a start
  3. $O(1)$ otherwise unknown
Some Initial Positions Are Easy!

- **CHOMP**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **CRAM**
  - $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.
  - $O(1)$ if I know it’s start
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **Cram**
  - $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.
  - $O(1)$ if I know it’s start
  - $O(4nm) = O(nm)$ otherwise
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **Cram**
  - $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.
  - $O(1)$ if I know it’s start
  - $O(4nm) = O(nm)$ otherwise

- **Sprouts** (maybe)
Some Initial Positions Are Easy!

- **Chomp**
  - \( G \in \mathcal{N} \) for any \( n \times m \) rectangle bigger than \( 1 \times 1 \)
  - \( O(1) \) if I know it’s start
  - \( O(nm) \) otherwise

- **Cram**
  - \( G \in \mathcal{P} \) for any \( 2n \times 2m \) rectangle.
  - \( O(1) \) if I know it’s start
  - \( O(4nm) = O(nm) \) otherwise

- **Sprouts (maybe)**
  - Sprouts Conjecture: \( n \text{ mod } 6 \leq 2 \iff \mathcal{P} \)
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **Cram**
  - $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.
  - $O(1)$ if I know it’s start
  - $O(4nm) = O(nm)$ otherwise

- **Sprouts (maybe)**
  - Sprouts Conjecture: $n \mod 6 \leq 2 \iff \mathcal{P}$
  - $O(\log(n))$ if I know it’s a start
Some Initial Positions Are Easy!

- **Chomp**
  - $G \in \mathcal{N}$ for any $n \times m$ rectangle bigger than $1 \times 1$
  - $O(1)$ if I know it’s start
  - $O(nm)$ otherwise

- **Cram**
  - $G \in \mathcal{P}$ for any $2n \times 2m$ rectangle.
  - $O(1)$ if I know it’s start
  - $O(4nm) = O(nm)$ otherwise

- **Sprouts (maybe)**
  - Sprouts Conjecture: $n \mod 6 \leq 2 \iff \mathcal{P}$
  - $O(\log(n))$ if I know it’s a start
  - (otherwise unknown)
General Algorithm (A) for Short Games

For any $G$, $A(G)$:

1. Draw the Game Tree
2. All the leaves are in $P$.
3. Use CGT rules to outcome classes all the way back up the tree until you evaluate $G$.
4. Return whether $G \in N \cup L$.

Worst case: have to evaluate all nodes of the game tree. So $A$ uses exponential time.
General Algorithm (A) for Short Games

For any $G$, $A(G)$:

- Draw the Game Tree

Worst case: have to evaluate all nodes of the game tree.

... so $A$ uses exponential time.
For any $G$, $A(G)$:

- Draw the Game Tree
- All the leaves are in $\mathcal{P}$. 

Worst case: have to evaluate all nodes of the game tree. So $A$ uses exponential time.
General Algorithm (A) for Short Games

For any \( G \), \( A(G) \):

- Draw the Game Tree
- All the leaves are in \( \mathcal{P} \).
- Use CGT rules to outcome classes all the way back up the tree until you evaluate \( G \).
General Algorithm (A) for Short Games

For any $G$, $A(G)$:
- Draw the Game Tree
- All the leaves are in $P$.
- Use CGT rules to outcome classes all the way back up the tree until you evaluate $G$.
- Return whether $G \in \mathcal{N} \cup \mathcal{L}$.
For any $G$, $A(G)$:

- Draw the Game Tree
- All the leaves are in $\mathcal{P}$.
- Use CGT rules to outcome classes all the way back up the tree until you evaluate $G$.
- Return whether $G \in \mathcal{N} \cup \mathcal{L}$.

Worst case: have to evaluate all nodes of the game tree.
General Algorithm (A) for Short Games

For any $G$, $A(G)$:

- Draw the Game Tree
- All the leaves are in $\mathcal{P}$.
- Use CGT rules to outcome classes all the way back up the tree until you evaluate $G$.
- Return whether $G \in \mathcal{N} \cup \mathcal{L}$.

Worst case: have to evaluate all nodes of the game tree. ... so $A$ uses exponential time.
Completeness

Problems where the *best* algorithm both:

- Runs in at most exponential time ("inclusion": in \( \text{EXPTIME} \))
- Requires exponential time in the worst cases ("hardness": \( \text{EXPTIME} \)-hard)

... are \( \text{EXPTIME} \)-complete.
Completeness

Problems where the best algorithm both:
  ▶ Runs in at most exponential time ("inclusion": in EXPTIME) and
Completeness

Problems where the best algorithm both:

- Runs in at most exponential time ("inclusion": in EXPTIME) and
- Requires exponential time in the worst cases ("hardness": EXPTIME-hard)
Completeness

Problems where the best algorithm both:

- Runs in at most exponential time ("inclusion": in \textit{EXPTIME}) and
- Requires exponential time in the worst cases ("hardness": \textit{EXPTIME}-hard)

... are \textit{EXPTIME}-complete.
Yes, there are rulesets that require exponential time to solve!

---

2 Hearn, Demaine - http://erikdemaine.org/papers/GPC/
3 Robson, "The Complexity of Go", IFIP Congress 1983
EXPTIME-complete Rule sets?

Yes, there are rulesets that require exponential time to solve!

- (Generalized) Chess\textsuperscript{1}

\begin{itemize}
  \item Fraenkel, Lichtenstein -
    \url{http://www.sciencedirect.com/science/article/pii/0097316581900169}
  \item Hearn, Demaine - \url{http://erikdemaine.org/papers/GPC/}
  \item Robson, "The Complexity of Go", IFIP Congress 1983
\end{itemize}
EXPTIME-complete Rulesets?

Yes, there are rulesets that require exponential time to solve!

- (Generalized) Chess\(^1\)
- Unbounded Constraint Logic\(^2\)

\(^2\)Hearn, Demaine - http://erikdemaine.org/papers/GPC/
\(^3\)Robson, "The Complexity of Go", IFIP Congress 1983
EXPTIME-complete Rulesets?

Yes, there are rulesets that require exponential time to solve!

- (Generalized) Chess\(^1\)
- Unbounded Constraint Logic\(^2\)
- Go (without Superko)\(^3\)

---


\(^3\)Robson, "The Complexity of Go", IFIP Congress 1983
EXPTIME-complete Rule sets?

Yes, there are rulesets that require exponential time to solve!

- (Generalized) Chess\(^1\)
- Unbounded Constraint Logic\(^2\)
- Go (without Superko)\(^3\)

Notice: All loopy games!

---


\(^3\)Robson, "The Complexity of Go", IFIP Congress 1983
Yes, there are rulesets that require exponential time to solve!

- (Generalized) Chess\(^1\)
- Unbounded Constraint Logic\(^2\)
- Go (without Superko)\(^3\)

Notice: All loopy games!
How do we know there’s no faster algorithm?

---

\(^1\)Fraenkel, Lichtenstein -

\(^2\)Hearn, Demaine -
http://erikdemaine.org/papers/GPC/

\(^3\)Robson, "The Complexity of Go", IFIP Congress 1983
Game Reductions!

Let’s say I want to prove a loopy game \textsc{Banane} is EXPTIME-hard.
Let’s say I want to prove a loopy game $\text{Banane}$ is EXPTIME-hard.

▶ I need to find a function

$f : \text{Chess positions} \rightarrow \text{Banane positions}$
Game Reductions!

Let’s say I want to prove a loopy game \textsc{Banane} is EXPTIME-hard.

- I need to find a function
  
  \[ f : \text{Chess positions} \rightarrow \text{Banane positions} \]

- \( f \) must be efficiently computable
Let’s say I want to prove a loopy game \texttt{Banane} is EXPTIME-hard.

- I need to find a function $f : \texttt{Chess positions} \rightarrow \texttt{Banane positions}$
- $f$ must be efficiently computable
- $x \in \mathcal{L} \cup \mathcal{N} \iff f(x) \in \mathcal{L} \cup \mathcal{N}$
Let’s say I want to prove a loopy game $\text{Banane}$ is EXPTIME-hard.

- I need to find a function $f: \text{Chess positions} \rightarrow \text{Banane positions}$
- $f$ must be efficiently computable
- $x \in \mathcal{L} \cup \mathcal{N} \iff f(x) \in \mathcal{L} \cup \mathcal{N}$
- Now $f$ is a reduction.
Game Reductions!

Let’s say I want to prove a loopy game \texttt{Banane} is \textsc{EXPTIME}-hard.

- I need to find a function
  \( f : \texttt{Chess positions} \rightarrow \texttt{Banane positions} \)
- \( f \) must be efficiently computable
- \( x \in \mathcal{L} \cup \mathcal{N} \iff f(x) \in \mathcal{L} \cup \mathcal{N} \)
- Now \( f \) is a \textit{reduction}. ... and \texttt{Banane} is \textsc{EXPTIME}-hard!
Game Reductions!

Let’s say I want to prove a loopy game $\text{Banane}$ is EXPTIME-hard.

- I need to find a function $f : \text{Chess positions} \rightarrow \text{Banane positions}$
- $f$ must be efficiently computable
- $x \in \mathcal{L} \cup \mathcal{N} \iff f(x) \in \mathcal{L} \cup \mathcal{N}$
- Now $f$ is a reduction. ... and $\text{Banane}$ is EXPTIME-hard!

How does this show hardness?
Hardness Follows a Reduction

Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$
Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$

- $\text{CHESS}$ EXPTIME-hard $\rightarrow \text{BANANE}$ EXPTIME-hard
Hardness Follows a Reduction

Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$

- $\text{CHESS}$ EXPTIME-hard $\rightarrow \text{BANANE}$ EXPTIME-hard
- Proof-by-contradiction
Hardness Follows a Reduction

Reduction \( f : \text{Chess} \rightarrow \text{Banane} \)

- \( \text{Chess} \) EXPTIME-hard \( \rightarrow \) \( \text{Banane} \) EXPTIME-hard
- Proof-by-contradiction
  - Assume \( \text{Banane} \) solvable in faster-than-exponential time by some algorithm \( A \)
Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$

- $\text{CHESS}$ EXPTIME-hard $\rightarrow \text{BANANE}$ EXPTIME-hard
- Proof-by-contradiction
  - Assume $\text{BANANE}$ solvable in faster-than-exponential time by some algorithm $A$
  - New $\text{CHESS}$-solving algorithm, $B(x)$: return $A(f(x))$
Hardness Follows a Reduction

Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$

- **CHESS** EXPTIME-hard $\rightarrow$ **BANANE** EXPTIME-hard
- Proof-by-contradiction
  - Assume **BANANE** solvable in faster-than-exponential time by some algorithm $A$
  - New **CHESS**-solving algorithm, $B(x) : \text{return } A(f(x))$
  - $B$ solves **CHESS**!
Hardness Follows a Reduction

Reduction $f : \text{CHESS} \rightarrow \text{BANANE}$

- $\text{CHESS}$ EXPTIME-hard $\rightarrow \text{BANANE}$ EXPTIME-hard
- Proof-by-contradiction
  - Assume $\text{BANANE}$ solvable in faster-than-exponential time by some algorithm $A$
  - New $\text{CHESS}$-solving algorithm, $B(x)$: return $A(f(x))$
  - $B$ solves $\text{CHESS}$!
  - $B$ solves $\text{CHESS}$ in faster-than-exponential time!
Reduction $f : \text{Chess} \rightarrow \text{Banane}$

- $\text{Chess}$ EXPTIME-hard $\rightarrow \text{Banane}$ EXPTIME-hard
- Proof-by-contradiction
  - Assume $\text{Banane}$ solvable in faster-than-exponential time by some algorithm $A$
  - New $\text{Chess}$-solving algorithm, $B(x)$: return $A(f(x))$
  - $B$ solves $\text{Chess}$!
  - $B$ solves $\text{Chess}$ in faster-than-exponential time!
  - Now $\text{Chess}$ is not EXPTIME-hard $\rightarrow \leftarrow$
What about Short Games?

- Assume longest game has polynomial turns.
What about Short Games?

- Assume longest game has polynomial turns.
- ... and polynomial options.
What about Short Games?

- Assume longest game has polynomial turns.
- ... and polynomial options.
- E.g. Domineering.
What about Short Games?

▶ Assume longest game has polynomial turns.
▶ ... and polynomial options.
▶ E.g. Domineering.
▶ Let’s do 3x3 case:
What about Short Games?

▶ Assume longest game has polynomial turns.
▶ ... and polynomial options.
▶ E.g. Domineering.
▶ Let’s do 3x3 case:

```
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
|   |   |   |
+---+---+---+
```
What about Short Games?
What about Short Games?

↓ Left’s potentially-winning options

∈R∪P (Right wins)
What about Short Games?

Left’s potentially-winning options

Right’s potentially-winning options

$\in \mathbb{R} \cup \mathbb{P}$ (Right wins)
What about Short Games?

Left’s potentially-winning options

Right’s potentially-winning options
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

\[ \in \mathbb{R} \cup \mathbb{P} \quad \text{(Right wins)} \]
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

∈R∪P (Right wins)
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

∈ \( \mathcal{R} \cup \mathcal{P} \) (Right wins)
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options ✓

So... previous move not a winner for Left. Back up and try the next one.
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

No Right move!
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

No Right move! (Left wins)
What about Short Games?

↓ Left’s potentially-winning options

↓ Right’s potentially-winning options

↓ Left’s potentially-winning options

No Right move! (Left wins) Go back and change Right’s last move...
What about Short Games?

- Left’s potentially-winning options
- Right’s potentially-winning options

How many rows of boards do I need? (polynomial)

How many boards in a row? (polynomial)

How much “workspace” do I need? (polynomial)
What about Short Games?

Left’s potentially-winning options

Right’s potentially-winning options

▶ Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
What about Short Games?

- Left’s potentially-winning options
- Right’s potentially-winning options

- Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
- How many rows of boards do I need?
What about Short Games?

Left’s potentially-winning options

Right’s potentially-winning options

- Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
- How many rows of boards do I need? (polynomial)
What about Short Games?

- Continued to decide $G \in \mathcal{L} \cup \mathcal{N}$
- How many rows of boards do I need? (polynomial)
- How many boards in a row?
What about Short Games?

- Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
- How many rows of boards do I need? (polynomial)
- How many boards in a row? (polynomial)
What about Short Games?

- Left’s potentially-winning options
- Right’s potentially-winning options

▶ Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
▶ How many rows of boards do I need? (polynomial)
▶ How many boards in a row? (polynomial)
▶ How much "workspace" do I need?
What about Short Games?

> Continue to decide $G \in \mathcal{L} \cup \mathcal{N}$
> How many rows of boards do I need? (polynomial)
> How many boards in a row? (polynomial)
> How much "workspace" do I need? (polynomial)
PSPACE: All problems solvable with a polynomial amount of space.
PSPACE: All problems solvable with a polynomial amount of space.

- $P \subsetneq \text{EXPTIME}$
PSPACE: All problems solvable with a polynomial amount of space.

- \( P \subsetneq \text{EXPTIME} \)
- \( P \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \)
PSPACE: All problems solvable with a polynomial amount of space.

- $P \subsetneq \text{EXPTIME}$
- $P \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$
- $P \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$
PSPACE-hard Problems

PSPACE-hard: Problems at least as hard as the hardest problem(s) in PSPACE
PSPACE-hard Problems

PSPACE-hard: Problems at least as hard as the hardest problem(s) in PSPACE

- Everything EXPTIME-hard.
PSPACE-hard Problems

PSPACE-hard: Problems at least as hard as the hardest problem(s) in PSPACE

- Everything EXPTIME-hard.
- Games: Amazons, Geography, Hex, Konane, Node Kayles, Snort, Toads and Frogs, etc.
PSPACE-hard Problems

PSPACE-hard: Problems at least as hard as the hardest problem(s) in PSPACE

- Everything EXPTIME-hard.
- Games: Amazons, Geography, Hex, Konane, Node Kayles, Snort, Toads and Frogs, etc.
- Non-Games: Deadlock detection, periodic scheduling, etc.
PSPACE-hard Problems

PSPACE-hard: Problems at least as hard as the hardest problem(s) in PSPACE

- Everything EXPTIME-hard.
- Games: Amazons, Geography, Hex, Konane, Node Kayles, Snort, Toads and Frogs, etc.
- Non-Games: Deadlock detection, periodic scheduling, etc.

PSPACE-complete: Both PSPACE-hard and in PSPACE.
Example: **Impartial Col** is PSPACE-hard

**Impartial Col**: 2-coloring placement game
Example: \textbf{Impartial Col} is \textsc{PSPACE}-hard

\textbf{Impartial Col}: 2-coloring placement game
Example: **Impartial Col** is PSPACE-hard

**Impartial Col**: 2-coloring placement game
Example: **Impartial Col** is PSPACE-hard

**Impartial Col**: 2-coloring placement game
Example: **Impartial Col** is PSPACE-hard

**Impartial Col**: 2-coloring placement game
Example: IMPARTIAL COL is PSPACE-hard

We’ll reduce from NODE KAYLES (known to be PSPACE-hard.)
Example: \textbf{Impartial Col} is PSPACE-hard

We’ll reduce from \textbf{Node Kayles} (known to be PSPACE-hard.)

\textbf{Node Kayles:} (Impartial) Each turn, place a token on an empty vertex not adjacent to another token.
Example: **Impartial Col** is PSPACE-hard

We’ll reduce from **Node Kayles** (known to be PSPACE-hard.)

**Node Kayles:** (Impartial) Each turn, place a token on an empty vertex not adjacent to another token.
Example: Impartial Col is PSPACE-hard

We’ll reduce from Node Kayles (known to be PSPACE-hard.)

Node Kayles: (Impartial) Each turn, place a token on an empty vertex not adjacent to another token.
Example: Impartial Col is PSPACE-hard

We’ll reduce from Node Kayles (known to be PSPACE-hard.)

Node Kayles: (Impartial) Each turn, place a token on an empty vertex not adjacent to another token.
Example: **Impartial Col** is PSPACE-hard

We’ll reduce from **Node Kayles** (known to be PSPACE-hard.)

**Node Kayles:** (Impartial) Each turn, place a token on an empty vertex not adjacent to another token.
Example: **Impartial Col** is PSPACE-hard

Proof: reduction from **Node Kayles** to **Impartial Col**.
Example: **Impartial Col** is PSPACE-hard

Proof: reduction from **Node Kayles** to **Impartial Col**.
Example: **Impartial Col** is PSPACE-hard

Proof: reduction from **Node Kayles** to **Impartial Col**.
Example: **Impartial Col** is PSPACE-hard

Proof: reduction from **Node Kayles** to **Impartial Col**.

Reduce!
Example: **Impartial Col** is PSPACE-hard

Proof: reduction from **Node Kayles** to **Impartial Col**.
Example: Snort is PSPACE-hard

Snort: Can’t play adjacent to opponent.
Example: Snort is PSPACE-hard

Snort: Can’t play adjacent to opponent.
Example: **Snort** is PSPACE-hard

**Snort**: Can’t play adjacent to opponent.
Example: **SNORT** is PSPACE-hard

**SNORT**: Can’t play adjacent to opponent.
Example: **Snort** is PSPACE-hard

Reduce from **Bigraph Node-Kayles** (known to be hard).
Example: **Snort** is PSPACE-hard

Reduce from **Bigraph Node-Kayles** (known to be hard).

**Bigraph Node-Kayles**: Kayles on Bipartite Graph where each player gets one side.
Example: **Snort** is PSPACE-hard

Reduce from **Bigraph Node-Kayles** (known to be hard).

**Bigraph Node-Kayles**: Kayles on Bipartite Graph where each player gets one side.
Example: **Snort** is PSPACE-hard

Reduce from **Bigraph Node-Kayles** (known to be hard).

**Bigraph Node-Kayles**: Kayles on Bipartite Graph where each player gets one side.
Example: **Snort** is PSPACE-hard

Reduce from **Bigraph Node-Kayles** (known to be hard). **Bigraph Node-Kayles**: Kayles on Bipartite Graph where each player gets one side.
Example: **Snort** is PSPACE-hard

Reduce **Bigraph Node Kayles** to **Snort**
Example: \textsc{Snort} is PSPACE-hard

Reduce \textsc{Bigraph Node Kayles} to \textsc{Snort}
Example: \textbf{Snort} is PSPACE-hard

Reduce \textbf{Bigraph Node Kayles} to \textbf{Snort}
Example: \textbf{Snort} is PSPACE-hard

Reduce \textbf{Bigraph Node Kayles} to \textbf{Snort}
Example: **GRAPH NoGo** is Hard

**GRAPH NoGo**: Go, without capture moves.
Example: $\text{GRAPH NoGo}$ is Hard

$\text{GRAPH NoGo}$: Go, without capture moves.
Example: **Graph NoGo** is Hard

**Graph NoGo**: Go, without capture moves.
Example: **GRAPH NoGo** is Hard

**GRAPH NoGo:** Go, without capture moves.
Example: GRAPH NoGo is Hard

Reduce: $\text{Col} \rightarrow \text{NoGo}$
Example: **GRAPH NoGo** is Hard

Reduce: $\text{Col} \rightarrow \text{NoGo}$

$\text{Col}$: Can’t play adjacent to yourself.
Example: **GRAPH NoGo** is Hard

**Reduce**: $\text{Col} \rightarrow \text{NoGo}$

*Col*: Can’t play adjacent to yourself.
Example: GRAPH NoGo is Hard

Reduce: Col $\rightarrow$ NoGo
Col: Can’t play adjacent to yourself.
Example: **GRAPH NoGo is Hard**

Reduce: **Col** $\rightarrow$ **NoGo**

**Col**: Can’t play adjacent to yourself.
Example: **GRAPH NoGo** is Hard

Reduce from **Col**
Example: \textbf{GRAPH NoGo} is Hard

Reduce from \textbf{Col}
Separate "gadgets" to replace vertices and edges.
Example: **GRAPH NoGo** is Hard

Reduce from **Col**
Separate "gadgets" to replace vertices and edges.
Here’s the gadget for each vertex:
Example: **GRAPH NoGo** is Hard

Reduce from **Col**
Separate "gadgets" to replace vertices and edges.
Here’s the gadget for each vertex:
Example: **GRAPH NoGO** is Hard

Reduce from **Col**
Separate "gadgets" to replace vertices and edges.
Here’s the gadget for each vertex:
Example: \textsc{Graph NoGo} is Hard

Reduce from \textsc{Col}
Separate "gadgets" to replace vertices and edges.
Here’s the gadget for each vertex:
Example: **GRAPH NoGo** is Hard

Here’s the reduction for each **Col** edge:
Example: **GRAPH NoGo** is Hard

Here’s the reduction for each **COL** edge:

\[ \text{Reduce!} \]

\[ x \rightarrow y \]
Example: \textbf{GRAPH NoGo} is Hard

Here’s the reduction for each \textsc{col} edge:

\[
\begin{array}{c}
x \\
\hline
y
\end{array}
\]

Reduce!
Example: **GRAPH NoGo** is Hard

Here’s the reduction for each *Col* edge:
Example: **GRAPH NoGo** is Hard

Here’s the reduction for each **COL** edge:

\[ x \rightarrow y \rightarrow x' \rightarrow y' \]

Reduce!
Locality is Tougher

For some games, we have to play adjacent to other moves.
Locality is Tougher

For some games, we have to play adjacent to other moves.

- Geography
Locality is Tougher

For some games, we have to play adjacent to other moves.

- **Geography**
- **Slimetrail**
Locality is Tougher

For some games, we have to play adjacent to other moves.

- **Geography**
- **Slimetrail**
- **Constraint Logic**
Locality is Tougher

For some games, we have to play adjacent to other moves.

- Geography
- Slimetrail
- Constraint Logic

Some of the first PSPACE-hard games were these, proven from QSAT.
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x}_1 \lor x_2) \land \cdots \land (\overline{x}_{27} \lor \overline{x}_1 \lor x_{12})\)
QSAT (is a game)

- **QSAT:** Quantified Boolean Satisfiability
- **3CNF:** \((x_0 \lor \overline{x}_1 \lor x_2) \land \cdots \land (\overline{x}_{27} \lor \overline{x}_1 \lor x_{12})\)
- **Play:** create an assignment of variables.
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x}_1 \lor x_2) \land \cdots \land (\overline{x}_{27} \lor \overline{x}_1 \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2\), etc
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2\), etc
- **Left wins** if formula is true;
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2\), etc
- Left wins if formula is true; Right otherwise
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- Play: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2\), etc
- Left wins if formula is true; Right otherwise
- Phrase winnability with quantifiers!
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2\), etc

- Left wins if formula is true; Right otherwise
- Phrase winnability with quantifiers!
- \(G \in \mathcal{L} \cup \mathcal{N} \iff\)
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x_1} \lor x_2) \land \cdots \land (\overline{x_{27}} \lor \overline{x_1} \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2, \text{etc}\)
- Left wins if formula is true; Right otherwise
- **Phrase winnability with quantifiers!**
- \(G \in \mathcal{L} \cup \mathcal{N} \iff \exists x_0 : \forall x_1 : \exists x_2 : \forall x_3 : \ldots : \forall x_{27} : \)
QSAT (is a game)

- **QSAT**: Quantified Boolean Satisfiability
- **3CNF**: \((x_0 \lor \overline{x}_1 \lor x_2) \land \cdots \land (\overline{x}_{27} \lor \overline{x}_1 \lor x_{12})\)
- **Play**: create an assignment of variables.
  - Left assigns to \(x_0\)
  - Right assigns to \(x_1\)
  - Left: \(x_2, \text{ etc}\)

- Left wins if formula is true; Right otherwise
- Phrase winnability with quantifiers!

\[ G \in \mathcal{L} \cup \mathcal{N} \iff \exists x_0 : \forall x_1 : \exists x_2 : \forall x_3 : \ldots : \forall x_{27} : (x_0 \lor \overline{x}_1 \lor x_2) \land \cdots \land (\overline{x}_{27} \lor \overline{x}_1 \lor x_{12}) \]
Sample Reduction: **GEOGRAPHY**

- Reduce QSAT to GEOGRAPHY
Sample Reduction: Geography

- Reduce QSAT to Geography
- Geography: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: **GEOGRAPHY**

- Reduce **QSAT** to **GEOGRAPHY**
- **GEOGRAPHY**: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: Geography

- Reduce QSAT to Geography
- Geography: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: Geography

- Reduce QSAT to Geography
- Geography: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: **GEOGRAPHY**

- Reduce QSAT to **GEOGRAPHY**
- **GEOGRAPHY**: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: Geography

- Reduce QSAT to Geography
- Geography: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: **GEOGRAPHY**

- Reduce **QSAT** to **GEOGRAPHY**
- **GEOGRAPHY**: Move around on a directed graph, but you can’t visit a vertex twice.
Sample Reduction: GEOGRAPHY

- Reduce QSAT to GEOGRAPHY
Sample Reduction: GEOGRAPHY

- Reduce QSAT to GEOGRAPHY
- Variable Gadget:
Sample Reduction: \textbf{GEOGRAPHY}

- Reduce QSAT to \textbf{GEOGRAPHY}
- Variable Gadget:

\[ \chi_0 \]
Sample Reduction: GEOGRAPHY

- Reduce QSAT to GEOGRAPHY
- Variable Gadget:

```
x_0
```

```
x_0 true
```

```
x_0 false
```
Sample Reduction: \text{GEOGRAPHY}

- Reduce QSAT to \text{GEOGRAPHY}
- Variable Gadget:
Sample Reduction: Geography

- Reduce QSAT to Geography
- Variable Gadget:
Sample Reduction: Geography

After all variables are chosen, Right will choose a clause,
Sample Reduction: Geography

After all variables are chosen, Right will choose a clause, ... then Left will choose a variable in that clause.
Sample Reduction: Geography

After all variables are chosen, Right will choose a clause, ... then Left will choose a variable in that clause.
Sample Reduction: Geography

After all variables are chosen, Right will choose a clause, ... then Left will choose a variable in that clause.
Issues!

At this point, I expect you have some problems:
Issues!

At this point, I expect you have some problems:

- **NoGo** is only played on grids! Not general graphs!
Issues!

At this point, I expect you have some problems:

- **NoGo** is only played on grids! Not general graphs!
- Same with **Snort**!
At this point, I expect you have some problems:

- **NoGo** is only played on grids! Not general graphs!
- Same with **Snort**!
- What about starting positions? What if we never reach these positions in the range of the reduction?
Issues!

At this point, I expect you have some problems:

- **NoGo** is only played on grids! Not general graphs!
- **Same with Snort!**

- **What about starting positions? What if we never reach these positions in the range of the reduction?**

Let’s address the starting positions problem first.
Starting Positions

▶ Just determining winnability, not strategy.
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. Chomp, Hex)

Reason: parameterized by number (e.g. $13 \times 13$ Cram)

Can describe with logarithmic information.

Drawing the board is exponential in that description.

Usually, starting positions are easy.
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. **Chomp**, **Hex**)
- Some are conjectured (e.g. **Sprouts**, **Cram**)

Reason: parameterized by number (e.g. $13 \times 13$ Cram)

Can describe with logarithmic information.

Drawing the board is exponential in that description.

Usually, starting positions are easy.
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. CHOMP, HEX)
- Some are conjectured (e.g. SPROUTS, CRAM)
- Reason: parameterized by number (e.g. 13 × 13 CRAM)
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. Chomp, Hex)
- Some are conjectured (e.g. Sprouts, Cram)
- Reason: parameterized by number (e.g. 13 × 13 Cram)
  - Can describe with logarithmic information.
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. Chomp, Hex)
- Some are conjectured (e.g. Sprouts, Cram)
- Reason: parameterized by number (e.g. $13 \times 13$ Cram)
  - Can describe with logarithmic information.
  - Drawing the board is exponential in that description.
Starting Positions

- Just determining winnability, not strategy.
- Some are known (e.g. Chomp, Hex)
- Some are conjectured (e.g. Sprouts, Cram)
- Reason: parameterized by number (e.g. 13 × 13 Cram)
  - Can describe with logarithmic information.
  - Drawing the board is exponential in that description.
- Usually, starting positions are easy.
"Snapping to a grid" is difficult.

\[^4\] Evans, Tarjan 1976
\[^5\] Reisch 1981
\[^6\] Few slides back.
\[^7\] B., Hearn unpublished
\[^8\] Schaefer 1978
\[^9\] B., Hearn unpublished
"Snapping to a grid" is difficult.

- Usual progression: General $\rightarrow$ more specific $\rightarrow \cdots \rightarrow$ Grid

\[\begin{align*}
4 & \text{Evans, Tarjan 1976} \\
5 & \text{Reisch 1981} \\
6 & \text{Few slides back.} \\
7 & \text{B., Hearn unpublished} \\
8 & \text{Schaefer 1978} \\
9 & \text{B., Hearn unpublished}
\end{align*}\]
"Snapping to a grid" is difficult.

- Usual progression: General $\rightarrow$ more specific $\rightarrow \cdots \rightarrow$ Grid
- $\text{HEX}$: Graph$^4$ $\rightarrow$ Hex-Grid$^5$

---

$^4$ Evans, Tarjan 1976
$^5$ Reisch 1981
$^6$ Few slides back.
$^7$ B., Hearn unpublished
$^8$ Schaefer 1978
$^9$ B., Hearn unpublished
Specific Board Geometry

"Snapping to a grid" is difficult.

- Usual progression: General $\rightarrow$ more specific $\rightarrow \cdots \rightarrow$ Grid
- HEX: Graph$^4 \rightarrow$ Hex-Grid$^5$
- NoGo: Graph$^6 \rightarrow$ Planar Graph$^7$

---

$^4$ Evans, Tarjan 1976
$^5$ Reisch 1981
$^6$ Few slides back.
$^7$ B., Hearn unpublished
$^8$ Schaefer 1978
$^9$ B., Hearn unpublished
Specific Board Geometry

"Snapping to a grid" is difficult.

- Usual progression: General $\rightarrow$ more specific $\rightarrow$ $\cdots$ $\rightarrow$ Grid
- $\text{Hex: Graph}^4 \rightarrow \text{Hex-Grid}^5$
- $\text{NoGo: Graph}^6 \rightarrow \text{Planar Graph}^7$
- $\text{Snort: Graph}^8 \rightarrow \text{Planar Graph}^9$

---

$^4$Evans, Tarjan 1976
$^5$Reisch 1981
$^6$Few slides back.
$^7$B., Hearn unpublished
$^8$Schaefer 1978
$^9$B., Hearn unpublished
"Snapping to a grid" is difficult.

- Usual progression: General $\rightarrow$ more specific $\rightarrow \cdots \rightarrow$ Grid
- **HEX**: Graph$^4 \rightarrow$ Hex-Grid$^5$
- **NoGo**: Graph$^6 \rightarrow$ Planar Graph$^7$
- **Snort**: Graph$^8 \rightarrow$ Planar Graph$^9$
- This seems so backwards! Usually the general case is strongest!

---

$^4$Evans, Tarjan 1976  
$^5$Reisch 1981  
$^6$Few slides back.  
$^7$B., Hearn unpublished  
$^8$Schaefer 1978  
$^9$B., Hearn unpublished
Supersets

Supersets of hard sets are hard.
Supersets
Supersets of hard sets are hard. Let $f$ be our \textsc{NoGo} reduction.
Supersets

Supersets of hard sets are hard. Let $f$ be our NoGo reduction.

Range($f$)

hard
Supersets

Supersets of hard sets are hard. Let $f$ be our NoGo reduction.

Also hard

Range($f$)

hard
Supersets

Supersets of hard sets are hard. Let $f$ be our NoGo reduction.

All Graph NoGo Boards (hard)
Supersets

Supersets of hard sets are hard. Let $f$ be our NoGo reduction.

All Graph NoGo Boards (hard)
State of the Art
What is known to be hard now?

10 www.sciencedirect.com/science/article/pii/0022000078900454
11 Not yet published.
12 Not yet published
13 https://link.springer.com/article/10.1007/BF00288964
14 https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
16 Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\textsuperscript{10})

\textsuperscript{10}www.sciencedirect.com/science/article/pii/0022000078900454
\textsuperscript{11}Not yet published.
\textsuperscript{12}Not yet published
\textsuperscript{13}https://link.springer.com/article/10.1007/BF00288964
\textsuperscript{14}https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\textsuperscript{15}https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\textsuperscript{16}Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\))
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\))
- **Arc Kayles**: no known hardness
- **Domineering**: none
- **Clobber**: NP-hard on general graphs. (AGNW 2005\(^{15}\))
- **NoGo**: planar graphs (B, Hearn \(^{16}\))
- **Sprouts**: none

---


\(^{11}\) Not yet published.

\(^{12}\) Not yet published

\(^{13}\) [https://link.springer.com/article/10.1007/BF00288964](https://link.springer.com/article/10.1007/BF00288964)

\(^{14}\) [https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49](https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49)

\(^{15}\) [https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf](https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf)

\(^{16}\) Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\))
- **Atropos**: hexagonal grid (B,Teng 2007\(^{14}\))
- **Arc Kayles**: no known hardness
- **Domineering**: none
- **Clobber**: NP-hard on general graphs. (AGNW 2005\(^{15}\))
- **NoGo**: planar graphs (B, Hearn \(^{16}\))
- **Sprouts**: none

\(^{10}\)www.sciencedirect.com/science/article/pii/0022000078900454
\(^{11}\)Not yet published.
\(^{12}\)Not yet published
\(^{13}\)https://link.springer.com/article/10.1007/BF00288964
\(^{14}\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\(^{15}\)https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\(^{16}\)Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^1\))
- **Snort**: planar graphs (B, Hearn \(^1\)\(^1\))
- **Col**: planar graphs (B, Hearn \(^1\)\(^2\))
- **Hex**: hexagonal grid (Reisch, 1981\(^1\)\(^3\)) ✓

\(^1\)www.sciencedirect.com/science/article/pii/0022000078900454
\(^1\)Not yet published.
\(^1\)Not yet published
\(^1\)https://link.springer.com/article/10.1007/BF00288964
\(^1\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\(^1\)https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\(^1\)Not yet published.
State of the Art

What is known to be hard *now*?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\)) ✓
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\)) ✓

---

\(^{10}\)www.sciencedirect.com/science/article/pii/0022000078900454
\(^{11}\)Not yet published.
\(^{12}\)Not yet published
\(^{13}\)https://link.springer.com/article/10.1007/BF00288964
\(^{14}\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\(^{15}\)https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\(^{16}\)Not yet published.
State of the Art

What is known to be hard *now*?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\)) ✓
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\)) ✓
- **Arc Kayles**: *no* known hardness

---

\(^{10}\)www.sciencedirect.com/science/article/pii/0022000078900454

\(^{11}\)Not yet published.

\(^{12}\)Not yet published

\(^{13}\)https://link.springer.com/article/10.1007/BF00288964

\(^{14}\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49

\(^{15}\)Not yet published.

\(^{16}\)Not yet published.
State of the Art

What is known to be hard *now*?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\)) ▶
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\)) ▶
- **Arc Kayles**: *no* known hardness
- **Domineering**: *none*

---

\(^{10}\)www.sciencedirect.com/science/article/pii/0022000078900454

\(^{11}\)Not yet published.

\(^{12}\)Not yet published

\(^{13}\)https://link.springer.com/article/10.1007/BF00288964

\(^{14}\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49

\(^{15}\)https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf

\(^{16}\)Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\)) ✓
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\)) ✓
- **Arc Kayles**: *no* known hardness
- **Domineering**: *none*
- **Clobber**: NP-hard on general graphs. (AGNW 2005\(^{15}\))

---

\(^{10}\) [Link](https://www.sciencedirect.com/science/article/pii/0022000078900454)

\(^{11}\) Not yet published.

\(^{12}\) Not yet published

\(^{13}\) [Link](https://link.springer.com/article/10.1007/BF00288964)

\(^{14}\) [Link](https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49)

\(^{15}\) [Link](https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf)

\(^{16}\) Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\textsuperscript{10})
- **Snort**: planar graphs (B, Hearn \textsuperscript{11})
- **Col**: planar graphs (B, Hearn \textsuperscript{12})
- **Hex**: hexagonal grid (Reisch, 1981\textsuperscript{13}) ✓
- **Atropos**: hexagonal grid (B, Teng 2007\textsuperscript{14}) ✓
- **Arc Kayles**: *no* known hardness
- **Domineering**: *none*
- **Clobber**: NP-hard on general graphs. (AGNW 2005\textsuperscript{15})
- **NoGo**: planar graphs (B, Hearn \textsuperscript{16})

\textsuperscript{10}www.sciencedirect.com/science/article/pii/0022000078900454
\textsuperscript{11}Not yet published.
\textsuperscript{12}Not yet published
\textsuperscript{13}https://link.springer.com/article/10.1007/BF00288964
\textsuperscript{14}https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\textsuperscript{15}https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\textsuperscript{16}Not yet published.
State of the Art

What is known to be hard now?

- **Node Kayles**: general graphs (Schaeffer, 1978\(^{10}\))
- **Snort**: planar graphs (B, Hearn \(^{11}\))
- **Col**: planar graphs (B, Hearn \(^{12}\))
- **Hex**: hexagonal grid (Reisch, 1981\(^{13}\)) ✓
- **Atropos**: hexagonal grid (B, Teng 2007\(^{14}\)) ✓
- **Arc Kayles**: *no known hardness*
- **Domineering**: *none*
- **Clobber**: NP-hard on general graphs. (AGNW 2005\(^{15}\))
- **NoGo**: planar graphs (B, Hearn \(^{16}\))
- **Sprouts**: *none*

\(^{10}\)www.sciencedirect.com/science/article/pii/0022000078900454
\(^{11}\)Not yet published.
\(^{12}\)Not yet published
\(^{13}\)https://link.springer.com/article/10.1007/BF00288964
\(^{14}\)https://link.springer.com/chapter/10.1007%2F978-3-540-77105-0_49
\(^{15}\)https://www.emis.de/journals/INTEGERS/papers/a1int2003/a1int2003.pdf
\(^{16}\)Not yet published.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

Why Classify?

Deep Learning untested for many games.

Deep Learning algorithms need huge data sets.

Is this game even competitive?

Not if it's in $\mathcal{P}$.

Hard games are better for competition.

Nature of $\mathcal{P}$ vs. $\mathcal{PSPACE}$.

Limits to $\mathcal{NP}$-approximation algorithms. Maybe to $\mathcal{PSPACE}$ as well.

Find the hardness, then use AI.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
  - Not if it’s in $P$. 
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
  - Not if it’s in P.
  - Hard games are better for competition.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
  - Not if it’s in P.
  - Hard games are better for competition.
- Nature of P vs. PSPACE.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
  - Not if it’s in $P$.
  - Hard games are better for competition.
- Nature of $P$ vs. $PSPACE$.
- Limits to $NP$-approximation algorithms. Maybe to $PSPACE$ as well.
"But... Deep Learning can solve all of this! Why should we bother to classify all these games?"

- Deep Learning untested for many games.
- Deep Learning algorithms need huge data sets.
- Is this game even competitive?
  - Not if it’s in $P$.
  - Hard games are better for competition.
- Nature of $P$ vs. $PSPACE$.
- Limits to $NP$-approximation algorithms. Maybe to $PSPACE$ as well.

Find the hardness, then use AI.
Thank you!
Thank you!

Thanks to Eric and GAG for hosting us!
Thank you!

Thanks to Eric and GAG for hosting us!
Extra thanks to Dan Burgess and Matt Ferland for proof-watching early versions of this talk.